Programmer Manual

Tektronix

BPA100 Series
Bluetooth Protocol Analyzer
Software Version 2.3

Application Programming Interface

071-1129-00

This document supports software version 2.3 and above.

www.tektronix.com

Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

HARDWARE WARRANTY

Tektronix warrants that the products that it manufactures and sells will be free from defects in materials and workmanship for a period of three (3) years from the date of shipment. If a product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; c) to repair any damage or malfunction caused by the use of non-Tektronix supplies; or d) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

SOFTWARE WARRANTY

Tektronix warrants that the media on which this software product is furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from date of shipment. If a medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which this software product is furnished, this software product is provided “as is” without warranty of any kind, either express or implied. Tektronix does not warrant that the functions contained in this software product will meet Customer’s requirements or that the operation of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period. If Tektronix is unable to provide a replacement that is free from defects in materials and workmanship within a reasonable time thereafter, Customer may terminate the license for this software product and return this software product and any associated materials for credit or refund.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX’ RESPONSIBILITY TO REPLACE DEFECTIVE MEDIA OR REFUND CUSTOMER’S PAYMENT IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

Getting Started

Setting Up the API

Programming Examples

Status and Events

Syntax and Commands

Appendices

Glossary and Index

List of Figures

List of Tables

Preface

This is the Programmer Manual for the BPA100 Series Bluetooth Protocol Analyzer. This manual provides information about the application programming interface of the BPA100 Series Bluetooth Protocol Analyzer.

Related Manuals

Table i: Related manuals

Language	Type	Part number
English Chinese Japanese	BPA100 Series Bluetooth Protocol Analyzer Installation Manual	071-1121-00
English	BPA100 Series Bluetooth Protocol Analyzer Software Version 2.3 User Manual	071-1128-00
English	BPA100 Series Bluetooth Protocol Analyzer Software Version 2.3 Application Programming Interfacer Manual	071-1129-00

Contacting Tektronix

Phone	1-800-833-9200*
Address	Tektronix, Inc. Department or name (if known) 14200 SW Karl Braun Drive P.O. Box 500 Beaverton, OR 97077 USA
Web site	www.tektronix.com
Sales support	1-800-833-9200, select option 1*
Service support	1-800-833-9200, select option 2*
Technical support	Email: techsupport@tektronix.com 1-800-833-9200, select option 3* 6:00 a.m. – 5:00 p.m. Pacific time

*** This phone number is toll free in North America. After office hours, please leave a voice mail message.**
Outside North America, contact a Tektronix sales office or distributor; see the Tektronix web site for a list of offices.

Getting Started

Introduction

The Application Programming Interface (API) of the BPA100 Series Bluetooth Protocol Analyzer is based on the Microsoft Component Object Model (COM). The API allows you to control the protocol analyzer with a user program on the Server or Remote Client. Figure 1-1 depicts various ways the protocol analyzer can be controlled.

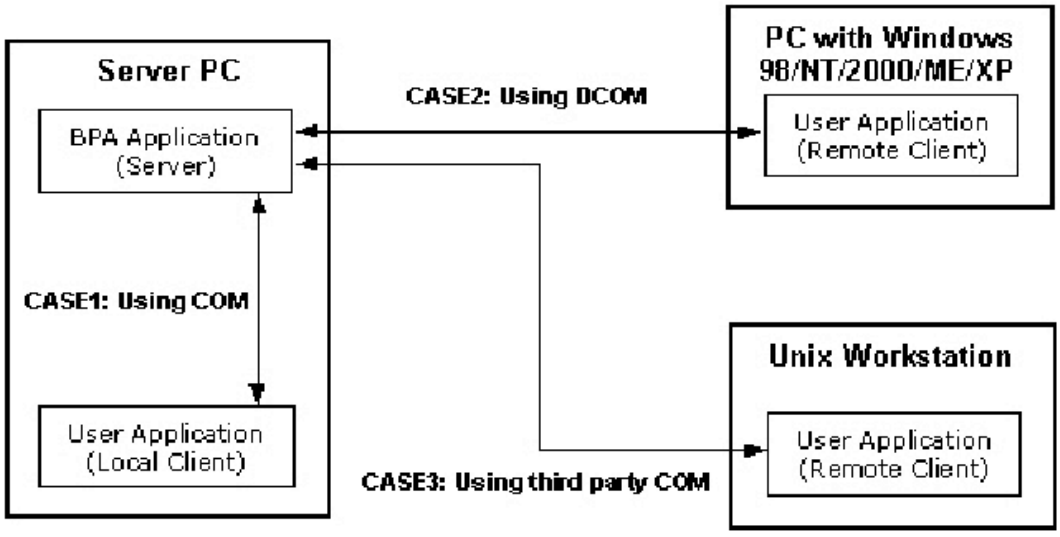


Figure 1-1: Controlling BPA100 Series Bluetooth Protocol Analyzer

Protocol analyzer Server: Computer where Tektronix Bluetooth Protocol Analyzer device is connected and Tektronix Bluetooth Protocol Analyzer application is installed.

Remote Clients: Computer that tries to connect to the Tektronix Bluetooth Protocol Analyzer device connected to the Protocol analyzer Server.

The Protocol analyzer application is called the server and the user program is called the client.

The following three cases are shown in Figure 1-1:

- Case 1 shows the user program running on the server and communicating with the Protocol analyzer application using Microsoft COM.
- Case 2 shows the user program running on another computer and communicating with the Protocol analyzer using Microsoft DCOM (Distributed COM).
- Case 3 shows the user program running on a UNIX workstation and communicating with the Protocol analyzer application using DCOM provided by a third party vendor.

The user program can be written in any language or programming environment that supports the Microsoft Component Object Model (COM). Some examples are Visual C++, Visual Basic, and LabView.

General Characteristic

The following is a list of the general characteristics of the API:

- API is consistent with programming interfaces exported by other Windows applications.
- All interfaces exported by the server are dual interfaces, which supports static and dynamic binding.
- The Protocol analyzer application must be initialized before a client tries to connect. This includes dismissing any errors that occur at startup time.

NOTE. *If a client attempts to connect before the application is initialized, an error message appears indicating access is denied.*

- A client either launches a session automatically or connects to an existing session of the server, if there is one.
- You must launch the server before the remote client can connect because Microsoft Windows 98 does not allow a client running on a remote host to launch a session on the server automatically.

- Clients hide the server's window using the programming interface. If the window is displayed, you can directly interact with the server and the status bar of the main window displays the connection status.
- The Protocol analyzer server continues to run after a client has disconnected. The server window is always visible even when clients are not connected.
- API operates within the main thread of the application.
- API supports multiple clients. Ensure that the clients do not interfere with one another.
- There is no provision for an API client to block other clients.

Software Requirements

API is supported by Tektronix Bluetooth Protocol Analyzer V2.2.1 or higher. If you have an older version of the Tektronix Bluetooth Protocol Analyzer application, you must upgrade to a recent version. You can download the new version of the Tektronix Bluetooth Protocol Analyzer application from the Tektronix Web site: http://www.tektronix.com/bpa_support.

Objects and Interfaces

Overview

The application programming interface for the Protocol analyzer consists of the following objects:

- BPAApplication. You can create a BPAApplication object to connect to the application and obtain a reference to additional objects. The BPAApplication object exports a single interface called IBPAApplication.
- BPASystem. The BPASystem object provides methods for controlling most of the functionality in the Tektronix Bluetooth-Protocol Analyzer GUI. The BPASystem object exports a single interface called IBPASystem.
- BPASystemEvents. The BPASystemEvents object provides methods for capturing events fired by the BPA application. The BPASystemEvents object exports a single interface called IBPASystemEvents.
- BPAAnalyzer. The BPAAnalyzer object provides methods for analyzing the log file provided by the Protocol analyzer Analyzer. The BPAAnalyzer object exports a single interface called IBPAAnalyzer.
- BPAHCISimple. The BPAHCISimple object provides methods for sending or receiving HCI command. The BPAHCISimple object exports a single interface called IBPAHCISimple.
- BPAHCISimpleEvents. The BPAHCISimpleEvents object provides methods for capturing events fired by the Protocol analyzer hardware. The BPAHCISimpleEvents object exports a single interface called IBPAHCISimpleEvents.

By default, all methods are synchronous and return the values after completing an operation.

The object hierarchy is depicted by Figure 1–2.

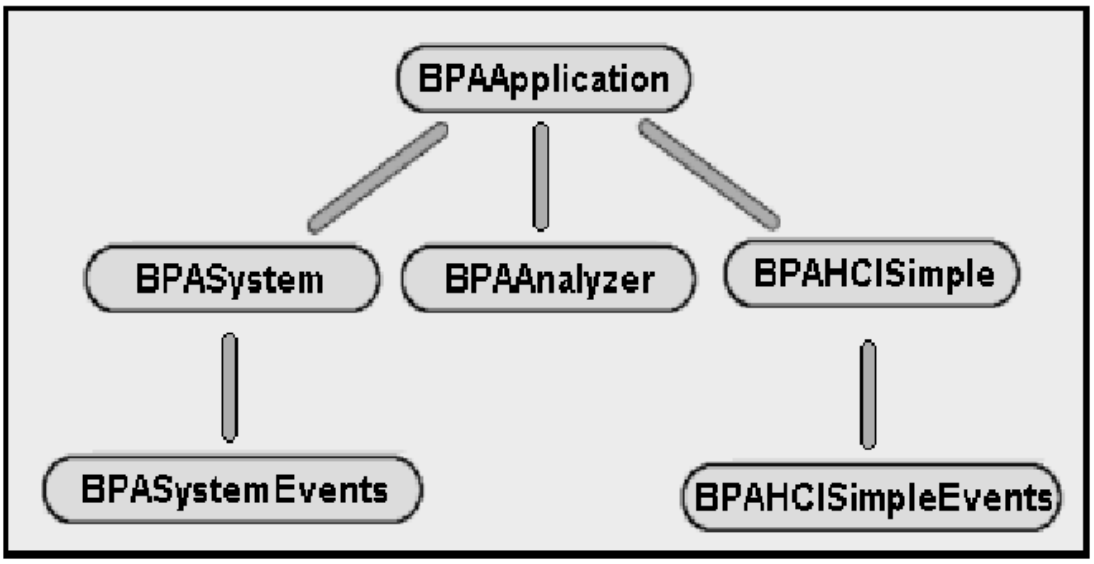


Figure 1-2: Object hierarchy

Setting up the API

Setting Up the API

In the following procedures, <install directory> refers to the directory where the API client application has been installed on your client machine. The install directory is **C:\Program Files\Tektronix Bluetooth Protocol Analyzer** by default.

The type library to be used with the API is BPA100.tlb. After you finish the following setup procedure, this file is located in **C:\Program Files\ Tektronix Bluetooth Protocol Analyzer\System\API** and in the <install directory>\System\API on your API client.

You have two methods of using the API with your client application:

- You can run the client application locally on the server where your Protocol Analyzer application is running. No special setup is required once the Protocol Analyzer application is installed. The following section, *Running a client application on the server machine,* shows how you can run a client application on the server.
- You can run your client application remotely across the network. In this case, both the server and the remote client require special setup procedures. Refer to the section, *Running a Client Application across the Network,* on page 2–3.

Running a Client Application on the Server Machine

If you are running a client application on the Server, no special setup is required if the Tektronix Bluetooth Protocol Analyzer application has already been installed. You can just run the client application that you have created.

To verify that a client application can connect to the Protocol Analyzer Server, do the following steps:

1. Start the Protocol Analyzer application on the computer where the Protocol Analyzer device is connected.
2. Run **testclient.exe** from <install directory>\Samples\API Samples\VC++\test client.

3. Click the connect button to check whether the client application connects to the Protocol Analyzer Server.
4. Click **Exit**.

Running a Client Application across the Network

If you want to run the client application remotely across the network, you must set up the API on the server machine and the client machine. Use the following procedures as appropriate:

Setting up the API on the Server Machine

Follow these steps to set up the API on the server:

1. Install and configure TCP/IP.

***NOTE**. If you have difficulty in configuring the network setup, contact your system administrator.*

---

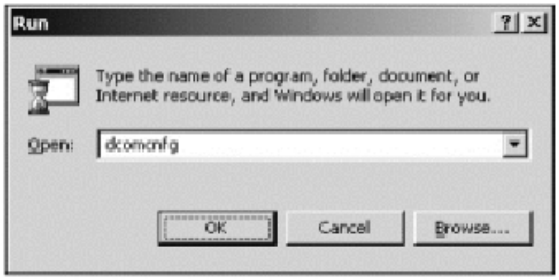
2. Select the appropriate access type. You may choose share-level access (page 2-4) or user-level access (page 2-12) to the server as provided by Microsoft Windows.

***NOTE**. For the API to work with share-level access, authentication is turned off and any COM client application can call in to a COM server running on the server.*

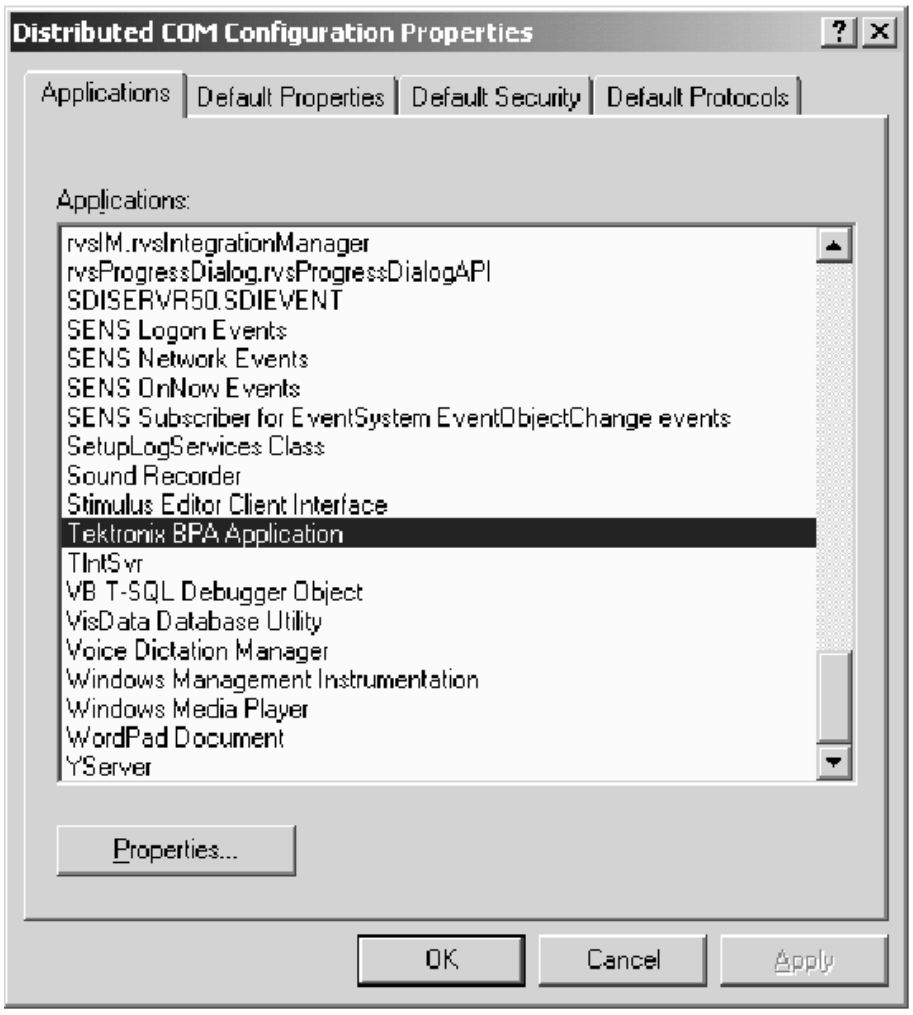
Share-Level Access

Follow these steps to set up the server to be shared among different users on a network:

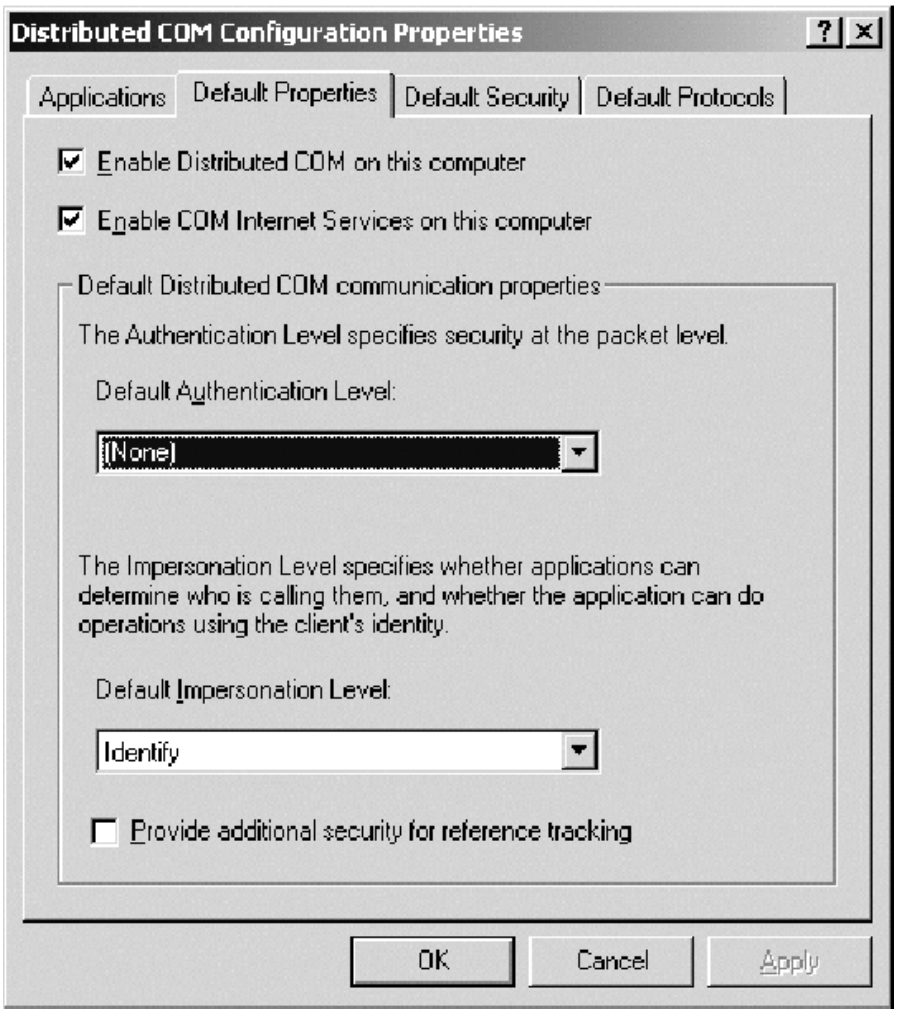
1. Click **Start> Run**. The Run dialog box appears.



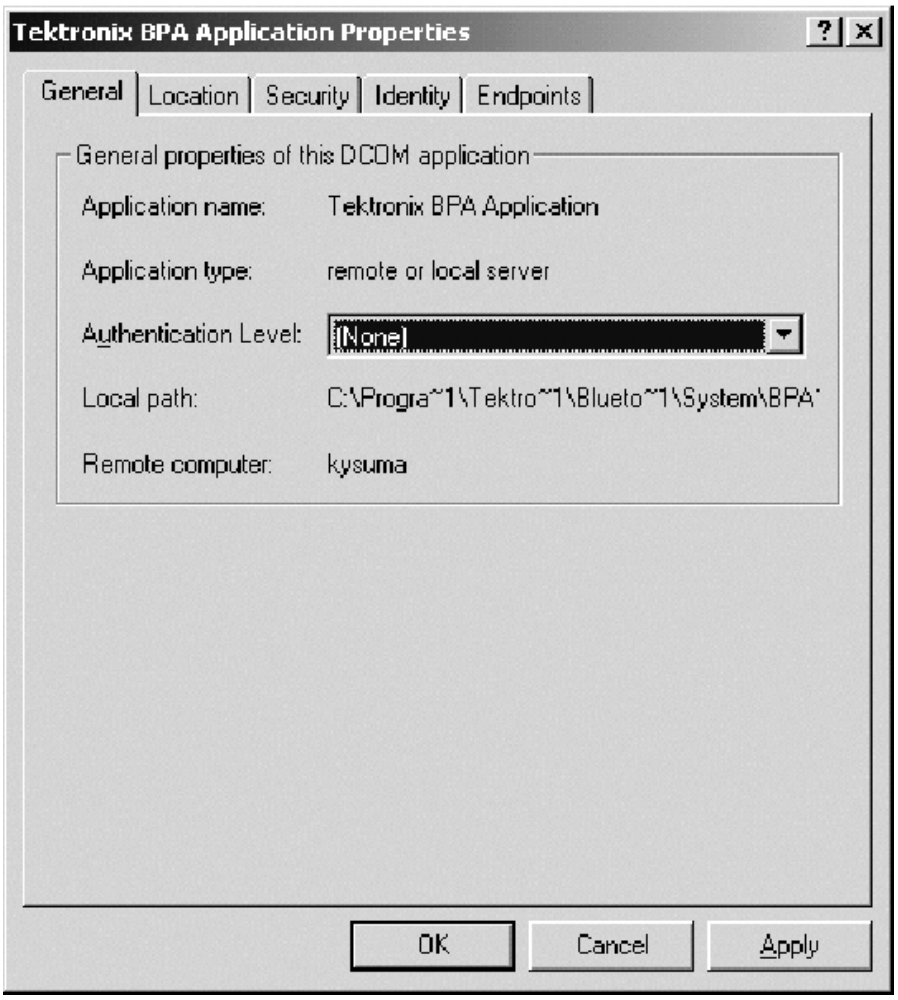
2. Type **dcomcnfg** in the Open field and click **OK**. The Distributed COM configuration properties box appears.


Distributed COM Configuration Properties
Applications
Default Properties
Default Security
Default Protocols
Applications:
rvslM.rvsIntegrationManager
rvsProgressDialog.rvsProgressDialogAPI
SDISERVR50.SDIEVENT
SENS Logon Events
SENS Network Events
SENS OnNow Events
SENS Subscriber for EventSystem EventObjectChange events
SetupLogServices Class
Sound Recorder
Stimulus Editor Client Interface
Tektronix BPA Application
TIntSvr
VB T-SQL Debugger Object
VisData Database Utility
Voice Dictation Manager
Windows Management Instrumentation
Windows Media Player
WordPad Document
YServer
Properties...
OK
Cancel
Apply

3. Click the **Default Properties** tab.

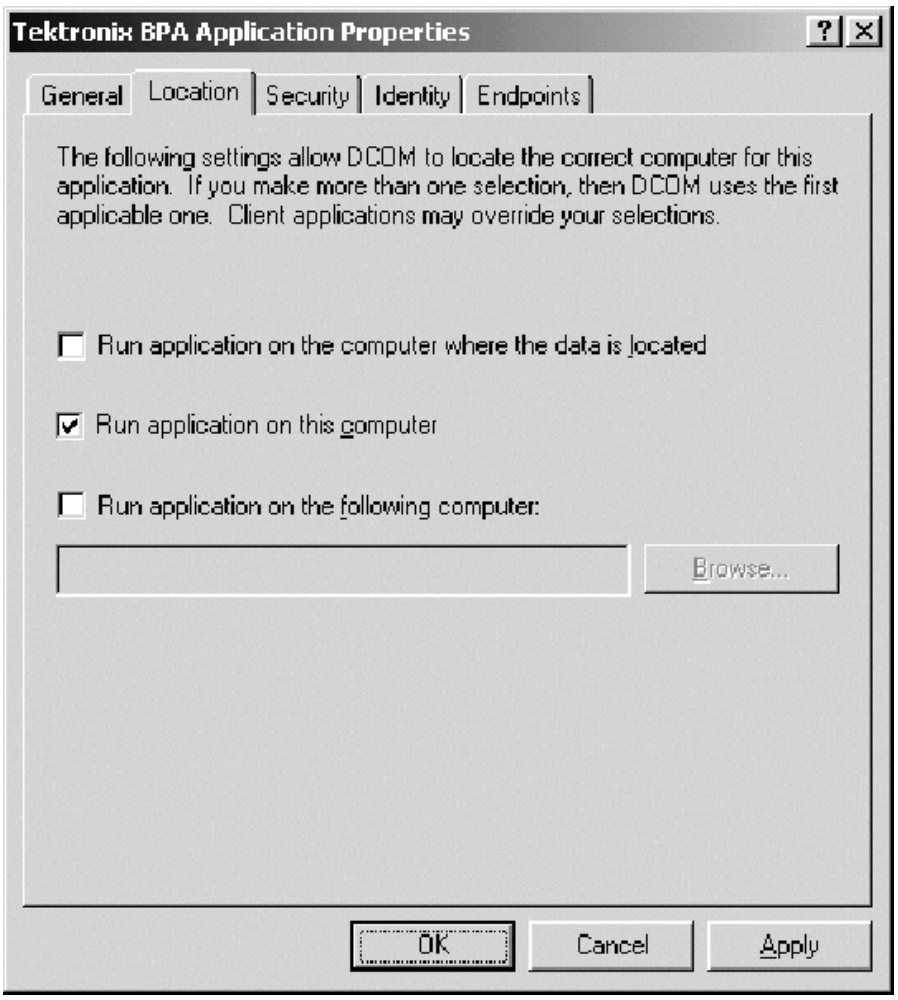


4. Select **None** in the Default Authentication Level drop down list.
5. Click the **Applications** tab and select the Tektronix Bluetooth Protocol Analyzer Application in the Applications list.
    a. Click the **Properties** button to display the Tektronix BPA Application Properties window as shown in the following figure.



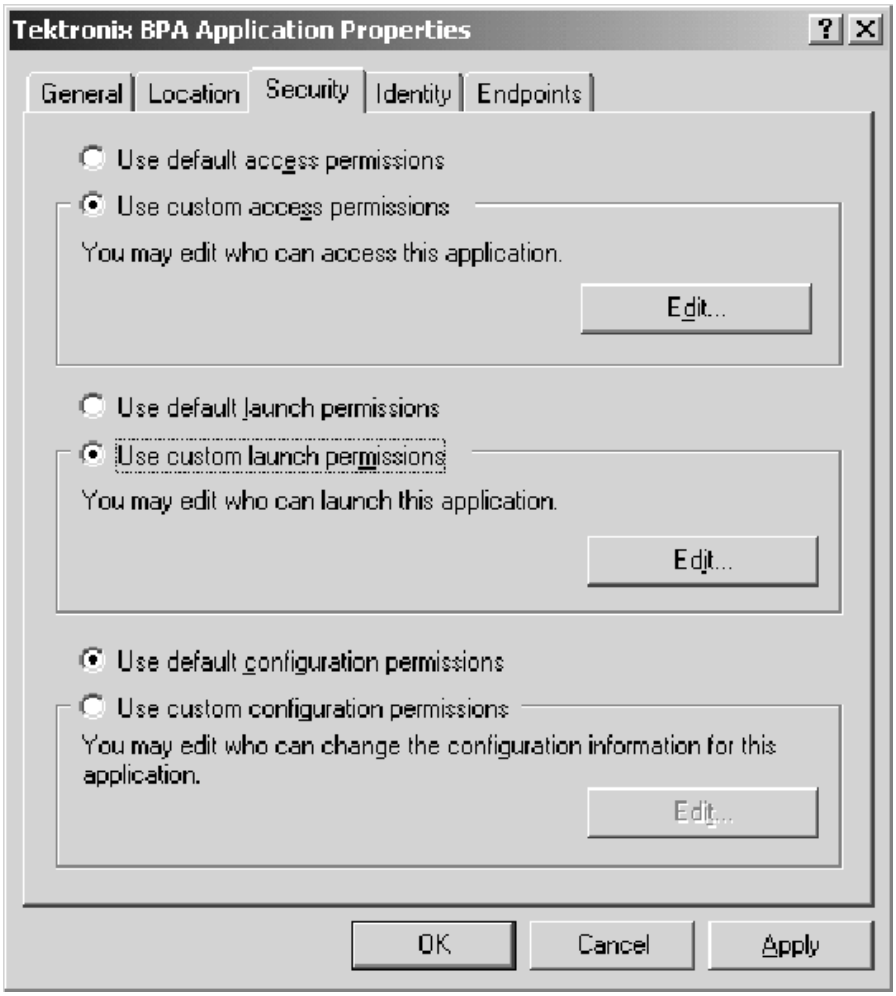
b. Select **None** in Authentication Level drop down list.

c. Click the **Location** tab.



d. Ensure that Run application on this computer check box is selected.

e. Click the **Security** tab.



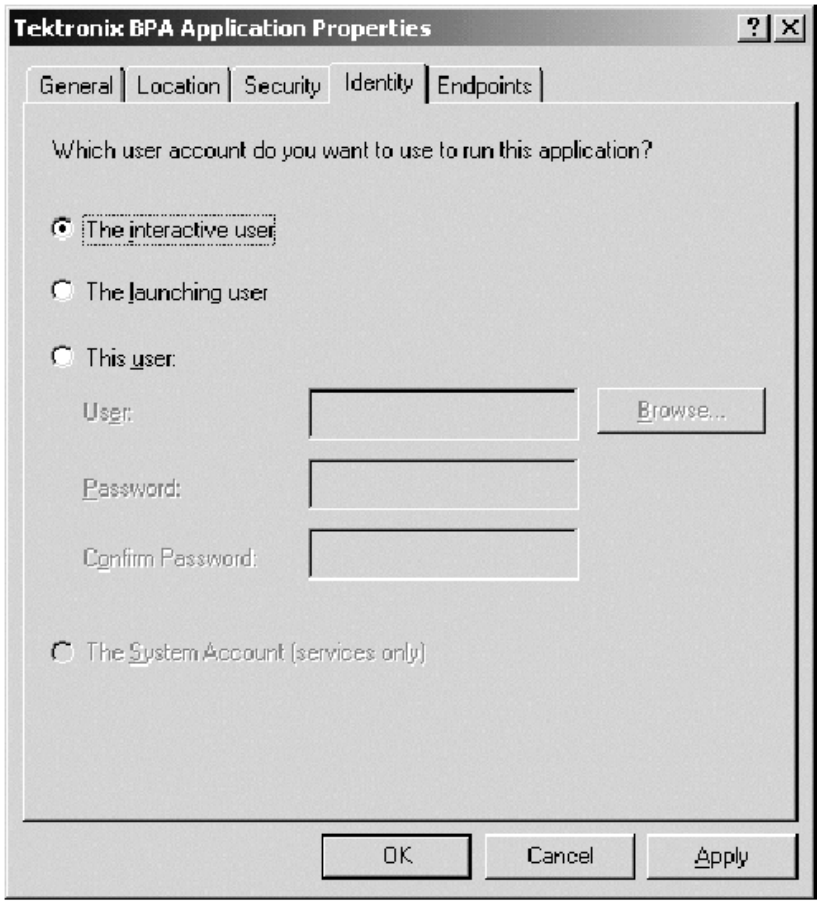
f. Select the **Use custom access permissions** check box.

- Click **Edit** to open Registry Value Permissions window.
- Click **Add** to display Add Users and Groups window.
- Select **Everyone** in the Names list.
- Click **Add** and then click **OK**.

g. Select the **Use custom launch permissions** check box.

- Click **Edit** to open Registry Value Permissions window.
- Click **Add** to display Add Users and Groups window.
- Select **Everyone** in the Names list.
- Click **Add** and then click **OK**.

h. Click the **Identity** tab.



i. Select **The interactive user** check box.

j. Click **Apply** and then click **OK**.

6. Exit the Protocol Analyzer application and restart. Do not attempt to make any connections before restarting the Protocol Analyzer application.

This completes the Protocol Analyzer Server setup for operation with a remote client application using share-level access.

NOTE*. You can switch between user-level access and share-level access later by repeating the procedure starting from step 2 of* Setting up the API on the Server Machine *on page 2-3.*

Next, you need to set up the client application to connect to the Protocol Analyzer Server. Select the appropriate setup:

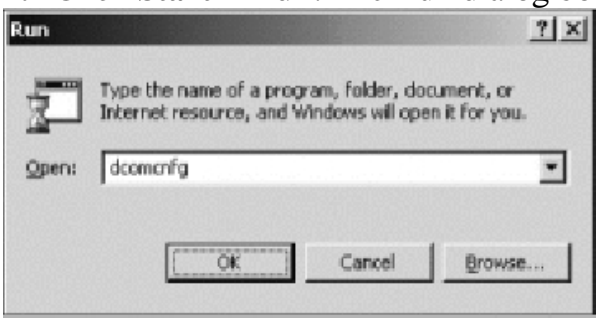
- *Setting up the Client Machine on Windows 2000/NT*, see page 2-20
- *Setting up the Client Machine on Windows 98,* see page 2-37
- *Setting up the Client Machine on Windows 95,* see page 2-46
- *Setting up the Client Application on Other Platforms,* see page 2-54

User-level access

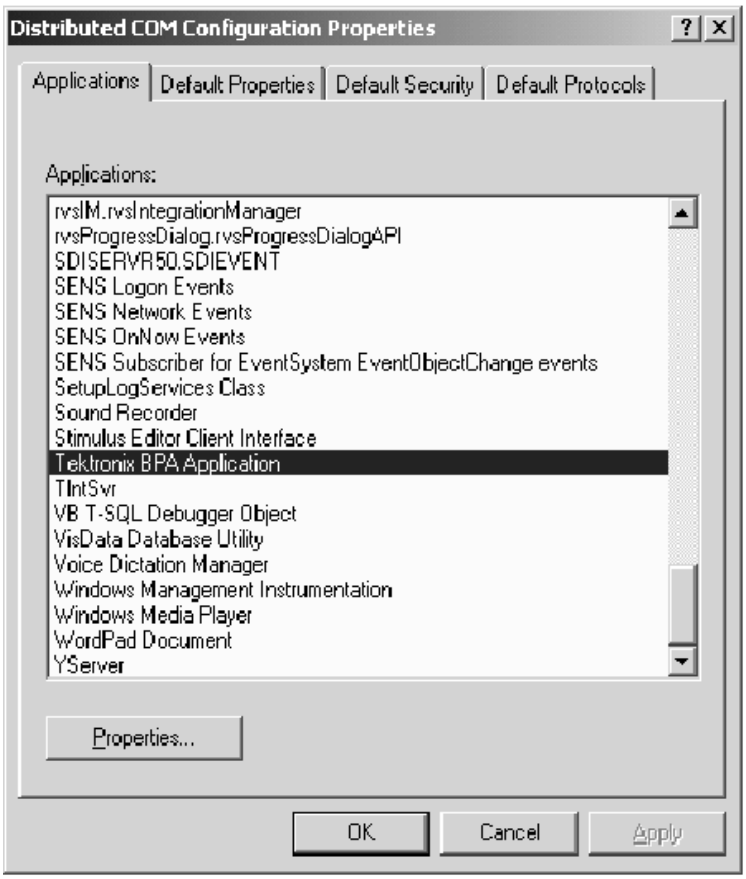
The default network setup for the Protocol Analyzer under Windows is compatible with clients operating with user-level access. With these settings, the client machine and server must be logged in to the same account and domain to make a connection. If this is too restrictive, use share-level access (see page 2–4) or talk to your network administrator.

To set up the Protocol Analyzer Server for user-level access (default settings), follow these steps:

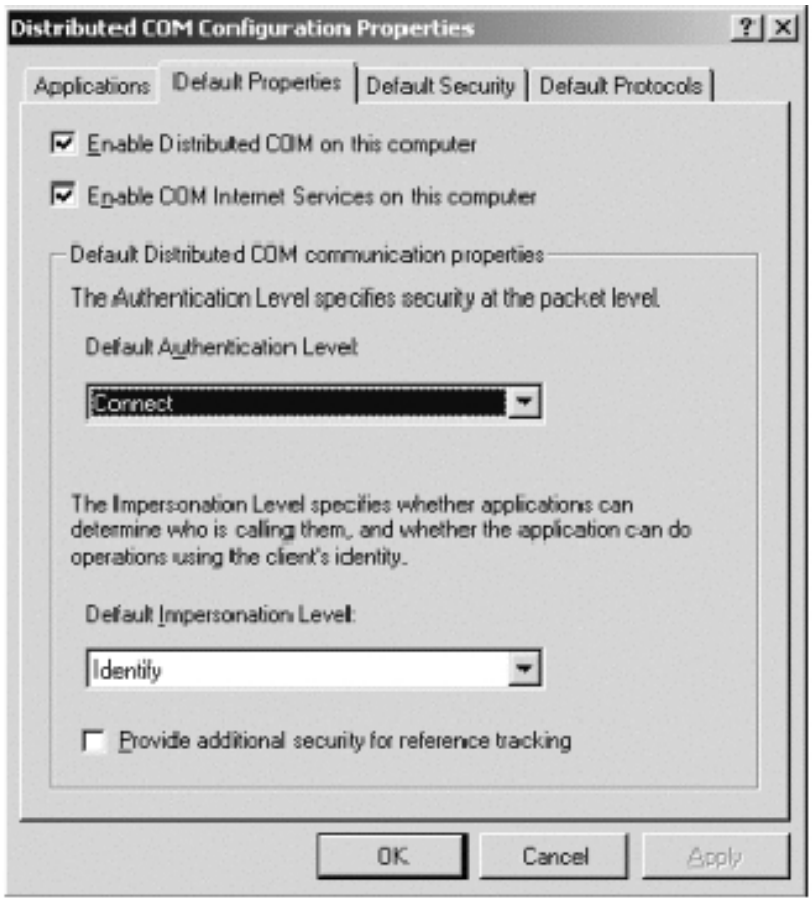
1. Click **Start > Run**. The Run dialog box appears.



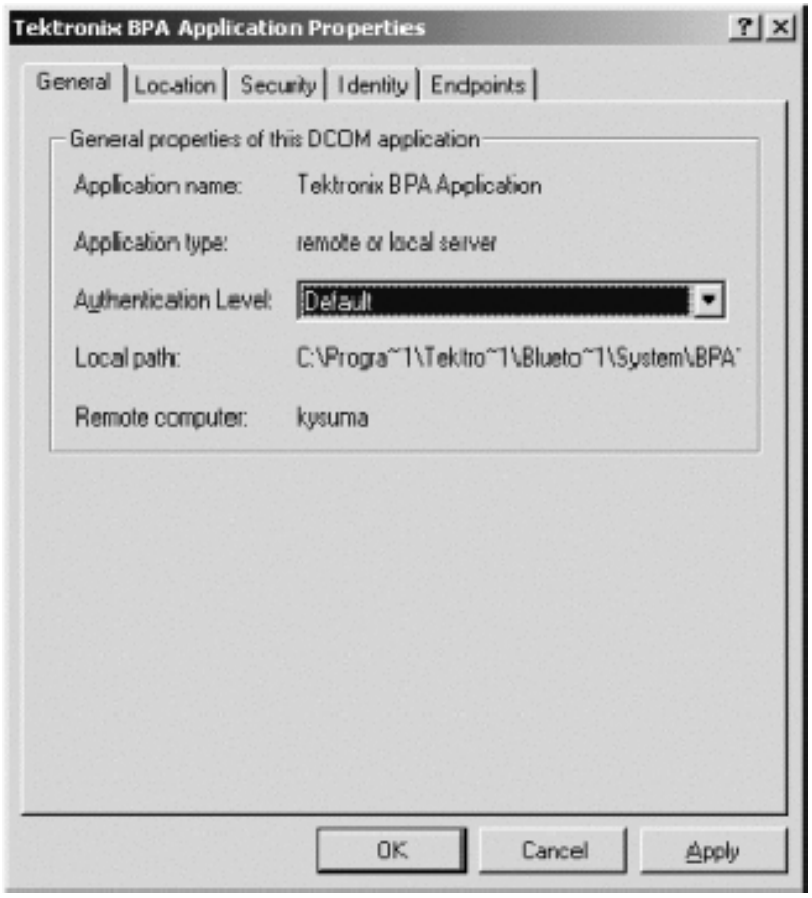
2. Type **dcomcnfg** in the Open field and click **OK**. The Distributed COM configuration properties box appears.


Distributed COM Configuration Properties
Applications
Default Properties
Default Security
Default Protocols
Applications:
rvsIM.rvsIntegrationManager
rvsProgressDialog.rvsProgressDialogAPI
SDISERVR50.SDIEVENT
SENS Logon Events
SENS Network Events
SENS OnNow Events
SENS Subscriber for EventSystem EventObjectChange events
SetupLogServices Class
Sound Recorder
Stimulus Editor Client Interface
Tektronix BPA Application
TIntSvr
VB T-SQL Debugger Object
VisData Database Utility
Voice Dictation Manager
Windows Management Instrumentation
Windows Media Player
WordPad Document
YServer
Properties...
OK
Cancel
Apply

3. Click the **Default Properties** tab.

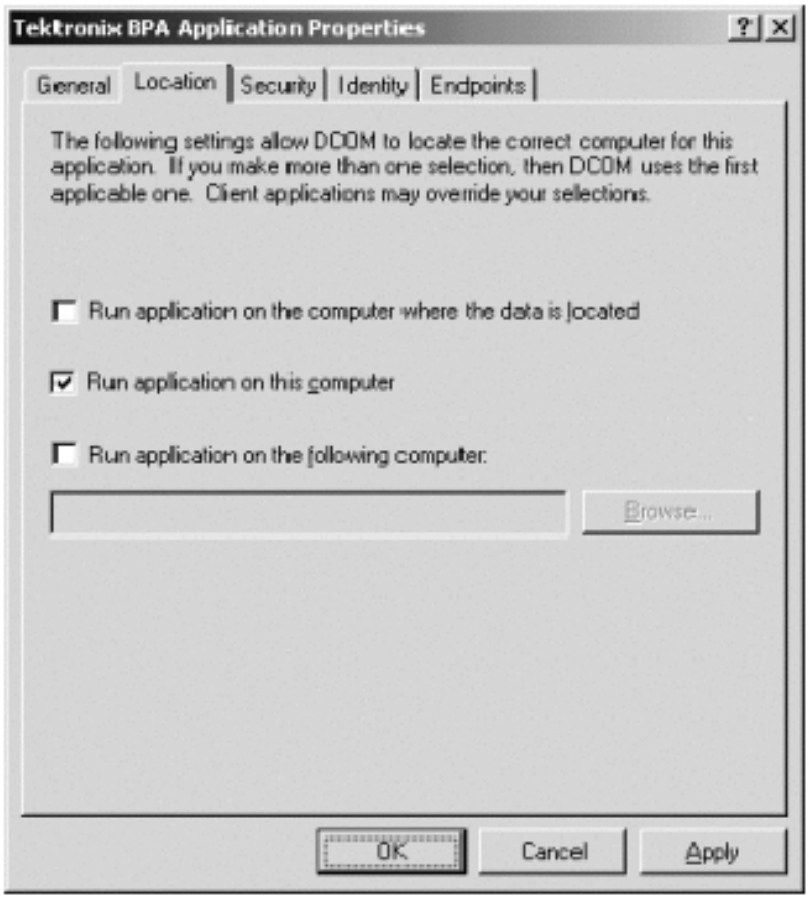


4. Select **Connect** in the Default Authentication Level drop down list.
5. Click the **Applications** tab and select the Tektronix Bluetooth Protocol Analyzer Application in the Applications list.
    a. Click the **Properties** button to display the Tektronix BPA Application Properties window as shown in the following figure.



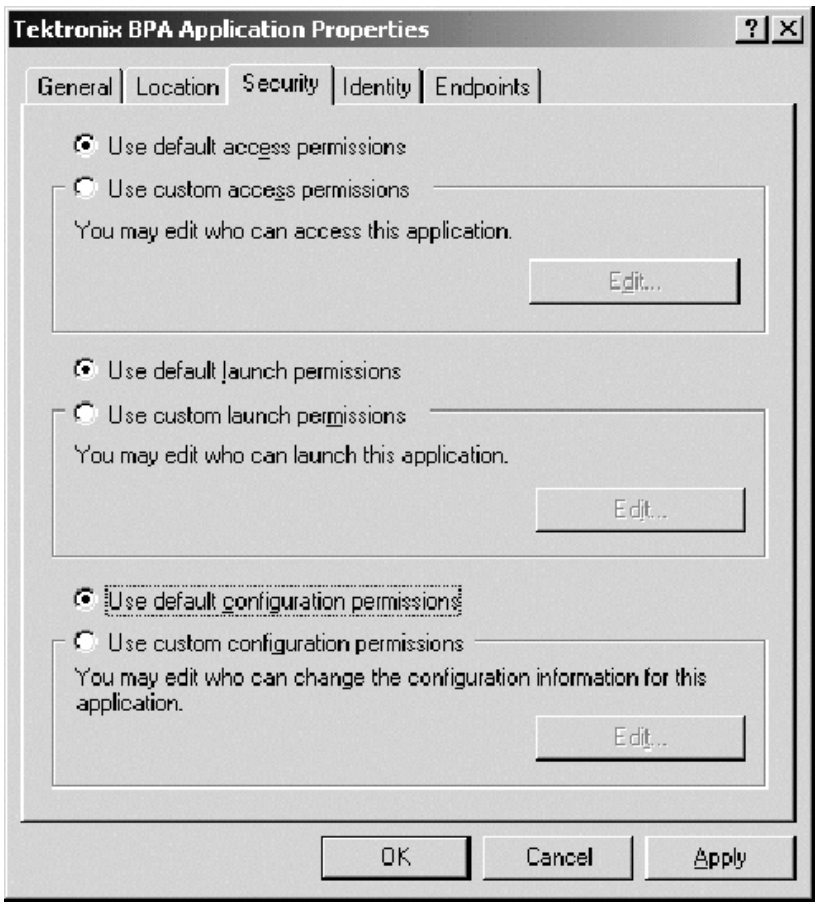
b. Select **Default** in Authentication Level drop down list.

c. Click the **Location** tab.



d. Ensure that the Run application on this computer check box is selected.

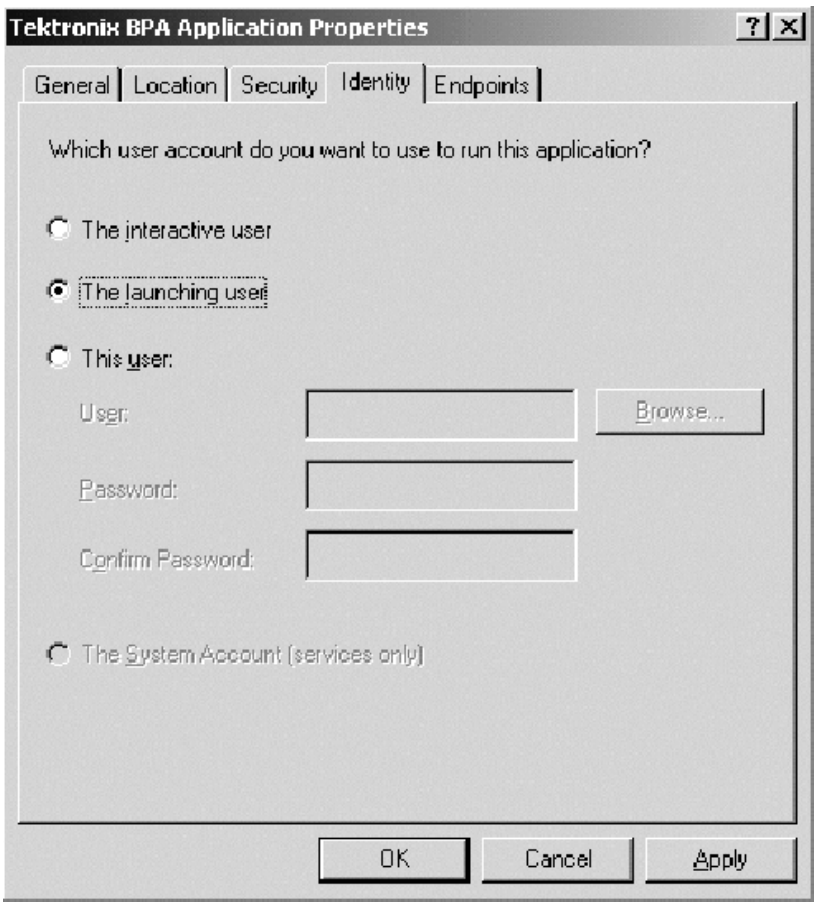
e. Click the **Security** tab.



f. Follow these steps:

- Select **Use default access permissions** check box.
- Select **Use default launch permissions** check box.
- Select **Use default configuration permissions** check box.

g. Click the **Identity** tab.



h. Select **The launching user** check box.

i. Click **Apply** and then click **OK**.

6. Exit the Protocol Analyzer application, and restart. Do not attempt to make any connections before restarting the Protocol Analyzer application.

This completes the setup of the Protocol Analyzer Server for operation with a remote client application using user-level access or Windows 2000/NT user authentication.

NOTE. *You can switch between user-level access and share-level access later by repeating the procedure starting from step 2 of* Setting up the API on the Server machine *on page 2-3.*

Next, you need to set up the client application to connect to the Protocol Analyzer Server. Select the appropriate setup:

- *Setting up the Client Machine on Windows 2000/NT*, see page 2-20
- *Setting up the Client Machine on Windows 98,* see page 2-37
- *Setting up the Client Machine on Windows 95,* see page 2-46
- *Setting up the Client Application on Other Platforms,* see page 2-54

Setting up the Client Machine on Windows 2000/NT

After you set up the Protocol Analyzer Server, you must set up the client application using the following procedure:

1. Install and configure TCP/IP.
2. Load the Tektronix Bluetooth Protocol Analyzer application software CD.
3. Double-click on **API client SW\Disk1\Setup.exe**.
4. Select the appropriate access type, share-level access (page 2-20) or user-level access (page 2-29). You must set up the client application to match the access level you chose for the Protocol Analyzer Server:

NOTE. *You can switch between user-level access and share-level access later by repeating the procedure from Step 4 onwards.*

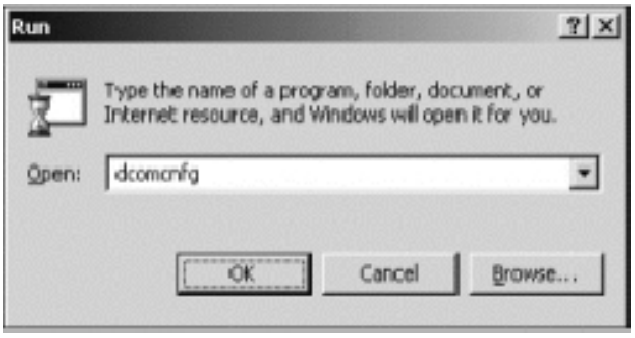
Share-level Access for Windows 2000/NT

To set up a client application to access a Protocol Analyzer server that is set up for share-level access, do the following procedure:

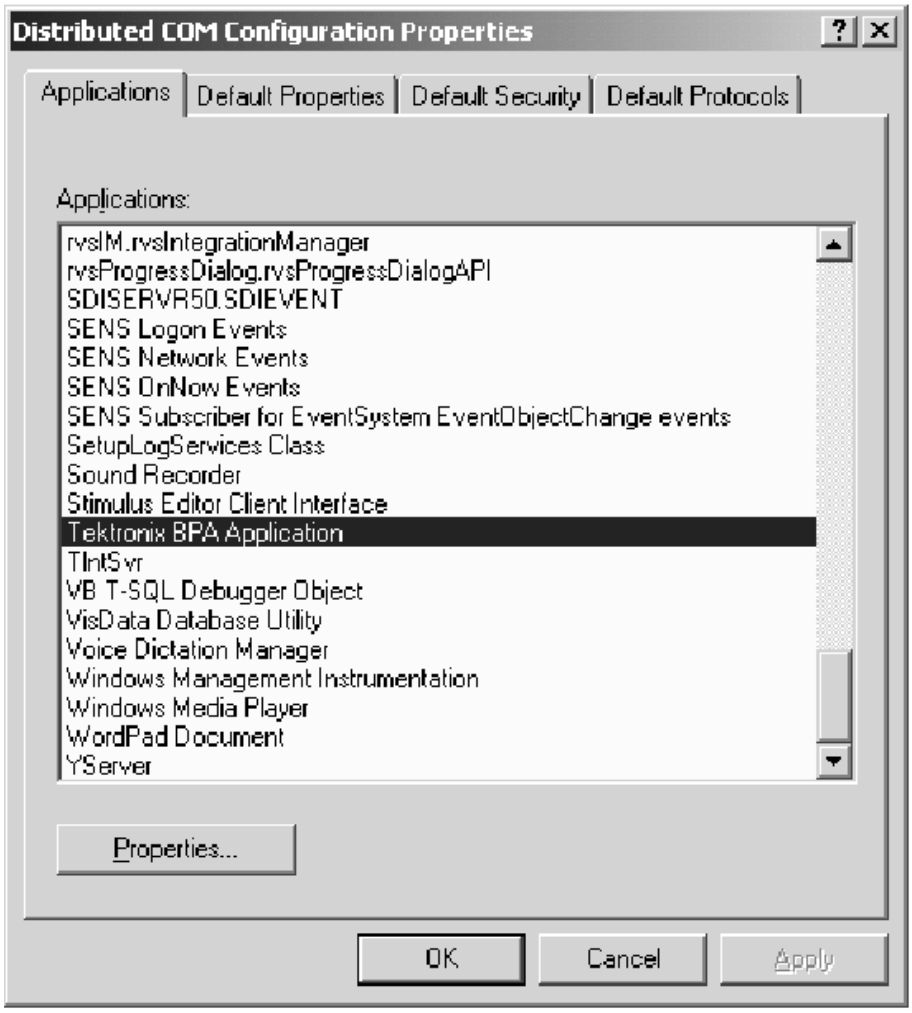
NOTE. *You must have administrative permissions on your computer.*

---

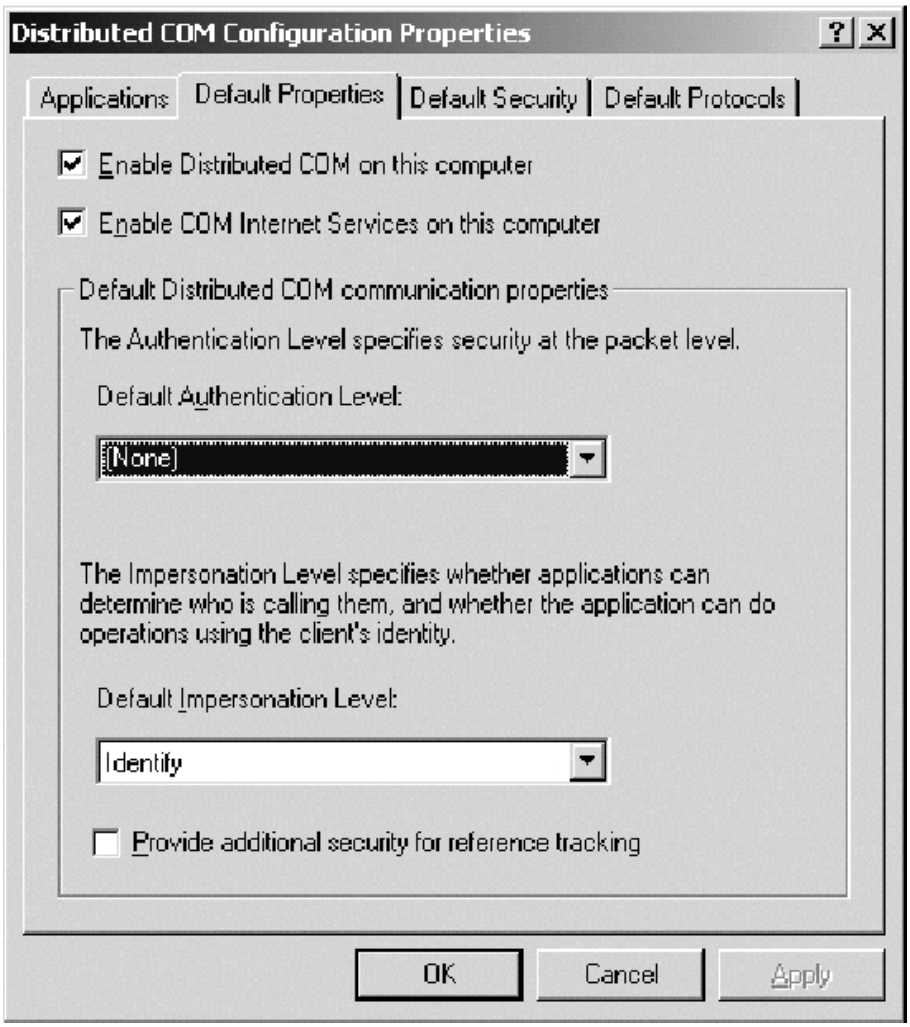
1. Click **Start > Run**. The Run dialog box appears.



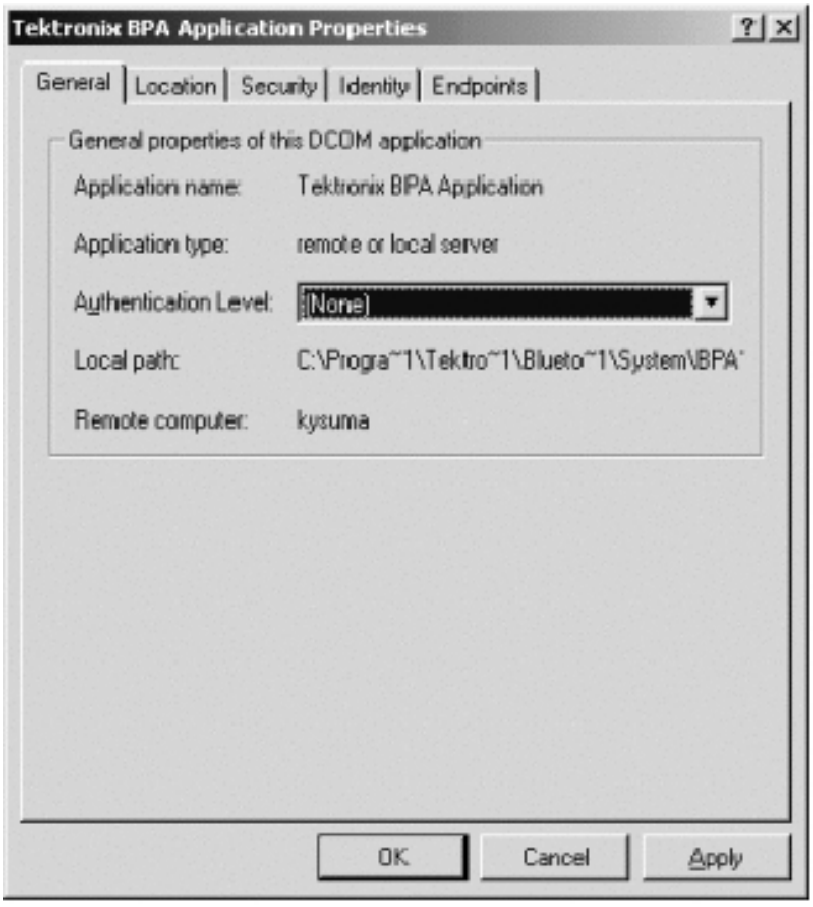
2. Type **dcomcnfg** in the Open field and click OK. The Distributed COM configuration properties box appears.



3. Click the **Default Properties** tab.

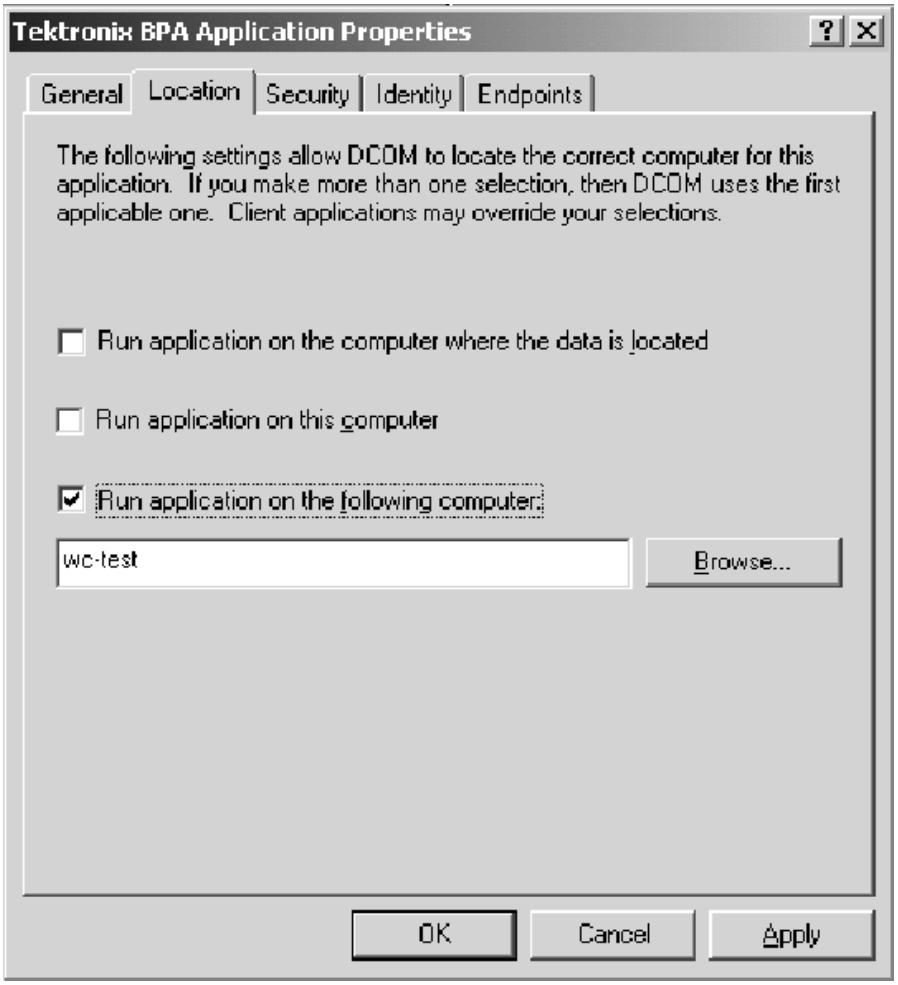


4. Select **None** in the Default Authentication Level drop down list.
5. Click the **Applications** tab and select the Tektronix Bluetooth Protocol Analyzer Application in the Applications list.
    a. Click the **Properties** button to display the Tektronix BPA Application Properties window as shown in the following figure.



b. Select **None** in Authentication Level drop down list.

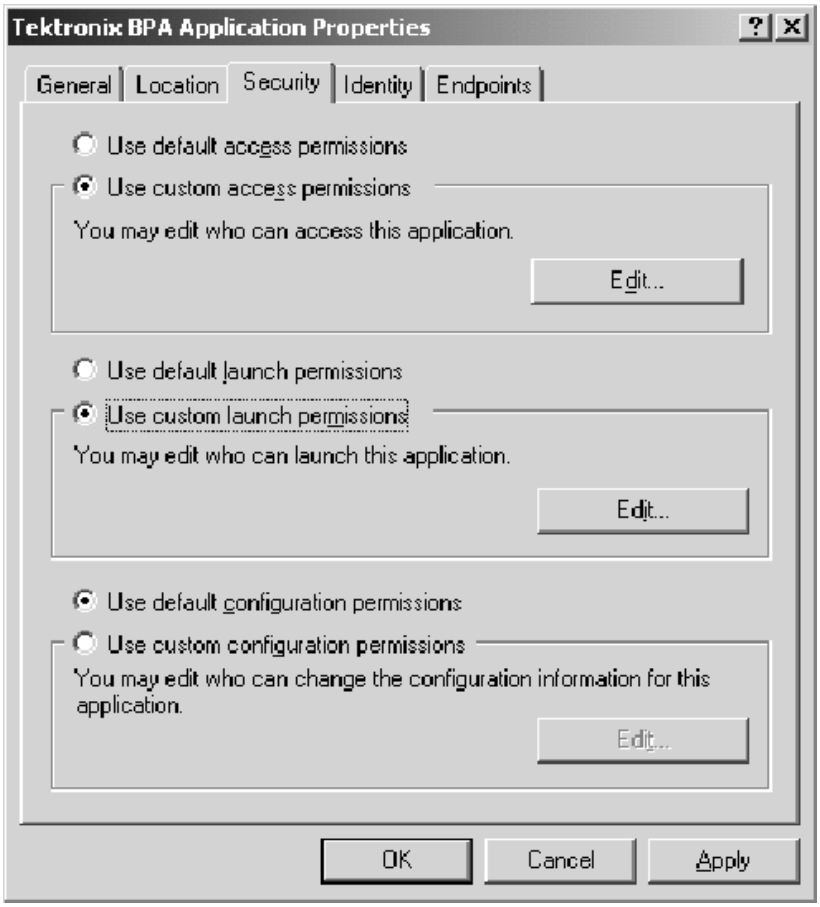
c. Click the **Location** tab.



d. Clear the Run application on this computer check box.

e. Select the Run application on the following computer check box and type the name of the Protocol Analyzer Server in the adjacent field.

f. Click the **Security** tab.



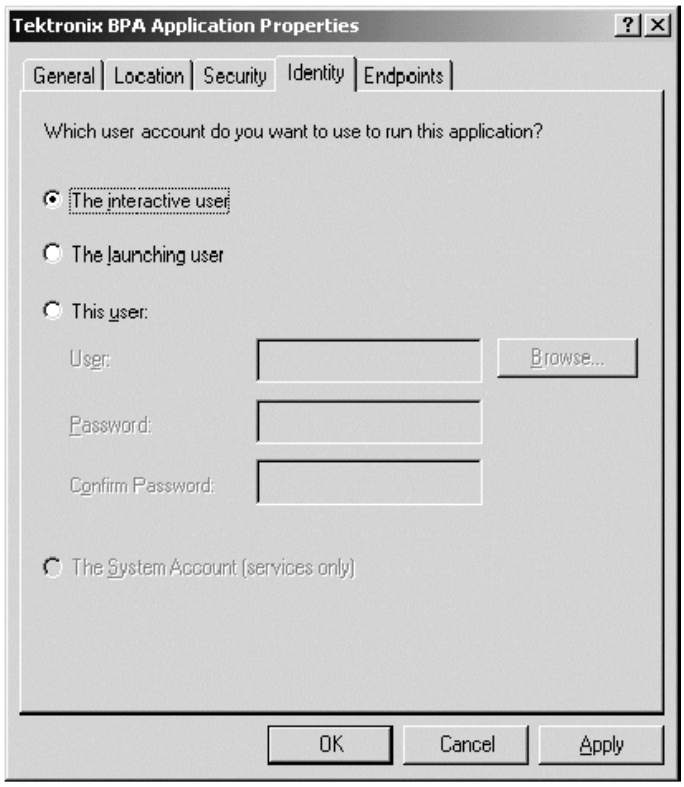
g. Select the **Use custom access permissions** check box.

- Click **Edit** to open Registry Value Permissions window.
- Click **Add** to open Add Users and Groups window.
- Select **Everyone** in the Names list.
- Click **Add** and then click **OK**.

h. Select the **Use custom launch permissions** check box.

- Click **Edit** to open Registry Value Permissions window.
- Click **Add** to open Add Users and Groups window.
- Select **Everyone** in the Names list.
- Click **Add** and then click **OK**.

i. Click the **Identity** tab.



j. Select **The Interactive user** check box.

k. Click **Apply** and then click **OK**.

6. To verify that the setup is complete:

   - Run **<install directory>\Samples\API Samples\VC++\test client\testclient.exe** on the client.

   - Click the connect button to check if the client application can connect to the Protocol Analyzer Server.

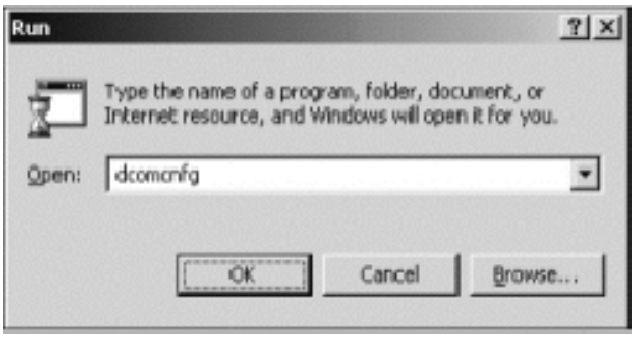
This completes the set up of the Protocol Analyzer Server and the client machine for running your client application across the network.

User-Level Access for Windows 2000/NT

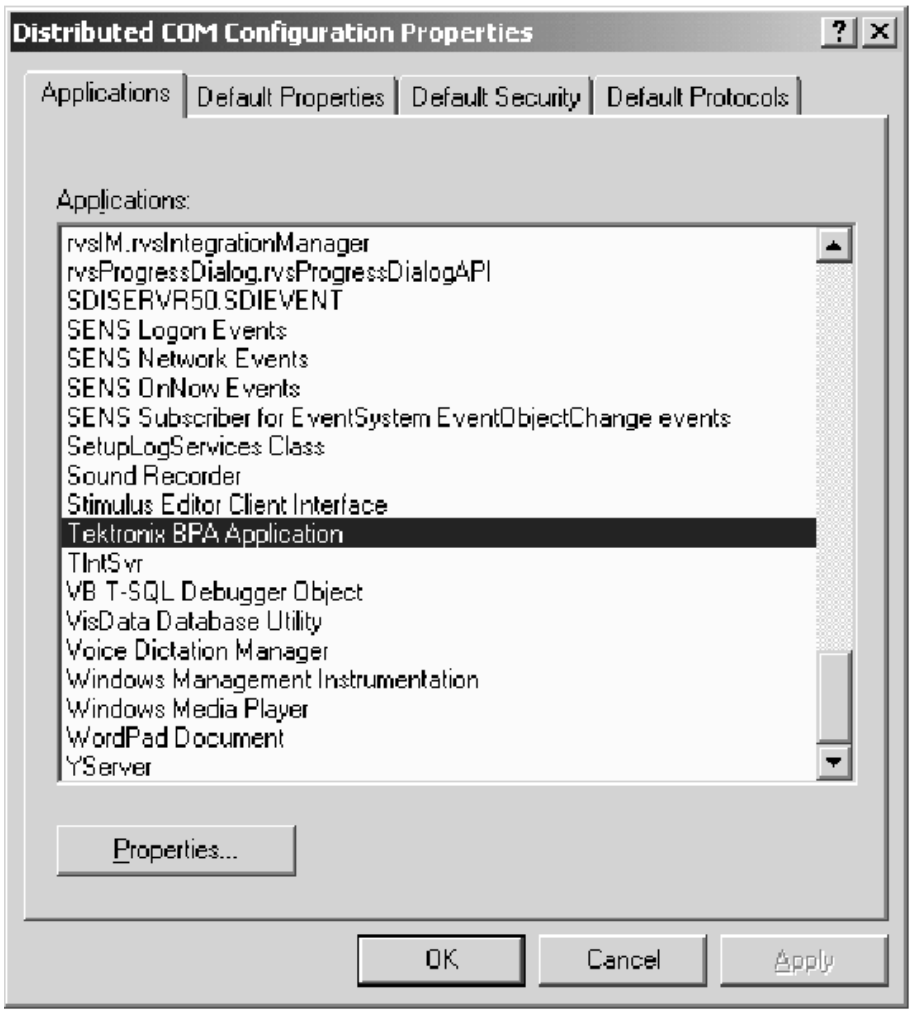
The default network settings are compatible with clients operating with user-level access. With these settings, the client machine and server must be logged in to the same account and domain to make a connection. If this is too restrictive, use share-level access (page 2-20) or talk to your network administrator.

To set up a client application to access a Protocol Analyzer server that is setup for user-level access, do the following procedure:

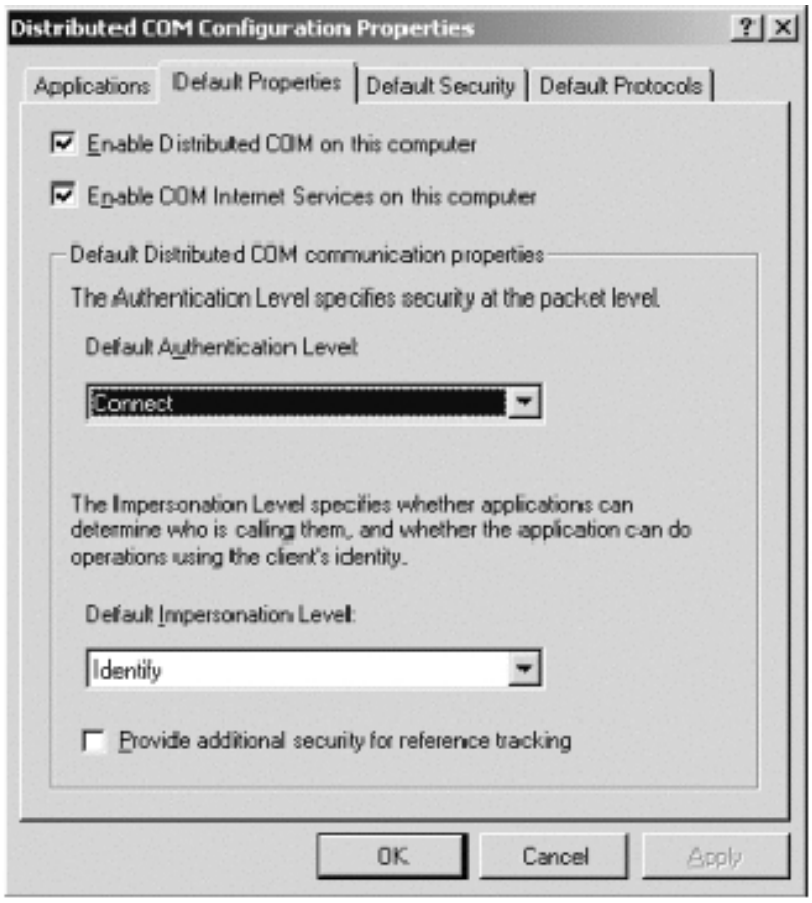
1. Click **Start > Run**. The Run dialog box appears.



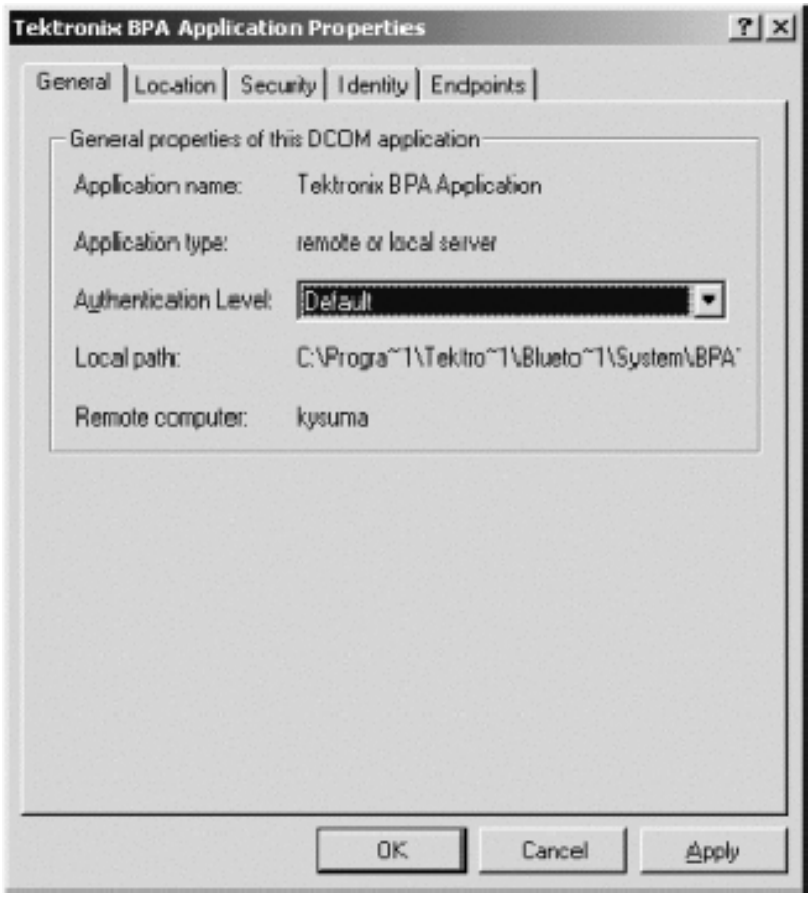
2. Type **dcomcnfg** in the Open field and click **OK**. The Distributed COM configuration properties box appears.


Distributed COM Configuration Properties
Applications
Default Properties
Default Security
Default Protocols
Applications:
rvsIM.rvsIntegrationManager
rvsProgressDialog.rvsProgressDialogAPI
SDISERVR50.SDIEVENT
SENS Logon Events
SENS Network Events
SENS OnNow Events
SENS Subscriber for EventSystem EventObjectChange events
SetupLogServices Class
Sound Recorder
Stimulus Editor Client Interface
Tektronix BPA Application
TIntSvr
VB T-SQL Debugger Object
VisData Database Utility
Voice Dictation Manager
Windows Management Instrumentation
Windows Media Player
WordPad Document
YServer
Properties...
OK
Cancel
Apply

3. Click the **Default Properties** tab.

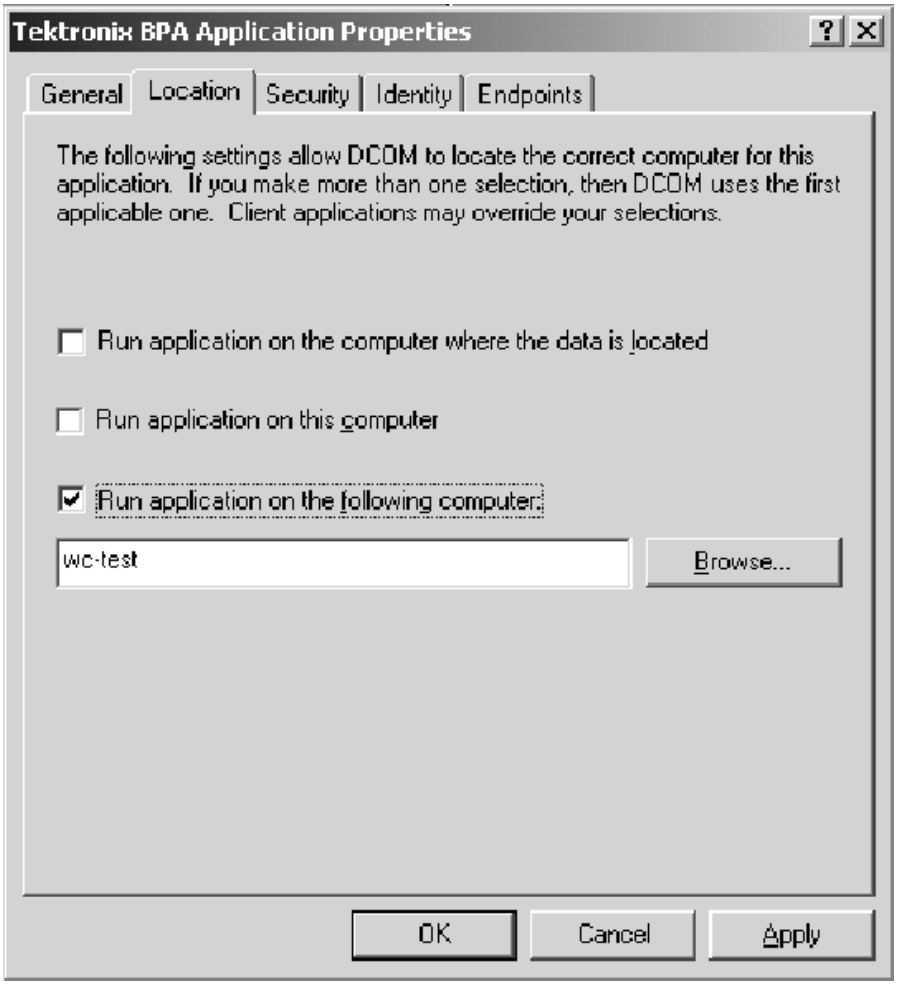


4. Select **Connect** in the Default Authentication Level drop down list.
5. Click the **Applications** tab and select the Tektronix Bluetooth Protocol Analyzer Application in the Applications list.
    a. Click the **Properties** button to display the Tektronix BPA Application Properties window as shown in the following figure.



b. Select **Default** in Authentication Level drop down list.

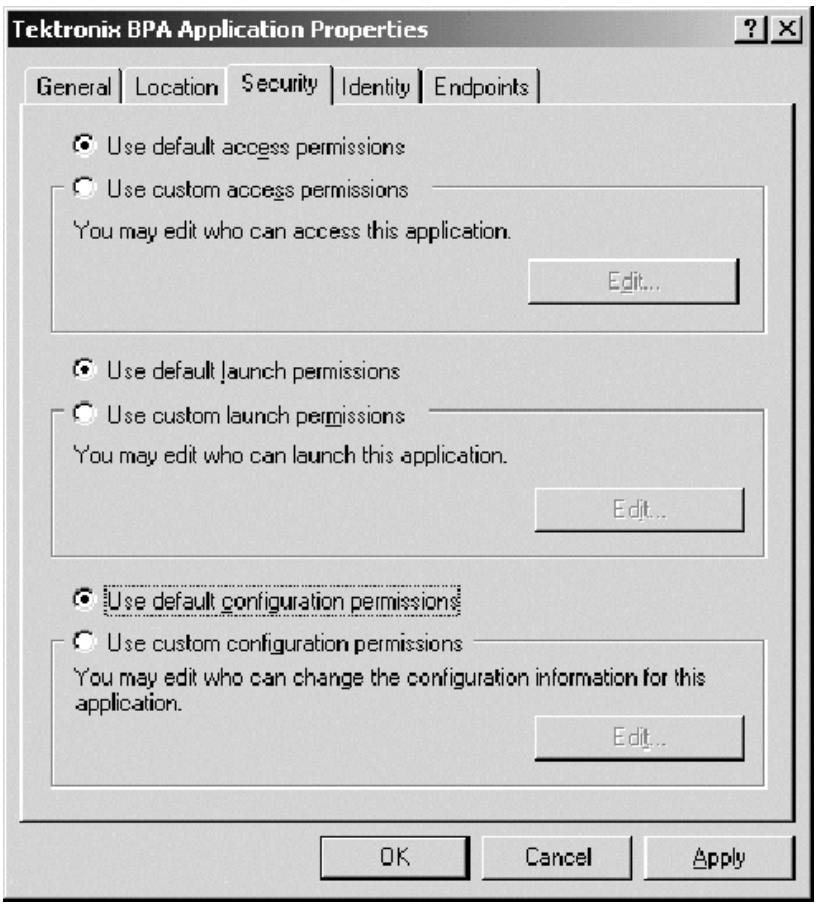
c. Click the **Location** tab.



d. Clear the **Run application on this computer** check box.

e. Select the **Run application on the following computer** check box and type the name of the Protocol Analyzer Server in the adjacent field.

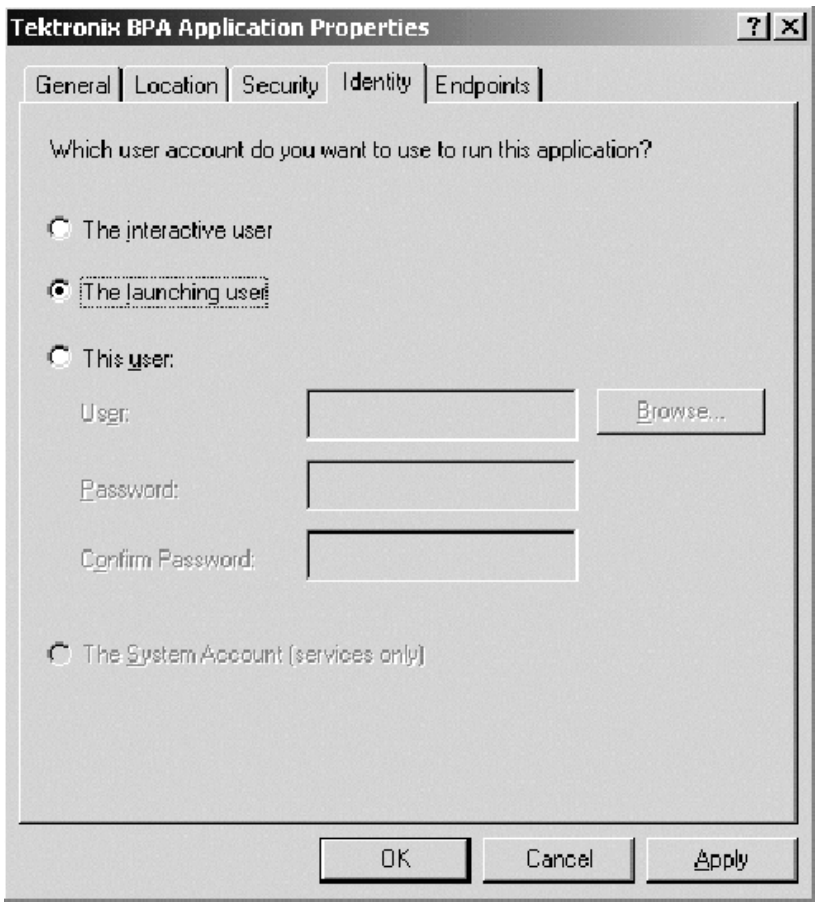
f. Click the **Security** tab.



g. Follow these steps.

- Select **Use default access permissions** check box.
- Select **Use default launch permissions** check box.
- Select **Use default configuration permissions** check box.

h. Click the **Identity** tab.



i. Select **The launching user** check box.

j. Click **Apply** and then click **OK**.

6. To verify that the setup is complete:

   - On the client machine, run **testclient.exe** from the directory <install directory>\Samples\API Samples\VC++\test client\. The Tektronix Test API client application window appears.

   - Click the Connect button to check if the client application can connect to the Protocol Analyzer Server.

This completes the setup of the Protocol Analyzer Server and the client machine for running your client application across the network.

Setting up the Client Machine on Windows 98

After you set up the Protocol Analyzer Server, you must set up the client application using the following procedure:

1. Install and configure TCP/IP.
2. Load the Tektronix Bluetooth Protocol Analyzer application software CD.
3. Double-click on **API client SW\Disk1\Setup.exe**.
4. Download and install the following from the Microsoft Web site. Restart the computer after each of these installations.
   - Distributed COM (DCOM) for Windows 98 (DCOM 98, version 1.1)
   - dcomcnfg (DCOM configuration utility)

***NOTE**. The dcomcnfg utility runs only if user-level access is enabled. See step 5.*

---

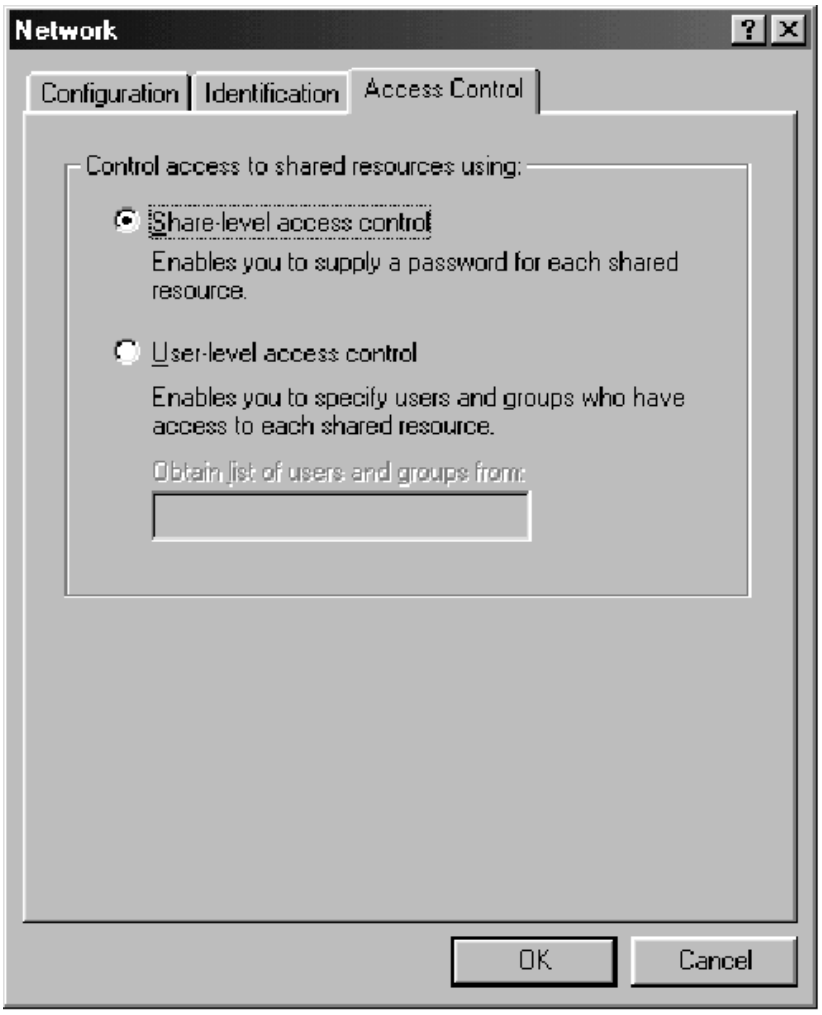
5. Select the appropriate access type, share-level access (page 2–37) or user-level access (page 2–41). You must set up the client application to match the access level you chose for the Protocol Analyzer Server.

***NOTE**. You can switch between user-level access and share-level access later by repeating the procedure from Step 5 onwards.*

Share-Level Access for Windows 98

To set up a client application to access a Protocol Analyzer server that is setup for share-level access, do the following procedure:

1. Click **Start> Settings> Control Panel** to open the Control Panel window.
2. Double-click **Network** to open the Network window as shown in the following figure.



3. Click the **Access Control** tab.
4. Select the **Share-level access control** check box.
5. Click **OK**.

***NOTE**. If prompted, insert the Windows 98 disk or provide a file path to the stored Windows 98 files.*

---

6. The **System Settings Change** dialog box appears. Click **Yes**.
7. Restart the client machine.

8. Go to the directory **<install directory>\System\API and run Share level Access Client.reg.** A dialog box appears, indicating successful registration. Click **OK**.
9. Restart the client machine.
10. Click **Start > Run**.
11. Either locate the regedit file using Browse or type regedit.
12. Click the following registry key: **HKEY_CLASSES_ROOT\AppID\{141DF06A-04FA-11d6-B2DC-00062912F3D2}**
13. Click **Edit > New > StringValue**, add a named value - RemoteServerName.
14. Click the new value - RemoteServerName and select **Edit > Modify**. The Edit String dialog box appears.
15. Enter the computer name of the Protocol Analyzer Server. This is the name used to identify the Protocol Analyzer Server on the network. Click **OK**.
16. To verify that the setup is complete:
    - On the client machine, run **testclient.exe** from the directory <install directory>\Samples\API Samples\VC++\test client\. The Tektronix Test API client application window appears.

    - Click the connect button to check if the client application can connect to the Protocol Analyzer Server.

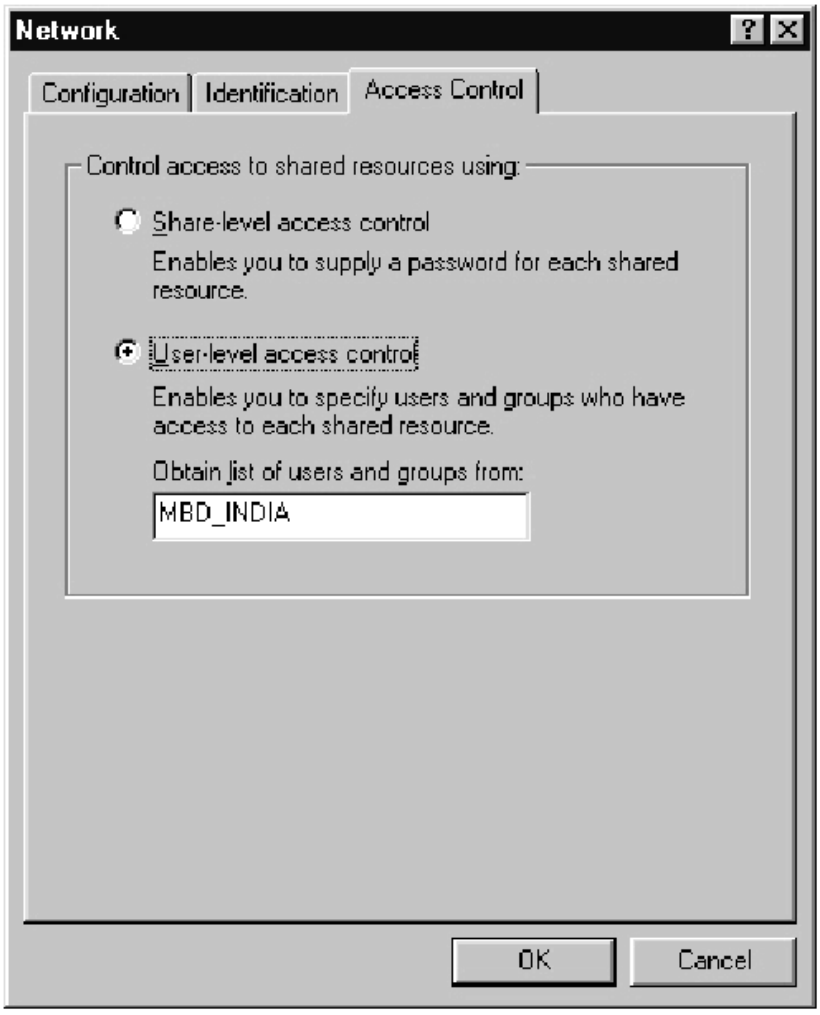
This completes the setup of the Protocol Analyzer Server and the client application for running a client application across the network.

***NOTE**. You can switch between user-level access and share-level access later by repeating the procedure from step 2 of* Setting up the Client Machine on Windows 98 *on page 2-37.*

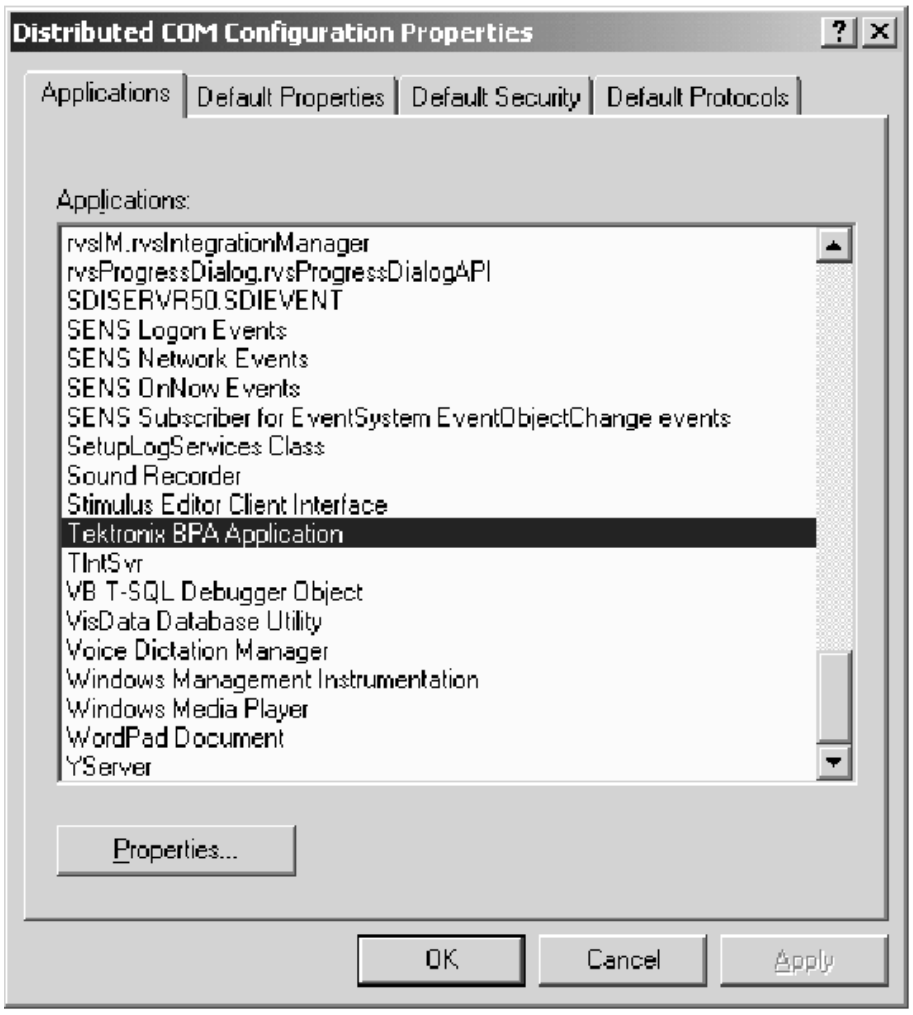
User-Level Access for Windows 98

To set up a client application to access a Protocol Analyzer server that is setup for user-level access, do the following procedure:

1. Click **Start > Settings > Control Panel** to open the Control Panel Window.
2. Double-click **Network** to display the Network window.
3. Click the **Access Control** tab.
4. Select the User-level access control check box.



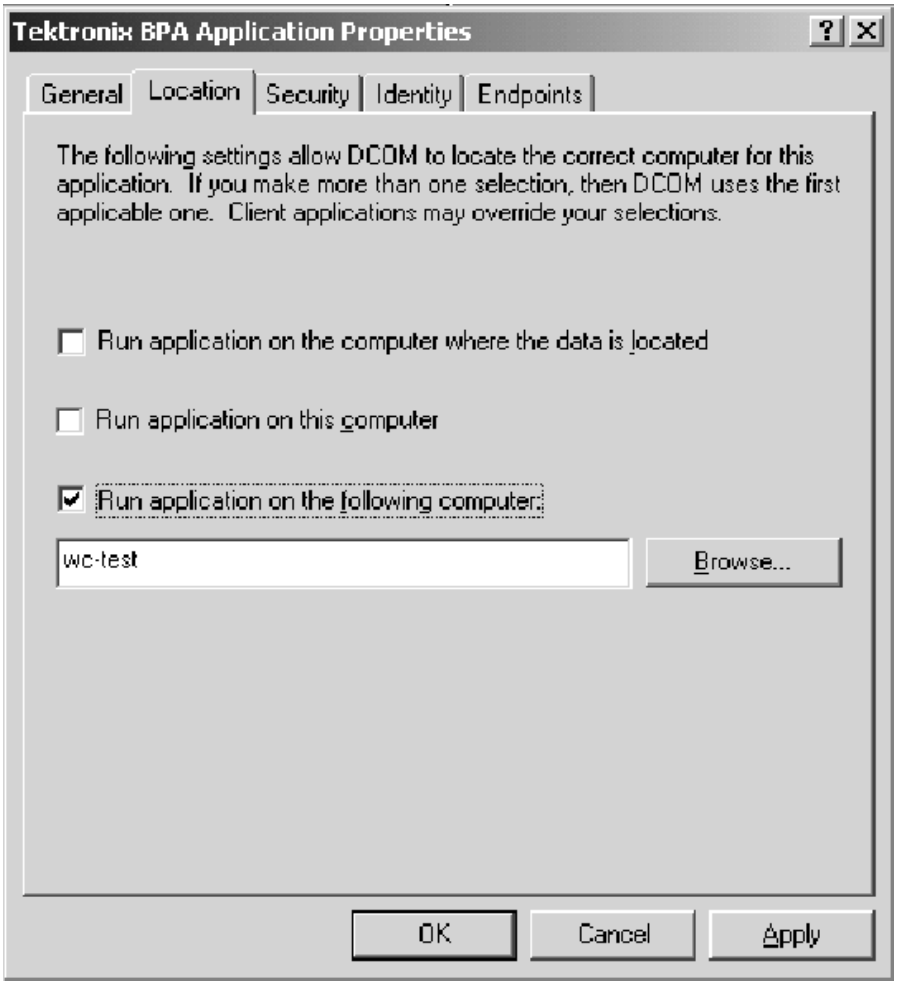
5. In the Obtain list of users and groups from field, type the name of the domain to validate user access.
6. Click **OK**.
7. Restart the client machine.
8. Go to the directory <install directory>\System\API and run User level Access Client.reg.
9. Restart the client machine.
10. Click **Start > Run.**
11. Either locate the dcomcnfg file using Browse or type dcomcnfg.
12. Click **OK**. The Distributed COM Configuration Properties window appears as shown in the following figure.



13. Double-click **Tektronix BPA Application** in the Applications tab.

14. Click the **Location** tab.

- Clear the Run application on this computer check box.
- Select the Run application on the following computer check box and type the name of the Protocol Analyzer Server in the adjacent field.



15. To verify that the setup is complete:

- On the client machine, run testclient.exe from the directory <install directory>\Samples\API Samples\VC++\test client\. The Tektronix Test API client application window appears.

- Click the connect button to check if the client application can connect to the Protocol Analyzer Server. (The first time you connect, it may take a few minutes.)

This completes the setup of the Protocol Analyzer Server and the client machine for running a client application across the network.

NOTE. *You can switch between user-level and share-level access later by repeating the procedure from step 2 of* Setting up the Client Machine on Windows 98 *on page 2-37.*

Setting up the Client Machine on Windows 95

After you set up the Protocol Analyzer Server, you must set up the client application using the following procedure:

1. Install and configure TCP/IP.
2. Load the Tektronix Bluetooth Protocol Analyzer software CD.
3. Double-click API client SW\Disk1\Setup.exe.
4. Download and install the following from the Microsoft Web site. Restart the computer after each of these installations.
   - Distributed COM (DCOM) for Windows 95 (DCOM 95, version 1.1)
   - dcomcnfg (DCOM configuration utility)

***NOTE**. The dcomcnfg utility runs only if user-level access is enabled. See step 5.*

---

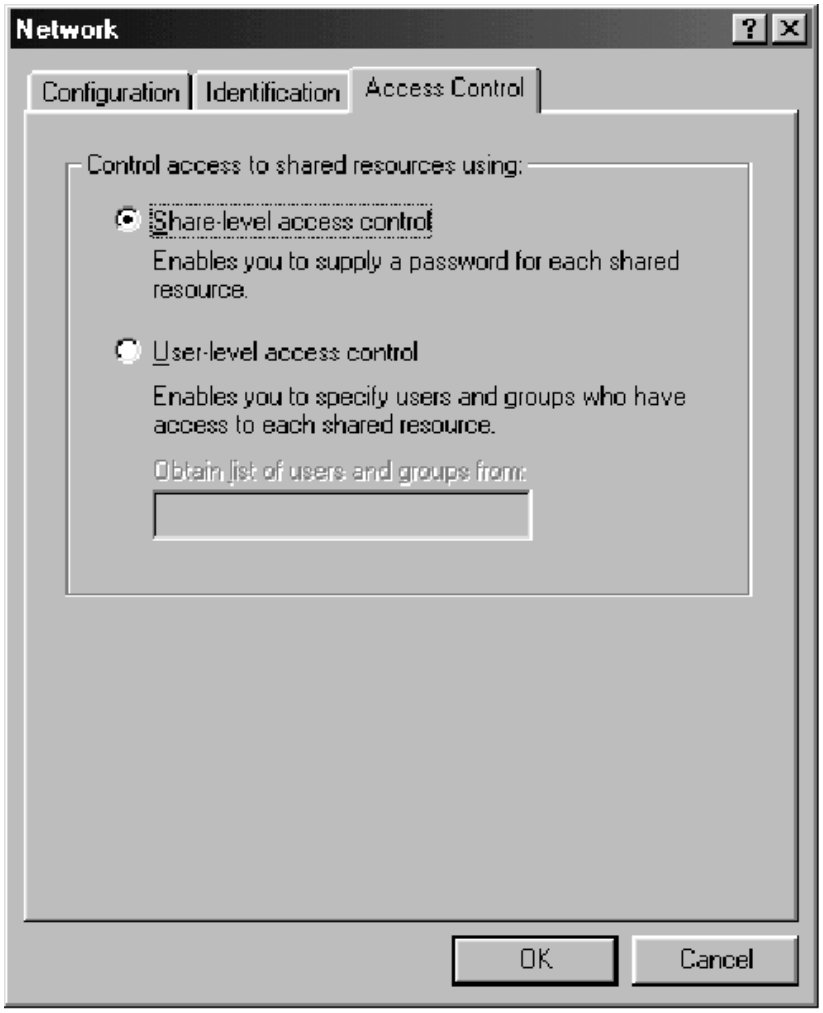
5. Select the appropriate access type, share-level access (page 2-47) or user-level access (page 2-49). You must set up the client application to match the access level you chose for the Protocol Analyzer Server.

***NOTE**. You can switch between user-level access and share-level access later by repeating the procedure from Step 2 onwards.*

Share-Level Access for Windows 95

To set up a client application to access a Protocol Analyzer server that is setup for share-level access, do the following procedure:

1. Click **Start > Settings > Control Panel** to open the Control Panel window.
2. Double-click **Network** to open the Network window.
3. Click the **Access Control** tab.



***NOTE**. If prompted, insert the Windows 95 disk or provide a file path to the stored Windows 95 files.*

4. Select the Share-level access control check box.
5. Click **OK.**
6. The System Settings Change dialog box appears. Click **Yes**.
7. Restart the client machine.
8. Go to the directory <install directory>\System\API and run **Share level Access Client.reg.** A dialog box appears, indicating successful registration. Click **OK**.
9. Restart the client machine.
10. Click **Start > Run**.
11. Either locate the regedit file using Browse or type regedit.
12. Click the following registry key: HKEY_CLASSES_ROOT\AppID\{141DF06A-04FA-11d6-B2DC-00062912F3D2}
13. Click **Edit > New > StringValue**, add a named value - RemoteServerName.
14. Click the new value - RemoteServerName and select **Edit > Modify**. The Edit String dialog box appears.
15. Enter the computer name of the Protocol Analyzer Server. This is the name used to identify the Protocol Analyzer Server on the network. Click **OK**.

To verify that the setup is complete:

- On the client machine, run **testclient.exe** from the directory <install directory>\Samples\API Samples\VC++\test client\. The Tektronix Test API client application window appears.

- Click the connect button to check if the client application can connect to the Protocol Analyzer Server.

This completes the setup of the Protocol Analyzer Server and the client application for running your client application across the network.

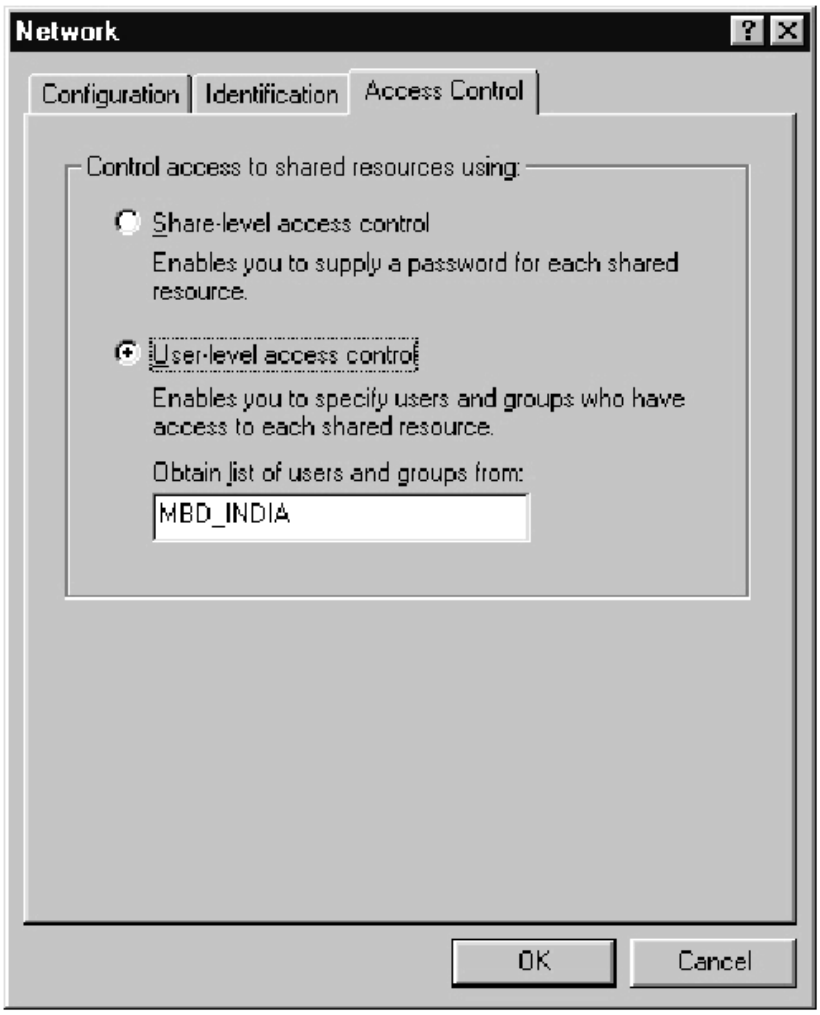
***NOTE**. You can switch between user-level access and share-level access later by uninstalling the Tektronix API client application and DCOM 95 using the Windows Control Panel and repeating the procedure from step 2 of* Setting up the Client Machine on Windows 95 *on page 2-46.*

User-Level Access for Windows 95

To set up a client application to access a Protocol Analyzer server that is set up for user-level access, do the following procedure:

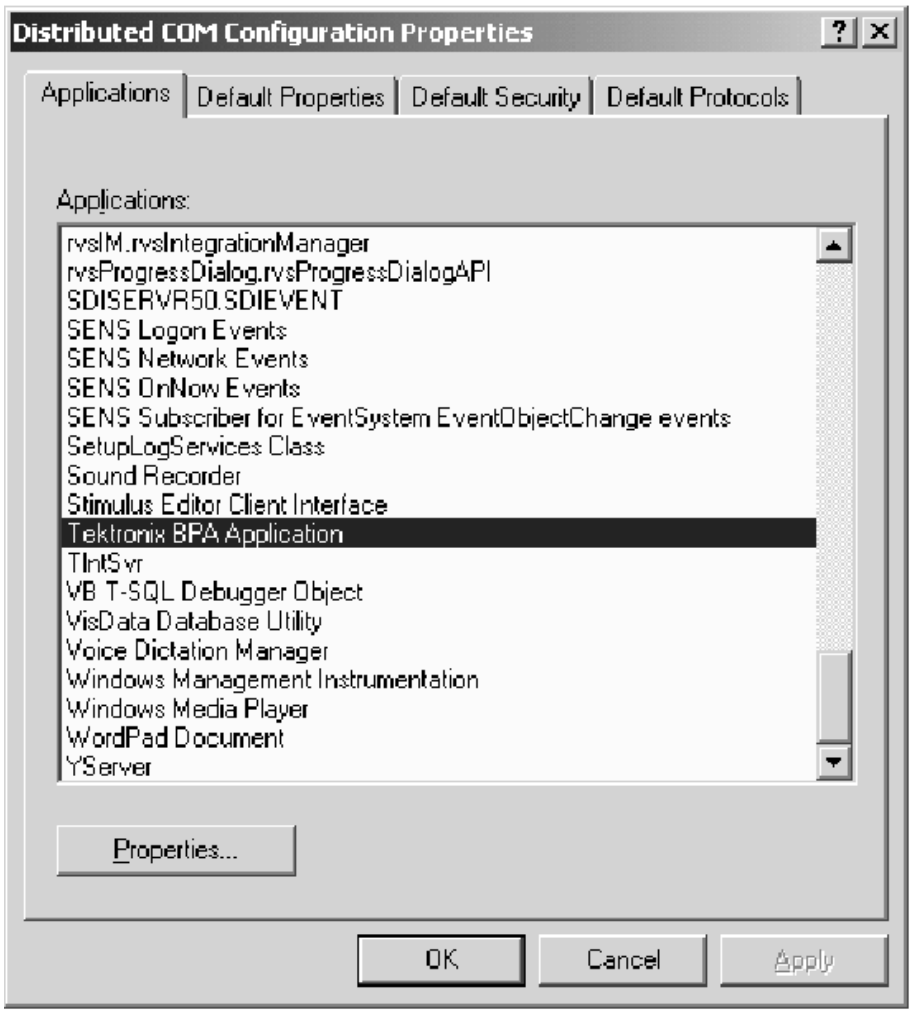
1. Click **Start > Settings > Control Panel** to open the Control Panel window.
2. Double-click **Network** to open the Network window.

3. Click the **Access Control** tab.



4. Select User-level access control check box.
5. In the Obtain list of users and groups from field, type the name of the domain to validate user access.
6. Click **OK**.
7. Restart the client machine.
8. Go to the directory <install directory>\System\API and run User level Access Client.reg.

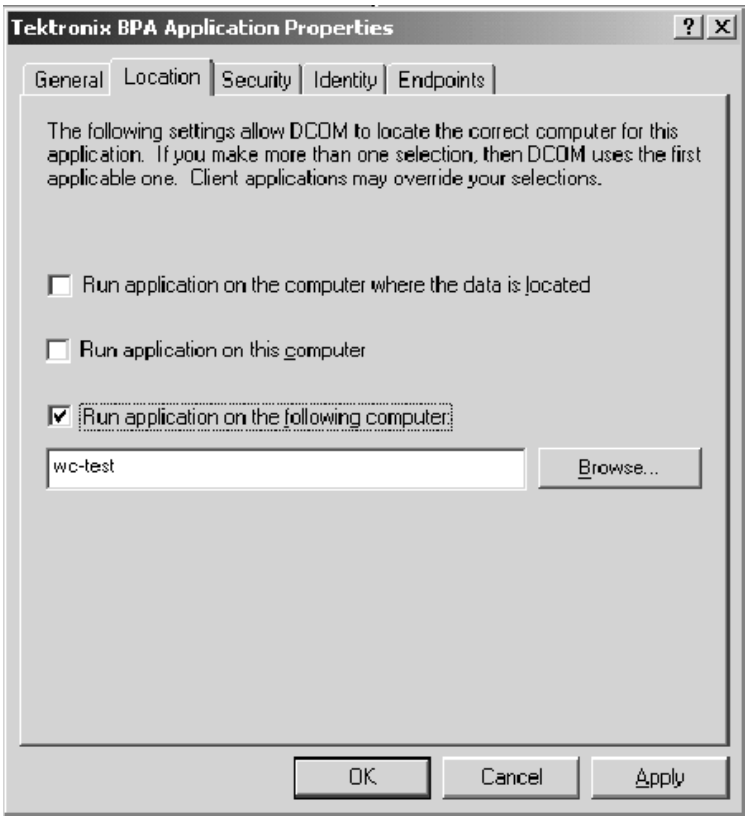
9. Restart the client machine.
10. Click **Start > Run**.
11. Either locate the dcomcnfg file using Browse or type dcomcnfg.
12. Click **OK**. The Distributed COM Configuration Properties window appears as shown in the following figure.



13. Double-click **Tektronix BPA Application** in the Applications tab.

14. Click the **Location** tab.

- Clear the Run application on this computer check box.
- Select the Run application on the following computer check box and type the name of the Protocol Analyzer Server in the adjacent field.



15. To verify that the setup is complete:

- Run <install directory>\Samples\API Samples\VC++\test client\testclient.exe on the client.

- Click the connect button to see if the client application can connect to the Protocol Analyzer Server. (The first time you connect, it may take a few minutes.)

This completes the set up of the Protocol Analyzer Server and the client machine for running a client application across the network.

NOTE. *You can switch between user-level access and share-level access later by repeating the procedure from step 2 of* Setting up the Client Machine on Windows 95 *on page 2-46*.

Setting up the Client application on Other Platforms

If the client application requires the type library, you can generate it on your platform using BPA100.idl in <install directory>\System\API\src.

Follow these steps:

1. Ensure that DCOM is working on your platform.
2. Merge <install directory>\System\API\Client.reg into your registry.
3. Merge <install directory>\System\API\Share-level Access Client.reg or User-level Access Client.reg into your registry, depending on the type of access control you chose for the Protocol Analyzer Server.
4. Add a string value named RemoteServerName to the key HKEY_CLASSES_ROOT\AppID\{141DF06A-04FA-11d6-B2DC-00062912F3D2}
5. Enter the computer name of the Protocol Analyzer Server as this string value. This is the name used to identify the Protocol Analyzer Server on the network.

This completes the set up of the protocol analyzer for operating with a remote client application using a platform other than Windows.

Connecting to the Protocol Analyzer Server

Client applications connect to the Protocol Analyzer Server by creating a BPAApplication Object. For example, in Microsoft Visual Basic:

Example

```
'Establish Connection to BPA
Dim App As Object
Set App = CreateObject("BPA100.BPAApplication")
```

Once the Application Object is created, the client application can call methods on it to get references to BPASystem and BPAAnalyzer Objects.

Disconnecting from the Protocol Analyzer Server

A client application connected to the Protocol Analyzer Server can disconnect by removing the reference to the BPAApplication Object. For example, in Microsoft Visual Basic:

Example

```
'Disconnect from BPA.
Set App = Nothing
```

References to any BPASystem or BPAModule Objects that were obtained must also be removed.

Programming Examples

Programming Examples

The Protocol Analyzer Server exports dual interfaces with both dispatch and vtable interface characteristics.The dispatch dual interface uses run time (dynamic) binding to resolve method calls. This type of interface is useful in interpretive and scripting environments, where you cannot use header files and type libraries for static binding.The vtable dual interface uses static binding to resolve method calls using header files and type libraries.

For more information, see these examples on the pages to follow:

- *Microsoft Visual Basic Client Using Dispatch Interfaces*
- *Microsoft Visual Basic Client Using VTABLE Interfaces*

***NOTE**. You can use the dispatch dual interface for additional code examples on specific methods.*

Microsoft Visual Basic Client Using Dispatch Interfaces

```
Dim Application As Object
Dim System As Object

Private Sub Form_Load()

   'Create the COM interfaces required
   Set Application = CreateObject("BPA100.BPAApplication")
   Set System= Application.GetSystem

End Sub

Private Sub StartLogButton_Click()
      'Start Acquisition
      System.Start ("C:\API Log File.tbpa")
End Sub

Private Sub StopLogButton_Click()
      'Stop Acquisition
      System.Stop
End Sub
```

Microsoft Visual Basic Client Using VTABLE Interfaces

```
Dim Application As BPAApplication
Dim System As BPASystem


Private Sub Form_Load()

   'Create the COM interfaces required
   Set Application = New BPA100.BPAApplication
   Set System = Application.GetSystem

End Sub

Private Sub StartLogButton_Click()
      'Start Acquisition
      System.Start ("C:\API Log File.tbpa")
End Sub

Private Sub StopLogButton_Click()
      'Stop Acquisition
      System.Stop
End Sub
```

Status and Events

Status and Events

All methods in all interfaces of API return an HRESULT (or SCODE). Refer to BPAErrors.h for possible error codes.

Additional error information is communicated as follows:

- Objects that use the dispatch portion of the dual interface can use the exception information argument of the Invoke method.
- Objects that use the Vtable portion of the dual interface can use error objects. When an HRESULT indicates an error, the client can call the standard function GetErrorInfo() to get more detailed information about the error.
- When a method returns an error, output arguments are undefined and should not be used.

Refer to the sample programs for examples on handling errors.

Error Handling

The API returns HRESULT (or SCODE) for all interface methods. Refer to the file “bpaerrors.h” in <install directory>\System\API\Src\bpaerrors.h> for possible error codes.

Additional error information is communicated as follows:

- Objects that use the dispatch dual interface can use the exception information argument of the IDispatch::Invoke method.
- Objects that use the vtable dual interface can use error objects. When an HRESULT indicates an error, the client can call the standard function GetErrorInfo() for detailed information about the error.
- When a method returns an error, output arguments are undefined and should not be used.

Server Message Boxes

In the Protocol Analyzer graphical user interface, there are instances where you are asked to confirm a particular operation. For example, once the acquisition is stopped in log file mode, you are asked whether to open the current log file.

Since it is not possible to ask questions through the API, the application always proceeds with the original operation as though the questions were never asked. In the previous example, the load operation proceeds without asking any confirmation.

Modal message boxes that are normally displayed in the Protocol Analyzer user interface will not be displayed when a client is connected to the server.

Syntax and Commands

Command Groups

This section lists the commands organized by functional group. The Command Descriptions section, starting on page 5–7, lists all commands alphabetically.

The BPA100 Series application programming interface conforms to Tektronix standard codes and formats excepts where noted.

BPAAplication Commands

BPAApplication commands connect to the application and obtain a reference to additional objects. The BPAApplication object exports a single interface called IBPAApplication.

Table 5-1: BPAAplication commands

Header	Description
ShowWindow	Shows/hides the BPA100 Series Server's main window
GetVersion	Retrieves the current version of the specified subsystem
GetBDAddress	Retrieves the address of the attached BPA100 Series device
GetAnalyzer	Returns the interface pointer for the BPAAnalyzer object
GetSystem	Returns the interface pointer for the BPASystem object
GetHCISimple	Returns the interface pointer for the BPAHCISimple object

BPASystem Commands

BPASystem commands control most of the functionality in the BPA100 Bluetooth Protocol Analyzer GUI. The BPASystem object exports a single interface called the IBPASystem.

Table 5-2: BPASystem commands

Header	Description
Start	Sets the analyzer to start mode
Stop	Sets the analyzer to stop mode
Pause	Pauses logging of the acquired data
Resume	Resumes logging of packet data from the current Protocol Analyzer session
GetDeviceStatus	Retrieves the sync status of the device
GetSessionInfo	Gets the session information for the current log session
SetHoppingMode	Sets the BPA100 Series hopping mode
GetHoppingMode	Gets the current settings of the hopping mode
SetCorrelation	Sets correlation settings for the acquisition
GetCorrelation	Gets correlation value from the acquisition setup
SetResync	Sets resync settings for the acquisition
GetResync	Gets resync values from the acquisition setup
SetDataWhitening	Sets the data whitening flag for acquisitions
GetDataWhitening	Gets the data whitening information from the acquisitions settings
SetAcquisitionMode	Configures the logging mode of the BPA100 Series

Table 5-2: BPASystem commands (Cont.)

Header	Description
GetAcquisitionMode	Gets the current settings of the logging mode in the acquisition settings
SetAcquistionDefault	Sets the acquisition settings to factory default values
SetLowLevelTrigger	Sets the low level trigger values from a file
GetLowLevelTrigger	Saves the current low level trigger values to the specified file
ActivateLowLevelTrigger	Activates or deactivates the current low-level trigger setup
SetHighLevelTrigger	Sets the high-level trigger values from a file
GetHighLevelTrigger	Saves the current high-level trigger values onto the specified file
ActivateHighLevelTrigger	Activates or deactivates the current high-level trigger setup
SetErrorPacketGeneration	Sets the error packet values from a file
GetErrorPacketGeneration	Saves the current error packet generated values to the specified file
ActivateErrorPacketGeneration	Activates or deactivates the current error packet generation setup
SetDataFilter	Sets baseband data filter
GetDataFilter	Retrieves the current baseband data filter settings
ActivateDataFilter	Activates or deactivates the current baseband data filter setup
SetDecryptionSettings	Sets the decryption settings to specified values
GetDecryptionSettings	Gets the current decryption settings
ActivateDecryption	Activates or deactivates decryption

Table 5-2: BPASystem commands (Cont.)

Header	Description
SetDecryptionDefault	Sets the decryption settings to factory default values
DoDeviceDiscovery	Searches for neighbouring Bluetooth devices
GetDeviceDiscoveryStatus	Returns the device discovery operation status
GetDevicesList	Returns the list of devices found during device discovery

BPASystemEvents Commands

BPASystemEvents commands capture events fired by the BPA application.

Table 5-3: BPASystemEvents commands

Header	Description
OnStateChange	This event is invoked whenever the BPA100 changes state
OnTriggetIn	This event is invoked whenever the Trigger In port is asserted
OnTriggerOut	This event is invoked whenever the Trigger Out port is asserted

BPAAnalyzer Commands

BPAAnalyzer commands analyze a log file captured by the BPA100 Series.

Table 5-4: BPAAnalyzer commands

Header	Description
Open	Opens the log file
Close	Closes the log file
GetPacketCount	Retrieves the number of packets in the specified protocol layer
GetPacket	Retrieves the specified packet for a given protocol layer
GetPacketInfo	Retrieves the decoded information of a specified packet depending on the specified information type
GetPrevPacketNumner	Searches for the packet previous to the specified packet and type in the Baseband layer and returns the index of the matching packet
GetNextPacketNumner	Searches for the packet next to the specified packet and type in the Baseband layer and returns the index of the matching packet
Export	Exports the specified protocol layer's packets of the current log file to a CSV or TXT or WAV file given beginning and ending packet number
GetAcquisitionReport	Gets the acquisition report that was created for the log file
SetL2CAPConnectionProperties	Assigns the L2CAP connection property type for a given packet
GetL2CAPConnectionProperties	Retrieves the current L2CAP connection property assignment for a given pac

Table 5-4: BPAAnalyzer commands (Cont.)

Header	Description
SetRFCOMMServerChannel	Assigns the RFCOMM server channel assignments for a specified packet.
GetRFCOMMServerChannel	Retrieves the RFCOMM server channel assignments for a specified packet

BPAHCISimple Commands

BPAHCISimple commands send or receive HCI commands.

Table 5-5: BPAHCISimple commands

Header	Description
Send	Sends an HCI message
Get	Gets the last received HCI event message

BPAHCISimpleEvents Command

BPAHCISimpleEvents command captures events fired by the BPA100 Series hardware.

Table 5-6: BPAHCISimpleEvents commands

Header	Description
HCIEvent	Captures events whenever the BPA100 receives an HCI event from the hardware

Command Descriptions

IBPAApplication::ShowWindow

This command shows/hides the Protocol Analyzer Server's main window.

Syntax

HRESULT ShowWindow([in] long Show)

Arguments

Show – This flag takes one of the following values:

Table 5-7: IBPAApplication::ShowWindow Show values

Value	Description
BPA_HIDE_WINDOW (0)	Hide the server window.
BPA_SHOW_WINDOW (1)	Show the server window.

Returns

Table 5-8: IBPAApplication::ShowWindow Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_SHOW	Invalid Show value.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Set App = CreateObject("BPA100.BPAApplication ")
```

```
'Hide the window.
App.ShowWindow 0
```

Remarks

The application window is shown by default when a client connects to the server.

IBPAApplication::GetVersion

This command retrieves the current version of the specified subsystem.

Syntax

HRESULT GetVersion([in] long Subsystem, [out, retval] BSTR* Version)

Arguments

Subsystem - Subsystem software version requested. This takes one of the following values:

Table 5-9: IBPAApplication::GetVersion Sybsystem values

Value	Description
BPA_SOFTWARE	Software
BPA_FIRMWARE	Firmware

Version - software version of requested subsystem.

Returns

Table 5-10: IBPAApplication::GetVersion Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_SUBSYSTEM	Invalid Subsystem value.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim SWVersion As String

Set App = CreateObject(”BPA100.BPAApplication”)
’Get software version.
SWVersion = App.GetVersion(BPA_SOFTWARE)
```

Remarks

The Protocol Analyzer Server will allocate the space for the returned string. The client is responsible for freeing it when it is no longer in use.

IBPAApplication::GetBDAddress

This command retrieves the address of the attached Protocol Analyzer device.

Syntax

HRESULT GetBDAddress([out, retval] BSTR* Address)

Arguments

Address - Bluetooth address of attached Protocol Analyzer device. Address is colon separated.
Example: 00:50:CD:00:92:B9

Returns

Table 5-11: IBPAApplication::GetBDAddress Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim BDAddress As String

Set App = CreateObject("BPA100.BPAApplication")
'Get software version.
BDAddress = App.GetBDAddress
```

Remarks

The Protocol Analyzer Server allocates the space for the returned string. The client is responsible for freeing it when it is no longer in use.

IBPAApplication::GetAnalyzer

This command returns the interface pointer for the BPAAnalyzer object.

Syntax

HRESULT GetAnalyzer([out, retval] IDispatch** ppAnalyzer)

Arguments

ppAnalyzer – The interface pointer for the BPAAnalyzer object.

Returns

Table 5-12: IBPAApplication::GetAnalyzer Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Sys As Object

Set App = CreateObject(”BPA100.BPAApplication”)
’Get Analyzer.
Set Sys = App.GetAnalyzer
```

IBPAApplication::GetSystem

This command returns the interface pointer for the BPASystem object.

Syntax

HRESULT GetSystem([out, retval] IDispatch** ppSystem)

Arguments

ppSystem - The interface pointer for the BPASystem object.

Returns

Table 5-13: IBPAApplication::GetSystem Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Sys As Object

Set App = CreateObject(”BPA100.BPAApplication”)
’Get system.
Set Sys = App.GetSystem
```

IBPAApplication::GetHCISimple

This command returns the interface pointer for the BPAHCISimple object.

Syntax

HRESULT GetHCISimple([out, retval] IDispatch** ppHCISimple)

Arguments

ppHCISimple – The interface pointer for the BPAHCISimple object.

Returns

Table 5-14: IBPAApplication::GetHCISimple Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Hci As Object

Set App = CreateObject(”BPA100.BPAApplication”)
’Get HCISimple.
Set Hci = App.GetHCISimple
```

IBPASystem::Start

This command sets the Analyzer into Start mode. If the Logging Mode in the Acquisition setup has been set to Piconet, the analyzer now begins logging any future data. If the Logging Mode has been set for Independent mode. Then the Protocol Analyzer tries to synchronize to another piconet.

Syntax

HRESULT Start ([in] BSTR Filename)

Arguments

Filename – Filename on to which the data to be logged. If NULL value is passed, use Freerun mode.

Returns

Table 5-15: IBPASystem::Start Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_FILE_NAME	Invalid file name.
BPA_E_FILE_OPEN_ERROR	Unable to open specified file. Cannot log data.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Start acquisition
System.Start ”C:\LogFile.tbpa”
```

Remarks

This method starts the acquisition operation. After calling this method, the method IBPASystem::GetDeviceStatus() can be used to find out the current run status of the system.

IBPASystem::Stop

This command takes the Protocol Analyzer out of Start mode and into Stop mode. If the unit is synchronized to another piconet, the Protocol Analyzer is set to Idle state and closes the currently open logging file.

Syntax

HRESULT Stop ()

Arguments

None.

Returns

Table 5-16: IBPASystem::Stop Returns

Value	Description
S_OK	The operation was successful.
BPA_E_NOT RUNNING	The Protocol Analyzer Server is not running.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem
’Start acquisition
System.Start ”C:\LogFile.tbpa”
...
’Stop acquisition
System.Stop
```

Remarks

This command stops the acquisition operation but does not wait for it to complete before returning. After calling this command, the command IBPASystem::GetDeviceStatus() can be used to find out the current run status of the system.

IBPASystem::Pause

This command pauses the data logging of the acquired data.

Syntax

HRESULT Pause ()

Arguments

None

Returns

Table 5-17: IBPASystem::Pause Returns

Value	Description
S_OK	The operation was successful.
BPA_E_NOT RUNNING	The Protocol Analyzer Server is not running.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem
’Start acquisition
System.Start ”C:\LogFile.tbpa”
...
...
’Pause acquisition
System.Pause
```

IBPASystem::Resume

This command resumes the logging of packet data from the current Protocol Analyzer session. Either Independent or Piconet mode.

Syntax

HRESULT Resume()

Arguments

None

Returns

Table 5-18: IBPASystem::Resume Returns

Value	Description
S_OK	The operation was successful.
BPA_E_NOT_PAUSED	The Protocol Analyzer Server is not paused.
BPA_E_NOT RUNNING	The Protocol Analyzer Server is not running.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem
’Start acquisition
System.Start ”C:\LogFile.tbpa”
...
...
’Pause acquisition
System.Pause
...
...
’Resume acquisition
System.Resume
’Stop acquisition
System.Stop
```

IBPASystem::GetDeviceStatus

This command retrieves the current software version of the specified subsystem.

Syntax

HRESULT GetDeviceStatus([out, retval] long* State)

Arguments

State – Current state of the Protocol Analyzer. This takes one of the following values:

Table 5-19: IBPASystem::GetDeviceStatus State Values

Value	Description
BPA_NO_SYNC	Protocol Analyzer is in idle state, not synchronized.
BPA_SYNC	Protocol Analyzer synchronized with another Bluetooth device.

Returns

Table 5-20: IBPASystem::GetDeviceStatus Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim Status As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem
’Start acquisition
System.Start ”C:\LogFile.tbpa”
...
...
’Get Protocol Analyzer Server status
Status = System.GetDeviceStatus
’Stop acquisition
System.Stop
```

IBPASystem::GetSessionInfo

This command gets the session information for the current log session.

Syntax

HRESULT GetSessionInfo([in] long SessionInfoType, [out, retval] BSTR* SessionInfo)

Arguments

SessionInfoType – This takes one of the following values:

Table 5-21: IBPASystem::GetSessionInfo SessionInfoType Values

Value	Description
BPA_SESSION_START_TIME	Session start time.
BPA_SESSION_END_TIME	Session end time.
BPA_SESSION_BYTES_ACQUIRED	Number of bytes acquired.
BPA_SESSION_BYTES_LOGGED	Number of bytes logged.
BPA_SESSION_PACKETS_ACQUIRED	Number of packets acquired.
BPA_SESSION_PACKETS_LOGGED	Number of packets logged.

SessionInfo – The session information.

Returns

Table 5-22: IBPASystem::GetSessionInfo Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_SESSION_INFO_TYPE	Invalid type of session information.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim PacketsCount As String

Set App = CreateObject( ”BPA100.BPAApplication” )

’Get System
Set System = App.GetSystem

’Set Data Filter Options
PacketsCount = System. GetSessionInfo(BPA_SESSION_PACK-
ETS_LOGGED)
```

IBPASystem::SetHoppingMode

This command sets the hopping mode of the Protocol Analyzer.

Syntax

HRESULT SetHoppingMode ([in] long Pattern, [in] long Frequency)

Arguments

Pattern - Hopping pattern. This takes one of the following values:

Table 5-23: IBPASystem::SetHoppingMode Pattern Values

Value	Description
BPA_NORMAL_HOPPING	Hopping pattern for Europe, USA, France, Spain and Japan.
BPA_SINGLE_FREQUENCY	Rx/Tx on single frequency.

If Pattern is BPA_NORMAL_HOPPING, then Frequency is one of the following values:

Table 5-24: IBPASystem::SetHoppingMode Pattern Frequency

Value	Description
BPA_HOPPING_EUROPE-USA	Hopping pattern for Europe and USA.
BPA_HOPPING_FRANCE	Hopping pattern for France.
BPA_HOPPING_SPAIN	Hopping pattern for Spain.
BPA_HOPPING_JAPAN	Hopping pattern for Japan.

If Pattern is BPA_SINGLE_FREQUENCY, then frequency value range from 2402 - 2480 MHz.

Returns

Table 5-25: IBPASystem::SetHoppingMode Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_PATTERN	Invalid pattern value.
BPA_E_INVALID_FREQUENCY	Invalid frequency.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Set the Hopping Pattern to Spain
System.SetHoppingMode BPA_NORMAL_HOPPING, BPA_HOP-
PING_SPAIN
```

Remarks

The new hopping pattern does not take effect until the next acquisition.

IBPASystem::GetHoppingMode

This command gets the current settings of the hopping mode.

Syntax

HRESULT GetHoppingMode ([out] long *Pattern, [out] long *Frequency)

Arguments

Pattern - Hopping pattern. This takes one of the following values:

Table 5-26: IBPASystem::GetHoppingMode Pattern Values

Value	Description
BPA_NORMAL_HOPPING	Hopping pattern for Europe, USA, France, Spain and Japan.
BPA_SINGLE_FREQUENCY	Rx/Tx on single frequency.

If Pattern is BPA_NORMAL_HOPPING, then Frequency is one of the following values:

Table 5-27: IBPASystem::GetHoppingMode Pattern Frequency

Value	Description
BPA_HOPPING_EUROPE-USA	Hopping pattern for Europe and USA.
BPA_HOPPING_FRANCE	Hopping pattern for France.
BPA_HOPPING_SPAIN	Hopping pattern for Spain.
BPA_HOPPING_JAPAN	Hopping pattern for Japan.

If Pattern is BPA_SINGLE_FREQUENCY, then frequency values range from 2402 - 2480 MHz.

Returns

Table 5-28: IBPASystem::GetHoppingMode Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim Pattern As Long
Dim Frequency As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Get Hopping Mode
System. GetHoppingMode Pattern, Frequency
```

IBPASystem::SetCorrelation

This command sets the correlation setting for the acquisition.

Syntax

HRESULT SetCorrelation ([in] long Correlation)

Arguments

Correlation - Correlation value. This value can range from 40 - 64.

Returns

Table 5-29: IBPASystem::SetCorrelation Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_DATA	Invalid correlation value.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Set Correlation value
System.SetCorrelation 54
```

Remarks

The new correlation value does not take effect until the next acquisition.

IBPASystem::GetCorrelation

This command gets correlation value from the acquisition setup.

Syntax

HRESULT GetCorrelation ([out, retval] long *Correlation)

Arguments

Correlation – Correlation value

Returns

Table 5-30: IBPASystem::GetCorrelation Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim CorValue As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Get Correlation Value
CorValue = System. GetCorrelation
```

IBPASystem::SetResync

This command sets the Resync setting for the acquisition.

Syntax

HRESULT SetResync ([in] long Resync)

Arguments

Resync - Drift value for resynchronizing. This value ranges from 0 - 500ppm.

Returns

Table 5-31: IBPASystem::SetResync Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_DATA	Invalid resync value.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Set Resync value
System.SetResync 40
```

IBPASystem::GetResync

This command gets resync values from the acquisition setup.

Syntax

HRESULT GetResync ([out] long *Resync)

Arguments

Resync - Resync value.

Returns

Table 5-32: IBPASystem::GetResync Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim Resync As Long

Set App = CreateObject(”BPA100.BPAApplication”)
’Get System
Set System = App.GetSystem

’Get Resync Value
Resync = System. GetResync
```

IBPASystem::SetDataWhitening

This command sets the data whitening flag for acquisitions.

Syntax

HRESULT SetDataWhitening ([in] long Whitening)

Arguments

Whitening - This takes one of the following values:

Table 5-33: IBPASystem::SetDataWhitening Whitening Values

Value	Description
BPA_WHITENING_ON	Data Whitening On.
BPA_WHITENING_OFF	Data Whitening Off.

Returns

Table 5-34: IBPASystem::SetDataWhitening Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_DATA	Invalid data whitening value.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Set Data Whitening
System.SetDataWhitening BPA_WHITENING_ON
```

Remarks

The new resync does not take effect until the next acquisition.

IBPASystem::GetDataWhitening

This command gets the data whitening information from the acquisition settings.

Syntax

HRESULT GetDataWhitening ([out, retval] long *Whitening)

Arguments

Whitening - This takes one of the following values:

Table 5-35: IBPASystem::GetDataWhitening Whitening Values

Value	Description
BPA_WHITENING_ON	Data Whitening On.
BPA_WHITENING_OFF	Data Whitening Off.

Returns

Table 5-36: IBPASystem::GetDataWhitening Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim Whitening As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Get Data Whitening
Whitening = System. GetDataWhitening
```

IBPASystem::SetAcquisitionMode

This command configures the logging mode of the Protocol Analyzer.

Syntax

HRESULT SetAcquisitionMode ([in] long Connect, [in] long SyncOption, [in] long Timeout, [in] BSTR BDAddr)

Arguments

Connect - command of connection. This takes one of the following values:

Table 5-37: IBPASystem::SetAcquisitionMode Connect Values

Value	Description
BPA_CONNECT_PICONET	Piconet Mode.
BPA_CONNECT_INDEPENDENT	Independent Mode.

SyncOption - Synchronization Option. This is valid only in Independent Mode. This takes one of the following values.

Table 5-38: IBPASystem::SetAcquisitionMode SyncOption Values

Value	Description
BPA_IND_MASTER_INQUIRY	Sync to piconet using master inquiry.
BPA_IND_SLAVE_INQUIRY	Sync to piconet using fake connection response.
BPA_IND_FAKE_CONNECTION	Sync to piconet using slave inquiry.

Timeout - Time in seconds. Valid in Independent Mode only.

BDAddr - Bluetooth address of master or slave to whom you wish to synchronize to. Valid in Independent Mode only.

Returns

Table 5-39: IBPASystem::SetAcquisitionMode Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CONNECT_MODE	Invalid mode of connection.
BPA_E_INVALID_SYNC_OPTION	Invalid synchronization option.
BPA_E_INVALID_TIMEOUT	Invalid timeout value.
BPA_E_INVALID_BDADDR	Invalid Bluetooth device address.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Set Acqsuisition log settings
System.SetAcquisitionLogging ( BPA_CONNECT_INDEPEN-
DENT, BPA_IND_MASTER_INQUIRY, 4,0 ”00:50:CD:00:95:3E” )
```

IBPASystem::GetAcquisitionMode

This command gets the current settings of the logging mode in the acquisition settings.

Syntax

HRESULT GetAcquisitionMode ([out] long *Connect, [out] long *SyncOption, [out] long *Timeout, [out] BSTR *BDAddr)

Arguments

Connect - command of connection. This takes one of the following values:

Table 5-40: IBPASystem::GetAcquisitionMode Connect Values

Value	Description
BPA_CONNECT_PICONET	Piconet Mode.
BPA_CONNECT_INDEPENDENT	Independent Mode.

SyncOption - Synchronization Option. This is valid only in Independent Mode. This takes one of the following values.

Table 5-41: IBPASystem::GetAcquisitionMode SyncOption Values

Value	Description
BPA_IND_MASTER_INQUIRY	Sync to piconet using master inquiry.
BPA_IND_SLAVE_INQUIRY	Sync to piconet using fake connection response.
BPA_IND_FAKE_CONNECTION	Sync to piconet using slave inquiry.

Timeout - Time in seconds. Valid in Independent Mode only.

BDAddr - Bluetooth address of master or slave to whom you wish to synchronize to. Valid in Independent Mode only.

Returns

Table 5-42: IBPASystem::GetAcquisitionMode Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim ConnectMode As Long
Dim IndMode As Long
Dim Timeout As Long
Dim BDAddr As String

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Get Acquisition mode settings
System. GetAcquisitionMode ConnectMode, SyncOption, Timeout,
BDAddr
```

IBPASystem::SetAcquisitionDefault

This command sets the acquisition settings to factory default values.

Syntax

HRESULT SetAcquisitionDefault ()

Arguments

None.

Returns

Table 5-43: IBPASystem::SetAcquisitionDefault Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Set Acquisition to factory defaults
System. SetAcquisitionDefault
```

Remarks

The new acquisition settings do not take effect until the next acquisition.

IBPASystem::SetLowLevelTrigger

This command sets the Low Level Trigger values from a file. This command activates the low level trigger by default.

Syntax

HRESULT SetLowLevelTrigger([in] BSTR FileName)

Arguments

Filename - File name from which the low level triggers values are to be loaded.

Returns

Table 5-44: IBPASystem::SetLowLevelTrigger Returns

Value	Description
S_OK	The operation was successful.
BPA_INVALID_FILE_FORMAT	File format has bad syntax.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Load the Low Level Trigger values from a file.
System.LoadLowLevelTrigger ”C:\My Documents\Trigger.llt”
```

Remarks

Use ActivateLowLevelTrigger command to enable or disable the low level trigger.

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPASystem::GetLowLevelTrigger

This command saves the current Low Level Trigger values onto the specified file.

Syntax

HRESULT GetLowLevelTrigger([in] BSTR FileName)

Arguments

Filename – Filename onto which the low level triggers values are to be saved.

Returns

Table 5-45: IBPASystem::GetLowLevelTrigger Returns

Value	Description
S_OK	The operation was successful.
BPA_FILE_SAVE_ERROR	Unable to save specified file.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Save the current Low Level Trigger values
System.GetLowLevelTrigger ”C:\My Documents\Trigger.llt”
```

Remarks

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPASystem::ActivateLowLevelTrigger

This command activates or deactivates the current low level trigger setup.

Syntax

HRESULT ActivateLowLevelTrigger([in] long Enable)

Arguments

Enable – Enable or Disable the current low level triggers. This takes one of the following values:

Table 5-46: IBPASystem::ActivateLowLevelTrigger Enable Values

Value	Description
BPA_ENABLE	Enable Low Level Trigger.
BPA_DISABLE	Disable Low Level Trigger.

Version – Software version of requested subsystem.

Returns

Table 5-47: IBPASystem::ActivateLowLevelTrigger Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Activate Low Level Trigger
System. ActivateLowLevelTrigger BPA_ENABLE
```

IBPASystem::SetHighLevelTrigger

This command sets the High Level Trigger values from a file. This command activates the high level trigger by default.

Syntax

HRESULT SetHighLevelTrigger([in] BSTR FileName)

Arguments

Filename - File name from which the high level triggers values are to be loaded.

Returns

Table 5-48: IBPASystem::SetHighLevelTrigger Returns

Value	Description
S_OK	The operation was successful.
BPA_INVALID_FILE_FORMAT	File format has bad syntax.
BPA_FILE_LOAD_ERROR	Unable to load specified file.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Load the High Level Trigger values from a file.
System.SetHighLevelTrigger ”C:\My Documents\Trigger.hlt”
```

Remarks

Use ActivateHighLevelTrigger command to enable or disable the high level trigger. All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPASystem::GetHighLevelTrigger

This command saves the current High Level Trigger values to the specified file.

Syntax

HRESULT GetHighLevelTrigger([in] BSTR FileName)

Arguments

Filename – Filename onto which the high level triggers values are to be saved.

Returns

Table 5-49: IBPASystem::SetHighLevelTrigger Returns

Value	Description
S_OK	The operation was successful.
BPA_FILE_SAVE_ERROR	Unable to save specified file.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Save the current High Level Trigger values
System.GetHighLevelTrigger ”C:\My Documents\Trigger.hlt”
```

Remarks

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPASystem::ActivateHighLevelTrigger

This command activates or deactivates the current high level trigger setup.

Syntax

HRESULT ActivateHighLevelTrigger([in] long Enable)

Arguments

Enable - Enable or Disable the current high level triggers. This takes one of the following values:

Table 5-50: IBPASystem::ActivateHighLevelTrigger Enable Values

Value	Description
BPA_ENABLE	Enable High Level Trigger.
BPA_DISABLE	Disable High Level Trigger.

Version - Software version of requested subsystem.

Returns

Table 5-51: IBPASystem::ActivateHighLevelTrigger Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem
’Activate High Level Trigger
System. ActivateHighLevelTrigger BPA_ENABLE
```

IBPASystem::SetErrorPacketGeneration

This command sets the error packet values from a file. This command activates the error packet generation by default.

Syntax

HRESULT SetErrorPacketGeneration([in] BSTR FileName)

Arguments

Filename – File name from which the error packet values are to be loaded.

Returns

Table 5-52: IBPASystem::SetErrorPacketGeneration Returns

Value	Description
S_OK	The operation was successful.
BPA_INVALID_FILE_FORMAT	File format has bad syntax.
BPA_FILE_LOAD_ERROR	Unable to load specified file.

Examples

```
Dim App As Object
Dim System As Object
Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem
’Load the Error Packet values from a file.
System.SetErrorPacketGeneration ”C:\My Documents\Error.epg”
```

Remarks

Use ActivateErrorPacketGeneration command to enable or disable the error packet generation.

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPASystem::GetErrorPacketGeneration

This command saves the current error packet generated values onto the specified file.

Syntax

HRESULT GetErrorPacketGeneration([in] BSTR FileName)

Arguments

Filename - File name from which the error packet values are to be saved.

Returns

Table 5-53: IBPASystem::GetErrorPacketGeneration Returns

Value	Description
S_OK	The operation was successful.
BPA_INVALID_FILE_FORMAT	File format has bad syntax.
BPA_FILE_SAVE_ERROR	Unable to save specified file.

Examples (Visual Basic)

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Save the Error Packet values from a file.
System.GetErrorPacketGeneration ”C:\My Documents\Error.epg”
```

Remarks

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPASystem::ActivateErrorPacketGeneration

This command activates or deactivates the current Error Packet Generation setup.

Syntax

HRESULT ActivateErrorPacketGeneration([in] long Enable)

Arguments

Enable - Enable or Disable the current Error Packet Generation. This takes one of the following values:

Table 5-54: IBPASystem::ActivateErrorPacketGeneration Enable Values

Value	Description
BPA_ENABLE	Enable Error Packet Generation.
BPA_DISABLE	Disable Error Packet Generation.

Version - Software version of requested subsystem.

Returns

Table 5-55: IBPASystem::ActivateErrorPacketGeneration Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Activate Error Packet Generation
System. ActivateErrorPacketGeneration BPA_ENABLE
```

IBPASystem::SetDataFilter

This command sets baseband data filter. This command activates the data filter by default.

Syntax

HRESULT SetDataFilter([in] long IDPacket, [in] long NULLPacket, [in] long POLLPacket, [in] long ErrorPacket)

Arguments

IDPacket – Enable/Disable

NULLPacket – Enable/Disable

POLLPacket – Enable/Disable

ErrorPacket – Enable/Disable

All the arguments take on of the following values:

Table 5-56: IBPASystem::SetDataFilter Values

Value	Description
BPA_ENABLE	Enable Data Filter.
BPA_DISABLE	Disable Data Filter.

Returns

Table 5-57: IBPASystem::SetDataFilter Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_DATA	Invalid data filter values.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Set Data Filter Options
System. SetDataFilter BPA_ENABLE, BPA_ENABLE, BPA_DIS-
ABLE, BPA_ENABLE
```

Remarks

Use ActivateDataFilter command to enable or disable the data filter.

IBPASystem::GetDataFilter

This command retrieves the current Baseband Data Filter Settings.

Syntax

HRESULT GetDataFilter([out] long* IDPacket, [out] long* NULLPacket, [out] long* POLLPacket, [out] long* ErrorPacket)

Arguments

IDPacket - Enable/Disable

NULLPacket - Enable/Disable

POLLPacket - Enable/Disable

ErrorPacket - Enable/Disable

All the arguments take one of the following values:

Table 5-58: IBPASystem::SetDataFilter Values

Value	Description
BPA_ENABLE	Enable Data Filter.
BPA_DISABLE	Disable Data Filter.

Returns

Table 5-59: IBPASystem::SetDataFilter Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim IDPacket As Long
Dim NULLPacket As Long
Dim POLLPacket As Long
Dim ErrorPacket As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Get Data Filter Options
System. GetDataFilter IDPacket, NULLPacket, POLLPacket,
ErrorPacket
```

IBPASystem::ActivateDataFilter

This command activates or deactivates the current baseband data filter setup.

Syntax

HRESULT ActivateDataFilter([in] long Enable)

Arguments

Enable - Enable or Disable the current baseband data filter setup.

Table 5-60: IBPASystem::ActivateDataFilter Values

Value	Description
BPA_ENABLE	Enable Data Filter.
BPA_DISABLE	Disable Data Filter.

Version - Software version of requested subsystem.

Returns

Table 5-61: IBPASystem::ActivateDataFilter Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Activate Data Filter Options
System. ActivateDataFilter BPA_ENABLE
```

IBPASystem::SetDecryptionSettings

This command sets the decryption settings to specified values.

Syntax

HRESULT SetDecryptionSettings ([in] long type, [in] BSTR masterAddr, [in] long session, [in] BSTR linkKey, [in] long asciiFlag, [in] BSTR pairing, [in] BSTR slaveAddr, [in] long amAddr)

Arguments

type – Authentication or Pairing type flag. This takes one of the following values:

Table 5-62: IBPASystem::SetDecryptionSettings Type Values

Value	Description
BPA_DECRYPTION_AUTHENTICATE	Authentication.
BPA_DECRYPTION_PAIRING	Pairing.

masterAddr – Master BD Address.

session – Session type flag. This takes one of the following values:

Table 5-63: IBPASystem::SetDecryptionSettings Session Values

Value	Description
BPA_DECRYPTION_SINGLE_SESSION	Single session.
BPA_DECRYPTION_MULTI_SESSION	Multi session.

linkKey – Link Key.

asciiFlag – ASCII Flag.

pairing – PIN Key.

slaveAddr – Slave BD Address.

amAddr - AM Address.

Returns

Table 5-64: IBPASystem::SetDecryptionSettings Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_DECRYPTION_TYPE	Invalid decryption type.
BPA_E_INVALID_BDADDR	Invalid Bluetooth device address.
BPA_E_INVALID_SESSION_TYPE	Invalid session type.
BPA_E_INVALID_ASCII_FLAG	Invalid ASCII flag.
BPA_E_INVALID_AM_ADDR	Invalid active member address.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Set Decryption settings to single session
System. SetDecryptionSettings ( BPA_DECRYPTION_AUTHENTI-
CATE,
”00:50:CD:00:95:3E”, BPA_DECRYPTION_SINGLE_SESSION,
”1A1B”, 0, ””, ”00:50:CD:00:96:3F”, 1 )
...
...
...
’ Set Decryption settings to multi-session
```

```
' Call SetDecryptionSettings for each AM address.
System. SetDecryptionSettings ( BPA_DECRYPTION_AUTHENTI-
CATE,
"00:50:CD:00:95:3E", BPA_DECRYPTION_MULTI_SESSION,
"1A1B", 0, "", "00:50:CD:00:96:2F", 1 )
System. SetDecryptionSettings ( BPA_DECRYPTION_AUTHENTI-
CATE,
"00:50:CD:00:95:3E", BPA_DECRYPTION_MULTI_SESSION,
"1A1B", 0, "", "00:50:CD:00:96:4F", 2 )
System. SetDecryptionSettings ( BPA_DECRYPTION_AUTHENTI-
CATE,
"00:50:CD:00:95:3E", BPA_DECRYPTION_MULTI_SESSION,
"1A1B", 0, "", "00:50:CD:00:87:3F", 3 )
```

Remarks

BD Addresses are specified by colon separated bytes with MSB first and LSB last (00:50:CD:00:93:4E where 00:93:4E is the LAP field, 00:50 is the NAP field, CD is the UAP field).

IBPASystem::GetDecryptionSettings

This command gets the current decryption settings.

Syntax

HRESULT GetDecryptionSettings ([out] long* type, [out] BSTR* masterAddr, [out] long* session, [out] BSTR* linkKey, [out] long* asciiFlag, [out] BSTR* pairing, [out] BSTR* slaveAddr, [out] long* amAddr)

Arguments

type – Authentication or Pairing type flag. This takes one of the following values:

Table 5-65: IBPASystem::GetDecryptionSettings Type Values

Value	Description
BPA_DECRYPTION_AUTHENTICATE	Authentication.
BPA_DECRYPTION_PAIRING	Pairing.

masterAddr – Master BD Address.

session – Session type flag. This takes one of the following values:

Table 5-66: IBPASystem::GetDecryptionSettings Session Values

Value	Description
BPA_DECRYPTION_SINGLE_SESSION	Single session.
BPA_DECRYPTION_MULTI_SESSION	Multi session.

linkKey – Link Key.

asciiFlag – Ascii Flag.

pairing – Pairing Flag.

slaveAddr – Slave BD Address.

amAddr - AM Address.

Returns

Table 5-67: IBPASystem::GetDecryptionSettings Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim type As Long
Dim session As Long
Dim linkkey As Long
Dim asciiFlag As Long
Dim pairing As String
Dim masterAddr As String
Dim slaveAddr As String
Dim amAddr as Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Get the current Decryption settings
System.GetDecryptionSettings ( type, masterAddr, session, linkkey,
asciiFlag, pairing, slaveAddr, amAddr )
```

Remarks

BD Addresses are specified by colon separated bytes with MSB first and LSB last (00:50:CD:00:93:4E where 00:93:4E is the LAP field, 00:50 is the NAP field, CD is the UAP field).

IBPASystem::ActivateDecryption

This command activates or deactivates decryption.

Syntax

HRESULT ActivateDecryption([in] long Enable)

Arguments

Enable - Enable or Disable the decryption settings. This takes one of the following values:

Table 5-68: IBPASystem::ActivateDecryption Enable Values

Value	Description
BPA_ENABLE	Enable Decryption.
BPA_DISABLE	Disable Decryption.

Returns

Table 5-69: IBPASystem::ActivateDecryption Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Activate Decryption
System.ActivateDecryption BPA_ENABLE
```

IBPASystem::SetDecryptionDefault

This command sets the decryption settings to factory default values.

Syntax

HRESULT SetDecryptionDefault ()

Arguments

None.

Returns

Table 5-70: IBPASystem::SetDecryptionDefault Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’ Set Decryption settings to single session
System. SetDecryptionSettings ( BPA_DECRYPTION_AUTHENTI-
CATE,
”00:50:CD:00:95:3E”, BPA_DECRYPTION_SINGLE_SESSION,
”1A1B”, 0, ””, ”00:50:CD:00:96:3F”, 1 )
...
...
’ Set Decryption settings to factory defaults
System.SetDecryptionDefault
```

IBPASystem::DoDeviceDiscovery

This command performs a device discovery.

Syntax

HRESULT DoDeviceDiscovery([in] long Timeout, [in] long AccessCode)

Arguments

Timeout - Inquiry Timeout (in seconds).
AccessCode - Inquiry Access Code (e.g. 0x9E8B33)

Returns

Table 5-71: IBPASystem::DoDeviceDiscovery Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_TIMEOUT	Invalid timeout value.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Start the device discovery operation
System.DoDeviceDiscovery (10390323, 12)
```

Remarks

This command starts the device discovery operation. After calling this function, you need to call GetDeviceDicoveryStatus and wait in a loop till the operation gets completed. Then you need to call GetDeviceList to get all the BD addresses.

IBPASystem::GetDeviceDiscoveryStatus

This command returns the device discovery operation status.

Syntax

HRESULT GetDeviceDiscoveryStatus([out, retval] long* Status)

Arguments

Status - Device discovery operation status.

Table 5-72: IBPASystem:GetDeviceDiscoveryStatus Status Values

Value	Description
BPA_DISCOVERY_RUNNING(1)	The operation is still in progress.
BPA_DISCOVERY_COMPLETED(0)	The operation is completed.

Returns

Table 5-73: IBPASystem::GetDeviceDiscoveryStatus Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim status As Long
Dim deviceList As Variant

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Start the device discovery operation
System.DoDeviceDiscovery ( 12, 10390323)

’wait in a loop till the device discovery operation gets ’completed
Do ’ start the loop
status = System.GetDeviceDiscoveryStatus
Loop Until status = 0 ’ status = 0, then exit

’Get the list of devices
deviceList = System.GetDeviceList)
```

IBPASystem::GetDeviceList

Returns the list of devices found in the device discovery operation.

Syntax

HRESULT GetDeviceList([out, retval] VARIANT* DeviceList)

***NOTE**. Device names are returned as a VARIANT. The variant is of type VT_ARRAY and points to SAFEARRAY. The SAFEARRAY has dimension 1 and its elements are of type VT_BSTR. The number of devices is equal to the number of elements in the SAFEARRAY. The device names are returned in the same order as they are appear.*

Arguments

DeviceList – List of Devices found in discovery operation.

Returns

Table 5-74: IBPASystem::GetDeviceList Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim System As Object
Dim status As Long
Dim deviceList As Variant

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set System = App.GetSystem

’Start the device discovery operation
System.DoDeviceDiscovery ( 12, 10390323 )

’wait in a loop till the device discovery operation gets ’completed
Do ’ start the loop
status = System.GetDeviceDiscoveryStatus
Loop Until status = 0 ’ status = 0, then exit

’Get the list of devices
deviceList = System.GetDeviceList

’Access the device names
For Each deviceName in deviceList
’use device name in deviceName
Next deviceName
```

Remarks

If there are no devices found, the function returns an empty SAFEARRAY.

IBPASystemEvents::OnStateChange

This event is fired whenever the Protocol Analyzer changes states.

Syntax

HRESULT OnStateChange ([out, retval] long State)

Arguments

State - Current state of the Protocol Analyzer. This takes one of the following values:

Table 5-75: IBPASystemEvents::OnStateChange State Values

Value	Description
BPA_NO_SYNC	Protocol Analyzer is in idle state, not synchronized.
BPA_SYNC	Protocol Analyzer synchronized with another Bluetooth device.

Returns

Table 5-76: IBPASystemEvents::OnStateChange Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As BPAApplication
Dim WithEvents Sys As BPASystem
Set App = CreateObject(”BPA100.BPAApplication”)
’Get System object
Set Sys = App.Getsystem
Private Sub Sys_OnStateChange(ByVal p_nState As Long)
                    ’TODO : Add your event handler code here
End Sub
```

IBPASystemEvents::OnTriggerIn

This event is fired whenever the Trigger In port is asserted.

Syntax

HRESULT OnTriggerIn ()

Arguments

None

Returns

Table 5-77: IBPASystemEvents::OnTriggerIn Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim app As BPAApplication
Dim WithEvents Sys As BPASystem
Set App = CreateObject(”BPA100.BPAApplication”)
’Get System object
Set Sys = App.Getsystem
Private Sub Sys_OnTriggerIn( )
                    ’TODO : Add your event handler code here
End Sub
```

IBPASystemEvents::OnTriggerOut

This event is fired whenever the Trigger Out port is asserted.

Syntax

HRESULT OnTriggerOut ()

Arguments

None

Returns

Table 5-78: IBPASystemEvents::OnTriggerOut Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim app As BPAApplication
Dim WithEvents Sys As BPASystem

Set App = CreateObject("BPA100.BPAApplication")
'Get System object
Set Sys = App.Getsystem

Private Sub Sys_OnTriggerOut( )
                    'TODO : Add your event handler code here
End Sub
```

IBPAAnalyzer::Open

This command opens the trace object.

Syntax

HRESULT Open([in] BSTR Filename, [out, retval] long* ClientId)

Arguments

Filename - File name where the log file exists.

ClientId - Each file is assigned an identifier by the Protocol Analyzer Server.

For all subsequent calls to BPAAnalyzer, this identifier must be passed in the first parameter.

Returns

Table 5-79: IBPAAnalyzer::Open Returns

Value	Description
S_OK	The operation was successful.
BPA_FILE_OPEN_ERROR	Unable to open specified file. Cannot log data.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer

’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId
```

Remarks

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPAAnalyzer::Close

This command closes the file.

Syntax

HRESULT Close ([in] long ClientId)

Arguments

ClientId - The identifier that was returned from the call to Open().

Returns

Table 5-80: IBPAAnalyzer::Close Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId


’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetPacketCount

This command retrieves the number of packets in the specified protocol layer.

Syntax

HRESULT GetPacketCount([in] long ClientId, [in] long PacketType, [out, retval] long* Packets)

Arguments

ClientId – The identifier that was returned from the call to Open().

PacketType – The type of packet to search for. This takes one of the following values:

Table 5-81: IBPAAnalyzer::GetPacketCount PacketType Values

Value	Description
BPA_PACKET_BASEBAND	Baseband packet
BPA_PACKET_LMP	LMP packet
BPA_PACKET_L2CAP	L2CAP packet
BPA_PACKET_RFCOMM	RFCOMM packet
BPA_PACKET_SDP	SDP packet
BPA_PACKET_TCS	TCS packet
BPA_PACKET_OBEX	OBEX packet
BPA_PACKET_HDLC	HDLC packet
BPA_PACKET_PPP	PPP packet
BPA_PACKET_BNEP	BNEP packet
BPA_PACKET_AT_COMMANDS	AT Commands packet
BPA_PACKET_HID	HID packet
BPA_PACKET_TRIGGER	TRIGGER packet

Packets - number of packets.

Returns

Table 5-82: IBPAAnalyzer::GetPacketCount Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_INVALID_PACKET_TYPE	Invalid packet type.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim PacketCount As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Get LMP Packet Count
PacketCount = Analyzer.GetPacketCount ClientId, BPA_PACK-
ET_LMP

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetPacket

This command retrieves the specified packet of a given protocol layer.

Syntax

HRESULT GetPacket ([in] long ClientId, [in] long PacketNum, [in] long PacketType, [out, retval] VARIANT* Packet)

***NOTE**. Packet is returned as a VARIANT. The variant is of type VT_ARRAY and points to SAFEARRAY. The SAFEARRAY has dimension 1 and its elements are of type VT_UI1. The number of bytes in packet is equal to the number of elements in the SAFEARRAY.*

Arguments

ClientId – The identifier that was returned from the call to Open ( ).

PacketNum – The packet number.

PacketType – The type of packet to search for. This takes one of the following values:

Table 5-83: IBPAAnalyzer::GetPacket PacketType Values

Value	Description
BPA_PACKET_BASEBAND	Baseband packet
BPA_PACKET_LMP	LMP packet
BPA_PACKET_L2CAP	L2CAP packet
BPA_PACKET_RFCOMM	RFCOMM packet
BPA_PACKET_SDP	SDP packet
BPA_PACKET_TCS	TCS packet
BPA_PACKET_OBEX	OBEX packet

Table 5-83: IBPAAnalyzer::GetPacket PacketType Values (Cont.)

Value	Description
BPA_PACKET_HDLC	HDLC packet
BPA_PACKET_PPP	PPP packet
BPA_PACKET_BNEP	BNEP packet
BPA_PACKET_AT_COMMANDS	AT Commands packet
BPA_PACKET_HID	HID packet
BPA_PACKET_TRIGGER	TRIGGER packet

Packet - The packet data. NULL if non-existent.

Returns

Table 5-84: IBPAAnalyzer::GetPacket Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_INVALID_PACKET_NUMBER	Invalid packet number.
BPA_E_INVALID_PACKET_TYPE	Invalid packet type.
BPA_E_NOT_FOUND	Packet not found.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim PacketData As String

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Get 100th Packet data
PacketData = Analyzer.GetPacket ClientId, 100,
BPA_PACKET_L2CAP

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetPrevPacketNumber

This command searches for the packet previous to the specified packet and type in the Baseband layer and returns the index of the matching packet.

Syntax

HRESULT GetPrevPacketNumber ([in] long ClientID, [in] long StartPacketNum, [in] long PacketType, [out, retval] long* PacketNum)

Arguments

ClientId – The identifier that was returned from the call to Open ( ).

StartPacketNum – Index of the packet from which the search has to start.

PacketType – The type of Baseband packet to search for. This takes one of the following values:

Table 5-85: IBPAAnalyzer::GetPrevPacketNumber PacketType Values

Value	Description
BPA_NULL_PACKET	Null packet
BPA_POLL_PACKET	Poll packet
BPA_FHS_PACKET	FHS packet
BPA_DM1_PACKET	DM1 packet
BPA_DH1_PACKET	DH1 packet
BPA_HV1_PACKET	HV1 packet
BPA_HV2_PACKET	HV2 packet
BPA_HV3_PACKET	HV3 packet
BPA_DV_PACKET	DV packet
BPA_AUX1_PACKET	AUX1 packet
BPA_DM3_PACKET	DM3 packet

Table 5-85: IBPAAnalyzer::GetPrevPacketNumber PacketType Values (Cont.)

Value	Description
BPA_DH3_PACKET	DH3 packet
BPA_DM5_PACKET	DM5 packet
BPA_DH5_PACKET	DH5 packet
BPA_ID_PACKET	ID packet
BPA_ERROR_PACKET	Access Error

PacketNum – Index of the matching packet.

Returns

Table 5-86: IBPAAnalyzer::GetPrevPacketNumber Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_INVALID_PACKET_NUMBER	Invalid packet number.
BPA_E_INVALID_PACKET_TYPE	Invalid packet type.
BPA_E_END_OF_FILE	Reached end of file.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim PacketNum As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get Analyzer
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Find a DM1 packet which is previous to 100th packet in Baseband
Layer
PacketNum = Analyzer.GetPrevPacketNumber(ClientID, 100,
BPA_DM1_PACKET

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetNextPacketNumber

This command searches for the next packet to the specified packet and type in the Baseband layer and returns the index of the matching packet.

Syntax

HRESULT GetNextPacketNumber([in] long ClientID, [in] long StartPacketNum, [in] long PacketType, [out, retval] long* PacketNum)

Arguments

ClientId – The identifier that was returned from the call to Open ( ).

StartPacketNum – Index of the packet from which the search has to start.

PacketType – The type of Baseband packet to search for. This takes one of the following values:

Table 5-87: IBPAAnalyzer::GetNextPacketNumber PacketType Values

Value	Description
BPA_NULL_PACKET	Null packet
BPA_POLL_PACKET	Poll packet
BPA_FHS_PACKET	FHS packet
BPA_DM1_PACKET	DM1 packet
BPA_DH1_PACKET	DH1 packet
BPA_HV1_PACKET	HV1 packet
BPA_HV2_PACKET	HV2 packet
BPA_HV3_PACKET	HV3 packet
BPA_DV_PACKET	DV packet
BPA_AUX1_PACKET	AUX1 packet
BPA_DM3_PACKET	DM3 packet

Table 5-87: IBPAAnalyzer::GetNextPacketNumber PacketType Values (Cont.)

Value	Description
BPA_DH3_PACKET	DH3 packet
BPA_DM5_PACKET	DM5 packet
BPA_DH5_PACKET	DH5 packet
BPA_ID_PACKET	ID packet
BPA_ERROR_PACKET	Access Error

PacketNum – Index of the matching packet.

Returns

Table 5-88: IBPAAnalyzer::GetNextPacketNumber Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_INVALID_PACKET_NUMBER	Invalid packet number.
BPA_E_INVALID_PACKET_TYPE	Invalid packet type.
BPA_E_END_OF_FILE	Reached end of file.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim PacketNum As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get Analyzer
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Find a DH5 packet which comes after the 100th packet in Baseband
Layer
PacketNum = Analyzer.GetNextPacketNumber(ClientID, 100,
BPA_DH5_PACKET )

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetPacketInfo

This command retrieves the decoded information of a specified packet depending on the specified information type.

Syntax

HRESULT GetPacketInfo([in] long ClientId, [in]long PacketType, [in] long PacketNum, [in] long InfoType, [in] long Radix, [out, retval] BSTR *Information)

Arguments

ClientId - The identifier that was returned from the call to Open().

PacketNum - The packet number

PacketType - The type of packet to search for. This takes one of the following values:

Table 5-89: IBPAAnalyzer::GetPacketInfo PacketType Values

Value	Description
BPA_PACKET_BASEBAND	Baseband packet
BPA_PACKET_LMP	LMP packet
BPA_PACKET_L2CAP	L2CAP packet
BPA_PACKET_RFCOMM	RFCOMM packet
BPA_PACKET_SDP	SDP packet
BPA_PACKET_TCS	TCS packet
BPA_PACKET_OBEX	OBEX packet
BPA_PACKET_HDLC	HDLC packet
BPA_PACKET_PPP	PPP packet
BPA_PACKET_BNEP	BNEP packet
BPA_PACKET_AT_COMMANDS	AT Commands packet

Table 5-89: (Cont.)IBPAAnalyzer::GetPacketInfo PacketType Values

Value	Description
BPA_PACKET_HID	HID packet
BPA_PACKET_TRIGGER	TRIGGER packet

InfoType – Type of Information to be request. This takes one of the following values:

Table 5-90: IBPAAnalyzer::GetPacketInfo InfoType Values

Value	Description
BPA_INFO_INDEX	Packet Index
BPA_INFO_TIME	Time
BPA_INFO_TIME_TICKS	Time ticks
BPA_INFO_SLAVE_MASTER	Slave/Master
BPA_INFO_AM_ADDR	AM Address
BPA_INFO_RX_TX	Rx/Tx
BPA_INFO_TYPE	Packet type
BPA_INFO_FLOW	Flow
BPA_INFO_ARQN	ARQN
BPA_INFO_SEQN	SEQN
BPA_INFO_HOPFREQ	Hop Frequency
BPA_INFO_TRAN_ID	Transaction ID
BPA_INFO_HEC	HEC
BPA_INFO_CRC	CRC
BPA_INFO_DESCRIPTION	Packet description
BPA_INFO_PAYLOAD	Payload data

Radix - Radix of Information to be requested. This takes one of the following values:

Table 5-91: IBPAAnalyzer::GetPacketInfo Radix Values

Value	Description
BPA_RAD_HEX	Hexadecimal
BPA_RAD_DECIMAL	Decimal
BPA_RAD_OCTAL	Octal
BPA_RAD_BINARY	Binary
BPA_RAD_ASCII	ASCII

Information - Information provided about specified packet as string value.

Returns

Table 5-92: IBPAAnalyzer::GetPacketInfo Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_INVALID_PACKET_TYPE	Invalid packet type.
BPA_E_INVALID_PACKET_NUMBER	Invalid packet number.
BPA_E_INVALID_INFO_TYPE	Invalid information type.
BPA_E_INVALID_RADIX	Invalid radix value.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim Info As String

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Get 100th Packet’s Type as string
Info = Analyzer.GetPacketInfo ClientId, 100, BPA_INFO_TYPE,
BPA_RAD_ASCII

’Close the Open log file
Analyzer.Close ClientId
```

Remarks

Use this function to get the decoded strings for the Index, Packet-Type, Description and Payload data column as it is displayed in the BPAServer GUI. With this function you can recreate the Protocol Analyzer Server GUI list window.

IBPAAnalyzer::Export

This command exports the specified protocol layer's packets of the current log file to a CSV or TXT or WAV file given beginning and end packet number.

Syntax

HRESULT Export ([in] long ClientId, [in] long FileType, [in] long ProtocolType, [in] long StartPacketNum, [in] long EndPacketNum, [in] BSTR Filename)

Arguments

ClientId - The identifier that was returned from the call to Open().

FileType - The file type to be created. This takes one of the following values:

Table 5-93: IBPAAnalyzer::Export FileType Values

Value	Description
BPA_FILETYPE_CSV	Comma seperated values.
BPA_FILETYPE_TEXT	Text
BPA_FILETYPE_WAV	Audio

ProtocolType - Type of protocol. This takes one of the following values:

Table 5-94: IBPAAnalyzer::Export ProtocolType Values

Value	Description
BPA_PACKET_BASEBAND	Baseband packet
BPA_PACKET_LMP	LMP packet
BPA_PACKET_L2CAP	L2CAP packet
BPA_PACKET_RFCOMM	RFCOMM packet

Table 5-94: IBPAAnalyzer::Export ProtocolType Values (Cont.)

Value	Description
BPA_PACKET_SDP	SDP packet
BPA_PACKET_TCS	TCS packet
BPA_PACKET_OBEX	OBEX packet
BPA_PACKET_HDLC	HDLC packet
BPA_PACKET_PPP	PPP packet
BPA_PACKET_BNEP	BNEP packet
BPA_PACKET_AT_COMMANDS	AT Commands packet
BPA_PACKET_HID	HID packet
BPA_PACKET_TRIGGER	TRIGGER packet

StartPacketNum - packet number of where to begin the export. Specify -1 to indicate from the first packet.

EndPacketNum - packet number of where to end the export. Specify a -1 to indicate the last packet.

Filename - string providing the full path name of file of where to export the data.

Returns

Table 5-95: IBPAAnalyzer::Export Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_INVALID_FILE_TYPE	Invalid file type.
BPA_E_INVALID_PACKET_TYPE	Invalid packet type.
BPA_E_INVALID_START_PACKET_NUMBER	Invalid start packet number.

Table 5-95: IBPAAnalyzer::Export Returns (Cont.)

Value	Description
BPA_E_INVALID_END_PACKET_NUMBER	Invalid end packet number.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Export all the LMP packets
Analyzer.Export ClientId, BPA_FILE_TYPE_CSV, BPA_PACK-
ET_LMP, -1,
-1, ”C:\LMPdata.txt”

’Close the Open log file
Analyzer.Close ClientId
```

Remarks

All file paths without machine qualifiers refer to drives mapped on the Protocol Analyzer Server.

IBPAAnalyzer::GetAcquisitionReport

This command gets the acquisition report created for the log file.

Syntax

HRESULT GetAcquisitionReport([in] long ClientId, [out, retval] BSTR* Report)

Arguments

ClientId – The identifier that was returned from the call to Open ( ).

Report – Acquisition report.

Returns

Table 5-96: IBPAAnalyzer::GetAcquisitionReport Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim Report As String

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Get the acquisition report
Report = Analyzer.GetAcquisitionReport ClientId

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::SetL2CAPConnectionProperties

This command assigns the given L2CAP connection property type for a given packet.

Syntax

HRESULT SetL2CAPConnectionProperties ([in] long ClientId, [in] long packetIndex, [in] long type)

Arguments

ClientId – The identifier that was returned from the call to Open().

packetIndex – index of the packet from the acquired data file.

type – Protocol type. This takes one of the following values:

Table 5-97: IBPAAnalyzer::SetL2CAPConnectionProperties Type values

Value	Description
BPA_L2CAPTYPE_RFCOMM	RFCOMM
BPA_L2CAPTYPE_SDP	SDP
BPA_L2CAPTYPE_NET	NET (Digianswer specific)
BPA_L2CAPTYPE_RFCOMM_FLOW	RFCOMM Flow Control
BPA_L2CAPTYPE_TCS	TCS
BPA_L2CAPTYPE_BNEP	BNEP
BPA_L2CAPTYPE_HID_CONTROL	HID control
BPA_L2CAPTYPE_HID_INTERRUPT	HID Interrupt
BPA_L2CAPTYPE_UNKNOWN	Unknown

Returns

Table 5-98: IBPAAnalyzer::SetL2CAPConnectionProperties Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long

Set App = CreateObject( “BPA100.BPAApplication“ )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open “C:\LogFile.tbpa“, ClientId

’Set the L2CAP Connection properties
Analyzer.SetL2CAPConnectionProperties( ClientId, 64,
BPA_L2CAPTYPE_SDP )

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetL2CAPConnectionProperties

This command retrieves the current L2CAP connection property assignment for a given packet.

Syntax

HRESULT GetL2CAPConnectionProperties ([in] long ClientId, [in] long packetIndex, [out, retval] long* type)

Arguments

ClientId – The identifier that was returned from the call to Open().

packetIndex – index of the packet from the acquired data file.

type – Protocol type. This takes one of the following values:

Table 5-99: IBPAAnalyzer::GetL2CAPConnectionProperties Type values

Value	Description
BPA_L2CAPTYPE_RFCOMM	RFCOMM
BPA_L2CAPTYPE_SDP	SDP
BPA_L2CAPTYPE_NET	NET (Digianswer specific)
BPA_L2CAPTYPE_RFCOMM_FLOW	RFCOMM Flow Control
BPA_L2CAPTYPE_TCS	TCS
BPA_L2CAPTYPE_BNEP	BNEP
BPA_L2CAPTYPE_HID_CONTROL	HID control
BPA_L2CAPTYPE_HID_INTERRUPT	HID Interrupt
BPA_L2CAPTYPE_UNKNOWN	Unknown

Returns

Table 5-100: IBPAAnalyzer::GetL2CAPConnectionProperties Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim type As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Set the L2CAP Connection properties
type = Analyzer.GetL2CAPConnectionProperties( ClientId, 64 )

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::SetRFCOMMServerChannel

This command assigns the RFCOMM server channel assignments for a specified packet.

Syntax

HRESULT SetRFCOMMServerChannel ([in] long ClientId, [in] long packetIndex, [in] long type)

Arguments

ClientId – The identifier that was returned from the call to Open().

packetIndex – index of the packet from the acquired data file.

type – Protocol type. This takes one of the following values:

Table 5-101: IBPAAnalyzer::SetRFCOMMServerChannel Type Values

Value	Description
BPA_RFCOMM_OBEX	OBEX
BPA_RFCOMM_HDLC	HDLC
BPA_RFCOMM_AT	AT
BPA_RFCOMM_AT_HDLC	AT+HDLC
BPA_RFCOMM_UNKNOWN	Unknown

Returns

Table 5-102: IBPAAnalyzer::SetRFCOMMServerChannel Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long

Set App = CreateObject( “BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Set the RFCOMM Server Channel assignment
Analyzer.SetRFCOMMServerChannel( ClientId, 64,
BPA_RFCOMM_OBEX )

’Close the Open log file
Analyzer.Close ClientId
```

IBPAAnalyzer::GetRFCOMMServerChannel

This command retrieves the RFCOMM server channel assignments for a specified packet.

Syntax

HRESULT GetRFCOMMServerChannel ([in] long ClientId, [in] long packetIndex, [out, retval] long* type)

Arguments

ClientId – The identifier that was returned from the call to Open().

packetIndex – index of the packet from the acquired data file.

type – Protocol type. This takes one of the following values:

Table 5-103: IBPAAnalyzer::GetRFCOMMServerChannel Type Values

Value	Description
BPA_RFCOMM_OBEX	OBEX
BPA_RFCOMM_HDLC	HDLC
BPA_RFCOMM_AT	AT
BPA_RFCOMM_AT_HDLC	AT+HDLC
BPA_RFCOMM_UNKNOWN	Unknown

Returns

Table 5-104: IBPAAnalyzer::GetRFCOMMServerChannel Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_CLIENT_ID	Invalid Client ID.
BPA_E_FAILED	The operation was unsuccessful.

Examples (Visual Basic)

```
Dim App As Object
Dim Analyzer As Object
Dim ClientId As Long
Dim type As Long

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get System
Set Analyzer = App.GetAnalyzer
’Open log file
Analyzer.Open ”C:\LogFile.tbpa”, ClientId

’Get the RFCOMM Server Channel assignment
type = Analyzer.GetRFCOMMServerChannel( ClientId, 64 )

’Close the Open log file
Analyzer.Close ClientId
```

IBPAHCISimple::Send

This command sends an HCI message.

Syntax

HRESULT Send([in] long ogf, [in] long ocf, [in] BSTR params)

Arguments

ogf - OpCode group field.

ocf - OpCode command field.

params - list of parameters to pass.

waitEvent - wait on this event type to a maximum of 1000 ms. Use NULL to specify not to wait for any event.

Returns

Table 5-105: IBPAHCISimple::Send Returns

Value	Description
S_OK	The operation was successful.
BPA_E_INVALID_OGF	Invalid OGF file.
BPA_E_INVALID_OCF	Invalid OCF file.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim Hci As Object

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get HCI simple
Set Hci = App.GetHCISimple

’Send HCI Command
Hci.Send( 1, 0104, ”16 40 00 CD 50 00 FF 00 00 00 00 00 01” )
```

IBPAHCISimple::Get

This command gets the last received HCI event message. This commands waits for the specified event to occur before either returning with a parameter list or timing out.

Syntax

HRESULT Get ([out, retval] BSTR* params)

Arguments

params – list of parameters passed back from the event.

Returns

Table 5-106: IBPAHCISimple::Get Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim Hci As Object
Dim eventParams As String

Set App = CreateObject( ”BPA100.BPAApplication” )
’Get HCI simple
Set Hci = App.GetHCISimple

’Send HCI Command
Hci.Send( 1, 0104, ”16 40 00 CD 50 00 FF 00 00 00 00 00 01” )
...
Get the event parameters
eventParams = Hci.Get
```

Remarks

After sending the HCI command, you need to wait in a loop till the event is received. Only after the event is received, you can call BPAHCISimple::Get() to get the event parameters.

IBPAHCISimpleEvents::HCIEvent

This event is fired whenever the BPA receives an HCI event from the hardware.

Syntax

HRESULT HCIEvent([in] VARIANT EventPacket)

NOTE. *EventPacket is returned as a VARIANT. The variant is of type VT_ARRAY and points to SAFEARRAY. The SAFEARRAY has dimension 1 and its elements are of type VT_UI1. The number of bytes in event packet is equal to the number of elements in the SAFEARRAY.*

Arguments

EventPacket – HCI Event from the hardware.

Returns

Table 5-107: IBPAHCISimpleEvents::HCIEvent Returns

Value	Description
S_OK	The operation was successful.
BPA_E_FAILED	The operation was unsuccessful.

Examples

```
Dim App As Object
Dim WithEvents Hci As Object

Set App = CreateObject(”BPA100.BPAApplication”)
’Get HCI simple object
Set Hci = App.GetHCISimple

Private Sub Hci_HCIEvent(ByVal EventPacket As Variant)
                    ’TODO : Add your event handler code here
End Sub
```

Appendices

Appendix A: Other Issues with DCOM

If the Protocol Analyzer Server is running on Windows 95/98, ensure that DCOM software is installed on the Server started before the clients connect.

Follow these steps to enable DCOM on Windows 95/98 computer:

1. Click **Start > Run**. The Run dialog box appears.
2. Type **Regedit**.
3. Go to the key HKEY_LOCAL_MACHINE\Software\Microsoft\Ole.
4. Double-click **EnableDCOM**. The Edit String dialog box appears.
5. In Value data, type **Y**.
6. Double-click EnableRemoteClient. The Edit String dialog box appears.
7. In Value data, type **Y**.

***NOTE**. If you do not find EnableDCOM and EnableRemoteClient string values in the Registry, create them with Value data as Y.*

Glossary

Glossary

BD_ADDR (Bluetooth Device Address)

The Bluetooth Device Address is a unique, 48-bit number used to identify a Bluetooth device. The Bluetooth device address is also used in encryption and in generation of frequency hop sequences. Sometimes referred to as the Ethernet MAC address of a Bluetooth device.

Bluetooth

An open specification for wireless communication of data and voice. It is based on a low-cost, short-range radio link facilitating protected ad hoc connections for stationary and mobile communication environments.

Bluetooth Protocol Layers

The Bluetooth protocol stack is a collection of lower-level, adaptation, and higher-level protocols. The following explains where the protocols implemented in the BPA100 analyzer are found in this rough breakdown of the Bluetooth protocol stack.

- Lower layer protocols. This protocol layer provides basic physical layer communication and link management among Bluetooth devices. Baseband and LMP protocols are used at this layer.
- Adaptation layer protocol. This protocol layer provides interoperability between higher-level and lower-level protocols through multiplexing, segmentation and reassembly, device discovery, and QoS protocols. L2CAP protocol is used at this layer.
- Upper layer protocols. This protocol layer provides device and service discovery and allows legacy devices to interact. SDP, RFCOMM, OBEX, HDLC, and PPP protocols are used at this layer.

Device Discovery

Before a link can be established, a Bluetooth device has to discover the other Bluetooth devices that are active within its range. Device discovery is the mechanism used to request and receive the Bluetooth address, clock, class of device, used page scan, and names of devices.

Encryption

Security mechanism that prevents eavesdropping and maintains link privacy.

Host Controller Interface (HCI)

Allows higher layers of the stack, including applications to access the baseband, link manager, and other hardware registers through a single, standard interface.

Master Device

The device that initiates a connection and, during this connection, controls all traffic in a piconet. The clock and hopping sequence of the master are used to synchronize all other devices in the piconet.

Piconet

A wireless network formed by two or more Bluetooth devices.

Scatternet

Multiple independent and nonsynchronized piconets form a scatternet.

TBPA

Tektronix Bluetooth Protocol Analyzer data file format (for instance, your_pod.tbpa)

Index

Index

A

B

C

D

E

G

H

I

L

O

P

R

S

T

U

W