Microsoft® Visual Studio®.net™

Microsoft Research
University Relations

Visual Studio .NET User Groups Use, Permissions, Security

By Jack Davis

Audience
Visual Studio .NET System Administrators.

Abstract
When Visual Studio .NET installs, two special user groups are created: "**VS Developers**" and "**Debugger Users**". This document provides an overview on the use, permissions, and security associated with Visual Studio group members for developing and debugging .NET Windows applications, web applications, and web services. Information includes configuration options for using Visual Studio from standard user accounts (non Administrator, non Power User), such as students programming Visual Studio on shared workstations in a computer learning lab.

.NET Development Server (dŏt–nĕt–dē•vĕl′ ŏp•ment–sĕrv′ er) *n.*

1) A Windows Advanced Server integrated with Visual Studio .NET, Terminal Services, Telnet Services, and academic software tools for programming and instruction use. Using a small terminal service client, developers on network workstations share access to the server to compile, debug and run .NET applications and web services using a variety of different programming languages.

2) A .NET programming playground.

CONTENTS

030129

Related Documents:

Visual Studio .NET User Groups – Use, Permissions, Security *(this document)*
Overview on the use, permissions, and security provided to Visual Studio users for developing and debugging Windows applications, ASP.NET web applications, and ASP.NET web services.
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevVSGROUPS.doc
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevVSGROUPS.pdf

.NET Development Server – Build Cookbook
Instructions for building a .NET Development Server. Step procedures for installation of Windows 2000 Advanced Server, Windows 2000 Service Pack updates, Terminal Services, and Visual Studio .NET.
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrCOOKBOOK.doc
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrCOOKBOOK.pdf

.NET Development Server – Cookbook Addendum
Instructions for installing optional .NET Development Server components including: Visual Studio .NET Academic teaching and student tools, MSDN Library, and Visual Studio 6 dual install with Visual Studio .NET.
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrADDENDUM.doc
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrADDENDUM.pdf

.NET Development Server – System Administrator Guide
IT system administrator overview; installing terminal service licenses; creating user accounts and developer work folders; access and security issues; receiving and applying software updates.
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrADMIN.doc
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrADMIN.pdf

.NET Development Server – Developer Jump Start
Developer guide – How to install remote desktop terminal service client; how to connect, log on and transfer files; stepped walk-throughs showing how to use the .NET Development Server to build, debug and run .NET console, .NET Windows, ASP.NET web service applications.
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrDEVELOPER.doc
http://msruniv.corp.bcentral.com/shared%20documents/dotNETDevSvrDEVELOPER.pdf

1 Visual Studio .NET User Groups

Q. Do developers need "Administrator" privileges to debug applications with Visual Studio .NET?

A. No. The "**Local Security Policy | User Rights Assignment**" of "**Debug programs**" assigned to **Administrators** is only needed when debugging external user and system processes not running within a local user account. For standard application development, including .NET Windows applications, web applications and web services, developers only need the capabilities provided as members of the "Debugger Users" and "VS Developers" user groups.

When Visual Studio .NET installs, two special user groups are created for development and debugging: "VS Developers" and "Debugger Users". These user groups provide the basic capabilities needed for normal application development and debugging. .NET development and debugging can be performed from standard user accounts (non Administrator, non Power User) with only the specific limited rights and permissions provided to members of the "Debugger Users" and "VS Developers" user groups. Sections 2 and 3 of this document outline the capabilities provided to members of the Debugger Users and VS Developers user groups.

For debugging ASP.NET web services, the Visual Studio debugger requires access to the ASPNET worker process, aspnet_wp.exe. By default the ASPNET worker process is configured to execute from a predefined "ASPNET" account. While developers with administrator privileges can access and debug any process on the system, to debug an ASP.NET web service, non-administrator developers need to log into the same account that runs the ASPNET worker process. The procedure for configuring the ASPNET worker process to execute from a shared user account is described in Section 4 of this document.

By default, Visual Studio is enabled with both local and remote debugging capabilities. Remote debugging is a special feature, and for standard application development is not normally required or used. Section 5 outlines the procedure to enhance security by disabling remote debugging and removing it as an external access point.

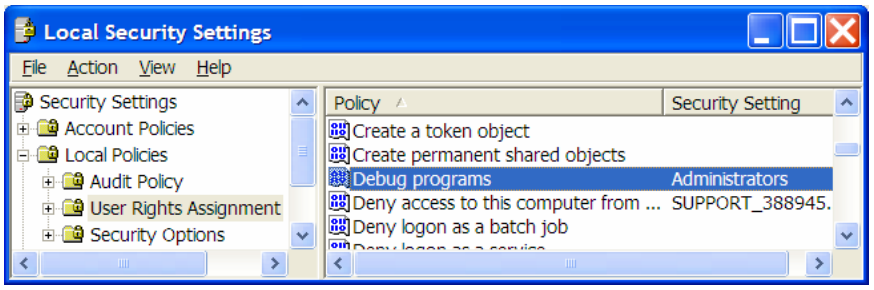
2 The "Debugger Users" User Group

Members in the **Debugger Users** user group are provided two basic capabilities: 1) permissions to communicate to the Visual Studio .NET debugging components (specifically DCOM communication to the Machine Debug Manager, MDM.exe), and 2) access to control the execution of processes within their user account. The user that installs Visual Studio .NET is included in the Debugger Users group by default. For cases where multiple developers share a workstation, such as students in a computer learning lab, each local user will need to be added as a member of the Debugger Users and VS Developers user groups. Only users that are members of the Debugger Users (and administrators) can use Visual Studio .NET for debugging.

Debugger Users permissions allow members to:

1. Launch processes within the user account (either locally or remotely).
2. Terminate processes within the user account (either locally or remotely).
3. Enumerate processes running on the system (either locally or remotely).
4. Communicate with the Machine Debug Manager (through DCOM access to MDM.exe).

Installing Visual Studio .NET does not modify the "Debug programs" security policy assigned to Administrators. The "Debug programs" security policy allows global debugging of all processes, even those running under different user or system accounts.



Policy	Security Setting
Create a token object	
Create permanent shared objects	
Debug programs	Administrators
Deny access to this computer from ...	SUPPORT_388945.
Deny logon as a batch job	

"Debug programs" permission is only needed for administrator-level developers to perform advanced system debugging. The "Debug programs" policy is not required for Visual Studio developers debugging standard Windows applications, web applications, or ASP.NET web services.

Non-administrator members of the Debugger Users group **cannot**:

1. Launch processes external to their user account.
2. Terminate processes external to their user account.
3. Modify or debug processes external to their user account.

2.1 Manually Creating the "Debugger Users" Group

In the event the Debugger Users group is deleted, or when building a custom Visual Studio .NET installation procedure with third-party installation tools or scripts; the Debugger Users group can be manually created by executing the following commands:

```
<path_to_mdm>\mdm.exe /regserver
<path_to_vs7jit>\vs7jit.exe /regserver
```

3 The "VS Developers" User Group

The Visual Studio .NET installation also creates a second user group called "VS Developers". This group provides the following permissions for development:

1. Read, Write, Delete and Change access to the wwwroot$ share (created on the server where the Visual Studio .NET server components are installed).
2. Read, Write, Delete and Change access to \wwwroot file folder (NTFS only).
3. Operator (administrative) access to the IIS Metabase.

The VS Developers user group provides limited permissions that allow editing and building web projects, and for the Visual Studio IDE system to set properties within the IIS server's web root.

4 Debugging ASP.NET Web Applications and Services

The following information pertains only when debugging ASP.NET web applications and services. It is not required for debugging Windows ,NET applications.

To debug an ASP.NET web application or web service the Visual Studio debugger needs to be able to attach to the ASPNET Worker Process, aspnet_wp.exe. The ASPNET Worker Process normally runs in a predefined "ASPNET" account created when .NET Framework is installed with Internet Information Server (IIS). While developers with administrator privileges can access and debug any process on the system, non-administrator developers need to log into the same account that runs the ASPNET worker process when debugging an ASP.NET web service. The following procedure outlines the steps needed to redirect the ASPNET Worker Process to execute within a shared "ASPUSER" account that non-administrator developers can log into when debugging an ASP.NET web service.

NOTE – A single IIS server only supports a single debug session at a time. If multiple developers are sharing the same IIS server, only one developer at a time will be able to debug an ASP.NET web service. This applies principally when developing with Visual Studio on a shared server, such as a .NET Development Server with multiple users connecting remotely by Terminal Services or Telnet services. For developers using workstations with both Visual Studio .NET and Internet Information Server (IIS) installed locally, each workstation provides its own independent IIS server for debugging.

Set "machine.config" Configuration Options

1. From the Administrator desktop, start Windows Explorer and locate the file "`C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705\CONFIG\machine.config`". Duplicate[1] the file and rename the copy "`machine.config-original`".

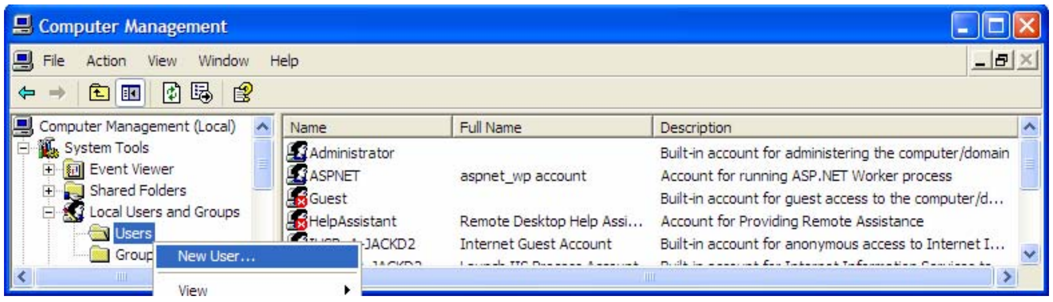
2. Using the Notepad text editor, open "**`machine.config`**" and from the menu click **Edit**, and then click **Find…**. Using the Find dialog, search for the string "<**processModel**" (be sure to include the starting less-than '<' sign). Once you have located the "`<processModel`" statement, click **Cancel** to close the "Find" dialog.

3. Reading along the `processModel` statement you will see the default settings of `userName="machine"` and `password="AutoGenerate"`. Change the userName and password settings to: **`userName="ASPUSER"`** and **`password="ASPNET_user"`** (casing matters). Once these changes have been made, click **File**, then click **Save** to save the edited machine.config file. Click **File**, and then **Exit** to quit Notepad.

   (With the processModel userName set to "`ASPUSER`", the aspnet_wp.exe process will run under the account "ASPUSER" which can be used by developers when debugging ASP.NET web services.)

Set up the ASPUSER Account

4. From the system desktop, click **Start**, click **Control Panel**, double-click **Administrative Tools**, double-click **Computer Management**, click-expand **Local Users and Groups**, right-click **Users**, and then click **New User…** to open the "New User" dialog.

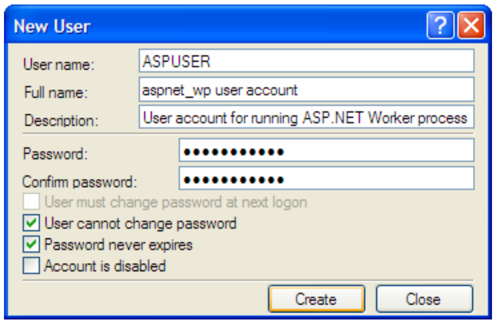


Name	Full Name	Description
Administrator		Built-in account for administering the computer/domain
ASPNET	aspnet_wp account	Account for running ASP.NET Worker process
Guest		Built-in account for guest access to the computer/d…
HelpAssistant	Remote Desktop Help Assi…	Account for Providing Remote Assistance
[illegible]-JACKD2	Internet Guest Account	Built-in account for anonymous access to Internet I…

5. On the "New User" dialog, for "User name" type "**ASPUSER**", for "Password" type "**ASPNET_user**" (casing matters), and then for "Confirm password" type "**ASPNET_user**" (casing matters). (This password must be the same as the processModel `password=` setting entered in the machine.config file, above.) Click to uncheck the **User must change password at next logon** checkbox, click to set the **User cannot change password** checkbox, click to set the **Password never expires** checkbox, click **Create**, and then click **Close**.

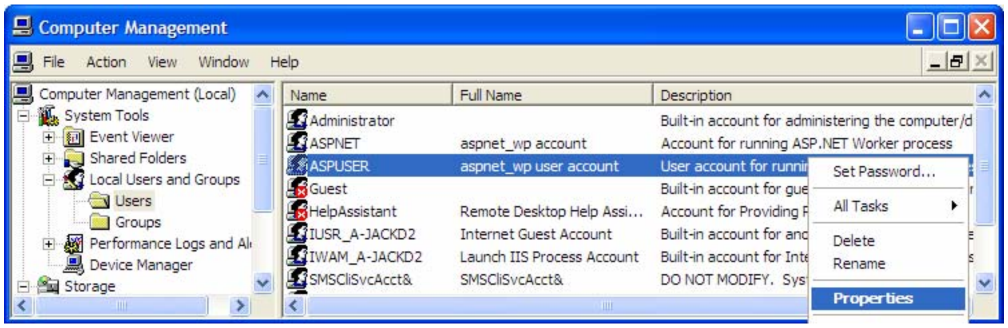
   SECURITY – Since the ASPNET worker process is configured to run under the ASPUSER account and password, users cannot be allowed to change the password. For security, the workstation system administrator should manually change the ASPUSER password periodically in both the machine.config file (step 3 above) and in the ASPUSER account. Both the machine.config "processModel" and ASPUSER passwords must always match.

[1] To make a duplicate file copy, click on the file name, then press **Ctrl-C** followed by **Ctrl-V** (this will create a duplicate file called "copy of machine.config"). Right-click on the duplicate file and rename it "**machine.config-original**".



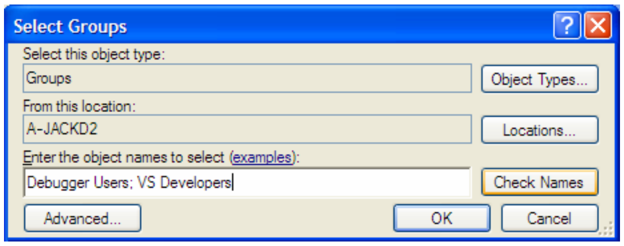
User name: ASPUSER
Full name: aspnet_wp user account
Description: User account for running ASP.NET Worker process
Password: •••••••••••
Confirm password: •••••••••••
☐ User must change password at next logon
☑ User cannot change password
☑ Password never expires
☐ Account is disabled

6. In the left pane of the “Computer Management” window, click **Users** to open the “Users” folder. In the right pane of the “Computer Management” window, right-click on the **ASPUSER** account name, and then click **Properties** to open the “ASPUSER Properties” dialog.



Name	Full Name	Description
Administrator		Built-in account for administering the computer/d
ASPNET	aspnet_wp account	Account for running ASP.NET Worker process
ASPUSER	aspnet_wp user account	User account for runni
Guest		Built-in account for gue
HelpAssistant	Remote Desktop Help Assi...	Account for Providing R
IUSR_A-JACKD2	Internet Guest Account	Built-in account for and
IWAM_A-JACKD2	Launch IIS Process Account	Built-in account for Inte
SMSCliSvcAcct&	SMSCliSvcAcct&	DO NOT MODIFY. Sys

7. On the “ASPUSER Properties” dialog, click the **Member Of** tab. On the “Member of” dialog, click **Add...**, on the “Select Groups” dialog enter “**Debugger Users; VS Developers**”, click **Check Names**, click **OK**. Click **OK** to close the “ASPUSER Properties” dialog.



Select this object type:
Groups
From this location:
A-JACKD2
Enter the object names to select (examples):
Debugger Users; VS Developers

8. In the “Computer Management” window, right-click in the title bar, and then click **Close**.

Enable ASPUSER Local Log on

9. From the system desktop click **Start**, click **Settings**, click **Control Panel**, double-click **Administrative Tools**, double-click **Local Security Policy**, click-expand **Local Policies**, then click **User Rights Assignment**. This brings up the “Local Security Settings” dialog.
10. From the Local Security Settings window, double-click **Log on locally**, click **Add...**, locate and click **ASPUSER**, click **Add**, click **Ok**, and then click **Ok** again.
11. From the Local Security Settings window, double-click **Log on as batch job** and make sure ASPUSER is included in the “Assigned To” list (if not, click **Add...** and add **ASPUSER** to the list). Click **Ok** to close the Local Security Policy Setting window.
12. Close the “Local Security Settings” and “Administrative Tools” windows.

Verify ASPUSER Debug Operation

13. Log out from “Administrator”, “Restart” the server to insure the updated settings are loaded, then log back in under the account “**ASPUSER**” (all uppercase) and password “**aspuser**” (all lowercase).
14. Using Notepad, enter the following two line C# ASP.NET web service program:

```
<%@ page language="C#" debug="true" %>
ASP.NET says: <%= " Hel" + "lo!" %>
```

Save this file as `D:\VSDev\test.aspx`. Also make sure that `D:\VSDev` is enabled as a "web shared" virtual directory[2]).

15. Start Internet Explorer, and enter the URL address "`http://localhost/VSDEV/test.aspx`" followed by **Enter.** The web service application will compile automatically and the following text should display on the web page:

    ASP.NET says: Hello!

16. Using Windows Explorer, navigate to the "`C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\GuiDebug\`" directory, and then double-click on `DbgCLR.exe` to start the CLR Debugger. From the CLR Debugger main menu, click **Tools**, and then click **Debug Processes….** In the "Processes" dialog make sure that "**Show system processes**" and "**Show processes in all sessions**" are both checked. In the "Available Processes" window select "**aspnet_wp.exe**", and then click **Attach….** "aspnet_wp.exe" should now show up in the lower "Debugged Processes" window. If the aspnet_wp.exe process successfully attaches to the debugger, then the .NET Server configuration options are properly set.

    To exit the CLR Debugger, in the "Processes" dialog click **Close**, click **File**, and then click **Exit.** Click **Yes** to the "Do you want to stop debugging?" message, and **No** to "Save changes to the following items".

Congratulations! You have successfully configured and verified that developers can run and debug ASP.NET web services from the ASPUSER account. (Remember that only one user will be able to log in and debug at a time.)

Additional information on configuring ASP.NET development and debugging is available in the following on-line documents:

ASP.NET Process Identity
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconApplicationIdentity.asp

Security Concerns of Visual Basic .NET and Visual C# .NET Programmers
Sections: **ASPNET Process Identity, Securing File Resources, and Debugging with ASPNET Identity**
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vbtchSecurityConcernsForVisualBasicNETProgrammers.asp

5 Securing Remote Debug Access

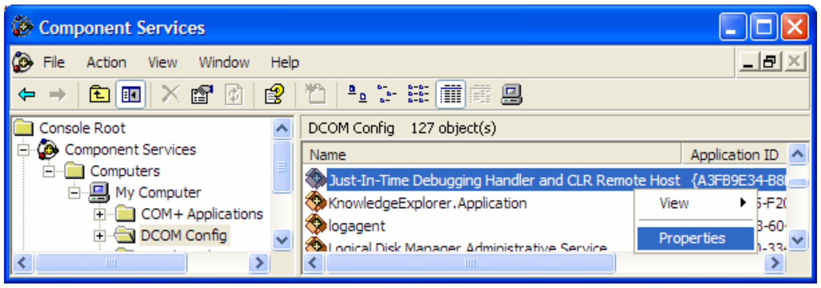
Remote debugging is a special facility typically used for debugging full-screen applications where the operation of a debug window interferes with the operation of the program running full-screen, or for debugging standalone systems where there is no keyboard or display available for local debugging.

When Visual Studio is installed both local and remote debugging are enabled by default. For standard application development, however, remote debugging is not normally required or used. To enhance security the following steps outline the procedure to disable remote debugging and remove it as an external access point.

1. From the administrator desktop, click **Start**, and then click **Run….** In the "Run" dialog, type "**dcomcnfg.exe**", and then click **OK** to open Component Services.

[2] To set `D:\VSDev` as a web shared directory, using Windows Explorer right-click on the **`D:\VSDev`** folder, click **Properties**, then click **Web Sharing**. "Share on:" *{workstation or server computer name}* should be set, and "Share this folder" should be selected with the alias "VSDev" (default Access permissions "Read" and Application permissions "Scripts" should also be set).

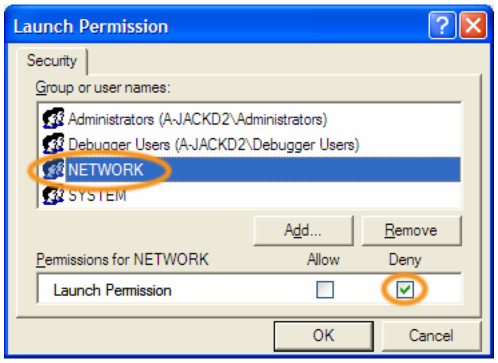
2. In the right pane of the "Component Services" window, click-expand **Component Services**, click-expand **Computers**, click-expand **My Computer**, and then click **DCOM Config**.
3. Under "DCOM Config", locate and right-click on **Just-In-Time Debugging Handler and CLR Remote Host**, and then click **Properties**.



DCOM Config 127 object(s)

Name	Application ID
Just-In-Time Debugging Handler and CLR Remote Host	{A3FB9E34-B8
KnowledgeExplorer.Application	
logagent	
Logical Disk Manager Administrative Service	

4. On the "Just-In Time Debugging Handler and CLR Remote Host Properties" dialog, click the **Security** tab. Under "Launch Permissions", click **Edit…**. On the "Launch Permission" dialog, click **Add…**, for the object name type "**NETWORK**", click **OK** (if a "Multiple Names Found" dialog appears click **OK** with the selected default "Network" name), set the NETWORK launch permission to **Deny**, and then click **OK** to close the "Launch Permission" dialog.

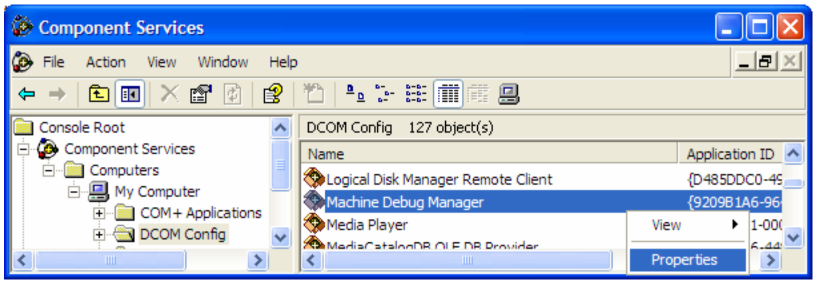


Group or user names:

- Administrators (A-JACKD2\Administrators)
- Debugger Users (A-JACKD2\Debugger Users)
- NETWORK
- SYSTEM

Permissions for NETWORK	Allow	Deny
Launch Permission	☐	✓

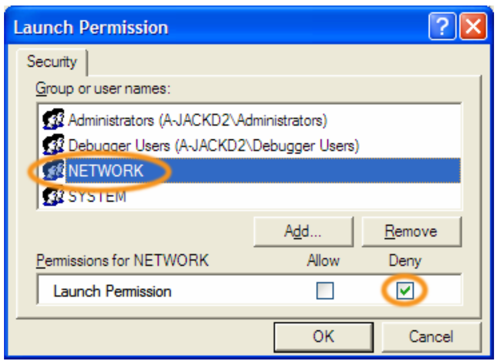
5. On the "Just-In Time Debugging Handler and CLR Remote Host Properties" dialog Security tab under "Access Permissions", click **Edit…**. On the "Access Permission" dialog, click **Add…**, for the object name type "**NETWORK**", click **OK** (if a "Multiple Names Found" dialog appears click **OK** with the selected default "Network" only name), set the NETWORK launch permission to **Deny**, and then click **OK** to close the "Access Permission" dialog. Click **OK** to close the "Just-In Time Debugging Handler and CLR Remote Host" dialog.
6. Under "DCOM Config", locate and right-click on **Machine Debug Manager**, and then click **Properties**.



DCOM Config 127 object(s)

Name	Application ID
Logical Disk Manager Remote Client	{D485DDC0-4
Machine Debug Manager	{9209B1A6-96
Media Player	
MediaCatalogDB OLE DB Provider	

7. On the "Machine Debug Manager Properties" dialog, click the **Security** tab. Under "Launch Permissions", click **Edit…**. On the "Launch Permission" dialog, click **Add…**, for the object name type "**NETWORK**", click **OK** (if a "Multiple Names Found" dialog appears click **OK** with the selected default "Network" name), set the NETWORK "Launch Permission" to **Deny**, and then click **OK** to close the "Launch Permission" dialog.



Administrators (A-JACKD2\Administrators)
Debugger Users (A-JACKD2\Debugger Users)
NETWORK
SYSTEM

Permissions for NETWORK	Allow	Deny
Launch Permission		✓

8. On the “Machine Debug Manager Properties” **Security** tab under “Access Permissions” click **Edit….** On the “Access Permission” dialog, click **Add…**, for the object name type “**NETWORK**”, click **OK** (if a “Multiple Names Found” dialog appears click **OK** with the selected default “Network” only name), set the NETWORK launch permission to **Deny**, and then click **OK** to close the “Access Permission” dialog. Click **OK** to close the “Machine Debug Manager Properties” dialog.

Remote debug access to the system is now disabled.

-End-