Windows xp

Microsoft Virtual PC

Developing and Debugging Software Inside of Virtual Machines

White Paper

By Ronald Beekelaar and John Paul Cook

Published: March 2006

For the latest information, please see http://www.microsoft.com/virtualpc

Acknowledgements

Ben Armstrong, Microsoft Corporation
Matthew Lapsen, Microsoft Corporation
Scott Woodgate, Microsoft Corporation
Carolyn Mader, Studio B Productions
Lisa Pere, Studio B Productions
David Talbott, Studio B Productions

Contents

Introduction

Microsoft® Virtual PC offers many features to facilitate the software development process. One of the classic problems in software development is that an application works on the developer's or tester's machine but doesn't work in the field. In the course of development, a developer's machine is likely to become altered in such a manner that the system is no longer a valid environment for testing. This can happen for a variety of reasons, including elevated security permissions and versions of DLLs and software that are not present on user machines.

By using Virtual PC to create a valid representation of an end-user machine, developers can test application changes on virtual end-user machines instead of on the developer's physical machine. This helps to mitigate the "works on my machine" syndrome, thus reducing overall development and test cycle times. Fixing a bug before it goes to the testing group is much faster than fixing it after testers detect it.

Sometimes a developer can't properly test a change because a complete testing environment can be more than just a developer-class workstation with Microsoft® Visual Studio® installed. Proper testing might require changes to middleware code and end-user code, which require at least two machines. You can use Virtual PC to create test environments that are both valid and complete.

In this white paper, you will learn how to create, copy, and configure virtual machines. Creating a virtual machine by copying files, sometimes referred to as *cloning* a virtual machine, can introduce problems. This white paper explains those problems and provides detailed workarounds.

Networking topologies for multi-machine testing scenarios can be complicated. You will learn how to configure a virtual network so that you can accurately recreate your physical network in a virtual environment.

Because a physical environment changes over time, a virtual environment must stay synchronized with the physical environment if it is to remain valid. This paper presents options for maintaining virtual machines so that you can properly plan for and accommodate changes. You will also learn options for restoring virtual machines to a pre-change state by using standard features as well as a custom implementation of repeatable saved state.

The final sections of the white paper address the topic of remote debugging. You will see step-by-step instructions for configuring both kernel debugging and application debugging.

The scope of this white paper is limited to Virtual PC 2004. Implementation of certain features in Microsoft® Virtual Server 2005 are slightly different, so if you are using Virtual Server, be sure to consult its product-specific documentation.

Managing multiple virtual machines

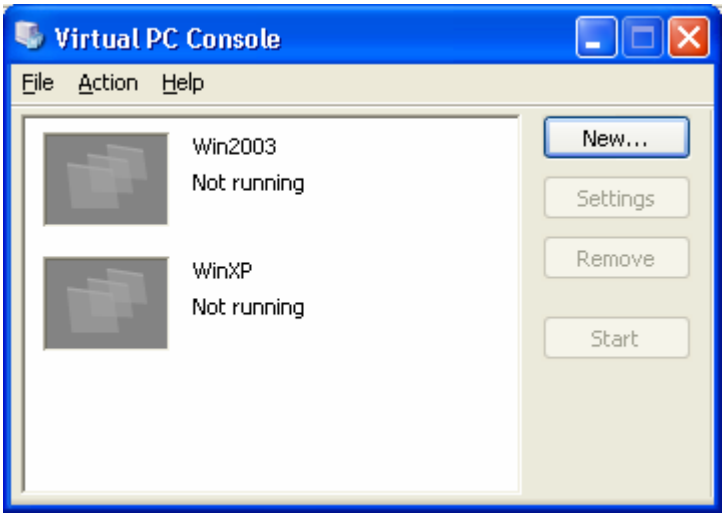
A virtual machine consists of multiple files. To manage multiple virtual machines effectively, you need to understand how these files interact. The table below includes a description of the three types of files that form a virtual machine (named WinXP in this example).

Virtual Machine Files

Sample file name	Description	Comments
WinXP.lnk	Shortcut file for a registered virtual machine	Points to the virtual machine's configuration (.vmc) file
WinXP.vmc	Virtual machine configuration text file (XML format)	Contains virtual machine configuration information, including the name of the virtual machine's .vhd file
Disk1.vhd	Virtual hard disk file	Represents the hard disk and its data on the virtual machine

When you register a virtual machine with Virtual PC, the application creates a shortcut file in the \Documents and Settings\<username>\Application Data\Microsoft\Virtual PC\Virtual Machines folder. This folder path is different for each user. The shortcut points to the virtual machine's configuration (.vmc) file.

All shortcuts appear in the Virtual PC Console as registered virtual machines. The name of a virtual machine in the Virtual PC Console is the same as the name of that machine's .vmc file.



Virtual PC Console listing two registered virtual machines

The .vmc file contains the virtual machine's configuration settings, such as the amount of RAM, the number of network adapters, and the full path of the virtual hard disk (.vhd) file.

The .vhd file represents the hard disk and its data on the virtual machine.

By copying these files and making a few changes, you can create new virtual machines very quickly.

Creating virtual machines

A common method for creating a virtual machine is to launch the New Virtual Machine Wizard from the Virtual PC Console, use the wizard to create an empty virtual machine, and then use one or more CDs or ISO-9660 CD image files images to install the operating system and needed applications on the newly created virtual machine.

Using the New Virtual Machine Wizard to create a virtual machine

To create a virtual machine by using the New Virtual Machine Wizard:

1. In the **Virtual PC Console**, click **New**.
2. On the **Welcome to the New Virtual Machine Wizard** page, click **Next**.
3. On the **Options** page, select **Use default settings to create a virtual machine**, and then click **Next**.

   Or, you can select **Create a virtual machine** to let the wizard guide you through the basic configuration of the new virtual machine.

   In either case, you can change all configuration settings later by using the **Settings** dialog box.

   Note: When you select **Create a virtual machine**, the wizard asks you to select the name of the operating system you plan to install on the new virtual machine. The choice of the selected operating system is not stored with the virtual machine. The wizard uses the information only to recommend appropriate default values for memory and hard disk sizes. You can change those values later.
4. On the **Virtual Machine Name and Location** page, type a name for the .vmc file, and then click **Next**.

   If you do not specify a path, the wizard creates the .vmc file in the \Documents and Settings\<username>\My Documents\My Virtual Machines folder. You can change this default location by creating a **MYVIRTUALMACHINES** environment variable to specify another location.

   Note: All shortcut files for registered virtual machines will be in the same \Documents and Settings\<username>\Application Data\Microsoft\Virtual PC\Virtual Machines folder as and will use the same file name as the .vmc file. Therefore, all .vmc files, even those stored in multiple folders, must have different names.
5. On the **Completing the New Virtual Machine Wizard** page, click **Finish**.

The Settings dialog box for the new virtual machine opens. The virtual machine configuration does not contain an associated .vhd file yet.

To create the .vhd file:

1. In the **Virtual PC Console**, on the **File** menu, click **Virtual Disk Wizard**.

   Or, you can open the **Settings** dialog box for the new virtual machine and click **Virtual Disk Wizard** in the **Hard Disk 1** configuration area.
2. On the **Welcome to the Virtual Disk Wizard** page, click **Next**.
3. On the **Disk Options** page, select **Create a new virtual disk**, and then click **Next**.
4. On the **Virtual Disk Type** page, select **A virtual hard disk**, and then click **Next**.

5. On the **Virtual Hard Disk Location** page, type the name and location for the .vhd file, and then click **Next**.

   The default location for the .vhd file is the \Documents and Settings\<username>\My Documents folder. For easier management of the virtual machine files, place the .vhd file in the same folder as the .vmc file that you created earlier.

6. On the **Virtual Hard Disk Options** page, select **Dynamically expanding**, and then click **Next**.

   When using a dynamically expanding .vhd file, Virtual PC stores all hard disk data for the virtual machine in a single file that increases in size until it reaches a configured maximum. This option maximizes portability of virtual machines from one host computer to another.

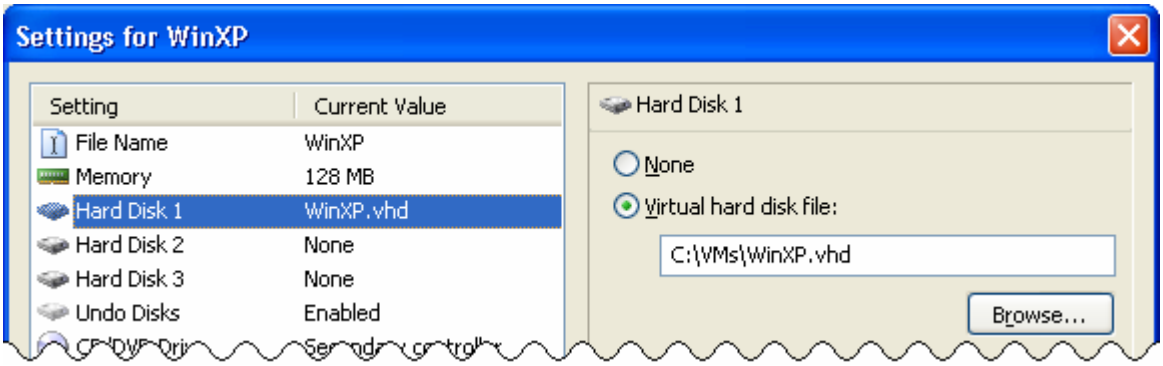
7. On the **Virtual Hard Disk Size** page, type **8000**, and then click **Next**.

   Note: This setting specifies the maximum size in megabytes (capacity) of the virtual hard disk. You cannot easily extend the maximum size later. The actual size of a dynamically expanding .vhd file on the host computer is dependent only on the amount of data on the virtual hard disk. You should select a large enough maximum size (8000 MB or larger) so that you do not quickly run out of hard disk space on the virtual machine.

8. On the **Completing the Virtual Disk Wizard** page, click **Finish**.

To associate the new, empty .vhd file with the new virtual machine:

1. In the **Settings** dialog box, in the **Hard Disk 1** area, select **Virtual hard disk file**, and then click **Browse**.
2. In the **Select Virtual Hard Disk** dialog box, select the new .vhd file, and then click **Open**.
3. Click **OK** to close the **Settings** dialog box.



Associating a virtual hard disk with a virtual machine

After you create the virtual machine and create and associate the virtual hard disk, you can use the virtual machine in Virtual PC.

The very first thing you need to do with the new virtual machine is start the virtual machine and install the operating system.

Methods for registering virtual machines

When you use the New Virtual Machine Wizard to create a virtual machine, the machine is automatically registered with Virtual PC. However, if you copy the virtual machine to another computer or if you want another user on the same computer to use the virtual machine, then you need to register the virtual machine with Virtual PC as a separate step. Virtual PC can start a virtual machine only after it is registered.

To register a virtual machine, you can use one of four methods:

- **Use the New Virtual Machine Wizard.** You can use the New Virtual Machine Wizard to register a virtual machine as well as to create a virtual machine. In the **Virtual PC Console**, click **New** to start the wizard. On the **Options** page, select **Add an existing virtual machine** and specify the virtual machine's .vmc file.
- **Use the Open command.** Right-click an unregistered .vmc file, and then click **Open** to register and immediately start the virtual machine. This action has the same effect as double-clicking the .vmc file.
- **Use Virtual PC.exe.** Open a Command Prompt window and navigate to the **\Program Files\Microsoft Virtual PC** folder, then use the **Virtual PC.exe -registervm *filename.vmc*** command to register the virtual machine. See the **Command-line reference** topic in the Virtual PC product Help for a list of all the command-line parameters.
- **Create a shortcut to the .vmc file.** When the Virtual PC application is not running, you can create a shortcut to the .vmc file, and then place that shortcut in the **\Documents and Settings\<username>\Application Data\Microsoft\Virtual PC\Virtual Machines** folder. When Virtual PC starts, it enumerates all the shortcuts in this folder to find out which virtual machines are registered. This method does not work if the Virtual PC application is already running.

You can also use the last two methods to automate registration of virtual machines with Virtual PC.

Right-clicking a .vmc file and then clicking **Open** (as in the second method above) registers the virtual machine with Virtual PC (if the machine is not already registered) and launches the virtual machine. To make it easy to register a virtual machine without immediately launching it, you can change the .vmc file context menu to include the **Register** command, which will execute the **Virtual PC.exe -registervm *filename.vmc*** command.

To make it easy to open .vmc files in Notepad, you can also add the **Edit** command, which will execute the **Notepad.exe *filename.vmc*** command.

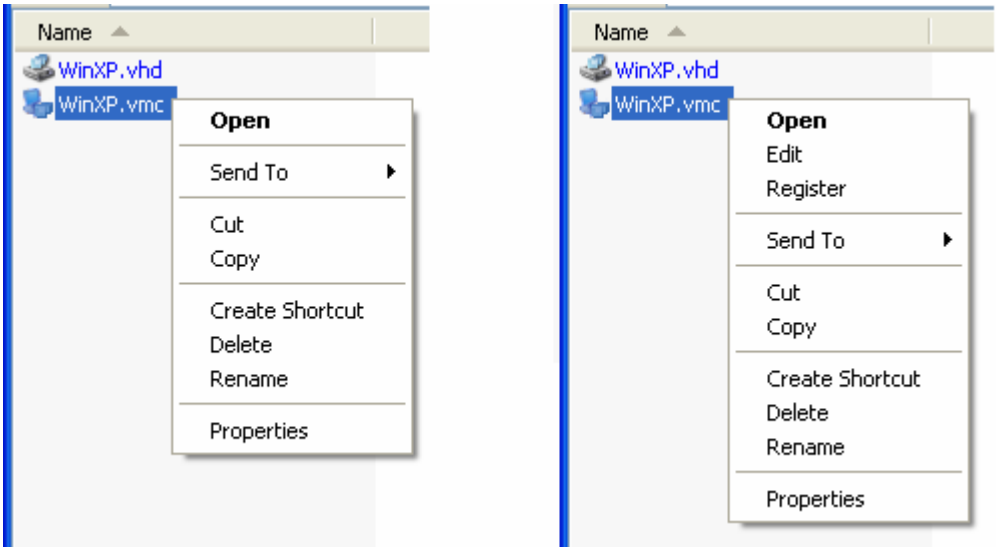
To add these two commands to the context menu:

1. Create a text file, named **vmcverbs.inf**, that contains the following text:

```
[Version]
Signature="$Chicago$"
[DefaultInstall.nt]
AddReg = Prog.reg.install
[Prog.reg.install]
HKCR,Virtual.Machine.VMC\shell\Edit\command, ,0x20000, ↵
  "%%SystemRoot%%\system32\Notepad.exe ""%%1"""
HKCR,Virtual.Machine.VMC\shell\Register\command, ,0x20000, ↵
  "%%ProgramFiles%%\Microsoft Virtual PC\Virtual PC.exe ↵
  -registervm ""%%1"""
```

Note: The `[Prog.reg.install]` section in this file contains two long lines that are wrapped in the text above. The `0x20000` parameter in the file indicates that the type of the new value in the registry is expandable string (**reg_expand_sz**).

2. Right-click the **vmcverbs.inf** file, and then click **Install**.



Edited context menu for .vmc files

Using existing virtual machine files to create more virtual machines

Once you have an existing virtual machine, you have two more options for creating virtual machines:

- **Use differencing disks.** You can create new virtual machines that use differencing disks by pointing to an existing .vhd file (parent disk) that contains an installed operating system and applications.
- **Copy virtual machines files.** You can create new virtual machines by copying the virtual machine files of the existing virtual machine.

The big advantage of creating new virtual machines based on an existing .vhd file is the speed with which you can create those new virtual machines. You do not have to wait for the completion of the installation of the operating system and applications. This method is especially useful in testing scenarios, when you want to create the new virtual machines only to run a test, and then discard those virtual machines.

Creating, installing and configuring a new virtual machine by using the New Virtual Machine Wizard can take an hour or more. Creating virtual machines by using either of the other methods can take a few minutes or less.

However, if you run two or more virtual machines that are based on the same .vhd file and you want to use those virtual machines in the same network, you might have to change some elements of the operating system on those machines to ensure that the machines are unique. These elements can include the computer name, computer security ID (SID), IP addresses, and in some cases the Media Access Control (MAC) address on the virtual network adapters. How to change the configuration of the virtual machines to make them unique is discussed later in this section.

Note: In all cases, independent of the method you use to create virtual machines, you need to have enough software licenses to cover the use of the virtual machines in accordance with the license terms of the software running on the virtual machines.

Using differencing disks to create virtual machines

A *differencing disk* is a .vhd file that is based on another .vhd file, called the *parent* .vhd file. You can use differencing disks to create one or more virtual machines, based on a parent file that already contains an installed operating system and applications. The differencing .vhd file contains only the changes (differences) from the parent file.

Scenario: Suppose you have a virtual machine named WinXP, which uses the .vhd file C:\VMs\WinXP.vhd. This file contains a fully installed copy of Microsoft® Windows® XP Service Pack 2 (SP2). You want to use both Microsoft® Internet Explorer 6 and Internet Explorer 7 to test a new Web application. You can do this by creating two virtual machines that use differencing disks and the parent file WinXP.vhd.

To create the new virtual machines:

1. Make WinXP.vhd read-only.

   Because the differencing disks contain only the changes from the parent file, you do not want any further changes to occur to the data on the parent .vhd file after you have created the differencing disks. One way to ensure that WinXP.vhd remains unchanged is to make it read-only.

2. Create two differencing disks: WinXP-IE6.vhd and WinXP-IE7.vhd.

3. Create and register two new .vmc files: WinXP-IE6.vmc and WinXP-IE7.vmc.

   You can create new .vmc files by using the **New Virtual Machine Wizard**, or you can copy an existing .vmc file, register the new .vmc files, and then update the associated .vhd files in the **Settings** dialog box of the new virtual machines. After you have created the new virtual machines, you can change the configuration for each virtual machine.

4. Install Internet Explorer 7 on the WinXP-IE7 virtual machine.

To create the differencing disk WinXP-IE6.vhd:

1. In the **Virtual PC Console**, on the **File** menu, click **Virtual Disk Wizard**.
2. On the **Welcome to the Virtual Disk Wizard** page, click **Next**.
3. On the **Disk Options** page, select **Create a new virtual disk**, and then click **Next**.
4. On the **Virtual Disk Type** page, select **A virtual hard disk**, and then click **Next**.
5. On the **Virtual Hard Disk Location** page, type **C:\VMs\WinXP-IE6.vhd**, and then click **Next**.

   The differencing .vhd file stores both a relative path and an absolute path to the parent .vhd file. When you place the differencing .vhd file in the same folder as the parent .vhd file, then the relative path is ".\". This path maximizes the portability of virtual machines from one host computer to another host computer, as long as you keep the two .vhd files in the same folder.

6. On the **Virtual Hard Disk Options** page, select **Differencing**, and then click **Next**.
7. On the **Differencing Virtual Hard Disk** page, type **C:\VMs\WinXP.vhd**, and then click **Next**.
8. On the **Completing the Virtual Disk Wizard** page, click **Finish**.

Creating a differencing disk

Use similar steps to create the WinXP-IE7.vhd differencing disk.

Notice that you do not specify the size of the differencing disk. Differencing disks have the same maximum size as the parent disk.

Initially, the new differencing .vhd files are very small — less than 100 KB. When you run the WinXP-IE6 and WinXP-IE7 virtual machines and make changes to the data on the hard disks, the size of the differencing .vhd files on the host computer increases. The parent .vhd file (WinXP.vhd) does not change.

Note: If you want to run the WinXP-IE6 and WinXP-IE7 virtual machines at the same time, you need to ensure that the machines use different computer names, different IP addresses, and possibly different computer SIDs. This process is discussed later in this section.

Copying files to create virtual machines

Because a complete virtual machine consists of only a few files, you can quickly create new virtual machines by copying those files.

The two main differences between creating virtual machines by using differencing disks and creating virtual machines by copying disk files are as follows:

- When using differencing disks, you should never change the parent .vhd file. When copying disks, you can change the original .vhd file without any consequences to the new virtual machines.
- When using differencing disks, you can move the new virtual machine only if you move the original .vhd file as well. A copied virtual machine does not use the original .vhd file, so you can move the copied files to other host computers, without moving the original .vhd file.

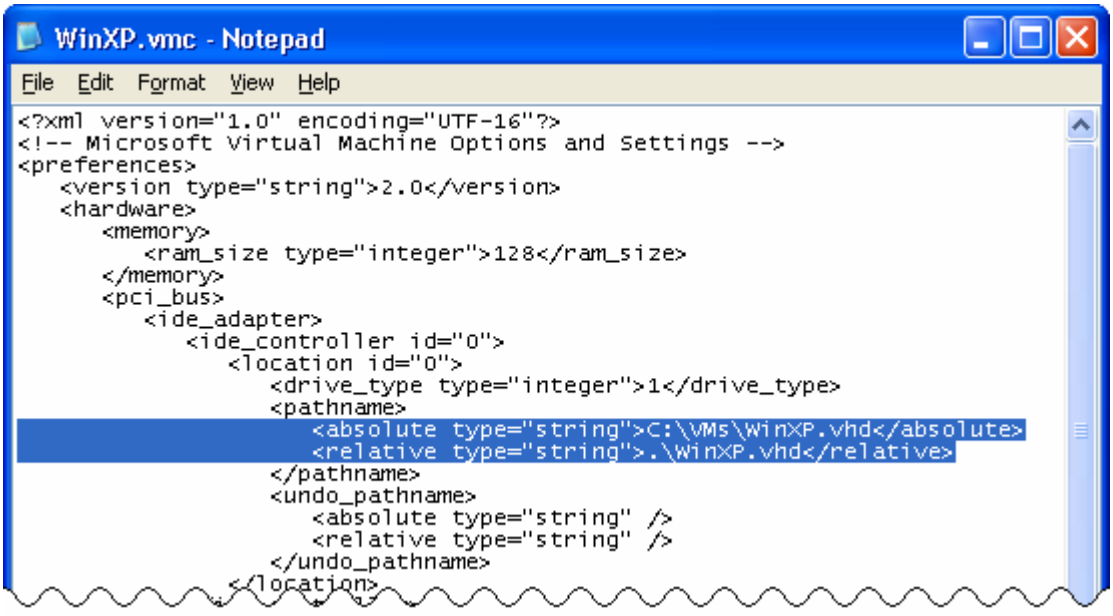
Scenario: Suppose you have a virtual machine, WinXP, that uses the .vhd file C:\VMs\WinXP.vhd. You want to use a copy of the WinXP virtual machine to test the installation steps of a newly developed application. .

To create a new virtual machine by copying files:

1. Copy the original .vhd file and give the copy a new name.
2. Create and register a new .vmc file that uses the copied .vhd file.

To create the new .vmc file, you can use the **New Virtual Machine Wizard** or you can copy an existing .vmc file, register that file, and then update the associated .vhd file in the **Settings** dialog box of the new virtual machine.

If you copy an existing .vmc file, do not forget to update the associated .vhd file in the new .vmc file. The .vmc file contains both a relative path and an absolute path to the associated .vhd file.



Relative and absolute .vhd file paths in a .vmc file

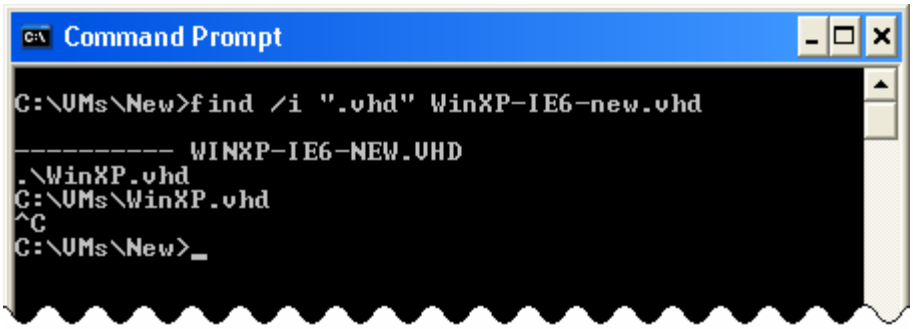
Virtual PC checks the relative path first, and then checks the absolute path, according to the following mechanism:

- **Relative path found.** If you copy a .vhd file and a .vmc file to another folder, rename just the .vmc file, and then register the .vmc file, then the new virtual machine starts correctly, using the relative path to the copied .vhd file. Virtual PC automatically updates the absolute path in the .vmc file.
- **Relative path not found, absolute path found.** If you copy a .vhd file and a .vmc file to new folder, rename both files, and then register the new .vmc file, then Virtual PC cannot find the renamed .vhd file by using the relative path. However, the virtual machine starts by using the absolute path to the original .vhd file. Virtual PC automatically updates the relative path in the .vmc file to point to the original .vhd file as well. The copied .vhd file is not used!
- **Relative and absolute path not found.** If Virtual PC cannot find the .vhd file by using the relative path or the absolute path, then it displays an error message, and the virtual machine cannot start.

If you use differencing disks, you can create a new virtual machine by copying just the differencing .vhd file or by copying both the differencing .vhd file and the parent .vhd file. (In either case, you need to create and register a new .vmc file for the new virtual machine.)

A differencing .vhd file stores both a relative path and an absolute to its parent .vhd file. As when dealing with .vhd file information in a .vmc file, Virtual PC checks the relative path first, and then checks the absolute path. However, Virtual PC does not have an easy way to find this information in a differencing vhd-file. If you want to verify whether a copied differencing .vhd file uses the intended parent .vhd file, you can use the Windows **find.exe** command-line tool to search for the text "**.vhd**" in the differencing .vhd file.

Differencing .vhd files can be very large (for example, several gigabytes). Because the relative and absolute parent .vhd file information is stored in the first 50 KB of the file, you can press CTRL+C immediately after the **find /i** command displays the found paths, to avoid having to search the entire file.



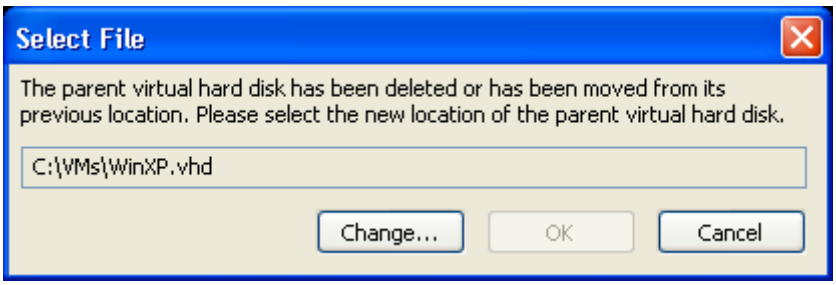
Relative and absolute parent .vhd file path in differencing .vhd file

To find the parent .vhd file referenced in a differencing .vhd file, Virtual PC uses the following mechanism:

- **Relative path found.** If you copy a differencing .vhd file and the parent .vhd file from one folder to another, then Virtual PC uses the relative path to find the copied parent .vhd file, and the new virtual machine starts correctly. Note that the absolute path to the original parent .vhd file remains in the differencing .vhd file; Virtual PC does not update that path.
- **Relative path not found, absolute path found.** If you copy just a differencing .vhd file to another folder, then Virtual PC cannot find the parent .vhd file by using the relative path.

However, Virtual PC can find the original parent .vhd file if the absolute path is correct, and the new virtual machine starts correctly. Virtual PC does not update the relative path in the differencing .vhd file.

- **Relative and absolute path not found.** If Virtual PC cannot find the parent .vhd file by using the relative path or the absolute path, the application asks you to select the new location of the parent .vhd file when the virtual machine starts. After you select the parent .vhd file, Virtual PC updates both the relative path and the absolute path in the differencing .vhd file—even if the virtual machine is configured to use undo disks.



Parent .vhd file is not found

Typically, Virtual PC updates neither the relative path nor the absolute path of the parent .vhd file if either the relative path or the absolute path is found in the differencing .vhd file. If you create a copy of a virtual machine by copying differencing disks and you want to update the relative path and absolute path in a differencing .vhd file, perform the following steps:

1. Rename the copied parent .vhd file and the original parent .vhd file to a temporary name.
2. Enable undo disks (see Restoring virtual machines to their pre-change state for information about enabling undo disks) and start the virtual machine. Virtual PC cannot find the parent .vhd file and prompts you to select the new location of the parent .vhd file. Select the renamed parent .vhd file. Virtual PC updates the relative path and the absolute path to use the temporarily renamed parent .vhd file
3. Close the virtual machine; do not save the undo changes.
4. Rename the copied parent .vhd file and the original parent .vhd file to the intended file name.
5. Start the virtual machine again. Because the differencing .vhd file was updated, Virtual PC cannot find the temporarily renamed parent .vhd file. Select the new parent .vhd file. Virtual PC updates the relative path and the absolute path to use the intended parent .vhd file.

Making copied virtual machines unique

When you create virtual machines that are based on an existing .vhd file, either by using differencing disks or by copying the .vhd file, you are actually creating clones of the original virtual machine.

Creating multiple virtual machines in this way is a great time saver, but cloned virtual machines might not always work well or might require additional configuration steps.

For example, several issues exist with cloned virtual machines that participate in Microsoft® Active Directory® replication on the network. You cannot successfully clone a virtual machine that is a domain controller (DC); Active Directory replication does not work in that case.

Another issue is caused by shared secrets on the network. When you copy a virtual machine that is a member of a domain, the cloned virtual machines share the same identity and computer account password in the domain. As soon as one of the machines changes the computer account password — something that happens periodically — the other clones can no longer authenticate to the domain. You can prevent this problem by temporarily removing the virtual machine from the domain before cloning it.

A third issue relates to the fact that all configuration settings on the cloned machines are the same as the settings on the original machine. If you intend to run two or more virtual machines that are based on the same .vhd file, this similarity might cause a problem. You can easily solve this issue by making a few configuration changes to the copied virtual machines so that all relevant configuration elements are unique.

The four configuration elements that need to be unique for each virtual machine are as follows:

- **Computer name.** For network communication, the computer name is often used to identify the computer. Each virtual machine that uses the network must have a unique computer name.
- **Computer SID.** The computer SID is used to form all local user and local group SIDs. When two computers have the same computer SID, authentication issues can arise in a domain environment, and unwanted access can occur if you use the NTFS file system (NTFS) to protect removable media.
- **IP address**. Cloned virtual machines might have the same fixed IP address. Virtual machines on the same network each need a unique IP address.
- **Network MAC address.** Each network adapter must have a unique MAC address. For physical computers, the MAC address is tied to a specific physical network adapter, ensuring a unique address. However, for virtual machines, the MAC address for the virtual network adapter is defined in the .vmc file. When you copy a .vmc file, you copy the MAC address as well.

To ensure that these configuration elements are unique, you can either prepare a *template* virtual machine before the copying process, or you can change the configuration elements of each clone after the copying process.

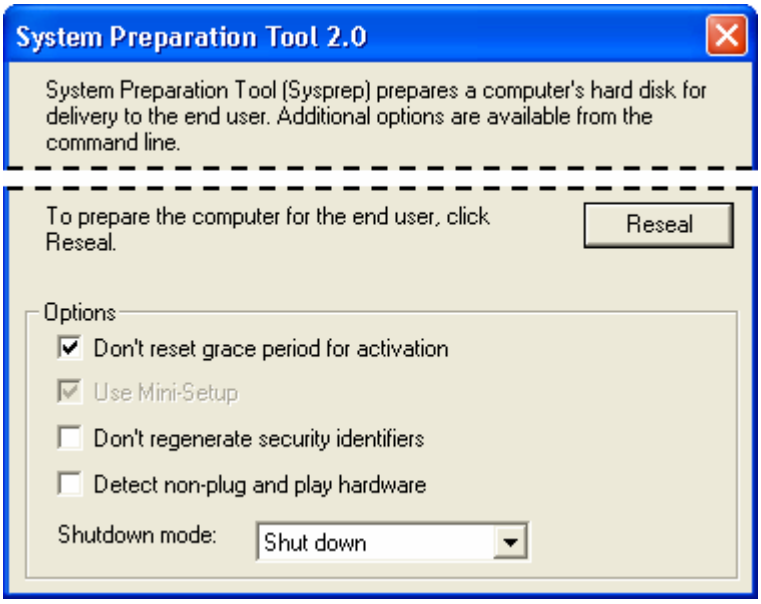
Preparing a template virtual machine by using the System Preparation Tool (Sysprep)

Preparing a virtual machine before copying is called preparing a template machine. One way to prepare a template machine is to run the **Sysprep** tool on the virtual machine. Sysprep is

part of the deployment tools in the **deploy.cab** file on the Windows XP and Microsoft® Windows Server™ 2003 product CD.

Using Sysprep ensures only that the computer name and the computer SID of the copied virtual machines are unique. Sysprep does not deal with the virtual machine IP address or network MAC address.

Running Sysprep on a virtual machine rolls back the last part of the operating system installation, and then shuts down the computer. The next time the computer starts, a new computer SID is created, and you are prompted to supply a new computer name. Sysprep is typically used to deploy duplicated desktop computers to different users in an enterprise environment, but you can just as easily use it to create a template machine that you can use as a basis for multiple virtual machines.



Sysprep in a Windows XP virtual machine

Note that using Sysprep can have other consequences. Depending on the Windows edition used, you might have to provide a Windows Product Key the first time you start the copied virtual machine.

Changing the computer name after copying

If you do not use Sysprep, you need to change the configuration of the copied virtual machine manually the first time you start it.

To change the computer name:

1. Start the copied virtual machine and log on.
2. On the **Start** menu, right-click **My Computer**, and then click **Properties**.
3. In the **System Properties** dialog box, on the **Computer Name** tab, click **Change**.
4. In the **Computer Name Change** dialog box, type a new computer name, and then click **OK**.
5. Click **OK** to acknowledge that you must restart the computer.
6. Click **OK** to close the **System Properties** dialog box.
7. Click **Yes** to confirm that you want to restart the computer now.

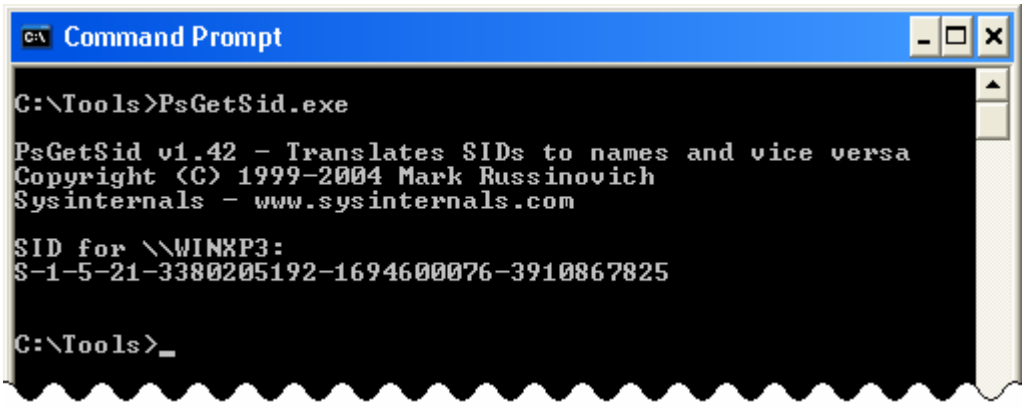
The virtual machine will reboot, using the new computer name.

Changing the computer SID after copying

Microsoft does not provide a tool to change the computer SID on a copied virtual machine manually, without using Sysprep. However, you can find several third-party tools that you can use to change the computer SID.

One well-known application is the **NewSID.exe** tool from Sysinternals. You can download this free tool from the Sysinternals Freeware Web site (http://www.sysinternals.com). When you run NewSID on a virtual machine, it stores a new computer SID in the registry; updates the SIDs on all the access control lists (ACLs) on files, folders, and registry keys; and then reboots the virtual machine.

For reference, you can use the **PsGetSid.exe** tool from Sysinternals to verify the local computer SID on a virtual machine.



PsGetSid from Sysinternals

Note that the local computer SID is not the same as the SID for the computer account in Active Directory, which is assigned when the virtual machine is added to a domain.

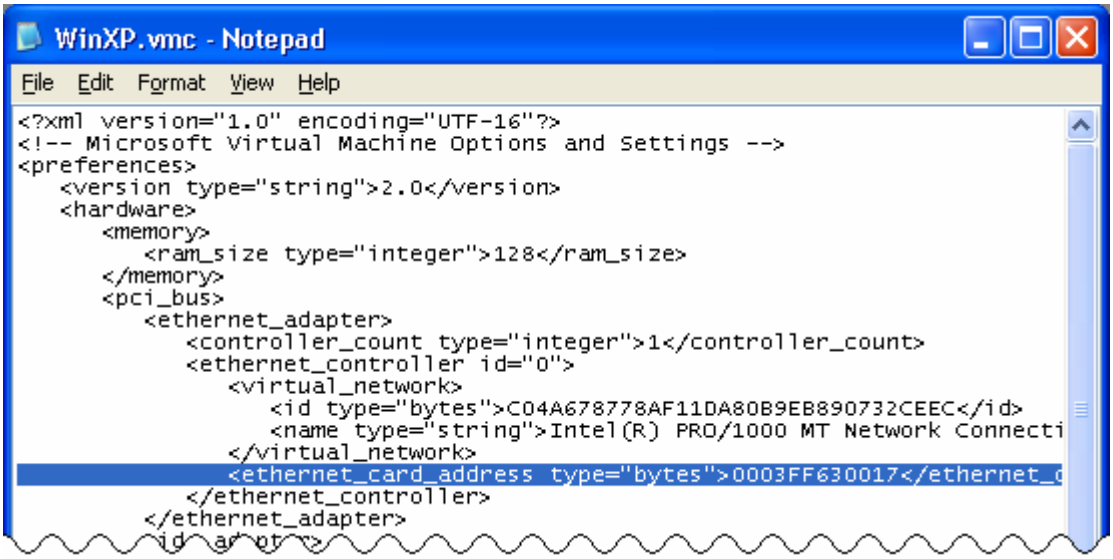
Changing the IP address after copying

Making sure that a copied virtual machine has a unique IP address is probably the easiest problem to solve. You do not even need to reboot the virtual machine.

If you assign fixed IP addresses to the network adapters, you need to change those IP addresses the first time you start the copied virtual machine. If you assign IP addresses dynamically by using Dynamic Host Configuration Protocol (DHCP), then the IP address of the network adapters in the copied virtual machine will automatically be unique.

Changing the network MAC address after copying

Like all of the hardware configuration settings for a virtual machine, the MAC address for a virtual network adapter is defined in the .vmc file. The Virtual PC MAC addresses all start with 00-03-FF, but you can configure any MAC address you want in the .vmc file.



MAC address in .vmc file

When you copy a virtual machine, including the .vmc file, you create multiple virtual machines that have the same MAC addresses. Before Virtual PC 2004 Service Pack 1 (SP1), this situation could lead to confusing networking issues if two or more virtual machines that had the same MAC address assigned to virtual network adapters attempted to communicate on the network.

Virtual PC 2004 SP1 includes several changes to the way MAC addresses are handled. These changes prevent duplicate MAC addresses from occurring on copied virtual machines. When using SP1, creating a situation in which two running virtual machines share the same MAC address on the same subnet is almost impossible. Virtual PC 2004 SP1 handles MAC address in several ways:

- **At registration.** When you register a new virtual machine by using the **New Virtual Machine Wizard**, or by using the **Virtual PC.exe -registervm** command, Virtual PC 2004 SP1 removes from the .vmc file the entire line containing the **<ethernet_card_address>** XML tag.
- **At startup when the MAC address is missing.** When you start the virtual machine, and the .vmc file does not contain a MAC address for the virtual network adapter, Virtual PC 2004 SP1 generates a new MAC address and adds it to the .vmc file.
- **At startup when the MAC address is in use.** If the MAC address that is defined in the .vmc file is already in use by a running virtual machine, then Virtual PC generates a new MAC address and stores it in the .vmc file.

These steps, especially checking whether the MAC address is already in use at startup time, prevent all common duplicate MAC address issues in Virtual PC.

Situations in which you want to keep the original MAC address when registering a new virtual machine do exist. This is the case when the actual MAC address is important for the correct operation of the operating system and applications on the virtual machine. For example, the MAC address of the network adapter is one of the hardware "signatures" used in both Windows Product Activation and Office Product Activation. A changed MAC address can be just enough change to require re-activation of Windows or Office on the virtual machine.

To ensure that the virtual machine is using the original MAC address, you can use one of two methods:

- **Register the virtual machine by creating a shortcut.** When the Virtual PC application is not running, you can register a new virtual machine by creating a shortcut to the .vmc file

and placing the shortcut file in the appropriate Virtual Machines folder in the user's profile. Virtual PC will not remove the MAC address from the .vmc file after registration.

- **Copy the original .vmc file.** After you register a new .vmc file by using the **New Virtual Machine Wizard** or by using the **Virtual PC.exe -registervm** command, overwrite the .vmc file with a copy of the file that still contains the original MAC address. Virtual PC removes the MAC address from the .vmc file only when you register the virtual machine.

Configuring advanced network scenarios

This section describes how to use Virtual PC networking settings to create advanced network scenarios that consist of one or more virtual machines.

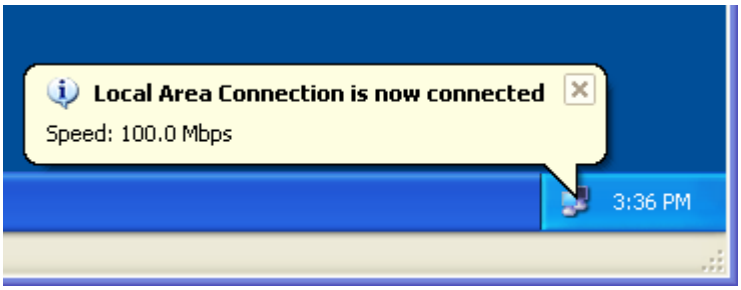
Understanding networking principles

Virtual PC provides extensive networking capabilities to virtual machines. You can configure network connections between multiple virtual machines, between a virtual machine and the host computer, and between a virtual machine and the external network and the Internet.

The network configuration of Virtual PC includes a built-in DHCP service, a built-in Network Address Translation (NAT) service, and an unlimited number of virtual switches to build advanced networking scenarios.

In the **Settings** dialog box of a virtual machine, you can specify as many as four network adapters. When a virtual machine starts, the operating system recognizes each network adapter as an **Intel 21440-Based PCI Fast Ethernet Adapter**.

Note: Physical network adapters are designed for certain maximum speeds. The virtual network adapters used in virtual machines do not have this hardware limitation and can run at any speed. However, the operating system reports the official maximum speed (100 MB per second) related to a physical Intel 21440-Based network adapter. When the processor and the disk components are fast enough, virtual machines might reach a higher network speed.

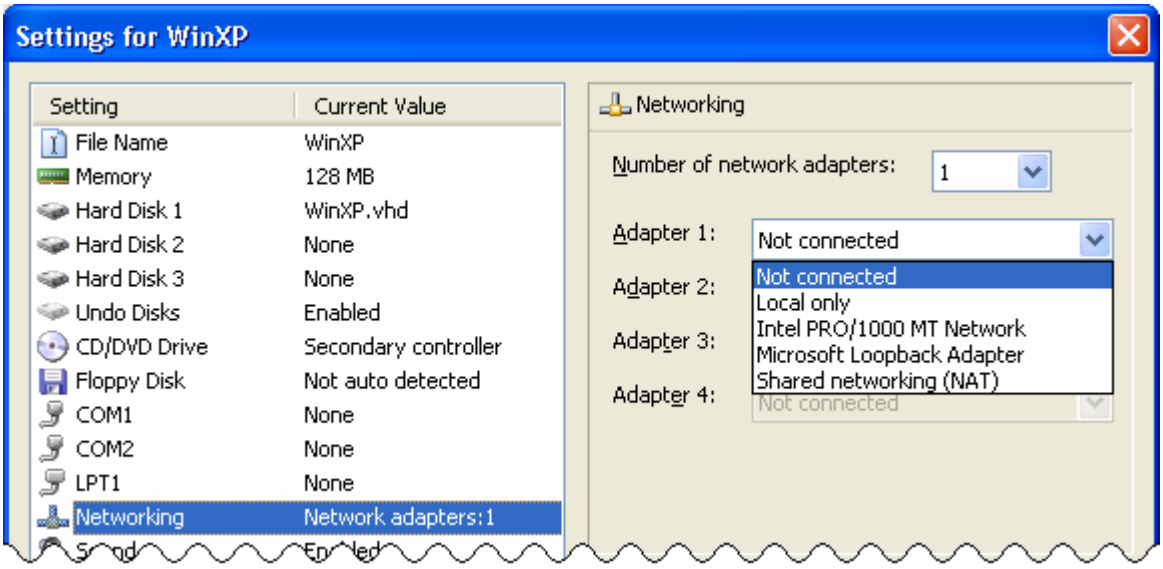


Reported network speed in a virtual machine

In the **Settings** dialog box for a virtual machine, you can configure each of the four network adapters to use a specific networking setting. You can think of the different networking settings as connecting to different virtual switches.

For each adapter, you can select one of five networking settings:

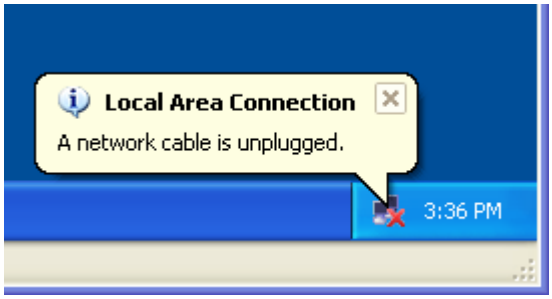
- Not connected
- Local only
- The host network adapter
- Microsoft Loopback Adapter
- Shared networking (NAT)



Networking settings per network adapter

Unlike many other hardware settings in the **Settings** dialog box, you can change each adapter's networking setting while the virtual machine is running (similar to unplugging a network cable from one switch, and then plugging the cable into another switch). The ability to change the networking settings while a virtual machine is running is helpful when you want to disable network connectivity from a virtual machine temporarily, without having to disable the network adapter in the virtual machine.

Network setting: Not connected

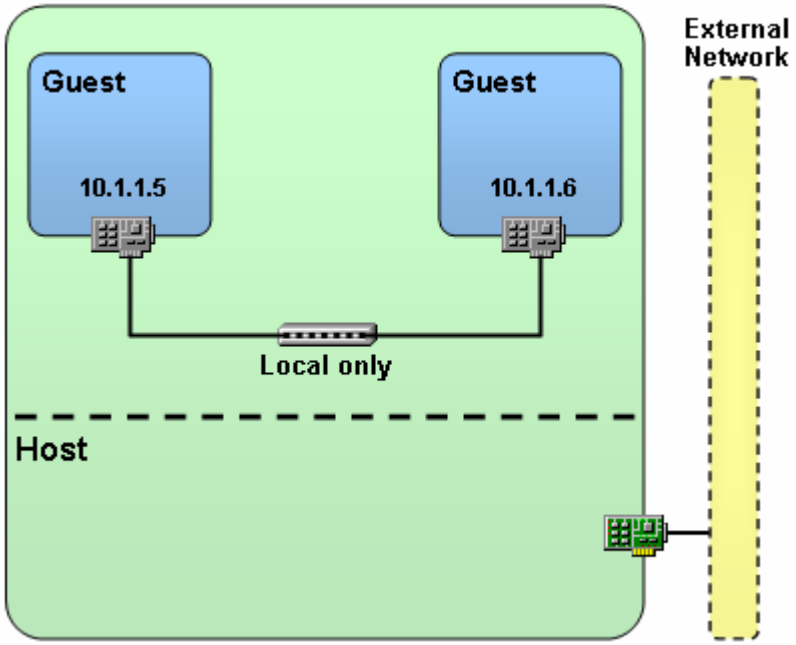


Network setting: Not connected

The **Not connected** networking setting is just what you would expect. The hardware detection process of the virtual machine will detect a network adapter, but the network adapter reports that the cable is unplugged.

If you do not want the virtual machine to have any networking at all, you can also set the **Number of network adapters** in the **Settings** dialog box to **None**. In that case, the virtual machine will not detect a network adapter at all.

Network setting: Local only



Network setting: Local only

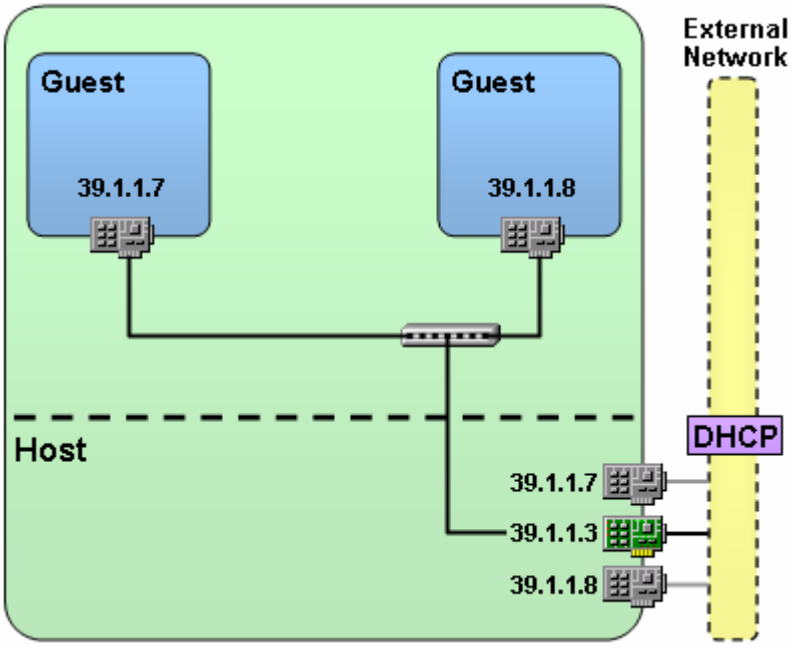
The **Local only** networking setting connects the network adapter in the virtual machine to a virtual switch that connects only to other virtual machine network adapters that use the **Local only** setting. The host computer cannot connect to this virtual switch. This means that virtual machines that use the **Local only** setting can communicate only with other virtual machines, never with the host computer.

This setting is convenient when you copy multiple networked virtual machines to other host computers. The setting does not rely on any specific IP address or network adapter on the host computer.

If the network adapter is configured to obtain a dynamically assigned IP address, then only a DHCP server running on another virtual machine will respond to the adapter's DHCP requests.

Virtual PC has only one **Local only** virtual switch. When you want to create multiple network segments for communication between virtual machines, you need to include the setting to use the host network adapter or the **Microsoft Loopback Adapter** setting.

Network setting: The host network adapter

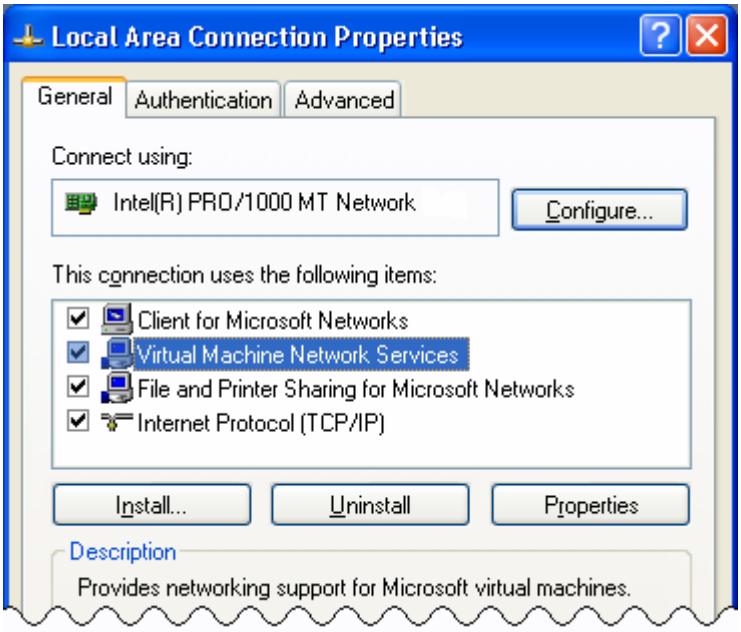


Network setting: the host network adapter

The networking setting to use the host network adapter maps the virtual machine network adapter directly to the host network adapter.

The host adapter is switched into promiscuous mode, and the MAC address and IP address of the virtual machine network appear directly on the external network connected to the host adapter. To other computers on the external network, the virtual machine and host machine appear as separate computers on the network, each with a separate MAC address and IP address.

If you want to use this setting, the Virtual Machine Network Services driver needs to be enabled on the host network adapter. This Open Systems Interconnection (OSI) layer 2 service switches the host network adapter in promiscuous modes, redirects incoming network traffic on the virtual machine MAC address to the correct virtual machine, and places the virtual machine MAC address in outbound network traffic from the virtual machine.



Virtual Machine Network Services on a host network adapter

Note: The Virtual Machine Network Services driver is a very low-level network driver. It runs even below the Windows Firewall driver on the host computer. Even if you have enabled Windows Firewall on the host network adapter, Windows Firewall on the host computer does not protect the connection to the virtual machine! If you want to protect the virtual machine network connection, you should enable Windows Firewall on the virtual machine.

If the virtual machine network adapter is configured to obtain a dynamically assigned IP address, then the DHCP server running on the external network will respond to the virtual machine's DHCP requests.

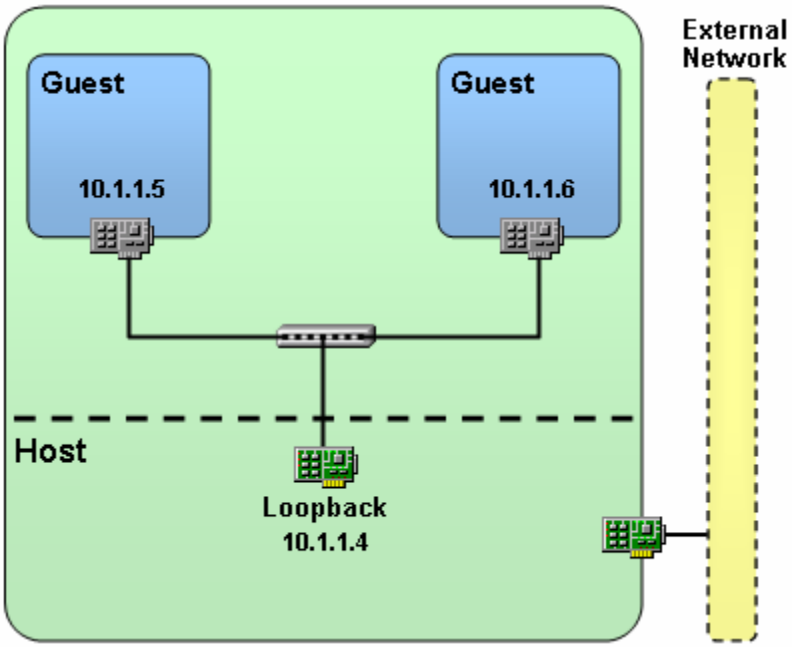
Virtual machines that use the host network adapter networking setting can communicate with other virtual machines that are configured to use the same host network adapter, with the host computer, and with all computers on the external network.

Network setting: Microsoft Loopback Adapter

When you connect multiple virtual machines by using the **Local only** networking setting, and you want to set up a more advanced networking scenario that consists of separate network segments, you can configure some virtual machine network adapters to use the host physical network adapter. In that way, the virtual machines can communicate with one another and are on a different network segment than the **Local only** virtual switch.

Connecting to the host network adapter exposes virtual machines to the external network, which might not be your intent. Also, you might not have enough physical network adapters on the host computer to support multiple network segments. Or, you might not have access to a physical network, such as when you are working while on an airplane.

The solution is to install one or more special software-only network adapters, called Microsoft Loopback Adapters, on the host computer, and then connect the virtual machines to those host network adapters by using the **Microsoft Loopback Adapter** networking setting.



Network setting: Microsoft Loopback Adapter

A Microsoft Loopback Adapter behaves exactly as a typical network adapter on the host computer, except that no external network is involved and the Loopback Adapter is not related to a physical network card. You can install an unlimited number of Loopback Adapters on a host computer.

To install a Microsoft Loopback Adapter on the host computer:

1. On the **Start** menu, click **Run**.
2. In the **Run** dialog box, type **hdwwiz.cpl**, and then click **OK** to start the Add Hardware Wizard starts.

   You can also start the wizard from **Control Panel**.
3. On the **Welcome to the Add Hardware Wizard** page, click **Next**.

   The wizard will briefly search for new hardware on the host computer.
4. On the **Is the hardware connected** page, select **Yes, I have already connected the hardware**, and then click **Next**.
5. In the **Installed hardware** list box, scroll to the bottom of the list, select **Add a new hardware device**, and then click **Next**.
6. Select **Install the hardware that I manually select from a list (Advanced)**, and then click **Next**.
7. In the **Common hardware types** list, select **Network adapters**, and then click **Next**.
8. In the **Manufacturer** list, select **Microsoft**; in the **Network Adapter** list, select **Microsoft Loopback Adapter**; and then click **Next**.
9. Click **Next** to start installing the Loopback Adapter.
10. On the **Completing the Add Hardware Wizard** page, click **Finish**.

If you want to install two or more Microsoft Loopback Adapters, repeat the steps above.

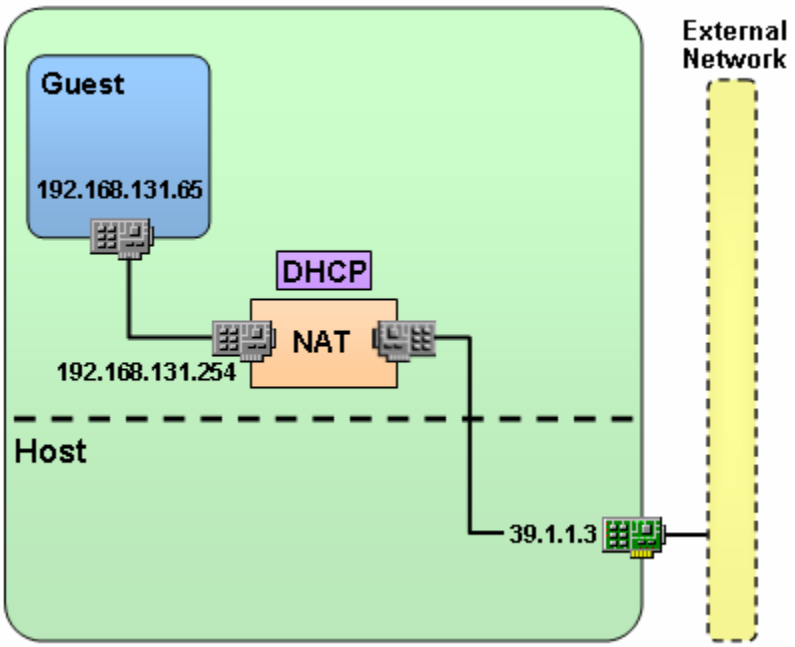
To automate the installation of Loopback Adapters on multiple host computers, you can use the **devcon.exe** command-line tool. See Microsoft Knowledge Base article 311272 for more information.

The installed Loopback Adapter will show up as a Local Area Connection network adapter in the Network Connections window.

By default, the Virtual Machine Network Services driver is enabled on the Loopback Adapter so that the virtual machine network adapters can connect to the Loopback Adapter.

After installation, the Loopback Adapter is configured to use DHCP. To avoid delays when starting the host computer and to allow communication from virtual machines to the Loopback Adapter, configure the Loopback Adapter to use a fixed IP address that is in the same range as the IP addresses configured for the virtual machine network adapters.

Network setting: Shared networking (NAT)



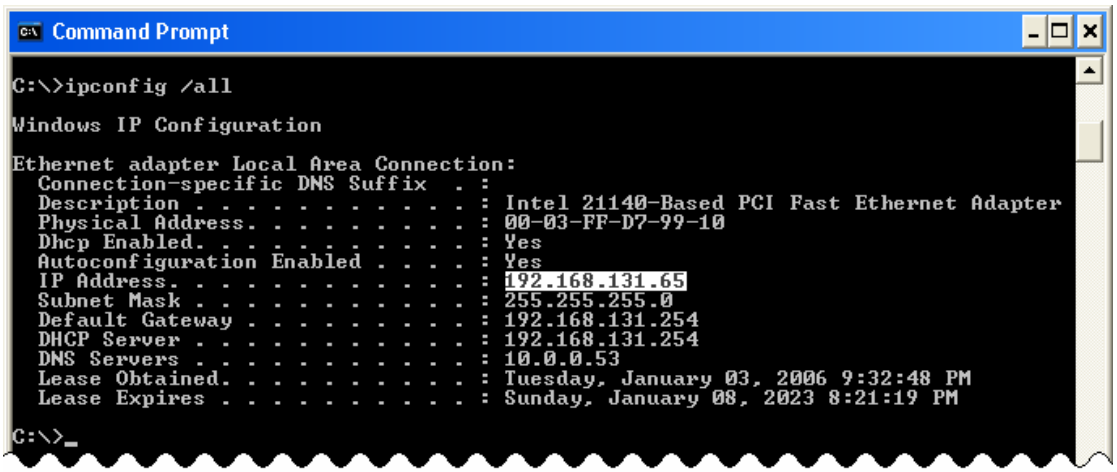
Network setting: Shared networking (NAT)

The **Shared networking (NAT)** networking setting is not comparable with the other networking settings.

When you configure this networking setting, the virtual machine is connected to a private network that includes a built-in DHCP service (192.168.131.254), and a built-in NAT service (192.168.131.254). This option is intended to provide a convenient way of connecting one or more virtual machines to the external network or Internet, without requiring any special DHCP or IP address configuration on the host computer.

On the virtual machine, you need to configure the network adapter to obtain a dynamically assigned IP address.

The Virtual PC built-in DHCP service will supply the virtual machine with an IP address that has a very long lease time, in the IP range 192.168.131.1 to 192.168.1.253. You cannot change this IP range. The default gateway is set to 192.168.131.254, which provides the NAT service to connect to the external network or Internet. The DNS server setting is copied from the host computer, although you can change the DNS server setting to 192.168.131.254 on the virtual machine.



Shared networking (NAT) IP configuration

Note: Shared networking (NAT) does not provide inbound port mapping to the virtual machine. Therefore, the host computer and other virtual machines cannot communicate to the virtual machine. The virtual machine can connect to the host computer and to computers on the external network.

You can configure the **Shared networking (NAT)** setting only for the first network adapter in a virtual machine.

Networking scenarios

You can use the various networking settings to implement networking for one or more virtual machines. Following is an overview of how to use the different networking settings to set up simple and advanced networking scenarios.

Scenario: Using a single network subnet

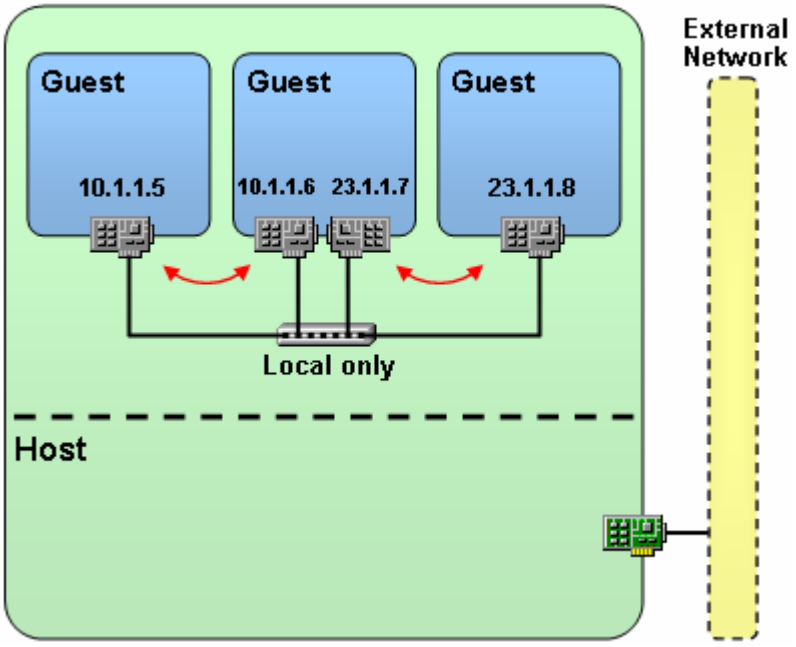
To set up a networking scenario in which multiple virtual machines use a single network subnet, you can use two solutions:

- If the virtual machines do not need to communicate with the host computer, then use the **Local only** setting for the network adapters on each virtual machine.
- If the virtual machines need to communicate with the host computer, then install a Microsoft Loopback Adapter on the host computer and select the **Microsoft Loopback Adapter** setting for the network adapters on each virtual machine.

Scenario: Using multiple network subnets

To set up a networking scenario in which multiple virtual machines use more than one network subnet, you can use two solutions:

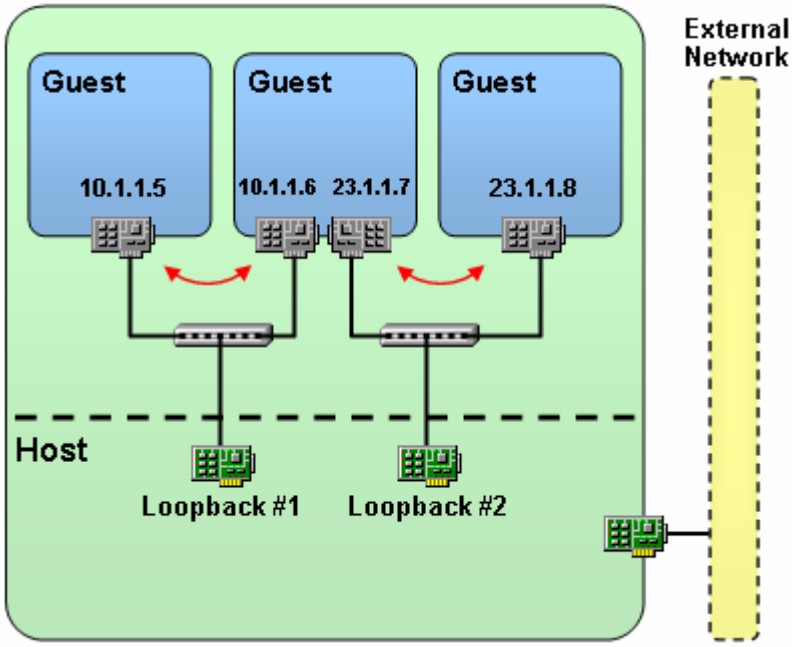
- If multiple IP subnets can share the same network segment, then you can use the **Local only** setting for all the network adapters. The advantage of using this setting is that you can easily copy the virtual machines to another host computer and use the same networking setup, without having to install or configure any IP addresses or network adapters on the host computer.



Different IP subnets on the same network segment

- If the multiple IP subnets must use separate network segments, then install a Microsoft Loopback Adapter for each network segment. When the virtual machines do not need to

communicate with the host computer, then the IP address of the Loopback Adapters is irrelevant.



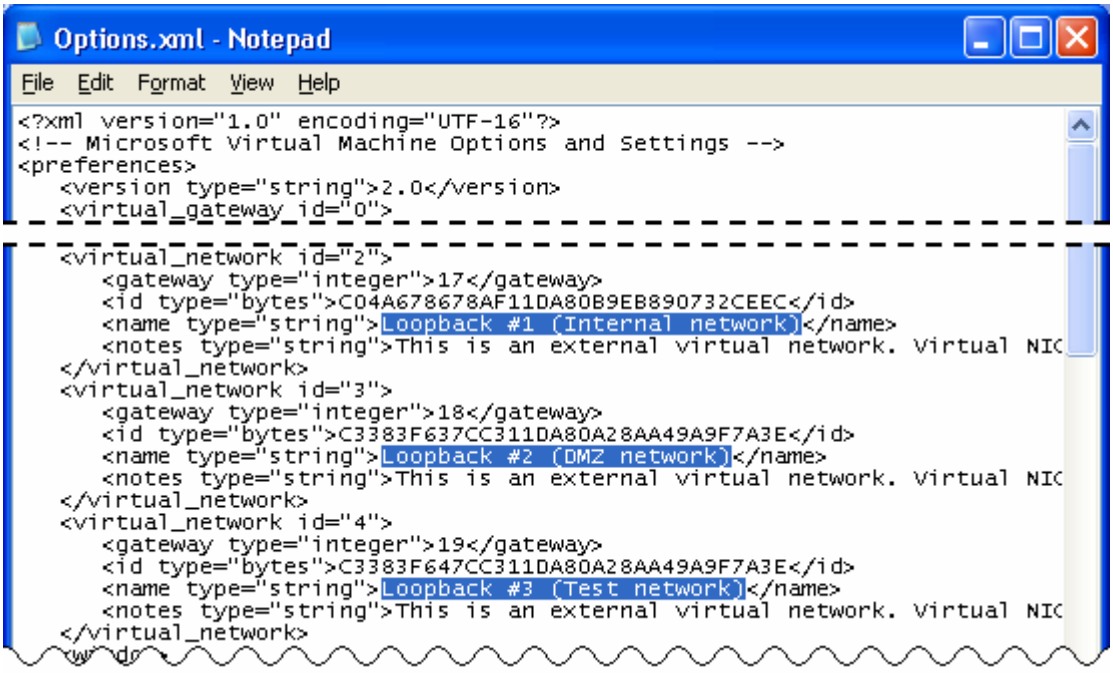
Different IP subnets on separate network segments

When you install more than one Loopback Adapter, you wind up with several similar network adapters in the **Settings** dialog box. The **Settings** dialog box derives the displayed names from the network adapter name as shown in Device Manager (for example, Microsoft Loopback Adapter #2), not the descriptive name that you can change in the Network Connections window (for example, Local Area Connection). The same issue occurs when you have two or more identical physical network adapters on the host computer. Figuring out which name in the list in the **Settings** dialog box refers to which Loopback Adapter or physical network adapter can be confusing. Fortunately, you can change the network adapter names that the **Settings** dialog box uses.

To change the network adapter names:

1. Close Virtual PC. Virtual PC must not be running when you edit the **Options.xml** file.
2. In the **\Documents and Settings\<username>\Application Data\Microsoft\Virtual PC** folder, create a backup copy of the **Options.xml** file.
3. Within the original file's **<virtual_network>** tags, change the value of the **<name>** tag for each network adapter that you want to rename.
4. **Note:** Do not change the value of the **<name>** tag within the **<virtual_gateway>** tags. Change only the **<virtual_network>** tags.
5. Save the changed **Options.xml** file.

The names of the network adapters in the **Settings** dialog box will be updated.



Rename network adapters names in Options.xml

Scenario: Connecting to the Internet

To set up a networking scenario in which one or more virtual machines connect to the external network or the Internet, you can use three solutions.

Use the host network adapter

Configuring the virtual machine network adapter to use the host network adapter is an easy way to connect a virtual machine to the external network. However, this solution might not work in all situations. If the host network adapter is a wireless network adapter, then the Virtual Machine Network Services driver will not be able to send out network packets with a different MAC address. For security reasons, the wireless networking standard does not allow a different MAC address.

Other limitations exist. Depending on the external network connection, using a different IP address on the external connection might not be possible. If you use a dial-up connection to the Internet or connect to the Internet from a hotel room, using multiple external IP addresses is not possible.

Exposing the virtual machine directly to the external network might also be dangerous, especially if you have a virtual machine that is configured to undo changes or that is not fully patched. Remember that Windows Firewall on the host computer does not protect network traffic to the virtual machine.

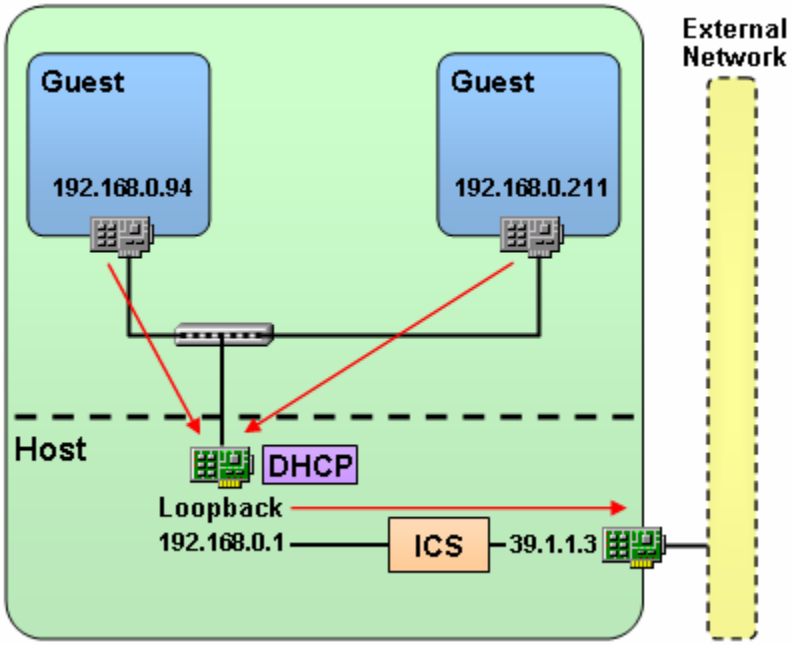
Use the Shared networking (NAT) setting

As an alternative to using the host network adapter, you can configure the virtual machine network adapter to use the **Shared networking (NAT)** setting. This solution does not have the disadvantages associated with using the host network adapter. When you have a wireless network adapter on the host computer, you can connect your virtual machines to the Internet by configuring them to use **Shared networking (NAT)**. This setting also works when you can have only one external IP address, such as when using a dial-up connection or a connection from a hotel room.

This solution also has a few disadvantages. When you use the **Shared networking (NAT)** setting, the virtual machine cannot connect to other virtual machines, nor can the host computer connect to the virtual machine. You cannot change the IP range that is used (192.168.131.1 to 192.168.131.253), and you must use dynamically assigned IP address on the network adapter on the virtual machine.

Use Internet Connection Sharing (ICS) on the host computer

The third way to let virtual machines connect to the Internet is to use a Microsoft Loopback Adapter and enable ICS on the host network adapter. Unlike using **Shared networking (NAT)**, using a Loopback Adapter and ICS on the host computer lets virtual machines connect to the host computer, other virtual machines, and the external network.



Connecting to the Internet by using ICS

To set up Internet connectivity from virtual machines by using ICS on the host computer:

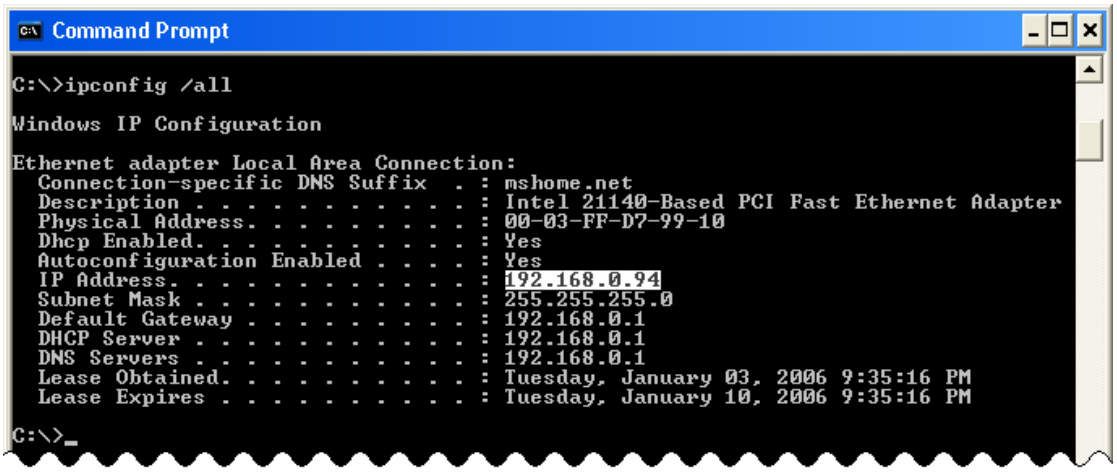
1. On the host computer, install the **Microsoft Loopback Adapter** according to the instructions provided earlier in this section.
2. Open the **Network Connections** window.
3. In the **Network Connections** window, right-click the network adapter that is connected to the external network, and then click **Properties**.
4. In the **Local Area Connection Properties** dialog box, on the **Advanced** tab, select the **Internet Connection Sharing** check box.
5. If you have more than two network adapters, then in the **Home networking connection** list, select the **Loopback Adapter**.
6. Click **OK** to close the **Local Area Connection Properties** dialog box.

   ICS will set the IP address of the Loopback Adapter to 192.168.0.1.
7. Close the **Network Connections** window.
8. In the Virtual PC **Settings** dialog box for the virtual machine, configure the network adapter to use the **Microsoft Loopback Adapter** setting.

9. Click **OK** to close the **Settings** dialog box.

On the virtual machine, configure the network adapter to obtain a dynamically assigned IP address.

The ICS built-in DHCP service will supply an IP address to the virtual machine in the IP range 192.168.0.2 to 192.168.0.255, with a lease time of 7 days. The default gateway is set to 192.168.0.1, which provides the ICS built-in NAT service. The DNS server is set to 192.168.0.1 as well and works as DNS proxy.



ICS-based IP configuration

You cannot change the IP range that ICS uses.

Note: When you want complete control over the IP address range that NAT uses, you can install Virtual PC on Windows Server 2003 Standard Edition, and then use the NAT/Basic Firewall functionality of the Routing and Remote Access service to provide DHCP and NAT services between the Loopback Adapter and the external network. You can change the IP address range that the DHCP Allocator uses in Routing and Remote Access.

Application Development

Virtual PC is a very effective tool for shortening development and test iterations. Creating a virtual environment is a cost-effective way to create a complete and valid test environment. In a multi-tier environment, each tier has unique testing requirements. By virtualizing each tier, a developer can test each tier without the cost and inconvenience of having one physical machine per tier.

Creating and maintaining a valid virtual environment

Developers need to carry out tests on machines that match the end-user environment as closely as possible. You can use physical machines for this purpose, creating and maintaining them by using well-known processes such as Remote Installation Service (RIS) and Windows Deployment Services (WDS) servers, image copying, Sysprep, Microsoft® Systems Management Server (SMS), and manual installations. But you can also use these techniques to create virtual machines that match the necessary physical machines. Virtual machines offer the following advantages over physical machines:

- You can create a complete multiple-machine environment on a developer's or tester's desktop.
- Virtual machines are portable and cost-effective.
- You can use the Shared Folders feature to simplify exchange of files between machines.
- You can quickly undo changes to restore virtual machines to a pre-change state.
- You can configure virtual machines to have an IP address and be on the network, just like physical machines (as discussed in Making copied virtual machines unique).
- You can configure virtual machines to use a specific amount of RAM that better matches the performance of RAM-constrained end-user machines.

Creating a complete virtual environment

Development often reaches beyond the developer's workstation. Modifications to server environments might be an integral part of implementing an application or making changes to an existing application. For example, new application code might be dependent on database changes such as new stored procedures or tables. For initial development of database object changes, having a local copy of the database can speed development. Depending on the size of the production database, having a full-scale copy might be neither feasible nor necessary. In some cases, a subset of data will suffice. In other cases, only the database schema is needed.

Take care when a developer needs production quantities or qualities of data. The developer might not be allowed access to confidential information that is subject to regulatory or legal restrictions on dissemination. Microsoft® SQL Server™ Integration Services (SSIS) can apply transformations to the data as it is loaded into the test database. More-complex data transformations might require custom programming.

Some server class applications, such as Microsoft® Office SharePoint® Portal Server and SQL Server, can be adversely affected if the machine on which they run is renamed. If renaming a virtual machine is not an option when multiple developers each need a separate copy, you will need to use network-isolation techniques such as configuring the virtual network adapter to use **Local only** or **Shared networking (NAT),** as described in the section Understanding networking principles to avoid duplicate name conflicts.

Developers should strive to test as many end-user scenarios as possible on virtual machines instead of on development machines. Too many variables exist to trust the outcomes of testing on a developer's machine. The machine might have elevated security permissions and versions of DLLs and software not present on user machines. Perform end-user testing scenarios on machines that match the actual user environments. By increasing the validity of the tests, cycle times can be reduced because developers and testers will discover bugs earlier in the process.

For an end-user machine to remain a valid environment for testing, you must maintain it properly. This fact is often neglected (if not overlooked completely) in virtual environments. Over time, end-user machines change. For example, Windows XP SP2 tightens security settings. If you test application changes on a virtual Windows XP machine that isn't running SP2, with the intention of deploying those changes to physical machines that are running Windows XP SP2, the test results cannot be trusted. Physical machines are updated through a variety of processes, including manual updates, SMS, Windows Updates, and third-party or custom maintenance applications. A virtual machine might be turned off for extended periods and miss critical updates that affect how the machine and its applications work. Even when a virtual machine is running, it might not be visible on the network and thus might not be included in SMS or another process's updates. The virtual networking settings might be disabled or set to **Local only**, making the virtual machine invisible to the process that performs maintenance updates. You should make provisions to keep virtual machines updated or to replace them with updated versions. As mentioned in Using existing virtual machine files to create more virtual machines, copying virtual hard drive files can introduce problems such as duplicate machine names and SIDs. With proper planning, you can make updates of virtual machines just as effective as updates of physical machines, by making the virtual machines available to the updating processes.

Using differencing disks to increase portability

All tiers of an application's virtual environment can benefit from the use of differencing disks (explained in Using differencing disks to create virtual machines), but usually the greatest benefit is in building multiple versions of the end-user environment. For example, a software vendor might have to support multiple versions (version 1 and version 2) of a product. Furthermore, the product requires certain configuration changes if it is to work properly with Windows XP SP2. You will need four virtual machines to provide a testing environment that covers all configurations:

- Software version 1 with Windows XP SP1
- Software version 1 with Windows XP SP2
- Software version 2 with Windows XP SP1
- Software version 2 with Windows XP SP2

The more permutations you need to test, the more of a problem finding available disk space for all the different virtual machines becomes.

You can use differencing disks to create multiple versions of similar virtual machines, thus minimizing disk space requirements. In this example, you could create a virtual machine equipped with Windows XP SP1. Make this machine's .vhd file the read only parent disk. Create four different virtual machines for each of the four configurations by using differencing disks pointing back to the parent disk. As explained in Using differencing disks to create virtual machines, a differencing disk is a .vhd file that stores only the differences between itself and its parent .vhd file. Any changes to the parent file invalidate all differencing disks that are based on that file, which is why you should make the parent file read only. Because differencing disks contain only changes from the parenting file, small changes will result in small differencing disks. The space savings can be quite substantial when compared to the space requirements of a complete, fully installed virtual machine.

Keep in mind that any change made to a virtual machine that uses differencing disks increases the size of the differencing disk. When the parent disk is badly fragmented, the differencing disk will be as well. But running a disk defragmenter on the differencing disk results in many, many changes relative to the parent disk, in turn causing the differencing disk to grow significantly. Therefore, you should defragment the parent .vhd file before setting it to read only and making differencing disks.

Using Shared Folders to simplify file exchange

The Virtual PC Shared Folders feature in provides a convenient way to exchange files between a host machine and a guest machine. When mapping a drive through Windows Explorer or the **NET USE** command, you need to configure permissions to enable the sharing so that the machine-to-machine communication will work across the network. Shared Folders does not involve the network, so you don't have to provide credentials or configure a share. The folder or volume that is specified when using Shared Folders appears as a mapped drive within the guest machine but does not use the TCP/IP stack as a true mapped drive does. When testing a scenario that relies on the use of an actual mapped drive on a production system, you should use mapped drives instead of Shared Folders to configure the virtual environment. Additionally, true mapped drives are the only way to make network drives available to the guest machine. You can use Shared Folders to map only the host's locally attached drives.

Restoring virtual machines to their pre-change state

One of the many benefits of using Virtual PC is the ability to undo changes through the use of undo disks, saved state, or repeatable save state restore. A developer can deploy changes to a virtual machine and test the changes. If the tests reveal problems, the developer can easily use one of the following methods to undo the changes.

Undo disks

Undo disks are valuable when fixing bugs that require a bit of trial and error to resolve. You can apply and test a proposed fix. If the fix doesn't work, you can undo changes to the virtual machine almost instantaneously, leaving the machine in a pristine state and ready for the next iteration. Restoring a physical machine to a known good state requires a lengthy process of rebuilding or reimaging.

When the undo disk feature is enabled, you can undo any changes by selecting **Turn off and delete changes** from the list on the **Close** dialog box when you close the virtual machine. This feature is not enabled by default, so if you want to use it you will need to take the following steps:

1. Open the **Virtual PC Console**.
2. Select the virtual machine you want to modify.
3. Click **Settings.**
4. In the Settings list, click **Undo Disks**.
5. Select the **Enable undo disk** check box.

If you are using undo disks on a Windows XP machine, you probably should turn off System Restore. Undo disks and System Restore offer somewhat redundant functionality. Turning off System Restore minimizes disk space requirements for the virtual machine.

Saved State

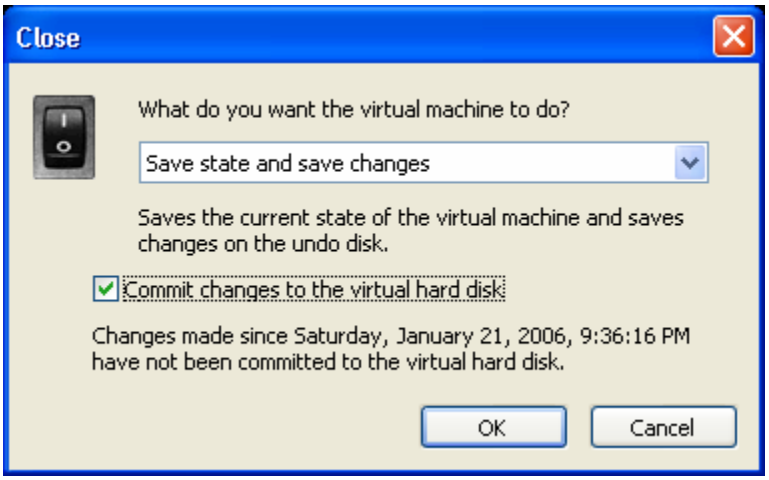
If a particular debugging or testing task takes longer than a single day or session to complete but running processes (such as background services) might affect the test results, saving state can be particularly valuable. Instead of leaving the virtual machine running overnight, you can specify **Save state and save changes** when you close the machine. You'll be able to resume the testing session exactly where you left off. You can also pause a virtual machine to suspend its execution by using the **Pause** item on the Action menu or pressing **Right Alt+P**.

Repeatable save state restore

When you want to start the machine from a saved state on a repeating basis, you can use repeatable save state restore. This type of restore is essentially a partial undo (i.e., an undo to the saved state, not an undo of all changes). This feature does not exist in Virtual PC 2004, but you can implement the feature through a little tweaking and a simple command script.

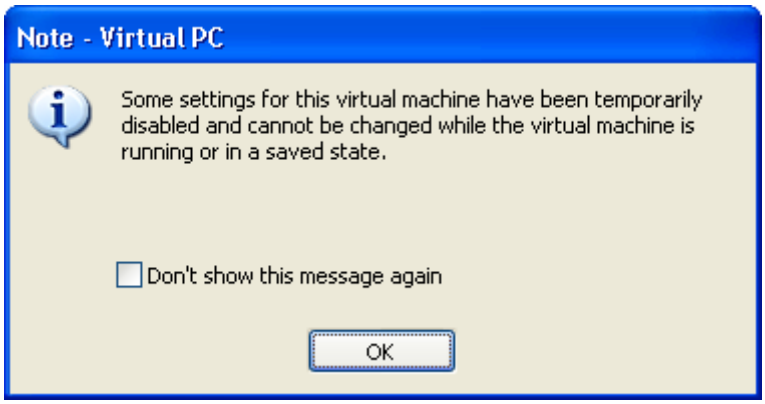
To enable a repeatable save state:

1. Follow the steps described above to enable undo disks.
2. Configure any other necessary settings, such as networking settings.
3. Start the virtual machine and get it to the state you want to save.
4. Turn off the virtual machine, choosing **Save state and save changes** and selecting the **Commit changes to the virtual hard disk** check box on the **Close** dialog box.



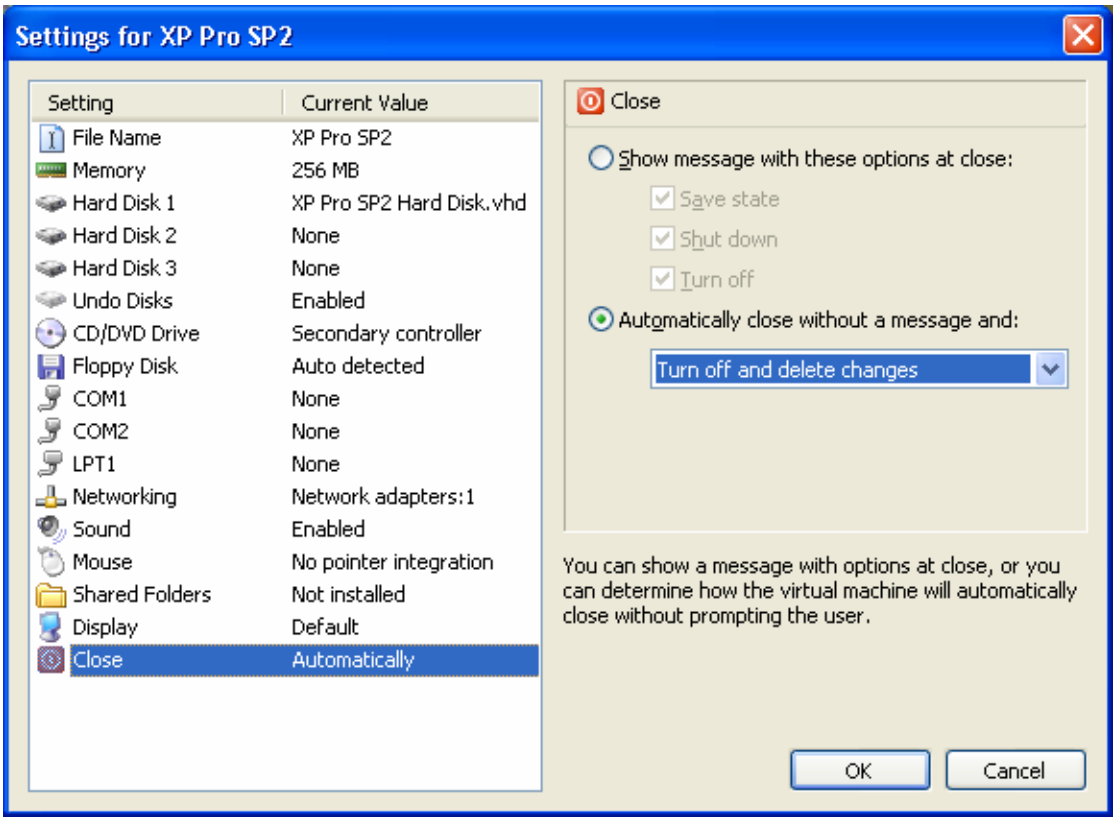
Saving state and committing changes

5. Open the settings for the virtual machine again. Ignore the message about how some settings cannot be changed while a virtual machine is in a saved state.



Repeatable saved state restore message

6. In the **Settings** dialog box, select the **Close** setting in the left pane. Click **Automatically close without a message and:**, then select **Turn off and delete changes** from the list. Click **OK** to close the **Settings** dialog box.



7. Go to the location on your file system in which the virtual machine .vmc file resides (the **My Documents\My Virtual Machines** folder by default). Within this folder, create a new subfolder called **backup**.

8. Copy the .vmc and virtual saved state (.vsv) files to the **backup** folder. By copying the files from this folder every time you run the virtual machine, you can start from the saved state as many times as you like.

9. Create the following command file to copy the files from the **backup** folder to the virtual machine folder:

```
@echo off

copy .\backup\. . /y > null

start "" "C:\Program Files\Microsoft Virtual PC\Virtual PC.exe"
   -startvm "C:\Documents and Settings\John\My Documents\
   My Virtual Machines\XP Pro SP2\XP Pro SP2.vmc"
```

Note: This command file contains only three lines; the last line is long and appears wrapped in this text. Make sure to replace the sample path in this code with your path to the .vmc file. You must use the full path to the file, not a relative path. The `@echo off` and `> null` statements suppress messages in the command window. The `start` statement causes the command window to close after Virtual PC starts.

Kernel Debugging

Kernel debugging tools are useful for debugging the Windows operating system, device drivers, memory-dump files, and applications. To use kernel debugging, you need two computers: the machine being debugged (called the *target* machine) and another machine to run the debugger (called the *host* machine). The term *host* is also used when discussing virtual machines: The physical machine that runs Virtual PC is referred to as the *host* machine, and the virtual machines running on the host are known as *guest* machines. To avoid confusing the two types of host machines during this discussion, the Virtual PC host machine is referred to as the *virtual host machine* and the machine that runs the kernel debugger is referred to as the *debugger host machine*. The debugger host can be either a virtual machine or a physical machine.

A debugging session requires that the host and target machines be connected. When both machines are physical machines, you need to use a physical connection such as a null-modem cable, an Institute of Electrical and Electronics Engineers (IEEE) 1394 cable, or a USB 2.0 cable. But if the target machine is a virtual machine on the debugger host, no physical connection is necessary because no second physical machine exists. Having a complete debugging environment on one physical machine makes debugging much more convenient.

Microsoft has a free debugging suite called Debugging Tools for Windows, which you can download from http://www.microsoft.com/whdc/devtools/debugging/debugstart.mspx. The suite is also available on the Windows Drive Development Kit (DDK), Platform Software Development Kit (SDK), and Customer Support Diagnostic CDs. You can install the tools on a physical machine or a virtual machine.

Virtual PC does not support IEEE 1394 ports, but it does support serial ports. Although you can emulate the behavior of a physical null modem cable, debugging a virtual machine over named pipes is more straightforward. This white paper focuses on configuring the kernel debugger to work in a virtual environment. For specific information about the debugging process, refer to the product documentation for the debugging tools you choose to use.

Configuring the debugger host

You can use the command-line kernel debugger **kd.exe** or the Windows application **WinDbg** for kernel debugging. Even when you choose to use WinDbg, you will probably find it easier to start the tool from a command line because of the options that you must specify: the path to the debug symbols and how to connect to the target.

Downloading debug symbols

The debug symbols are found in Program Debug Database (.pdb) files, which you can obtain from a symbol server or from a file system cache. The symbol server can be a corporate symbol server or the Microsoft symbol server at http://msdl.microsoft.com/download/symbols. Please note that the Microsoft server is not browsable and does not respond to ping requests. Using the Microsoft symbol server is a good idea because Microsoft keeps the server current with the most recent symbols for service packs and Windows updates. When using a symbol server, you need to specify a local or network directory in which to cache the symbols that you download from the symbol server. If the directory you specify does not exist, it will be created for you. The syntax for using the Microsoft symbol server is as follows:

```
kd –y srv*symbolStoreDirectoryPath*
   http://msdl.microsoft.com/download/symbols
```

Note: This command should be typed on one line.

Although using the Microsoft symbol server is a good idea and the approach that the authors recommend, it is not always practical because of connectivity issues, such as those that exist

when doing a demo or working while on an airplane. If you choose to download symbols for local installation, you can do so at http://www.microsoft.com/whdc/devtools/debugging/symbolpkg.mspx. The default location for the symbols is **C:\WINDOWS\Symbols**, which you specify to the kernel debugger by using the following syntax:

```
kd -y C:\WINDOWS\Symbols
```

Be sure that you download the correct symbols for your operating system and service pack. If you download the wrong symbols, you'll see an error message such as the following:

```
C:\Program Files\Debugging Tools for Windows>kd -y C:\WINDOWS\Symbols
-k com:pipe,port=\\.\pipe\debugPipe,resets=0,reconnect

Microsoft (R) Windows Debugger  Version 6.5.0003.7
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\debugPipe
Waiting to reconnect...
Connected to Windows XP 2600 x86 compatible target, ptr64 FALSE
Kernel Debugger connection established.
Symbol search path is: C:\WINDOWS\Symbols
Executable search path is:
*** ERROR: Symbol file could not be found.  Defaulted to export
symbols for ntoskrnl.exe -
```

To troubleshoot a problem with downloaded symbols, modify the command line to use the **-n** (noisy) switch. By using this switch, you can see that the specified symbol path does not contain the correct symbols, which explains the error message.

```
C:\Program Files\Debugging Tools for Windows>kd -y C:\WINDOWS\Symbols
-n -k com:pipe,port=\\.\pipe\debugPipe,resets=0,reconnect

Microsoft (R) Windows Debugger  Version 6.5.0003.7
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\debugPipe
Waiting to reconnect...
Connected to Windows XP 2600 x86 compatible target, ptr64 FALSE
Kernel Debugger connection established.
DBGHELP: Symbol Search Path: C:\WINDOWS\Symbols
Symbol search path is: C:\WINDOWS\Symbols
Executable search path is:
DBGHELP: SharedUserData - virtual symbol module
DBGHELP: C:\WINDOWS\Symbols\ntoskrnl.pdb - file not found
DBGHELP: C:\WINDOWS\Symbols\symbols\exe\ntoskrnl.pdb - file not found
DBGHELP: C:\WINDOWS\Symbols\exe\ntoskrnl.pdb - mismatched pdb
DBGHELP: ntoskrnl.pdb - file not found
DBGHELP: Couldn't load mismatched pdb for ntoskrnl.exe
*** ERROR: Symbol file could not be found.  Defaulted to export
symbols for ntoskrnl.exe -
```

You can prevent this type of error by using the Microsoft symbol server during your first debugging session. Doing so will ensure that you download the correct symbols, which will be cached locally. Later, you can point to your local cache of symbols. Assuming that you cached the symbols at **C:\websymbols**, you can use this syntax to point the kernel debugger to the local cache:

```
kd -y C:\webymbols
```

Another advantage this approach has over downloading and installing symbols is that using a symbol server caches only the symbol files that you actually use, reducing disk space requirements.

Connecting to the target

The first example uses a physical debugger host and a virtual target, using a named pipe connection. The syntax for specifying a named pipe connection to the client is as follows:

```
kd -k com:pipe,port=\\.\pipe\userSpecifiedNameForNamedPipe,
   resets=0,reconnect
```

Note: This command should be typed on one line.

You'll need to open a command prompt to run either debugger. If you haven't modified your path to point to the location of the Debugging Tools for Windows files, you'll need to first change your directory to **C:\Program Files\Debugging Tools for Windows** (or whatever location you specified during installation). Here are the commands issued to run the test cases described in this document:

```
kd -y srv*c:\websymbols*http://msdl.microsoft.com/download/symbols
   -k com:pipe,port=\\.\pipe\debugPipe,resets=0,reconnect

windbg -y srv*c:\websymbols*http://msdl.microsoft.com/
   download/symbols -k com:pipe,port=\\.\pipe\debugPipe,
   resets=0,reconnect
```

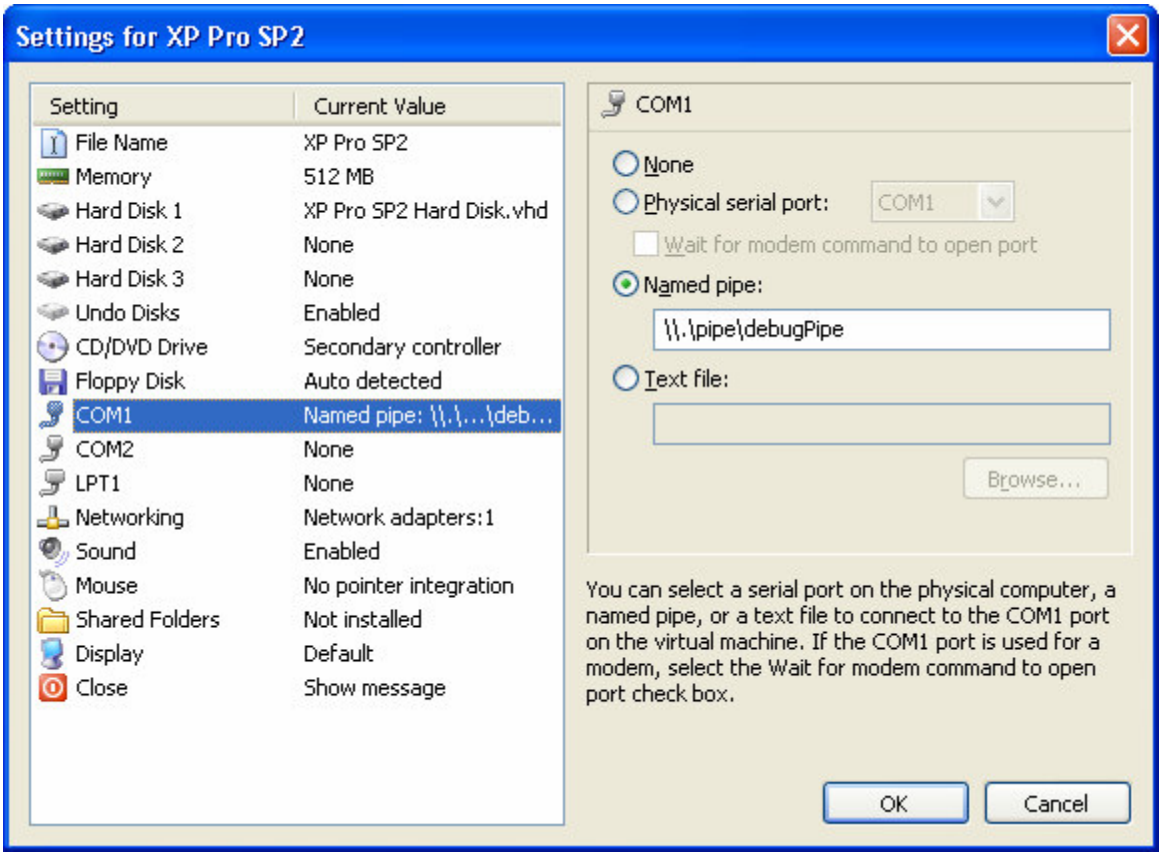
Note: Each command above is actually one line.

If you are going to use this command often, you'll probably find it more convenient to put the command in a shortcut or .cmd file so that you don't have to type it every time you use it.

Configuring the target

To begin the process of configuring the virtual debugging target:

1. Open the **Settings** dialog box for the target.
2. Select the **COM1** setting, then select **Named pipe**. Specify the named pipe that you specified in the **kd.exe** or **WinDbg** command.



Using Virtual PC settings to configure named pipes

3. Start the virtual target machine and open its boot.ini file. This file is in the root of C: but is hidden by default, so you might need to change your Windows Explorer settings to see the file. (Even when the file is hidden, you can bring it up in Notepad by going to **My Computer** | **Properties** | **Advanced** | **Settings** | **Edit**.) You will see a line similar to the following:

```
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP
   Professional" /noexecute=optin /fastdetect
```

   Add the following code to the end of the line:

```
/debug /debugport=com1
```

   Save your changes. The final result will be something like this:

```
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP
   Professional" /noexecute=optin /fastdetect /debug
   /debugport=com1
```

   Note: The commands above are each one line.

4. Shut down the target virtual machine. (Do not turn off and close the machine; only shut it down. The virtual machine will be restarted to use the changes you just made.)

Debugging a target virtual machine from a physical debugger host

To debug a target virtual machine from a physical debugger host machine:

1. Run **kd.exe** or **WinDbg** from a command prompt on the debugger host machine. You will see the following output:

```
C:\Program Files\Debugging Tools for Windows>kd -y
srv*c:\websymbols*http://msdl.microsoft.com/download/symbols -k
com:pipe,port=\\.\pipe\debugPipe,resets=0,reconnect

Microsoft (R) Windows Debugger  Version 6.5.0003.7
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\debugPipe
Waiting to reconnect...
```

The kernel debugger will wait until the boot.ini debug-enabled virtual target boots up. Once the target machine starts the boot process, the debugger continues with output similar to this:

```
Connected to Windows XP 2600 x86 compatible target, ptr64 FALSE
Kernel Debugger connection established.
Symbol search path is:
srv*c:\websymbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows XP Kernel Version 2600 UP Free x86 compatible
Built by: 2600.xpsp_sp2_gdr.050301-1519
Kernel base = 0x804d7000 PsLoadedModuleList = 0x8055a420
System Uptime: not available
Break instruction exception - code 80000003 (first chance)
```

2. Press **CTRL+C** (if you are using **kd.exe**) or **CTRL+BREAK** (if you are using **WinDBG**) to bring up a **kd** command prompt. The following output appears:

```
*******************************************************************
*                                                                 *
*   You are seeing this message because you pressed either        *
*       CTRL+C (if you run kd.exe) or,                            *
*       CTRL+BREAK (if you run WinDBG),                           *
*   on your debugger machine's keyboard.                          *
*                                                                 *
*                   THIS IS NOT A BUG OR A SYSTEM CRASH           *
*                                                                 *
* If you did not intend to break into the debugger, press the "g" *
* key, then press the "Enter" key now.  This message might        *
* immediately reappear.  If it does, press "g" and "Enter" again. *
*                                                                 *
*******************************************************************
nt!RtlpBreakWithStatusInstruction:
804e3592 cc              int     3
```

3. At the resulting command prompt, enter **!process 00** to confirm that the kernel debugging connection was successful. You will see output similar to the following:

```
**** NT ACTIVE PROCESS DUMP ****

PROCESS 823c8a00  SessionId: none  Cid: 0004    Peb: 00000000  ParentCid:
0000

    DirBase: 00039000  ObjectTable: e1001d28  HandleCount: 285.

    Image: System
```

Debugging a target virtual machine from a virtual debugger host

Configuring a virtual machine to act as a debugger host requires the same named pipe configuration in the Virtual PC settings as described for the virtual target in Configuring the debugging target. You need to install Debugging Tools for Windows on the virtual debugger host just as you do on a physical host. You will use a slightly different syntax on the virtual debugger host to connect to the target machine:

```
kd -y srv*c:\websymbols*http://msdl.microsoft.com/download/symbols
   -k com:port=com1
```

Note: This command is one line.

Once the kernel debugger has started, follow the same procedure as for a physical debugger host. Be sure to start the virtual debugger host machine first, and then start the virtual target.

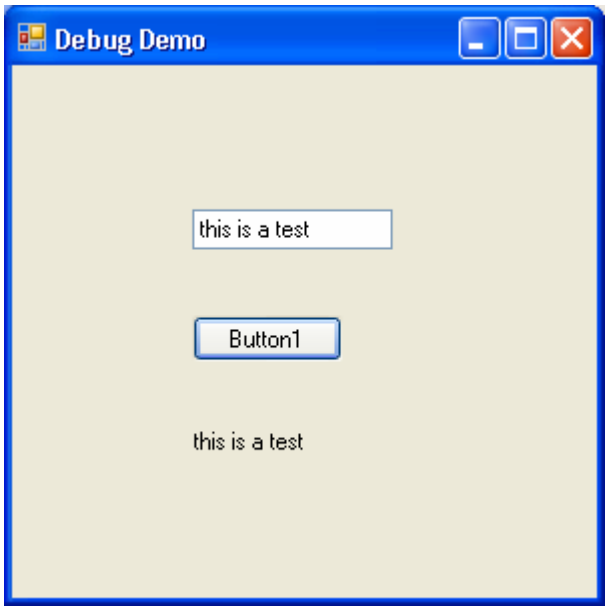
Remote Application Debugging

You can debug an application locally or remotely. Local debugging is the easiest method, but the easy way out isn't always best. A developer's machine often isn't configured the same as an end user's machine. You might not even be able to match an end user's configuration because developers and end users might be using different versions of Windows.

By configuring a virtual machine with the same version of Windows and the same applications and settings as an end user machine, a valid debugging target environment can be brought to the developer's desktop. A virtual machine used as a debugging target is more convenient than a physical machine. It doesn't take up physical space and it offers fast recovery and repeatability through the use of Undo Disks. Debugging scenarios often need to be repeated. With a physical machine, repeatability can require reinstalling or reimaging to restore the machine to a pristine state.

Even when Virtual PC runs on the host machine's local desktop, guest virtual machines appear as other machines on the network when they are bound to the host's network adapter. Debugging a virtual machine involves making the same configuration and security changes as are necessary to remotely debug a physical machine on the network. The remote debugging principles described are not specific to Virtual PC; they can be applied to physical machines as well. This section focuses on debugging managed code. Debugging unmanaged code is not covered.

The sample debugger host is a Windows XP Professional SP2 system running Visual Studio 2005 Team Suite. The sample target is a virtual Windows XP Professional SP2 system running the .NET Framework 2.0 within Virtual PC 2004 SP1. Windows Firewall is enabled on the target. To demonstrate remote debugging, this example uses a simple Windows application named DebugDemo, which consists of a text box, button, and a label. When you click the button, the contents of the text box are displayed as the label control's text property. DebugDemo was developed using Visual Studio 2005.



Sample application

Both the debugger host machine and the virtual target machine have identical users: **Dbug** and **Tester**. User **Dbug** is a member of the local Administrators group. User Tester is an ordinary user. The passwords for the users are the same on both machines. The physical debugger host machine is WXP. The remote virtual debugger target machine is WXPSP2.

Visual Studio 2005 debugging

Visual Studio 2005 is easier to configure for remote debugging than earlier versions of Visual Studio. Remote debugging requires Visual Studio 2005 Professional or Team System and is not supported on Standard or Express editions.

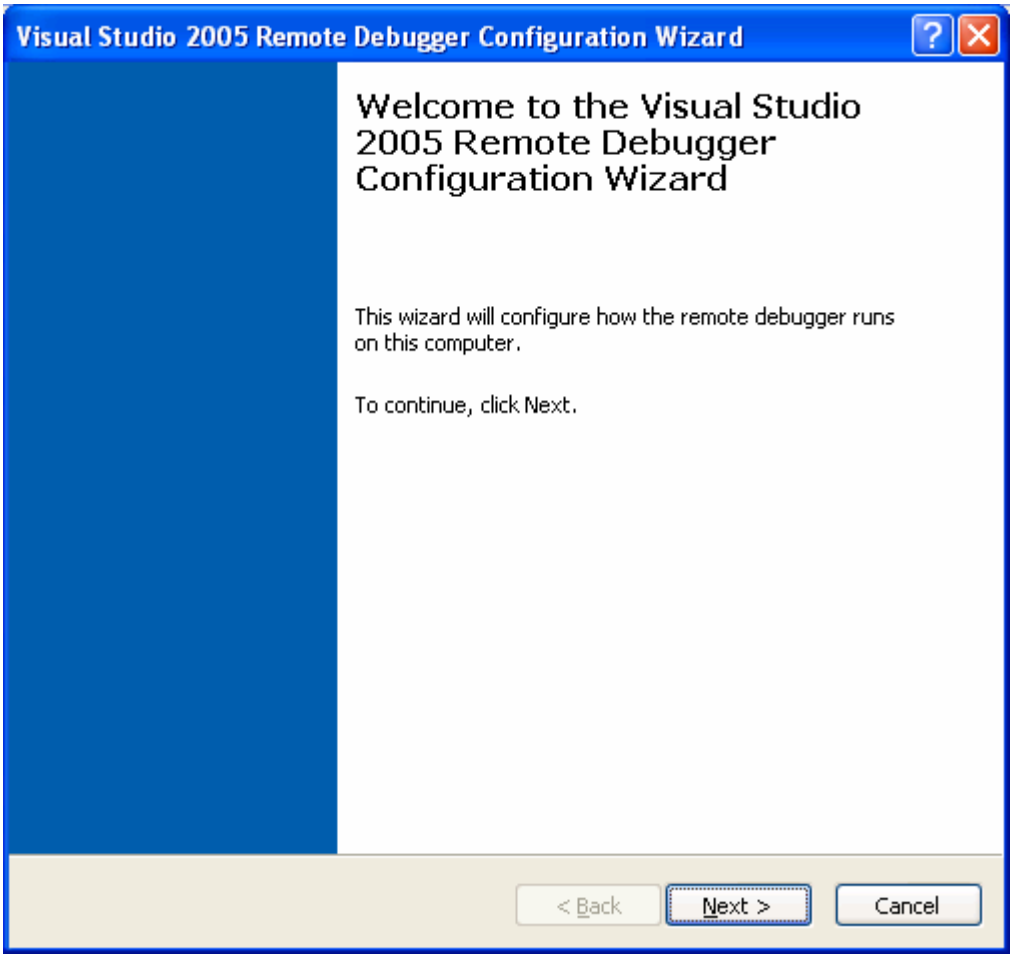
To use remote debugging with Visual Studio 2005, you will use the **Remote Debugging Monitor**, which replaces earlier versions' Machine Debug Manager. You run this tool on the remote debugger target machine. To do so, you must install the tool on the target, run it from a share that is accessible to the target, or run it from the Visual Studio installation media. Running from a share is an appealing option because it does not require installation of any additional components, but it does have a few limitations. Running from a share does not support debugging of Microsoft® Windows® 98 or Windows® Me. Additionally, you need to manually attach the target machine to a process to step into an XML Web service, Active Template Library (ATL) server-based Internet Server API (ISAPI) extension, or ASP.NET Web service.

You can debug applications on a variety of operating system without having multiple physical machines. You can leverage the power of Visual Studio 2005 debugging by remotely connecting to virtual machines running other Windows operating systems.

Installing the remote debugging components

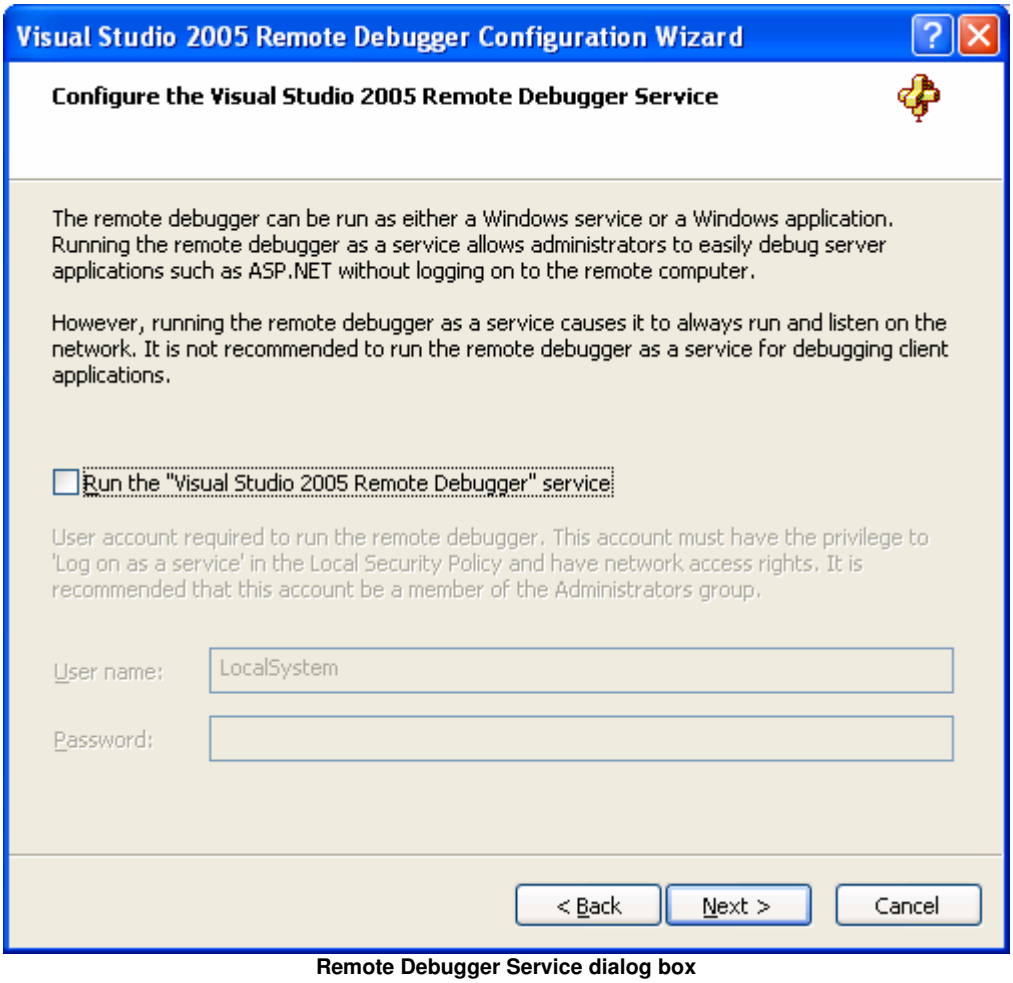
The remote debugging components are on the Visual Studio 2005 installation media. (If you obtain Visual Studio 2005 on DVD, the **Remote Debugging Monitor** files are in the **vs**/**Remote Debugger** folder. If you have CDs, the files are on the last CD in the set, in the **Remote Debugger** folder.) Open the appropriate folder and then open the correct folder for the target operating system (**x86** for a 32-bit system; **x64** for a 64-bit system). Run **rdbgsetup.exe** to install the remote debugging components, then follow these steps:

1. On the **Welcome to Microsoft Visual Studio 2005 Remote Debugger Setup** page, click **Next**.
2. Accept the license agreement and click **Install**.
3. Click **Next** to start the **Remote Debugger Wizard**.
4. On the **Welcome to the Visual Studio 2005 Remote Debugger Configuration Wizard** page, click **Next**.



Visual Studio 2005 Remote Debugger Configuration Wizard

5. You will need to choose whether to run the Remote Debugger as an application or a service. Running as an application is the default. If you must debug an ASP.NET application, select the **Run the "Visual Studio 2005 Remote Debugger" service** check box to run the debugger as a service. Otherwise, choose the default and leave the check box clear. Click **Next**.



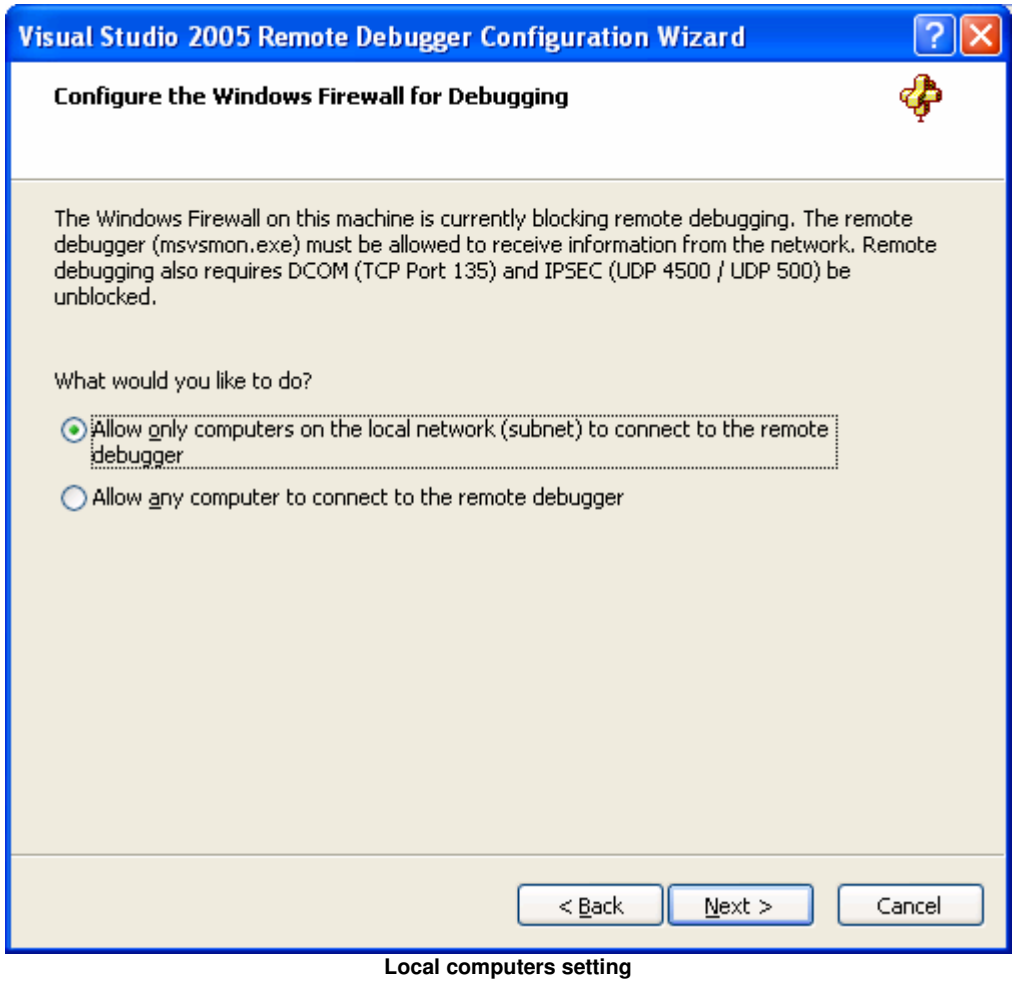
Remote Debugger Service dialog box

6. Because the debugging target machine has Windows Firewall enabled, you need to configure Windows Firewall to permit debugging. The requirements are as follows:

   - Configure msvsmon.exe to receive data from the network.
   - Unblock DCOM on TCP port 135.
   - Unblock Internal Protocol Security (IPSec) on UDP ports 500 and 4500.

   In the next step, the wizard will automatically make the necessary changes to the Windows Firewall settings.

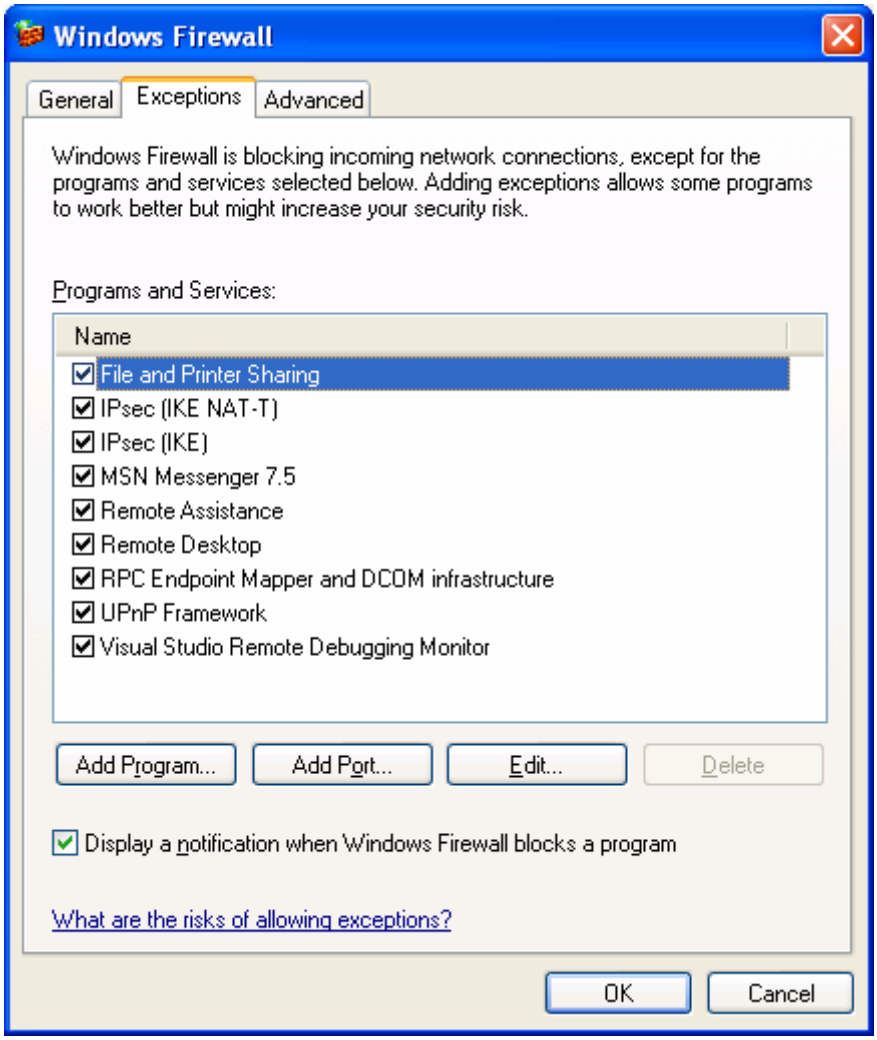
   The default option is to only allow computers on the same subnet to connect to the Remote Debugger Monitor. This is a more secure option than allowing any computer to connect to the remote debugger. Accept the default option and click **Next**.



Local computers setting

7. Click **Finish** to close the wizard.

The modified Windows Firewall settings are shown below:



Windows Firewall settings for remote debugging

The changes to the Windows Firewall settings enable

- File and Printer Sharing
- IPsec (IKE NAT-T)
- IPsec (IKE)
- RPC Endpoint Mapper and DCOM infrastructure
- Visual Studio Remote Debugging Monitor

Keep in mind that not only must these settings be enabled, Windows Firewall must allow exceptions.

An Administrator can run the Visual Studio 2005 Remote Debugger Configuration Wizard at any time should the Windows Firewall settings need to be reset. To invoked the wizard, go to **Start** | **All Programs** | **Microsoft Visual Studio 2005** | **Visual Studio Tools** | **Visual Studio 2005 Remote Debugger Configuration Wizard**.

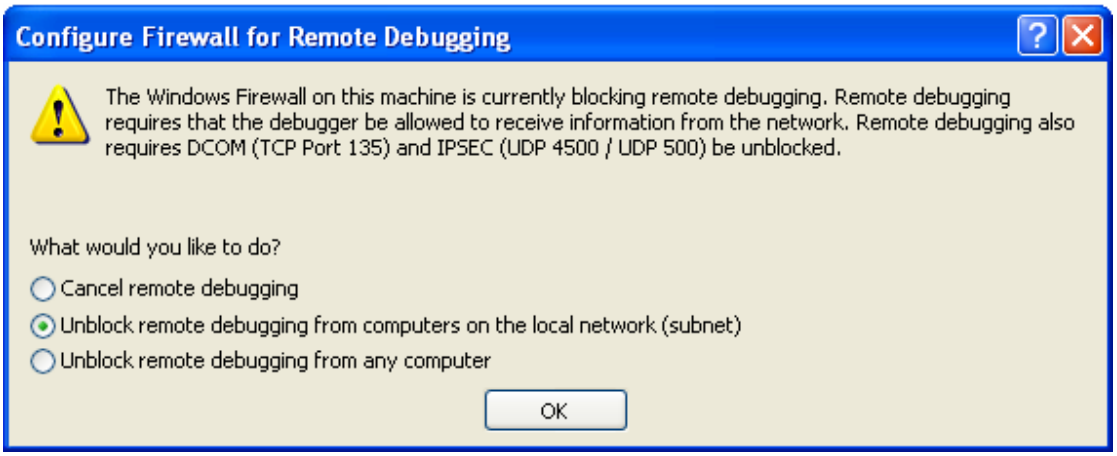
The remote debugger is now installed and Windows Firewall is configured on the virtual debugger target.

Configuring remote debugging permissions

When you debug an application, you are actually debugging a process. Each process runs under specific security credentials. When you run Visual Studio 2005 with the same credentials as the process being debugged, no additional configuration is needed. However, running Visual Studio 2005 with the same credentials as the process being debugged isn't always possible or desirable. For example, if you are debugging an ASP.NET worker process (aspnet_wp.exe), you can't run your machine as the ASP.NET user. In the example that follows, the host Dbug user will debug the DebugDemo application, which is running as the Tester user. This is possible because the remote machine has matching Dbug user credentials. When the host and remote computers do not have matching credentials, you must configure remote debugging to work with any computer, which is a less-secure option.

Configuring local debugging permissions

If Windows Firewall is enabled on the debugger host machine running Visual Studio 2005, you need to configure Windows Firewall in a manner compatible with the remote debugger target machine. If the debugger host's Windows Firewall settings are incorrect, you will see the following message when you attempt to start debugging:



Windows Firewall on host, blocking remote debugging

You will need Administrator privileges to reconfigure Windows Firewall to enable remote debugging. If you encounter this error message and you are attempting to debug without having Administrator privileges, you will need to log on as an Administrator to reset Windows Firewall.

Compiling the Visual Studio 2005 project files on the host

To configure the Visual Studio 2005 debugger host:

1. Open the sample project in Visual Studio. Check the project properties and make sure it is configured for Debug.
2. Build the project to create the ,exe, .dll, and .pdb files..
3. Copy the .exe, .dll, and .pdb files to the remote target machine.

Configuring the Remote Debugging Monitor

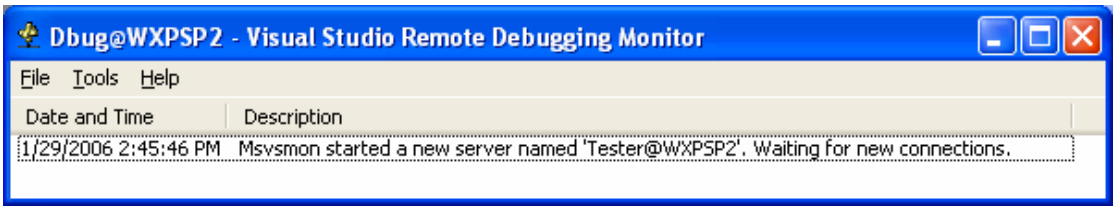
If Visual Studio 2005 is running on the host machine as user Tester with the same password as the target machine, no further configuration of the Remote Debugging Monitor would be

necessary. To make this example more meaningful, the host machine is logged on as user Dbug. Because the users on the two machines do not match, additional configuration of the Remote Debugging Monitor is required.

To configure the Remote Debugging Monitor on the virtual machine serving as the debugger target:

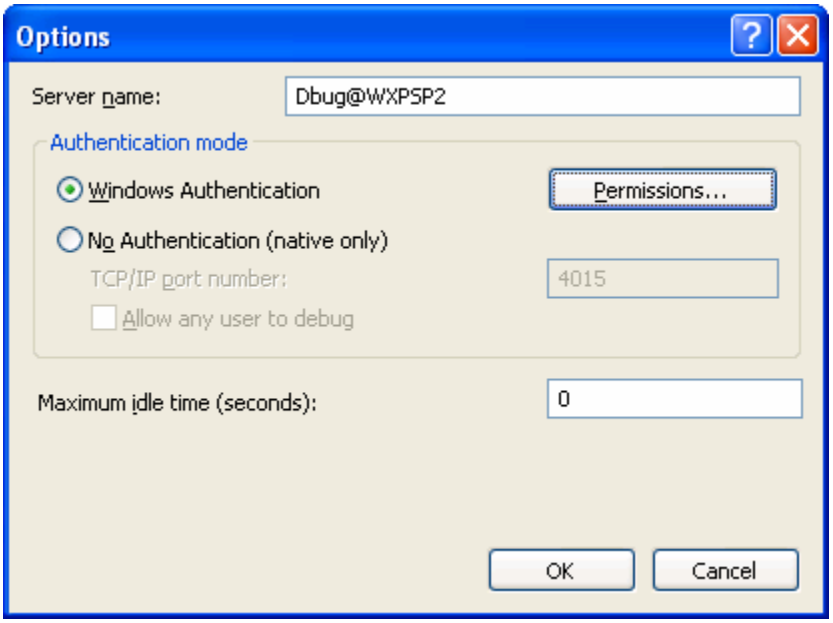
1. Go to the remote target machine, log on as user Tester, and select **Start | Microsoft Visual Studio 2005 | Visual Studio Tools | Visual Studio 2005 Remote Debugger** to start the **Remote Debugging Monitor**.

   You will see a dialog box similar to the following:



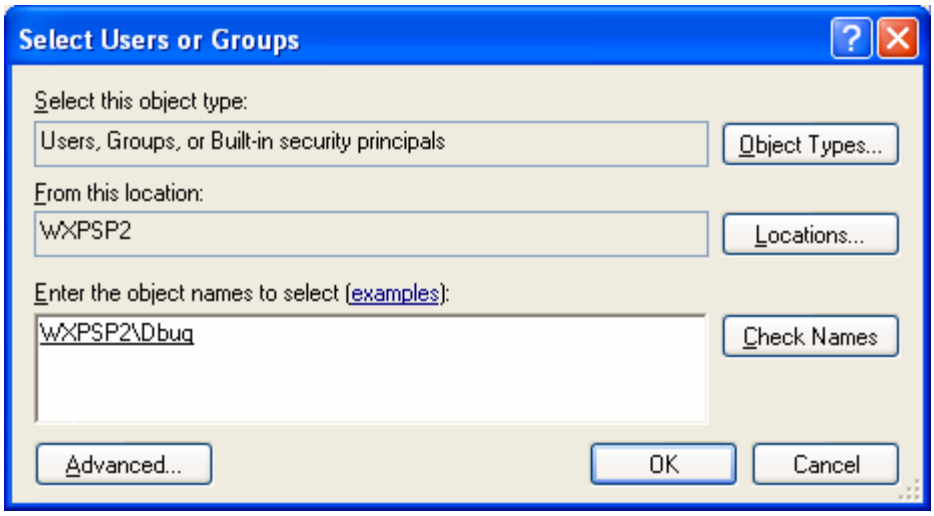
Remote Debugging Monitor waiting for a connection from Visual Studio 2005

2. In the **Tools** menu, click **Options**.
3. In the **Server name** box, change the user from **Tester** to **Dbug**.
4. Click **Permissions** to open the **Permissions for Remote Debugging** dialog box.



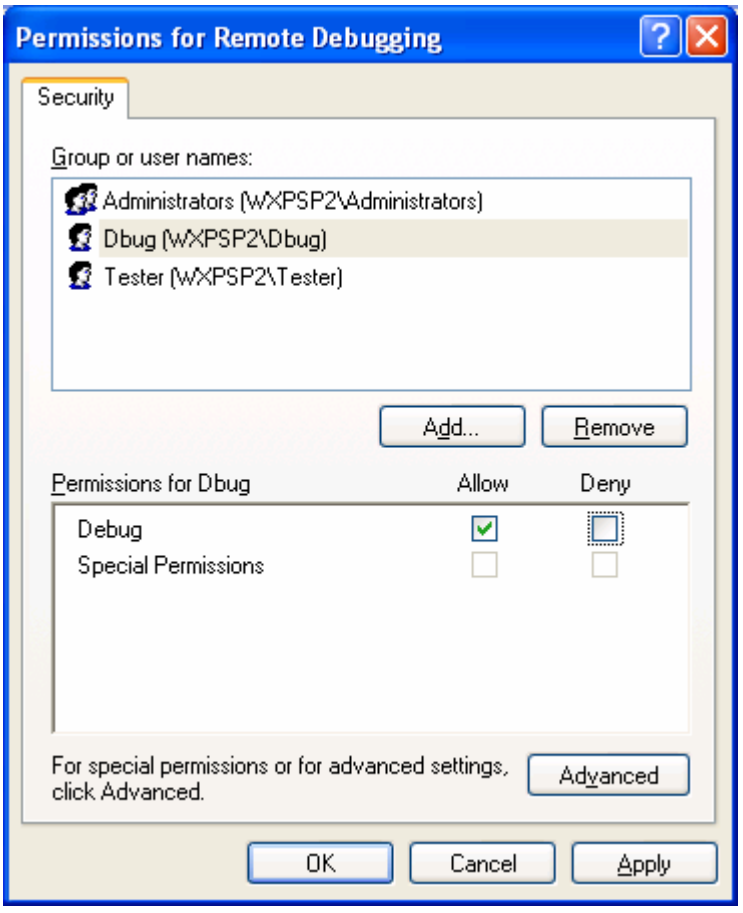
Remote Debugging Monitor options

5. Click **Add** to open the **Select Users or Groups** dialog box. Add the Dbug user and click **OK**.



Adding the Dbug user

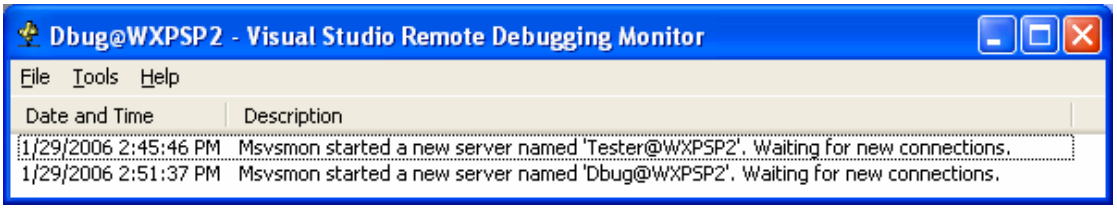
The **Permissions for Remote Debugging** dialog box appears:



Dbug user with Debug permissions

6. Click **OK** to close the **Permissions for Remote Debugging** dialog box.
7. Click **OK** to close the **Options** dialog box.

   The Remote Debugging Monitor is now ready to accept a connection from Visual Studio 2005 running as user Dbug on the host machine.



Remote Debugging Monitor ready for a connection from Visual Studio 2005 running as the Dbug user

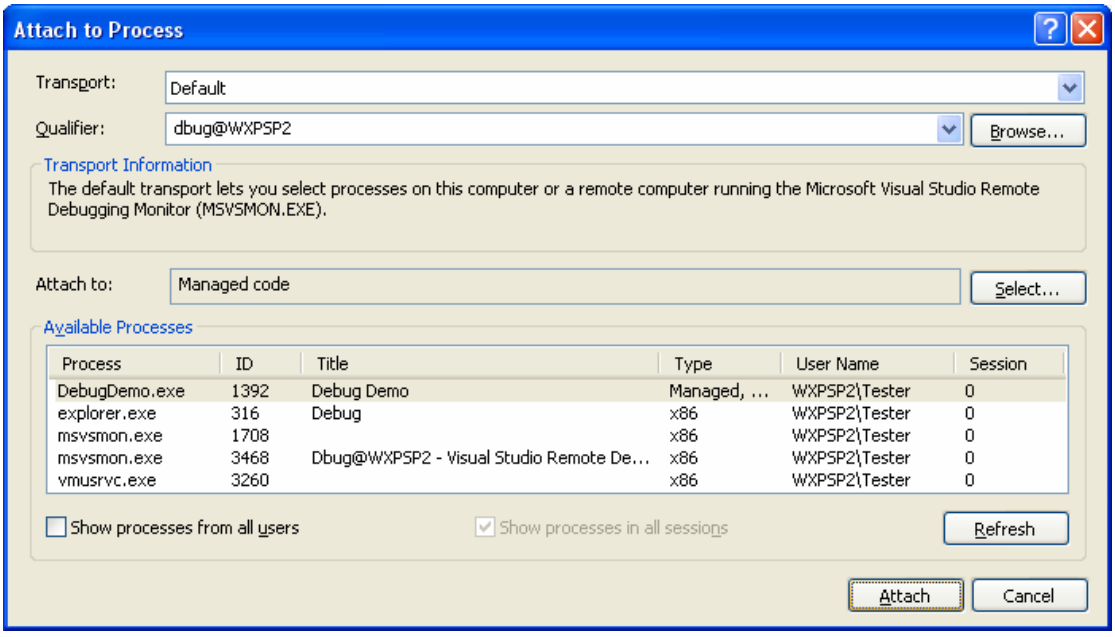
Attaching to the remote process from Visual Studio 2005

To attach to and debug the remote process:

1. Start **DebugDemo.exe** on the virtual machine running the Remote Debugging Monitor.
2. While logged on as the Dbug user on the debugger host machine, open the **DebugDemo** project in Visual Studio 2005.
3. On the **Debug** menu, click **Attach to Process**.
4. In the **Attach to Process** dialog box, click **Browse**.
5. In the **Browse for Computer** dialog box, select the virtual remote debugger target. Click **OK**.

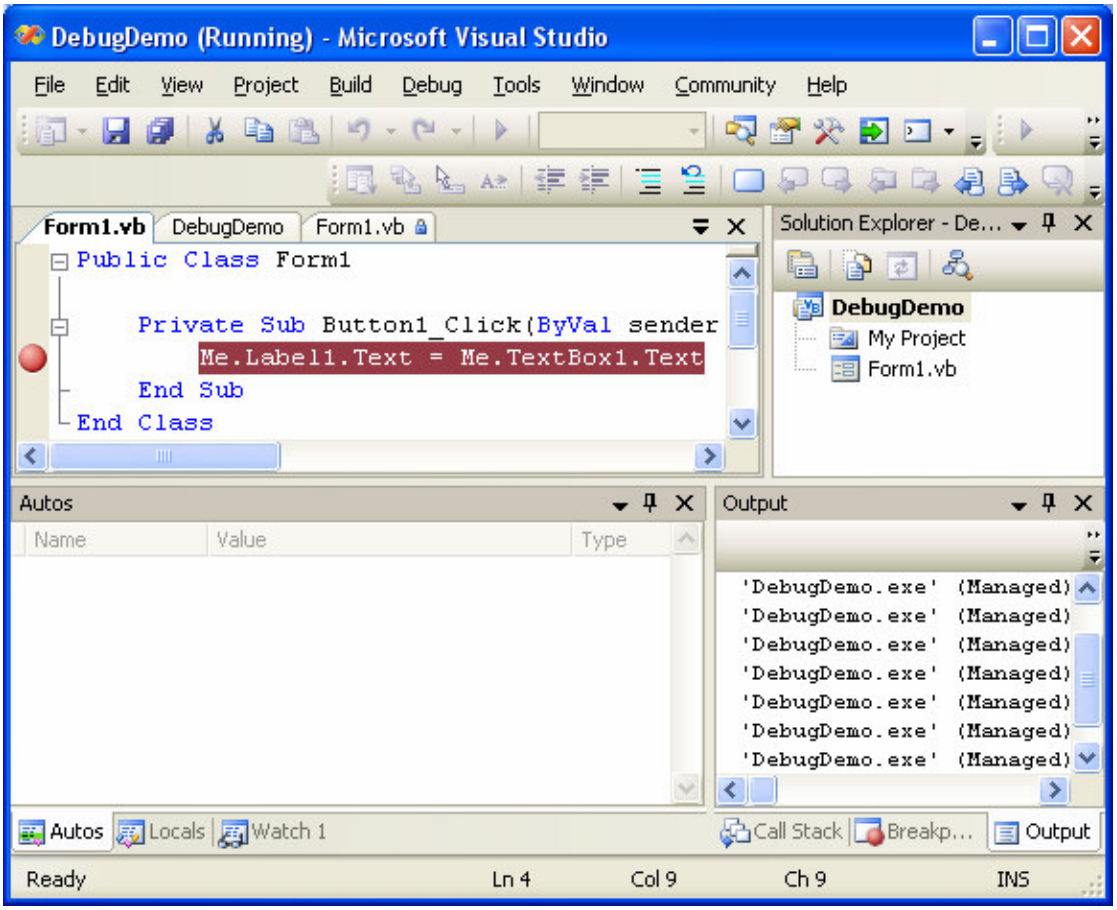
Selecting the remote debugging target machine

6. In the **Attach to Process** dialog box, under **Available Processes**, select the **DebugDemo.exe** process. Click **Attach**.



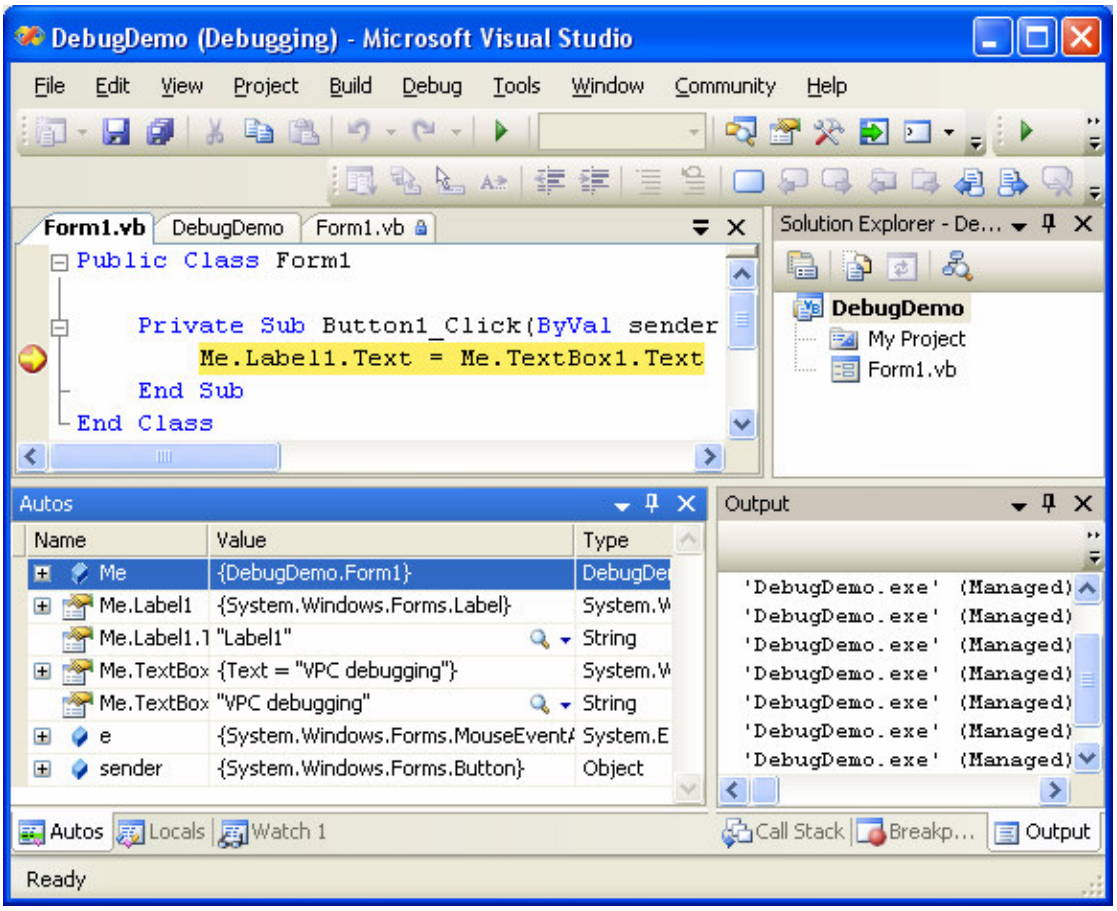
Selecting the DebugDemo.exe process

7. Within Visual Studio 2005, select the code file you want to debug — Form1.vb in this example. Open the file in Code View. Set a breakpoint as shown:



Setting a breakpoint

8. Go to the virtual machine running the application being debugged. Enter **VPC debugging** in the text box and click **Button1**. Visual Studio will break at the breakpoint and you can examine the contents of the variables just as you would when debugging a local application.



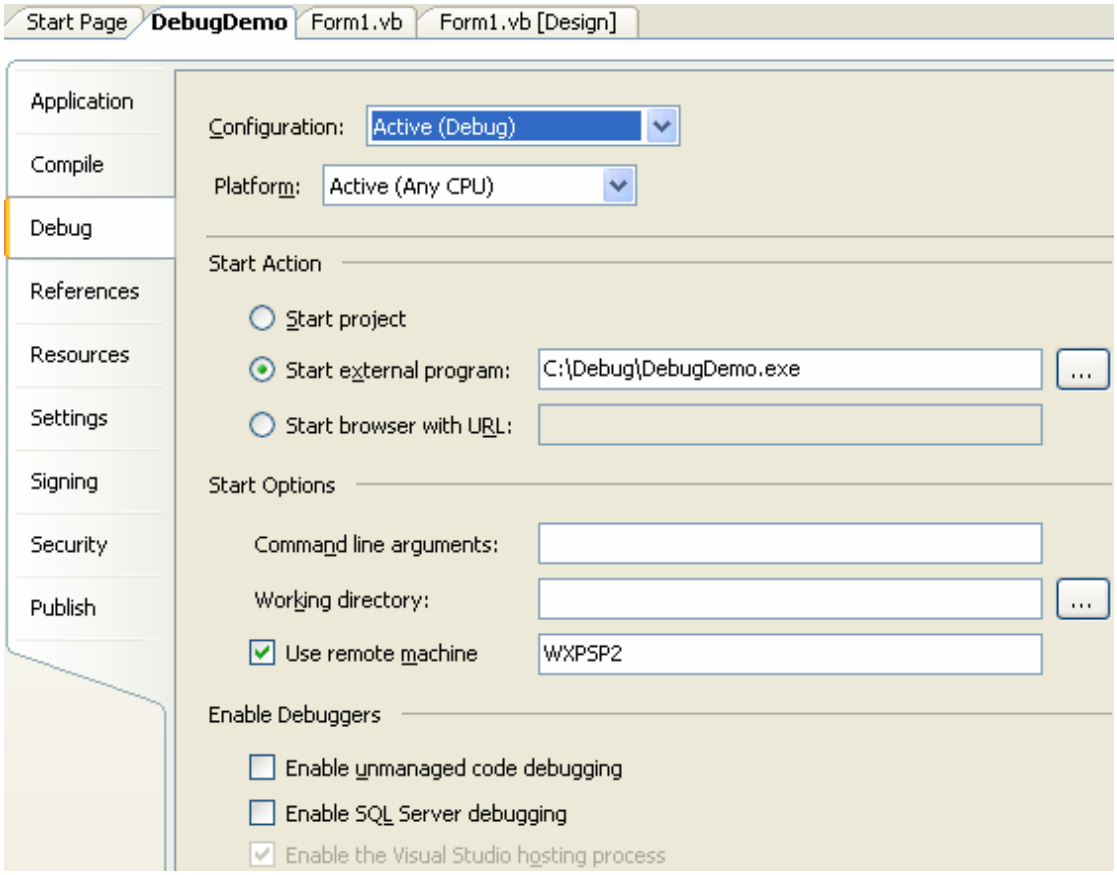
Debug output at the breakpoint

Starting a remote debugging session from Visual Studio 2005

Instead of attaching to an already running remote process, Visual Studio 2005 can start a process on a remote machine and debug it. You must use the same credentials on the debugging host and debugging target (Note: the user on the target machine does not have to have Administrator privileges). Additionally, the directory structures on both machines must match exactly. To keep things simple and directory paths short, the project's entire debug folder was copied to the root of the C drive on both the host and the target machines. Here are the steps to starting a remote debugging session on Visual Studio 2005:

1. Start the Remote Debugging Monitor on the remote virtual machine.
2. Open the **DebugDemo** project in Visual Studio 2005 on the host machine.
3. Open the **Properties** pages for the project (either right-click the project name in Solution Explorer and select Properties or go to the menu bar and select **Properties** | **DebugDemo Properties**).
4. Select the **Debug** tab on the Properties pages.

5. Under the **Start Action**, select **Start external program** and enter **C:\Debug\DebugDemo.exe**.
6. Under **Start Options**, check the checkbox for Use remote machine and enter the name of the remote machine (WXPSP2 in this example).
7. Under **Enable Debuggers**, check the checkbox for **Enable the Visual Studio hosting process**. The completed configuration should look something like this:



Configuring the project's Properties settings for starting a remote debugging session

8. Click the Continue icon (green arrow), press F5, or go to the Debug menu and select Continue to start the debugging session. After a momentary delay, the application will begin execution on the remote machine.

Using a share instead of copying files for a remote session

Starting a remote debugging session requires matching directory structures. Instead of copying folders to matching locations on both the target and host, it's more convenient to use a file share on the target machine. This eliminates the need to copy files and ensures complete consistency of content and directory structure. Keeping in mind how Code Access Security (CAS) treats network resources, it is easier to work with the share if it appears as a local resource on both machines. The **subst** command is used to map the share as a drive letter. Here are the steps to starting a remote debugging session using a share:

1. Start the Remote Debugging Monitor on the remote virtual machine.

2. Create a share on the remote target machine (this requires Administrator rights). Read privileges are sufficient, no additional privileges need to be granted. In this example, the name of the share is **Debug**.
3. On the target machine, open a Command Prompt window and enter the following command:

   ```
   C:\>subst R: \\WXPSP2\Debug
   ```

   This maps the **Debug** share on virtual machine **WXPSP2** to drive letter **R**.
4. On the host machine, open a Command Prompt window and enter the following command:

   ```
   C:\>subst R: \\WXPSP2\Debug
   ```

   This maps the **Debug** share on virtual machine **WXPSP2** to drive letter **R** on the host machine. Now both machines have exactly the same files (instead of identical copies) with exactly the same directory structures.
5. In Visual Studio 2005, go to the project's Property pages and select the **Debug** tab. Under **Start Action**, change the external program to **R:\DebugDemo.exe**.
6. Click the Continue icon (green arrow), press F5, or go to the Debug menu and select Continue to start the debugging session. After a momentary delay, the application will begin execution on the remote machine.

Conclusion

Virtual PC provides a highly configurable environment for application development. You can use Virtual PC to create both single-machine and networked virtual environments in which developers can safely try out operating system and application changes. It offers a repeatable and reproducible environment for testing. Advanced debugging techniques are simplified by using Virtual PC as a substitute for physical machines.

In this white paper, you have learned how to configure Virtual PC to create virtual environments that are both valid and complete. You have also learned how to debug processes running in Virtual PC.

0905 Part No. xxx-xxxxxx (if applicable)

For More Information

- Microsoft Virtual PC 2004 Web site
  http://www.microsoft.com/virtualpc
- Microsoft Virtual PC 2004 Deployment White you paper
  http://www.microsoft.com/windows/virtualpc/techinfo/deploy.mspx
- Virtual PC Guy's Weblog (Ben Armstrong)
  http://blogs.msdn.com/Virtual_PC_Guy
- Download Microsoft Virtual PC 2004 Free 45-Day Trial Edition
  http://go.microsoft.com/?linkid=2615769
- Download Microsoft Virtual PC 2004 Service Pack 1
  http://www.microsoft.com/windows/virtualpc/downloads/sp1.mspx
- Microsoft Knowledge Base article 311272
  The DevCon command-line utility functions as an alternative to Device Manager
  http://support.microsoft.com/?kbid=311272
- Microsoft Knowledge Base article 833144
  Overview of the technical specifications of virtual machines in Virtual PC 2004
  http://support.microsoft.com/?kbid=833144
- Microsoft Knowledge Base article 887790
  How to improve virtual machine performance in Virtual PC 2004
  http://support.microsoft.com/?kbid=887790
- Debugging Tools for Windows
  http://www.microsoft.com/whdc/devtools/debugging/debugstart.mspx
- Windows Debug Symbols
  http://www.microsoft.com/whdc/devtools/debugging/symbolpkg.mspx