RTX MiniTutorial

REMOTE DEBUGGING WITH 64-BIT WINDOWS HOST AND 32-BIT TARGET

RTX 2012

IntervalZero

RTX

MiniTutorial: Remote Debugging with 64-bit Windows Host and 32-bit Target	IZ-DOC-X86-0058-R3	July 2015

IntervalZero

400 Fifth Avenue
Fourth Floor
Waltham, MA 02451
Phone: 781-996-4481

www.intervalzero.com

Overview

64-bit Windows 7-based PCs are becoming the standard development platform. With this in mind, IntervalZero designed the RTX 2012 SDK to install on a 64-bit Windows operating system.

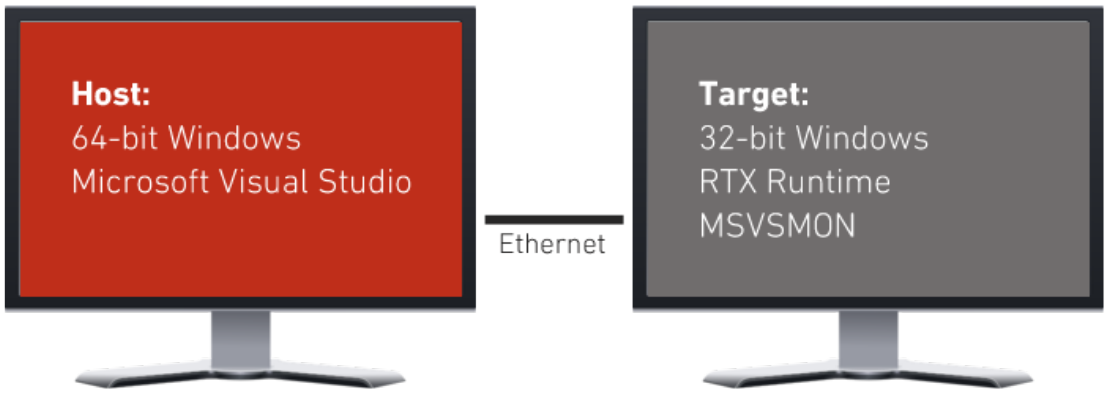
Although the IntervalZero RTX runtime currently only supports 32-bit Windows, you can develop and debug your RTX 2012 applications with a 64-bit Windows platform. This can be accomplished through a Host / Target debug scenario. This allows you to:

- Build and Debug using any Windows based 64-bit Host
- Remotely connect to any Windows based 32-bit Target
- Use Ethernet between the Host and Target

NOTE: The instructions in this MiniTutorial correspond with RTX 2012.

System Requirements

Host Requirements	Target Requirements	Shared Requirements
Windows 64-bit Operating System	Windows 32-bit Operating System	Ethernet connection between the Host and Target
Microsoft Visual Studio 2010, 2008, or 2005	RTX 2012 Runtime	
	Microsoft Remote Debugging Monitor (MSVSMON.exe)	



Splitting a Bundled Activation Key

A bundled activation key for RTX combines multiple activation keys for different RTX components. For instance, a bundled activation key may include activation keys for the RTX SDK, RTX Runtime, and RTX RT-TCP/IP, as shown in this example:

```
RTX-110-1234-1234-1234-1234-TCP-110-1234-1234-1234-1234-SDK-110-1234-1234-1234-1234
```

You can split a bundled activation key into separate keys in order to activate the RTX SDK (headers, libraries, and wizards) on the Host machine, and activate the RTX Runtime on the Target machine.

You will want to break the bundled key so that one key includes the RTX Runtime and RT-TCP/IP activation, and the other includes just the RTX SDK activation. Using the example above, you would split the bundled key as follows:

- Runtime key: `RTX-110-1234-1234-1234-1234-TCP-110-1234-1234-1234-1234`
- SDK key: `SDK-110-1234-1234-1234-1234`

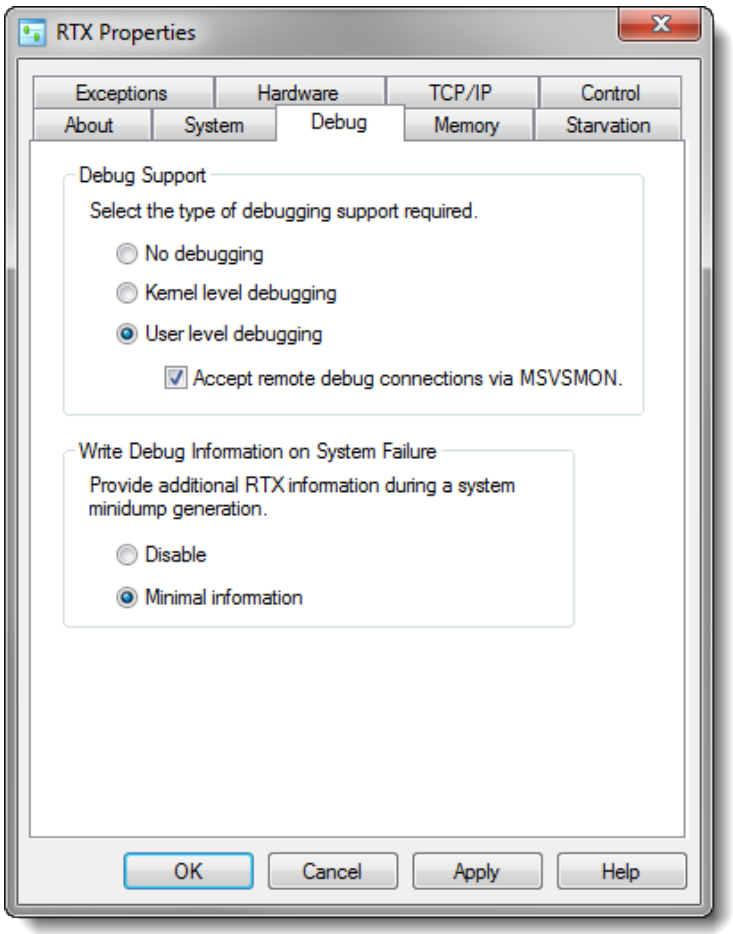
Setup

Target Setup

Setup of the Target system requires the following steps.

Configure RTX remote debug support:

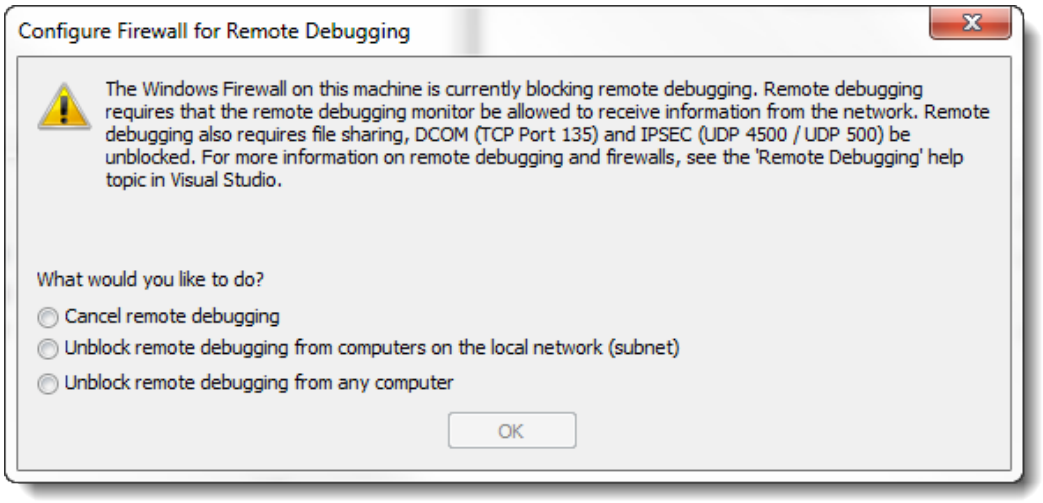
1. Click **Start > All Programs > IntervalZero > RTX 2012 > RTX Properties**.
2. On the **Debug** tab, click the **User level debugging** radio button, and then check the **Accept remote debug connections via MSVSMON** box.



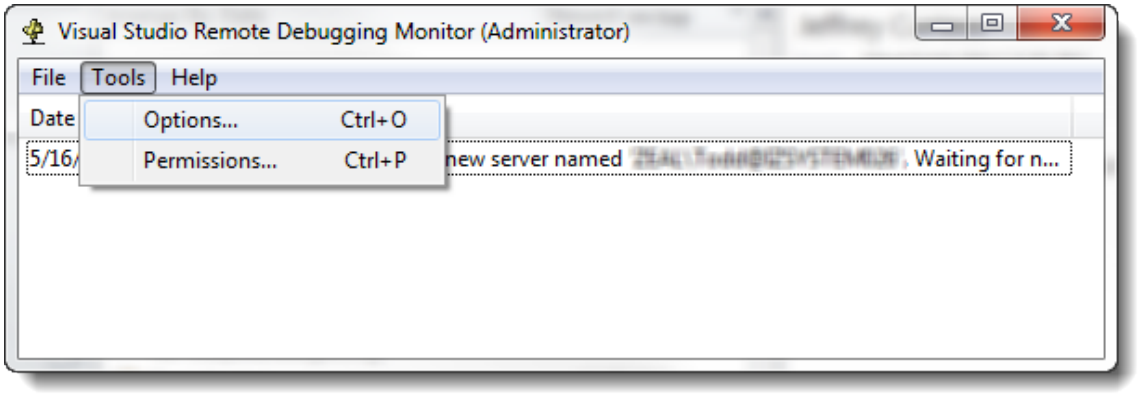
3. Click **Apply**.
4. On the **Control** tab, ensure that the RTX subsystem is running. If RTX Subsystem drivers are not started, click the **Start** button.

Start MSVSMON:

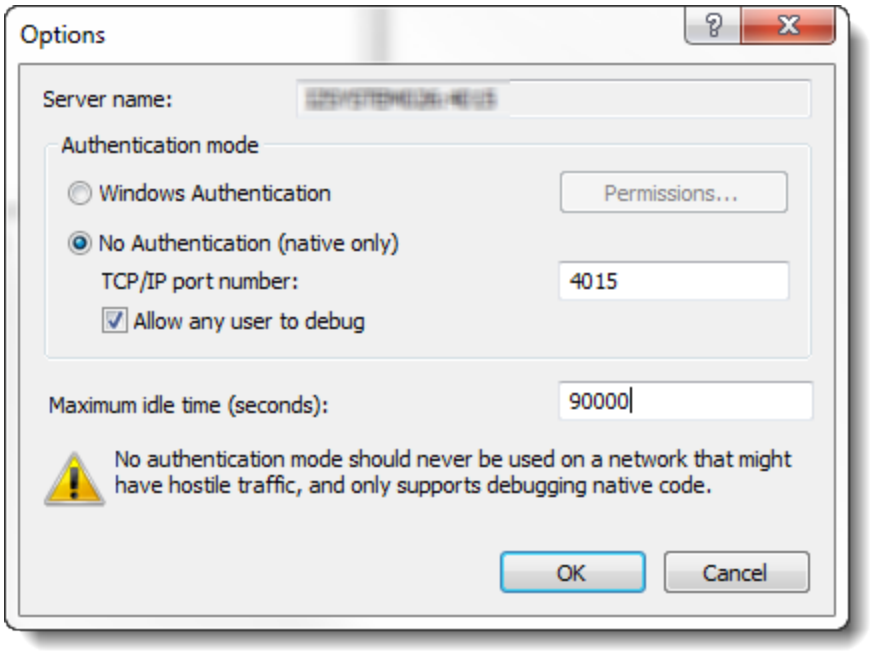
1. On the Host machine, locate the Visual Studio installation directory (`C:\Program Files\Microsoft Visual Studio <version>` by default) and navigate to `\Common7\IDE\Remote Debugger\x86`.
2. Copy the x86 directory to the Target machine.
3. On the target system, in the x86 folder, double-click the file `msvsmon.exe`.
4. If Windows Firewall is enabled on the Target machine, the **Configure Firewall for Remote Debugging** prompt appears.



5. Choose one of the following:
   - Unblock remote debugging from computers on the local network (subnet)
   - Unblock remote debugging from any computer
6. Click **OK**.
7. In the Visual Studio Remote Debugging window, click **Tools > Options**.



The **Options** dialog appears:



8. Do the following:
    a. Copy the server name.
    b. Click **No Authentication (native only)** to allow for an open TCP connection.
    c. Click **Allow any user to debug**.
    d. Increase the **Maximum idle time** as needed.
9. Click **OK**.

Run the RTSS application:

1. Launch the Command Prompt as Administrator.
2. Type `rtssrun /y` followed by the full path to the application. If the path name contains spaces, it must be enclosed in quotes. For example:

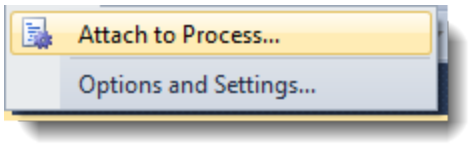
```
C:\>rtssrun /y "C:\Users\Todd\Documents\Visual Studio 2010\Projects\test.rtss"
```

NOTE: The `/y` flag is required to run the application so Visual Studio can attach to it.

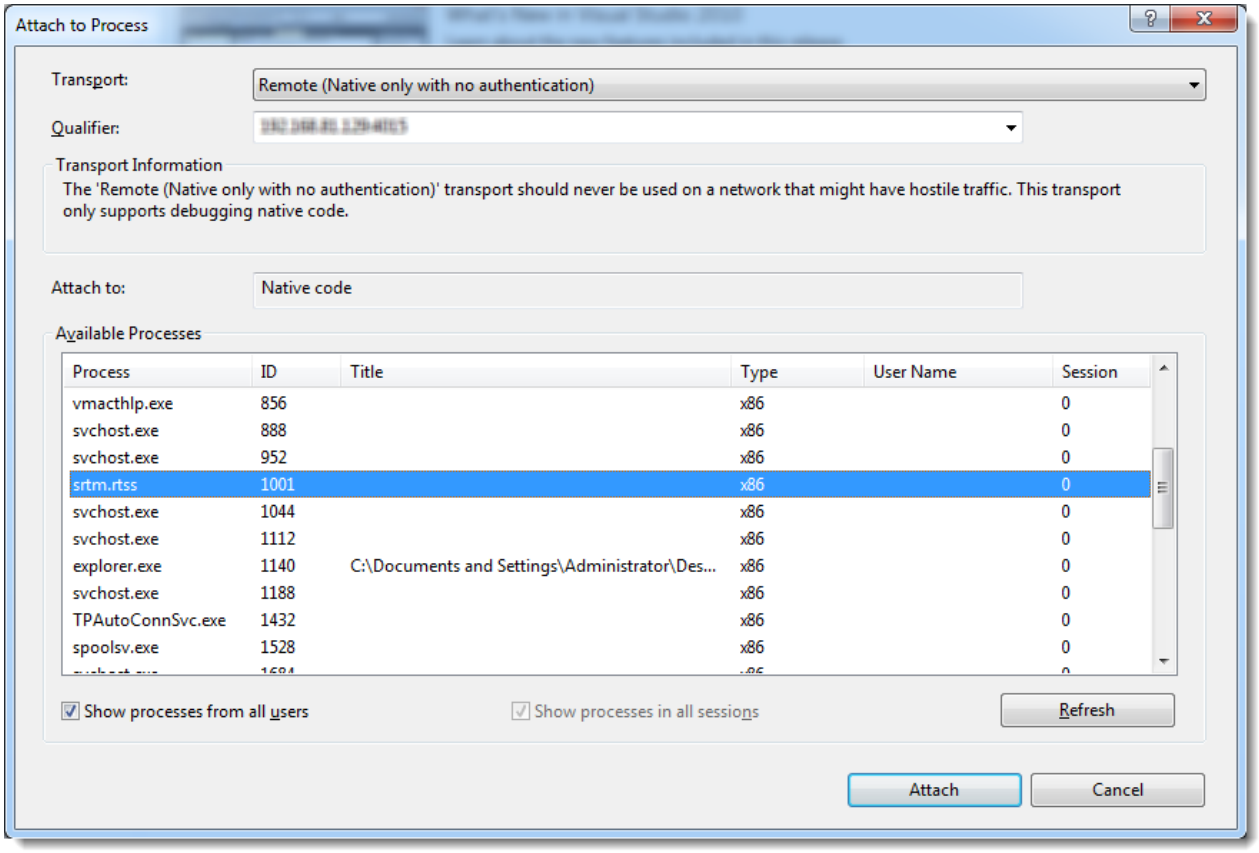
Host Setup

Start the Visual Studio session:

1. Open the solution in Visual Studio.
2. In the **Debug** menu, click **Attach to Process**.



3. The **Attach to Process** dialog appears. Configure the Attach settings as follows:
   - **Transport** – Set to **Remote (Native only with no authentication)**
   - **Qualifier** – Enter the Server Name you copied during MSVSMON setup (**Step 8a** under *Start MSVSMON* above)
   - Enable **Show processes from all users**
4. Click **Refresh**.
5. Under **Available Processes**, select the process to attach to.



6. Click **Attach**.

Connection Troubleshooting

If there are any problems connecting the host and target systems, check to ensure:

- The Server Name in the Attach to Process window matches the Server Name in the MSVSMON options on the target system
- The RTX subsystem has been started on the target system

NOTE: While remote debugging, if the connection is lost, Visual Studio may close and restart.

Debugging

After you have configured the host and target systems and attached to the process, you can debug your application as you normally would within Visual Studio.

Resources

For more information on RTX, visit the IntervalZero website at http://www.intervalzero.com/

For information on MSVSMON setup, see the MSDN article *How to: Set Up Remote Debugging* at http://msdn.microsoft.com/en-us/library/bt727f1t.aspx