

# IntervalZero

---

## **MiniTutorial: RTX 2011 SDK with 64-bit Windows and 32-bit Target**

Copyright © 1996-2011 by IntervalZero Inc. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means, graphic, electronic, or mechanical, including photocopying, and recording or by any information storage or retrieval system without the prior written permission of IntervalZero Inc., unless such copying is expressly permitted by federal copyright law.

While every effort has been made to ensure the accuracy and completeness of all information in this document, IntervalZero Inc. assumes no liability to any party for any loss or damage caused by errors or omissions or by statements of any kind in this document, its updates, supplements, or special editions, whether such errors, omissions, or statements result from negligence, accident, or any other cause. IntervalZero Inc. further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. IntervalZero Inc. disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, including implied warranties of merchantability or fitness for a particular purpose.

IntervalZero Inc. reserves the right to make changes to this document or to the products described herein without further notice.

IntervalZero RTX is a trademark of IntervalZero, Inc.

Microsoft, MS, and Win32 are registered trademarks and Windows 7, Windows XP, Windows 2000, Windows Server 2003, and Windows NT are trademarks of Microsoft Corporation.

All other companies and product names may be trademarks or registered trademarks of their respective holders.

MiniTutorial: RTX 2011 SDK with 64-bit Windows and 32-bit Target

Document Number: DOC-RTX-019

March 2011

# Developing RTX 2011 applications with 64-bit Windows

## Overview

64-bit Windows 7-based PCs are becoming the standard development platform. With this in mind, IntervalZero designed its recently released RTX 2011 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 2011 applications with a 64-bit Windows platform. This can be accomplished through a Host / Target debug scenario (see the diagram on the following page). Local debug on 64-bit Windows will be available in the IntervalZero RTX64 product, available early in 2012.

The following tutorial describes the steps required to develop a 32-bit RTX application with a 64-bit Windows 7 development system. The process will take about 15 minutes and when completed you will be able 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

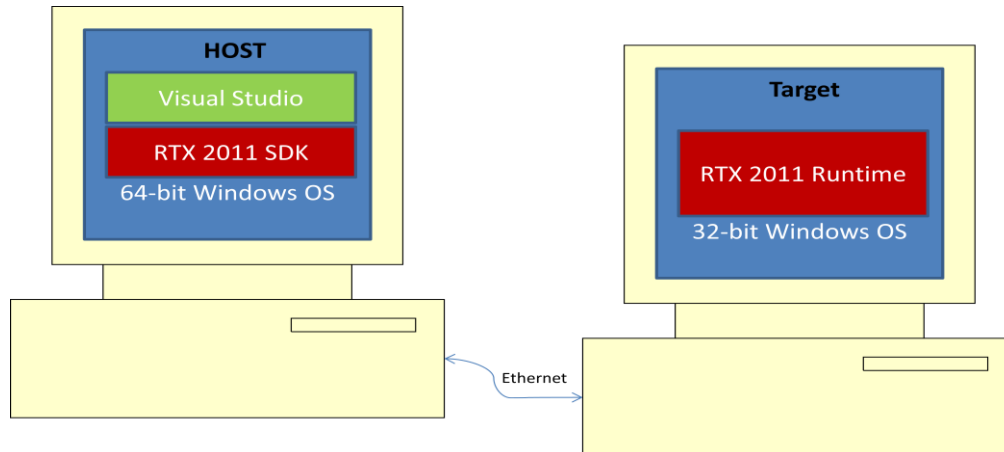
## System Requirements

1. 1 x Host PC with Windows 64-bit Operating System
2. 1 x Target PC with RTX supported Windows 32-bit Operating System
3. RTX 2011 SDK with PAC key (Evaluation version of RTX 2011 SDK will not work as the Runtime is required to be installed)
4. RTX 2011 Runtime with PAC key (\*\*Contact sales for a runtime PAC key).

**NOTE: Only full versions should be used. Evaluation versions will not work in this configuration.**

5. Ethernet connections between host and target.

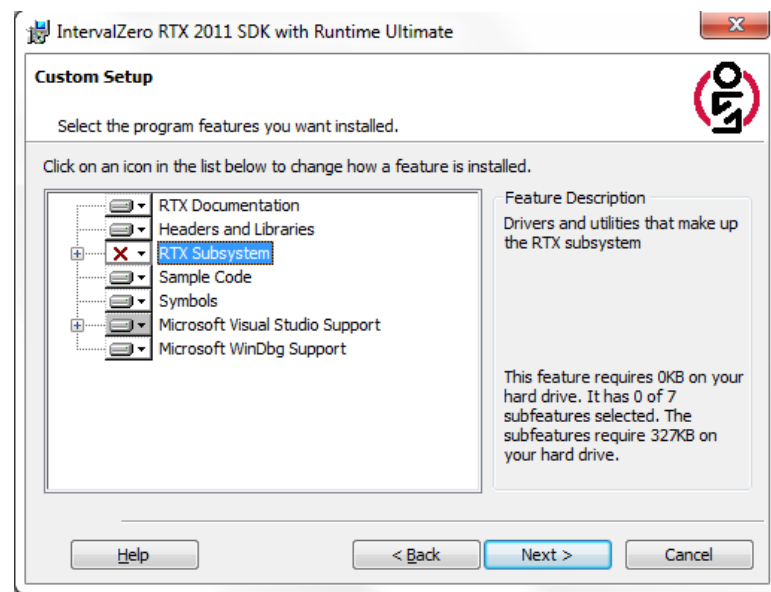
## Host/Target Diagram



## Installation

1. **Host:** Install RTX SDK on the Windows 64-bit host following the standard installation prompts. Enter the SDK PAC key when prompted.

**NOTE: Because the RTX Runtime requires a 32-bit OS, you must deselect the “RTX Subsystem” option from the custom setup options.**



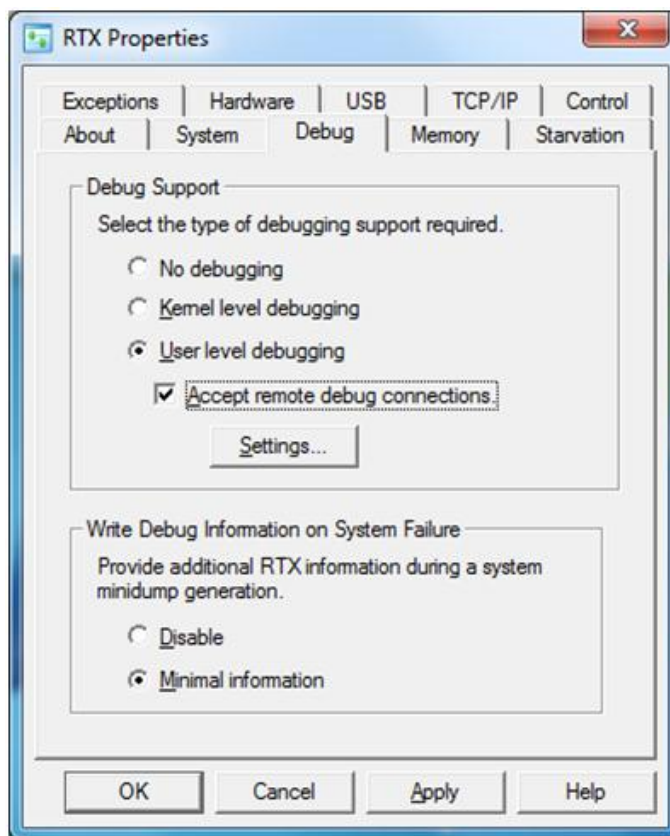
2. **Target:** Install RTX Runtime on the Windows 32-bit target following the standard installation prompts. Enter the Runtime PAC key when prompted. During the installation process, make sure that the **Target Debugging Support** feature is selected.

## Target Setup

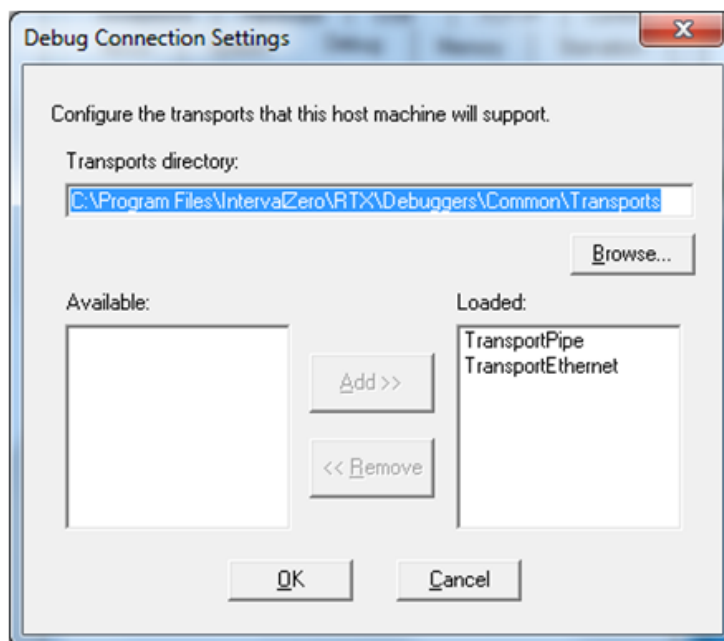
### Configuring RTX Debug Target Properties

To configure the RTX Runtime target, use the RTX Properties control panel as follows.

1. Click **Start > All Programs > IntervalZero > RTX 2011 > RTX Properties**.
2. On the **Debug** tab, click the **User level debugging** radio button, check the **Accept remote debug connections** box and then click the **Settings** button.



3. In the **Debug Connection Settings** window, make sure the **Transports Directory** is set to the location of the transport DLLs. Keep the default location, unless you moved the transport directory after you installed RTX.



4. If required, **Add** or **Remove** transport types for the debugger to load.

**NOTES:**

- **TransportPipe** is used only for local debugging; pipes are not supported for remote debugging.
- For more information on writing a custom debug transport DLL, “About the RTX Debug Transport Interface” in the *RTX SDK Reference Manual*.

5. Click **Apply**, **OK**, or **Cancel**.
6. Copy the binaries (`.rtss` and/or `.rtdll`) for the application to be debugged into an execution directory on the target system (for example `C:\targetdebug\test1`). If an RTDLL is used, it must be registered with the target system using `rtssrun /d`, so that it can be found by the debugging `.rtss` program.

**TIP:** Make a note of the directory that binaries were copied to on the target, so that this location can be entered into the **Host-Target Connection** window during setup of the host system.

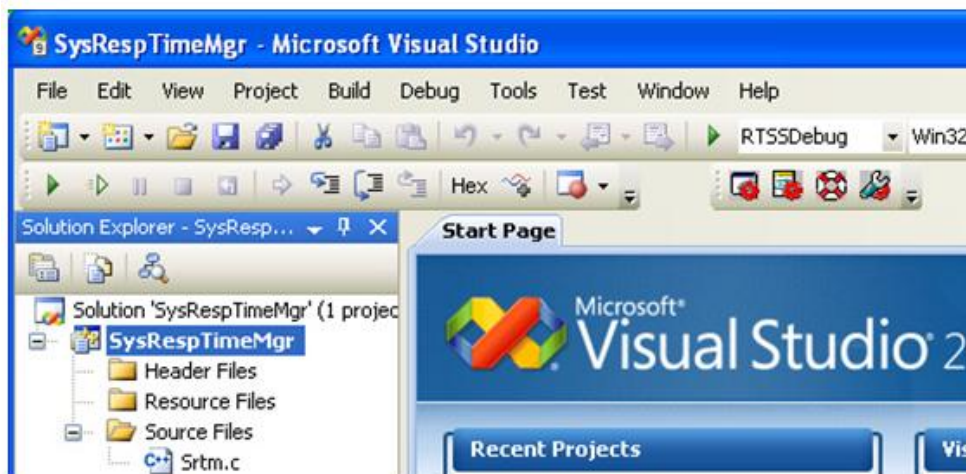
7. Ensure that the RTX subsystem is running on the target system by checking the **Control** tab in the **RTX Properties** control panel. If RTX Subsystem drivers are not started, click the **Start** button.

**NOTE: Each time the application program is changed or rebuilt; new binary files must be copied over to the target system. Any RTDLLs that are used must be reregistered.**


## Host Setup

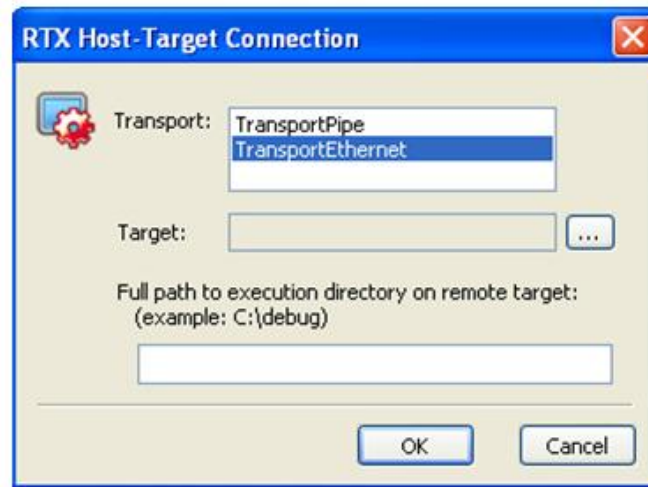
### Start the Visual Studio Session

Start Visual Studio and open the project containing the debug source code files.

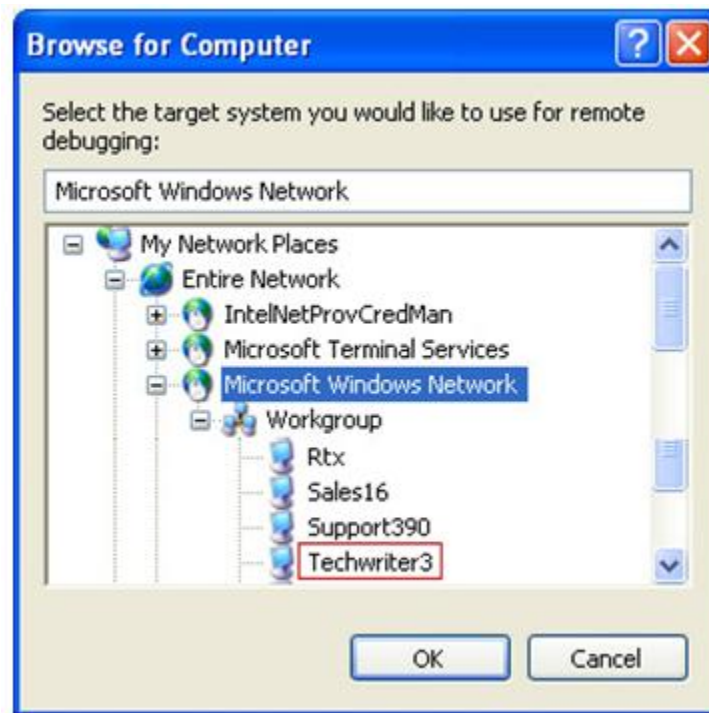


### Specifying the Host-Target Connection

1. While in Visual Studio, click **RTX Debug Configuration**  in the **RTX Toolbar**. The **RTX Host-Target Connection window** appears.
2. Select **TransportEthernet** as the transport type.
3. Click the browse button (...) next to the **Target** box to select a target PC.

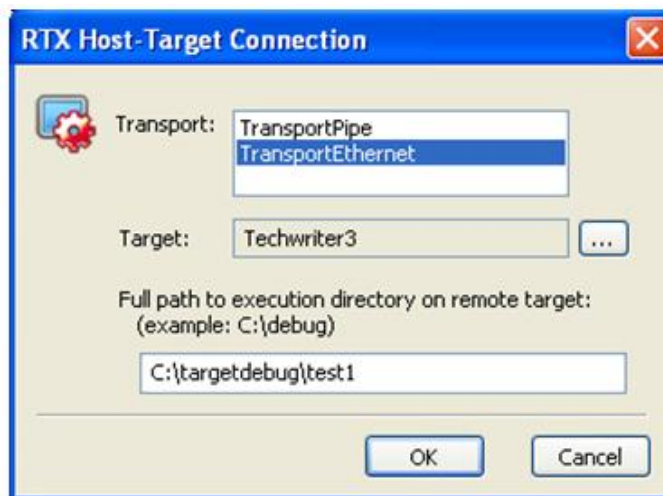


4. Browse for the target computer, select it by highlighting the system name, and then click **OK**.



5. Enter the full path of the execution directory on the target system where binary files were copied. For example, `C:\targetdebug\test1`.
  - o The executable files (.rtss) must be copied to this directory before you can begin to debug.

- When using remote debugging, the .rtdll must be registered with the target system using `rtssrun /d`, so that it can be found by the debugging .rtss program.



6. Click **OK** to save your settings.

## Debugging

After you have configured the host and target systems, you can debug your application as you normally would within Visual Studio by setting up break points and selecting **Start Debugging** from the Debug menu.

## Connection Troubleshooting

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

- No firewall is active or, if active, an exception has been included for `<RTXDIR>\debuggers\Common\RtxDbgManager.exe`
- The target system name in the RTX Host-Target Connection window matches the Computer Name found in the System Properties window of the target system
- The RTX subsystem has been started on the target system
- The code path on the host is the full path to the .rtss binary on the target system and that it is less than 128 characters long
- The host and target systems are not booted into a kernel debug enabled configuration (make sure your boot configuration does not contain `/debug` or `/baudrate`)

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

## Hints and Tips

When using Host/Target debugging in Visual Studio, the Visual Studio IDE will display the local path instead of the remote location.

## Resources

For detailed setup instructions, follow the procedure outlined in the Host/Target debug sections of the RTX help at:

**RTX Application Debugging Guide > Debugging with Visual Studio > Host/Target Debugging with Visual Studio**

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