



WHITE PAPER

# Using Windows Server 2008 as a Platform for Legacy Integration

## C O N T E N T S

- INTRODUCTION
- EXTENSIVE KNOWLEDGE OF .NET, ACTIVEX, VISUAL BASIC, VB SCRIPT, ASP AND MORE
- NON-INTRUSIVE USE OF HOST SCREEN AS AN API SPEEDS DELIVERY OF APPLICATION TO MARKET
- EXTRA SECURITY DELIVERED THROUGH SSL/TLS AND SSH
- DEVELOPER INSTRUCTIONS ON USING PASSPORT HIO
- 3270 IS OUR BUSINESS
- OUR CLIENT LIST IS IMPRESSIVE

Given the proliferation of Microsoft Windows servers and desktops within large IBM mainframe accounts, the Windows platform can be the perfect place for low-cost, low-risk, screen-based legacy integration.

Large insurance companies, banks, retailers and others have succeeded with Microsoft-based 3270 and 5250 legacy integration projects where much more complex and expensive Java-based legacy integration solutions have disappointed.

---

Research shows that the larger the IBM mainframe account, the more likely they are to extensively use Microsoft Windows servers and desktops.

In this document, we will consider the strategic benefits of using Microsoft Windows Server 2008 as a platform for legacy integration projects and offer examples of how Zephyr's PASSPORT Host Integration Objects (HIO) can be used as a proven legacy integration solution.

Some of the legacy integration projects Zephyr have participated with:

- Three of the largest banks in the U.S. use PASSPORT to integrate vital branch banking applications with their IBM 3270 mainframe applications. These mission critical applications, used daily by tens of thousands of users at each bank, offer an example of how the Microsoft Server can play a secure, intermediary role between the user and the IBM mainframe.
- One of the largest healthcare system providers in the U.S. built a Medicare access solution with PASSPORT that offers web-based access to Medicare systems and services for hospitals and healthcare organizations.

The Medicare applications run on IBM mainframes. This solution has been enormously successful and brought immediate sources of revenue for the organization that developed the solution.

- PASSPORT has also been used by car rental companies, telecom providers, government agencies and others to integrate 3270 and 5250 host applications with Windows SQL Server applications, Visual Basic applications and more.

#### **Zephyr: The Source for Microsoft-based Legacy Integration and Terminal Emulation**

Visit [www.zephyrcorp.com/products](http://www.zephyrcorp.com/products) to view Case Studies of how PASSPORT has been used in various legacy integration projects.

For more information on our products and services, or to contact a legacy integration or terminal emulation specialist, visit [www.zephyrcorp.com](http://www.zephyrcorp.com).

### Extensive knowledge of .NET, ActiveX, Visual Basic, VB Script, ASP and more

One of the principal benefits of using a Microsoft platform for legacy integration is the extensive knowledge of Microsoft programming languages and the broad amount of resources at your disposal for an integration project involving legacy applications and assets.

### Non-intrusive use of host screen as an API speeds delivery of application to market

Though certainly not new, scraping 3270 and 5250 host screens as a method of passing data to and from legacy applications offers one of the fastest, if not the fastest, platform for legacy application integration. This is particularly true if developers have knowledge of the IBM host application or if they have access to mainframe application managers or power end users that do.

### Extra security delivered through SSL/TLS and SSH

As an extra layer of security, Microsoft-based integration applications have access to SSL/TLS and SSH security found on IBM zSeries (Mainframe), iSeries (AS/400) and UNIX servers and PASSPORT supports the use of SSL/TLS and SSH security to reach these legacy applications.

### PASSPORT Host Integration Objects

PASSPORT Host Integration Objects (HIO) offers a path to IBM 3270 and 5250 mainframe applications at the screen buffer level, allowing you to read and write data to the host presentation space and input fields, open and close sessions, get specific text strings from the screen, send function keys to the host, and much more.

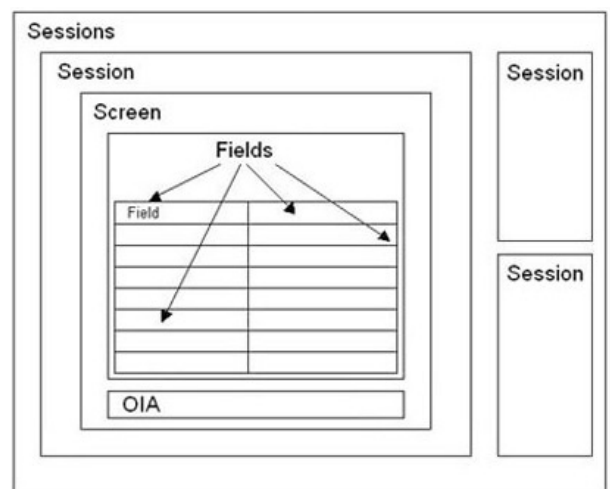
PASSPORT HIO enables programmers to write .NET applications deployed from Windows Server 2008 that integrate with these IBM 3270 and 5250 host applications.

Though many deploy PASSPORT HIO from a Windows 2008 Server, PASSPORT HIO can also

be used to develop client-side integration solutions on Microsoft Windows 7 and XP desktops, and features the PASSPORT HIO Terminal Control object to expose the 3270/5250 screen and keyboard interface to the user.

### PASSPORT HIO Hierarchy

PASSPORT HIO has the following containment hierarchy:



Sessions: Contains a collection of Session objects.

Session: A host session containing one Screen object.

Screen: Encapsulates the host presentation space containing OIA and Fields objects.

OIA: The operator information area of a host session.

Fields: Contains a collection of Field objects.

Field: A field in the presentation space. A field is the fundamental element of a virtual screen.

### Starting a C# .NET Project

To begin, Start Microsoft Visual Studio

1. From the menu select File->New->Project... to create a new project.
2. Under Project types, choose Other Languages->Visual C#->Windows

3. Under Templates, choose Windows Forms
4. For the Name field, enter "TestHIO" or a name of your choice.
5. Visual Studio will create a skeleton project for you.
6. Next, we will need to add reference to the PASSPORT HIO object.
7. Open the menu Project->Add Reference, choose the Browse tab, and browse to the PASSPORT HIO installation folder (e.g. C:\Program Files\PASSPORT Host Integration Objects) and select the file PASSHIO.dll

We can now reference the PASSPORT HIO objects:

```
using PASSHIOLib;
```

### Zephyr Communication Configuration (.ZCC) File

You will need a Zephyr Communication Configuration (.ZCC) file to specify the connection information to your 3270 or 5250 host. You can copy the LOCIS.zcc file from the TermControlSample folder. At the very least, you will need to fill in the IPHostName and TCPPort for your host connection. You can read more information about the .ZCC file format [here](#). An example of a .ZCC file:

```
[Connection]
IPHostName=127.0.0.1
TCPPort=23
EmulType=TN3270
TimeOut=30
TN3270ESupport=Yes
ConnectMethod=Generic
ResourceName=
ScreenSize=2
ExtendedAttributes=Yes
AutoReconnect=Yes
HostCodePage=037
```

### Establishing Host Sessions

Before you can create a host session, you will need to create a Sessions object. The Sessions object contains and manages a collection of individual Session objects. All applications should allocate a single Sessions object.

```
hioSessMgr = new PASSHIOLib.OhioSessions();
```

After creating this manager object you will need to create a Session object for your particular host. To do this, call AddSession and specify the .ZCC configuration file. For example:

```
hioSession = hioSessMgr.AddSession(@"C:\Program Files\PASSPORT Host Integration Objects\locis.zcc", 1);
```

Note the second parameter should set to 1 if you want to receive the OnSessionChanged, OnScreenChanged and OnOIACChanged events. We will discuss events later. You should also enclose it with a try-catch block to handle

exceptions such as out of license or other errors.

To start a connection to the host, use the Connect method.

```
hioSession.Connect();
```

The Connect method only initiates a connection to the host and returns immediately. Before going further, you need to wait for the connection to be fully established. You may do that with a simple loop or you may wait for the OnSessionChanged event to be fired. An example of doing a simple wait loop:

```
while ( hioSession.isConnected() == 0 )  
{  
    System.Threading.Thread.Sleep(500);  
}
```

To disconnect a session from the host, use the Disconnect method:

```
hioSession.Disconnect();
```

When you are done with the host session, use the CloseSession method to release all resources associated with it. This method requires the name of the session you are closing. For example:

```
hioSessMgr.CloseSession( hioSession.SessionName );  
Marshal.ReleaseComObject( hioSession );
```

Note: Because PASSPORT HIO is a COM object, the Session object must be explicitly released by calling Marshal.ReleaseComObject. Otherwise, the .NET framework will not release the object until the next garbage collection causing the PASSPORT HIO license to not be released.

### Reading Data from the Mainframe Screen

Once the host session is connected, you can access the mainframe screen data through the Screen object. The Screen object provides methods that allow you to read the host screen text, search the screen, send keystrokes to the host, and work with the cursor. The cursor is an offset into the host screen and ranges from 0 to the last character of the screen buffer (1919 for a 3270 model 2 screen). All the methods of the Screen object use this cursor offset for accessing data.

To obtain the Screen object:

```
OhioScreen hioScreen = hioSession.Screen;
```

There are different ways you can read the text currently displayed on the host screen. The simplest and recommended way is to use the String property which contains the entire host screen as a single string.

```
Debug.Print("Entire Host Screen={0}", hioScreen.String);
```

You can use the Substring function to extract different area of the host screen.

```
Debug.Print("First Row Of Host Screen={0}", hioScreen.String.Substring(0, hioScreen.Columns));
```

The following example checks if the specified text appears at the specified row and column position:

```
public bool IsTextXY(int row, int col, string strFind)
{
    int nPos = (row - 1) * hioScreen.Columns + col - 1;
    string strScreen = hioScreen.String.Substring(nPos, strFind.Length);
    return strScreen.Equals(strFind, StringComparison.OrdinalIgnoreCase);
}
```

### Alternative Method of Reading Data

The Fields property of the Screen object provides access to the collection of fields on the host screen. A field is the fundamental element of the screen. A field includes both data (sequence of characters) and attributes describing the field.

To get the Fields object, use the Fields property of the Screen object:

```
OhioFields hioFields = hioScreen.Fields;
```

For example, to print the texts of all the fields in the order of their appearance on the screen:

```
for (int i=1; i<= hioFields.Count; i++)
    Debug.Print("Field{0}={1}", i, hioFields[i].String);
```

### Writing Data to the Mainframe Screen

The mainframe screen contains protected and unprotected fields and only unprotected fields can be written with user data. To write data to the host screen, you can use the PutString and the SendKeys methods. PutString puts a string on a particular screen location overlaying only unprotected fields. Any parts of the string which fall into protected fields will be discarded. The cursor is not changed after a PutString. On the other hand, SendKeys simulates keystrokes typed by the user so the cursor position may be changed after a SendKeys.

```
hioScreen.SendKeys("login", hioScreen.Cursor);
hioScreen.PutString("testuser", hioScreen.Cursor);
```

### Sending Data to the Mainframe

The text you place on the mainframe screen using PutString or SendKeys is not sent to the host until you issue a SendAid with certain Attention Identifier (AID) keys. The SendAid method sends an AID key to the host screen. These keys can be thought of as special non-textual keystrokes.

Note that not all AID keys will cause data to be sent to the 3270 or 5250 host. For example, the OHIO\_KEY\_TABFORWARD key only moves the cursor to the next field position. When AID keys that do cause transmissions to the host (such as ENTER key, ATTN key and PF keys) are used, PASSPORT HIO encodes all text data you have entered and sends it to the host. The following example sends the text "login" to the host:

```
hioScreen.SendKeys("login", hioScreen.Cursor);
hioScreen.SendAid( OHIO_KEY.OHIO_KEY_ENTER );
```

After data is sent to the host, you must then wait for the host to process and respond to you. There are various

ways to wait for this to happen:

**WaitForStr** – Wait for certain text to appear at certain location on the host screen.

**WaitForNoX** – Wait for the X () or X SYSTEM to be removed from the OIA line .This is not always reliable because the host may send an unpredictable numbers of buffers back. Do not use this method unless you are certain of the outcome.

**OnScreenChanged** – Wait for this event to be called and use this method if you design your application in an event-driven way.

We recommend using a combination of **WaitForStr** and **WaitForNoX**. For example:

```
hioScreen.WaitForStr("USERID", 18, 5000);  
hioScreen.WaitForNoX(5000);
```

Note: It is highly recommended to call **WaitForNoX** before calling **SendAid**. If the host is in an input inhibited state, the call to **SendAid** may fail silently. Additionally, **Screen.Cursor** property can be checked to increase the reliability of the screen detection. Once we recognize the newly received screen, we can continue to do the writing and sending of data to the mainframe and repeat this process until we are finished.

### Operator Interface Area (OIA)

The operator information area of a host session is used to provide status information regarding the state of the host session and location of the cursor.

The OIA object can be obtained using the OIA property on an instance of the Screen object:

```
OhioOIA hioOIA = hioScreen.OIA;
```

An example to make sure the session is in a ready state (not in an input inhibited state) to send the ENTER key:

```
if (hioOIA.InputInhibited == OHIO_INPUTINHIBITED.OHIO_INPUTINHIBITED_NOTINHIBITED)  
    hioScreen.SendAid(OHIO_KEY.OHIO_KEY_ENTER);
```

### Events

If you prefer to design your application in an event-driven model, PASSPORT HIO provides the following events that your application can listen for:

- **OnSessionChanged** - This event is generated when the session state changes. The session state can be either connected or disconnected. It can be used to tell when the connection process is completed after issuing the **Session.Connect** and also after the mainframe disconnects you (when you log off from the host or when you are inactive for a certain period of time)
- **OnScreenChanged** - This event is generated when the virtual screen is modified whether it is from the host or from the user inputting text on the host screen.

For example, when the **Screen.SendKeys** (simulating a keystroke) is called, it will trigger this event.

- **OnOIAChanged** - This event is generated when anything on the Operator Information Area (OIA) changes.

These events can be subscribed by:

```
hioSession.OnSessionChanged += new
_IOhioSessionEvents_OnSessionChangedEventHandler(OnSessionChangedHandler);

hioSession.OnScreenChanged += new
_IOhioSessionEvents_OnScreenChangedEventHandler(OnScreenChangedHandler);

hioSession.OnOIAClanged += new
_IOhioSessionEvents_OnOIAClangedEventHandler(OnOIAClangedHandler);
```

### Screen Scraping - A Sequential Approach

Navigating through your 3270 or 5250 host application and collecting data from it can be done with a straightforward sequential program without the use of events. It may not be as efficient as the event driven approach but it is much easier to program and understand. Take a look at the following example to understand the structure of writing one (the wrapper functions WaitStrXY, WaitCursorXY and SendText are not shown.)

```
try
{
    hioSessMgr = new PASSHIOLib.OhioSessions();
    hioSession = hioSessMgr.AddSession(@"C:\Program Files\PASSPORT Host Integration Objects\locis.zcc", 1);

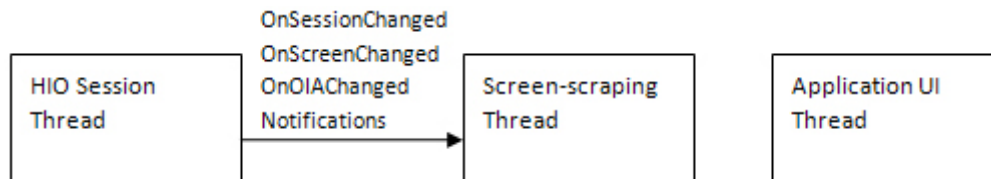
    // wait for connection to the host to be established

    while (hioSession.isConnected() == 0)
        System.Threading.Thread.Sleep(500);

    WaitStrXY("Choose", 16, 7);
    WaitCursorXY(24, 2);
    SendText("tso");
    WaitStrXY("USERID", 1, 18);
    WaitCursorXY(2, 1);
    SendText("zephyr"); // send login ID
    WaitStrXY("Password", 8, 5);
    WaitCursorXY(8, 20);
    SendText("1234"); // send login password
    WaitStrXY("****", 24, 2);
    WaitCursorXY(24, 6);
    :
    :
}
catch (Exception ex)
{
    Debug.Print("Exception: {0}", ex);
}
```

## Screen Scraping - An Event Driven State Machine

An event driven approach is more efficient but it involves the use of threads which is more difficult to program and debug. PASSPORT HIO maintains an internal pool of threads to handle all the host communications and decoding. Each session has a dedicated thread to handle host communications and decodings.



A separate thread for handling the notifications is highly recommended because you do not want to tie up the PASSPORT HIO session thread from doing its job (namely handling host communications and decodings). In the following example, `AutoResetEvent` is used to wake up the waiting thread running the `ScreenScrapeThread` function when a new host screen is ready to be processed.

```
public void ScreenScrapeThread()
{
    // wait until the screen is ready to be processed
    while (eventScreenReady.WaitOne(15000, false))
    {
        try {
            if (!ScreenScrape())
            {
                Debug.WriteLine("Completed");
                return;
            }
        }
        catch (Exception ex)
        {
            Debug.Print("ScreenScrapeThread Exception: {0}", ex);
        }
    }

    // timeout condition, perhaps set a larger timeout value?
    Debug.Print("Timed-out");
}

public void OnScreenChanged(PASSHIOLib.OHIO_UPDATE iUpdate, int istance, int iEnd)
{
    // host updated the screen
    if (iUpdate == OHIO_UPDATE.OHIO_UPDATE_HOST)
        OnOIACChanged();
}
```

```
public void OnOIChanged()
{
    if (hioScreen.OIA.InputInhibited == 0)
        eventScreenReady.Set();
}

public bool ScreenScrape()
{
    try
    {
        switch (nScrState)
        {
            case STATE.LOGIN:
                if (IsCursorXY(2, 1) && IsTextXY(1, 12, "ENTER USERID"))
                {
                    SendTexts("p390r");
                    nScrState++;
                }

                break;

            case STATE.MAINMENU:
                if (IsCursorXY(22, 14) && IsTextXY(3, 29, "ISPF Primary Option Menu"))
                {
                    SendAID (OHIO_KEY.OHIO_KEY_PF3);
                    nScrState++;
                }

                break;
                :
                :

            case STATE.FINISHED:
                if (IsCursorXY(24, 2) && IsTextXY(16, 7, "Choose from the following commands:"))
                {
                    return false;
                }

                break;

        }

    }

    catch (Exception ex)

    {
        Debug.Print("ScreenScrape Exception: {0}", ex);
    }

    return true;
}
```

### How to Enable a Secure SSL Session

To enable a secure SSL connection between the 3270 or 5250 mainframe host and your PASSPORT HIO application, your host must be configured properly for a SSL connection. Usually a different port is used for the SSL connection. You must also enable SSL by setting Security=SSL in your .ZCC file. For example:

```
[Connection]
PHostName=127.0.0.1
TCPPort=9923
EmulType=TN3270
TimeOut=30
TN3270ESupport=Yes
ConnectMethod=Generic
ResourceName=
ScreenSize=2
ExtendedAttributes=Yes
AutoReconnect=Yes
HostCodePage=037
Security=SSL
```

### How to Use International Host Code Page

Note: this option is only available if you purchased the international version of PASSPORT HIO. You can configure the appropriate host code page for your country, or the country in which the host mainframe is located. This determines which translation tables are used to convert between the host's EBCDIC character set and the PC's Windows ANSI character set. For example, to specify the host code page for Italy:

```
[Connection]
PHostName=127.0.0.1
TCPPort=23
EmulType=TN3270
TimeOut=30
TN3270ESupport=Yes
ConnectMethod=Generic
ResourceName=
ScreenSize=2
ExtendedAttributes=Yes
AutoReconnect=Yes
HostCodePage=1144
```

### How to Trace What Your Application is Doing

The PASSPORT HIO trace feature is designed to assist in the diagnosis of problems that can occur during communications and terminal emulation. Information captured in a trace file can be analyzed by the Zephyr technical support and programming staff to assist with problem resolution.

PASSPORT HIO supports the following types of tracing:

Data Stream Trace: If enabled, this trace captures the TN3270 or TN5250 data stream that is transmitted

between the host and PASSPORT HIO. This type of trace is very helpful in resolving problems involving the SNA BIND attributes, as well as other problems involving the host data stream.

**Low Level Trace:** If enabled, this trace captures the lowest level of communication between the PASSPORT HIO communications driver and the underlying communications subsystem. This type of trace is very helpful in resolving problems involving the establishment of a connection.

These traces are enabled by calling the EnableTrace method of the Session object. The trace files that are generated are stored in the folder where the .ZCC is stored. For example:

```
hioSessMgr = new PASSHIOLib.OhioSessions();
hioSession = hioSessMgr.AddSession(@"C:\Program Files\PASSPORT Host Integration Objects\locis.zcc", 1);
hioSession.EnableTrace(1, 1);
hioSession.Connect();
```

Will produce the following trace files (the number 1567 is the session ID so you will most likely have a different number):

```
C:\Program Files\PASSPORT Host Integration Objects\locis__1567_ds.txt
C:\Program Files\PASSPORT Host Integration Objects\locis__1567_ll.txt
```

You may also implement your own methods of tracing. For example, you may dump the entire host screen to a file before sending an AID key by using the Screen.String property.

### How to Deploy Your Application

In addition to your own application executable and any other files it uses, you will need to deploy the following PASSPORT HIO files:

- PASSHIO.dll (this file needs to be registered)
- PassHIO.key (license key file)
- YourConfig.zcc (the .ZCC file you specify in the AddSession method)
- Host?????.tab (translation tables for the host code page you use)

**Note:** You need to deploy all translation tables being used by your application. For example, if your application only uses host code page 1140 (US/Canada Euro), you only need to deploy the Host1140.tab file. If you are unsure of which host code pages/translation tables your application uses, Zephyr recommends deploying all .TAB files with your application. If your application uses the PASSPORT HIO Terminal Control object, which we do not cover in this white paper, you will need to deploy additional files. Please refer to the "Files to Include when Deploying Client Application" knowledge base article for more details.

### Conclusion

With the PASSPORT Host Integration Objects core set of classes, methods, properties and events, software developers using Microsoft development tools and platforms can easily develop and deploy client or server based integration software providing access to 3270 and 5250 legacy applications running on IBM mainframe systems.

## Connecting to Legacy Applications is our Business

Zephyr specializes in TN3270E, TN5250E and UNIX connectivity and legacy integration solutions, whether from a Windows 7 or XP desktop, Windows 2008 Server, a standard web server or virtual server. If you seek reliable, industry-standard host access solutions for Microsoft Windows, Zephyr is the number one supplier to consider.

Founded in 1985, Zephyr is an employee owned company that is debt-free and consistently profitable. The large majority of our revenue recurs annually and the company maintains an almost 100% renewal rate of its products and services. Zephyr maintains offices and distributors in the U.S., UK, Germany, South Africa and throughout Latin America.

## Our Client List is Impressive

Our client portfolio includes large banks such as Bank of America Merrill Lynch, Comerica, U.S. Bank and Wachovia, as well as large insurance companies such as Liberty Mutual, Nationwide Insurance and Progressive.



ZEPHYR



### CORPORATE

3355 West Alabama  
Suite 1220  
Houston Texas 77098  
USA

800.966.3270  
tel 713.623.0089  
fax 713.623.0091

### EUROPE

71 High Street  
Harrold, Bedfordshire  
MK43 7BJ UK

tel 44 (0) 1234 721755  
fax 44 (0) 1234 420317

Zephyr, PASSPORT and PASSPORT Host Integration Objects are trade marks of Zephyr Development Corporation. All other trademarks and trade names are the property of their respective owners.

Zephyr excels at connecting Microsoft Windows desktops and servers to IBM mainframe, AS/400 and UNIX host applications. Whether using our PASSPORT terminal emulation for Windows 7 or the PASSPORT host integration client or server for legacy integration, Zephyr helps organizations lower host access costs through secure, standard TN3270E, TN5250E, VT, SCO ANSI, Wyse 60 and FTP solutions.

Our impressive client list features many notable U.S. and international organizations, including Banco BPI (Portugal), Bank of America Merrill Lynch, Comerica Bank, First National Bank (South Africa), Huntington Bank, Landesbank Baden-Württemberg (Germany), Liberty Mutual, Nationwide Insurance, Progressive Insurance, Otis Elevator, Saks Fifth Avenue, State of California, Wachovia and Wyndham Vacation Ownership.

WEB [zephyrcorp.com](http://zephyrcorp.com)  
E-MAIL [info@zephyrcorp.com](mailto:info@zephyrcorp.com)