## Windows® 10 IoT Core Native Remote Debugging

By Sean D. Liming and John R. Malin Annabooks – <u>www.annabooks.com</u>

November 2016

Two application types are support in Windows 10 IoT Core: Universal Windows Platform (UWP) and native console applications C/C++. Remote debug support (VSGraphicsRemoteEngine.exe) for UWP is part of the core OS. Native Remote Debugging for C/C++ applications is not included since Visual Studio and Windows 10 will update at different intervals. In this paper, we will explore how to add native remote debugging, system setup for remote debugging, and a few items to help with native code development.

### **Deploy the Remote Debug Files**

The remote debug tools come with the installation of Visual Studio 2015 and higher. Make sure you include Phone development during the installation. Windows 10 supports different processor architectures, ARM and Intel Architecture (x86 and x64). The binaries are found at this path:

C:\Program Files (x86)\Common Files\Microsoft Shared\Phone Tools\14.0\Debugger\target

With separate folders for the supported processor architectures:

← → ~ ↑	This PC → Local Disk (C:) → Program Files	(x86) > Common Files > Micro	osoft Shared > Pl	none Tools > 14.0 > [	Debugger > target
	Name	Date modified	Туре	Size	
> 📌 Quick access	armv4i	9/12/2016 3:29 PM	File folder		
> 👧 OneDrive	🔚 lib	9/12/2016 3:29 PM	File folder		
This PC	🔜 x64	9/12/2016 3:29 PM	File folder		
> Desktop	<mark>  </mark> x86	9/12/2016 3:29 PM	File folder		

Figure 1 - Different Binaries for Processor Architecture

Only use the binaries that go with your target's processor architecture.

Note: if Unified Write Filter (UWF) is available in the image, make sure UWF is disabled. You can check if UWF is enabled, by running UWFMGR get-config from a remote PowerShell session. If UWF is enabled, disable UWF and reboot the target.

Here are the steps for putting the binaries on your target platform:

- 1. Make sure your target and host development system are on the same network.
- 2. Boot the target system to Windows 10 IoT Core.
- 3. On the host system, start the IoT Dashboard application.
- 4. In the My Devices section, right click on your target's IP address, and select Open Network Share from the context menu.

- 🗆 X
OS
10.0.14393.0

Copyright © 2016 Annabooks, LLC. All rights reserved

## Figure 2 - Opening Network Share

- 5. Enter the credentials to access the remote target, and a file explorer window appears showing the remote target's C drive.
- 6. In the file explorer window, create a folder called RDBG.
- 7. Copy all the debug files for the target's processor architecture to the \RDBG folder on the target.

÷ → × ↑ 📑 × N	letwork > 192.168.1.103 > c\$ > RDBG			
	Name	Date modified	Type	Size
📌 Quick access			- 21-	
	1033	11/10/2016 2:19 PM	File folder	
🐔 OneDrive	AsyncCollectionAgent.dll	7/26/2016 10:23 PM	Application extens	65 KB
This PC	🚳 dbgshim.dll	1/27/2016 6:51 PM	Application extens	135 KB
_	DiagnosticsHub.CollectorBridge.dll	7/26/2016 10:23 PM	Application extens	154 KB
🛖 AB1TB (E:)	DiagnosticsRemoteHelper.dll	7/26/2016 10:23 PM	Application extens	186 KB
	DiagnosticsTap.dll	7/26/2016 10:23 PM	Application extens	448 KB
	GpuProfilingCollectionAgent.dll	6/20/2016 1:51 PM	Application extens	68 KB
AB500MB (F:)	🚳 msdbg2.dll	7/26/2016 10:23 PM	Application extens	152 KB
A	MSDIA140.dll	6/10/2016 7:22 AM	Application extens	1,175 KB
	MSVCP140.dll	6/10/2016 7:22 AM	Application extens	589 KB
🔩 Homegroup	📧 msvsmon.exe	7/26/2016 10:23 PM	Application	1,845 KB
	msvsmon.exe.config	7/26/2016 10:05 PM	CONFIG File	2 KB
	💿 pdm.dll	7/26/2016 10:23 PM	Application extens	547 KB
	pdmproxy100.dll	7/26/2016 10:23 PM	Application extens	102 KB
	pdmproxy140.dll	7/26/2016 10:23 PM	Application extens	98 KB
	VCRuntime140.dll	6/10/2016 7:22 AM	Application extens	78 KB
	💿 vsdebugeng.dll	7/26/2016 10:23 PM	Application extens	2,547 KB
	💿 vsdebugeng.impl.dll	7/26/2016 10:23 PM	Application extens	2,484 KB
	vsdebugeng.script.dll	7/26/2016 10:23 PM	Application extens	355 KB
	VSDebugLaunchNotify.exe	7/26/2016 10:23 PM	Application	32 KB
	SDebugScriptAgent140.dll	7/26/2016 10:23 PM	Application extens	157 KB
	VsEtwService.dll	7/26/2016 10:23 PM	Application extens	72 KB
	VSGraphicsCaptureEngine.dll	7/26/2016 10:23 PM	Application extens	114 KB

Figure 3 - New RDBG Folder and Remote Debug Files

Msvsmon.exe should be in the root of \RDBG folder.

#### Configuring the Firewall and Starting the Debugger

The firewall is enabled by default, thus the next step is to configure the firewall. Msvsmon.exe is the remote debug utility. You can configure the firewall via PowerShell remoting.

- 1. From the IoT Dashboard, right click on the target's IP address and select Launch Power Shell.
- 2. A PowerShell window opens, and you will have to enter the target's credentials again.
- 3. There are two options to configure the firewall. The first is to create a rule that allows msvsmon.exe to allow communications to come through the firewall:

netsh advfirewall firewall add rule name="Remote Debug" dir=in action=allow
program="C:\RDBG\msvsmon.exe" enable=yes

The second option is just to disable the firewall:

netsh advfirewall set allprofiles state off

4. Enter either of these in the PowerShell window and hit enter. You might want to create a .cmd file with either command if you ever have to run the remote debugger in a new image.

The next step is to start the debug session

- 5. In the PowerShell window, change directory to the \RDBG folder.
- 6. Enter the following to start the debugger:

msvsmon.exe /nowowwarn /noauth /anyuser /nosecuritywarn /timeout:36000

You might want to create a .cmd file with the command if you ever have to rerun the remote debug. The .cmd file would be best placed in the \RDBG folder.

- 7. You can see the debugger running in the target image, in IoT Dashboard, right click on your target's IP address and select Open in Device Portal.
- 8. An Internet Explorer or EDGE window will open. You will have to enter the credentials for the target.
- 9. In the Device Portal, click on Processes. You should see msvsmon.exe running.

<b>Windows</b>	Run	ning Pr	ocesses		() Power	Eeedback	? Help	2:43 PM
	^	644	awm.exe	window wa	nager\uvvi	VI-U U	U.UU%	A'2 IAR ~
UTILITIES /	$\times$	688	svchost.exe	NT AUTHOR		D 0	0.00%	2.1 MB
Home	$\times$	748	svchost.exe	NT AUTHOR	RITY\LOCAL	S 0	0.00%	5.4 MB
Apps	$\times$	832	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	11.1 MB
App File Explorer	$\times$	860	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	5.0 MB
Processes	$\times$	916	svchost.exe	NT AUTHOR	RITY\LOCAL	S 0	0.00%	5.4 MB
Debugging	$\times$	924	WUDFHost.exe	NT AUTHOR	RITY\LOCAL	S 0	0.00%	1.0 MB
Debugging	$\times$	980	svchost.exe	NT AUTHOR	RITY\LOCAL	S 0	0.00%	3.8 MB
ETW Dorf Tracing	$\times$	1036	svchost.exe	NT AUTHOR	RITY\LOCAL	S 0	0.00%	1.1 MB
Devices	$\times$	1116	svchost.exe	NT AUTHOR	RITY\LOCAL	S 0	0.00%	1.3 MB
Bluetooth	$\times$	1452	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	936.0 KE
Audio	$\times$	1508	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	3.0 MB
Networking	$\times$	1516	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	2.1 MB
Windows Update	$\times$	1524	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	3.5 MB
loT Onboarding	$\times$	1604	svchost.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	808.0 KE
TPM configuration	$\times$	1632	WebManagement.exe	NT AUTHOR	RITY\SYSTE	0 N	0.73%	2.9 MB
Remote	$\times$	1664	ebootpinger.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	452.0 KE
	$\times$	2160	SearchIndexer.exe	NT AUTHOR	RITY\SYSTE	0 N	0.00%	5.2 MB
	$\times$	2260	sihost.exe	MINWINPC	DefaultAcc	0	0.00%	2.4 MB
	$\times$	2408	IoTShell.exe	MINWINPC	DefaultAcc	o 0	0.00%	5.1 MB
	$\times$	2612	RuntimeBroker.exe	MINWINPC	DefaultAcc	o 0	0.00%	2.4 MB
	$\times$	2692	msvsmon.exe	MINWINPC	Administra	tor 0	0.00%	760.0 KE
	$\times$	2732	IoTCoreDefaultApp.exe	MINWINPC	DefaultAcc	o 0	0.00%	11.1 MB
	$\times$	2828	wsmprovhost.exe	MINWINPC	Administra	tor 0	0.00%	29.9 MB
	$\times$	3036	cmd.exe	MINWINPC	Administra	tor 0	0.00%	408.0 KE
	$\times$	3064	msvsmon.exe	MINWINPC	Administra	tor 0	0.00%	704.0 Ke

#### Figure 4 - Running Processes

The system is ready for a native remote debug session.

## **Visual Studio Project Settings**

Make sure that you have installed the IoT Core project Templates. The templates add the options to create Windows IoT Core applications. For C++ applications, there are three project types. The Blank Windows IoT Core Console Application creates a simple Hello World application that you can edit to create your own console application.

New Project			? ×
₽ Recent	.NET Framework 4.5 - Sort by: Default	• # IE	Search Installed Templates (Ctrl+E)
▲ Installed	Background Application (IoT)	Visual C++	Type: Visual C++
Templates     Visual C#     Visual Rasic	Blank Windows loT Core Console Application	Visual C++	A project for creating a Win32 console application for Windows IoT Core
<ul> <li>Visual Dasic</li> <li>Visual C++</li> <li>Visual C++</li> <li>Windows Driver</li> <li>Windows</li> <li>Universal</li> <li>Windows IoT Core</li> <li>ATL</li> <li>CLR</li> <li>General</li> </ul>	Arduino Wiring Application for Windows IoT Core	Visual C++	

Figure 5 - Creating Console Application

The final step is to configure the C/C++ project debug settings in Visual Studio:

- 1. With your project open in Visual Studio, select Project-><Project name> Properties...
- 2. Select Debugging under Configuration Properties.
- 3. Set the Debugger to Launch: "Remote Windows Debugger".
- 4. For both Active and Debug configurations, fill in the following information:
  - Remote Command: c:\<path>\Application name.
  - Working Directory: c:\<path>.
  - Remote Server Name: IP address of the target.
  - Connection: Remote with no authentication.
  - Debugger Type: Native only.
  - Deployment Directory: c:\<path>.
  - Deploy Visual C++ Debug Runtime Libraries: No.

## Annabooks

Configuration Properties General       Debugger to launch:         Debugging       Remote Windows Debugger         VC++       Dirther Windows Debugger         VC++       Dirther Windows Debugger         Variation Properties       Remote Command         C/C++       Variation Properties         Mainfest Tool       Remote Command Arguments         Mainfest Tool       Remote Server Name         Browse Information       Debugger Type         Native Only       Environment         Ocde Analysis       SOL Debugging         No       SOL Debugging         No       Deployment Directory         Cideploy         Additional Files to Deploy         Deployust C++ Debug Runtime Librari No         Amp Default Accelerator       WARP software accelerator	onfiguration:	Active(Release)	<ul> <li>Platform:</li> </ul>	Active(x64)	~	Configuration Manager
General       Petrodypting          Debugging       Petrodypting          VC++ Directories       C/C++       Remote Command Arguments          Manifest Tool       Remote Command Arguments       C/deploy         Working Directory       c:\deploy         B Manifest Tool       Remote Server Name       192.168.1.136         Build Step       Native Only       Environment         Debugger Type       Native Only       Environment         Code Analysis       SQL Debugging       No         SQL Debugging       No       SQL Debugging         Additional Files to Deploy       Deploy Visual C++ Debug Runtime Librari         Amp Default Accelerator       WARP software accelerator	Configuratio	on Properties	Debugger to launch:			
Debugging         Periods Prinder Proceedings           VC++         Directories         C/C++           E C/C++         Remote Command Arguments         Remote Command Arguments           Mainfest Tool         Remote Server Name         192.168.1.136           Mult Document Generator         Browse Information         Remote Server Name           Build Events         Debugger Type         Native Only           Code Analysis         SQL Debugging         No           SQL Debugging         No         Deployment Directory           Additional Files to Deploy         Deploy Visual C++ Debug Runtime Librari No           Amp Default Accelerator         WARP software accelerator	General		Remote Windows Debu	ager		<i></i>
VC++       Directories         © C/C++       Remote Command       c:\deploy\sysinfo14393.exe         > Manifest Tool       Remote Command Arguments         > Manifest Tool       Remote Server Name       192.168.1.136         > MUL Document Generator       Remote Server Name       192.168.1.136         > Build Events       Debugger Type       Native Only         > Custom Build Step       Attach       No         > Code Analysis       SQL Debugging       No         > Deployment Directory       c:\deploy         Additional Files to Deploy       DeployUsual C++ Debug Runtime Librari         Amp Default Accelerator       WARP software accelerator	Debuggir	ng	nemote mindows bebu	990		
	VC++ Directories > C/C++ > Linker > Manifest Tool > XML Document Generator > Rowse Information > Build Events > Custom Build Step > Code Analysis		Remote Command Remote Command Working Directory Remote Server Nam Connection Debugger Type Environment Attach SQL Debugging Deployvment Direct Additional Files to I Deploy Visual C++ Amp Default Accele	Arguments ie pry Peploy Debug Runtime rator	c:\deploy\sysinfo14393.exe c:\deploy 192.168.1.136 Remote with no authentication Native Only No c:\deploy Librari No WARP software accelerator	

Figure 6 – Configuring the Project for Remote Debug

- 5. The final step is to make sure Deploy is enabled. From the menu, Build->Configuration Manager...
- 6. Make sure Deploy is checked for both Retail and Debug.

Configuration Manager						?	×
Active solution configuration:	Active solution platf	form:	:				
Debug		~	x64				$\sim$
Project contexts (check the proj	ect configurations to buil	d or de	ploy):				
Project	Configuration		Platform		Build	Deploy	
sysinfo2016	Debug	$\sim$	хб4	$\sim$	$\checkmark$	$\checkmark$	

Figure 7 - Enabling Deployment

You can now deploy and remotely debug your native application. If your code sends any output to the console, the output will be displayed in PowerShell

# Annabooks

sysinfo2016 (Debugging)	) - Microsoft Visual Studio			$\nabla$	Quick Launch (Ctrl+Q)	₽ = ¤ ×
File Edit View Project	Build Debug Team Tools Arc	hitecture Test Driver	Analyze Window Help	р		Sean Liming 👻 SL
G - O 🔞 - 🏜 🖬	P 🤊 - 🔍 - Debug - x64	- 🕨 Continue -	🗧 📮 🗉 🔳 👌 😌 🕒	→ <b>! ? !</b> \A (	Code Map 🛛 🤏 📮 🎜 💭 🖉	이 哈 师 ( 제 개 이 이
Process: [352] sysinfo2016.e	exe 🔹 💽 Lifecycle Events 👻 T	hread: [1316] Main Thread	- 🔻 🔻 100	Stack Frame: main	n	•
Sysinfo2016.cpp ↔ ×						<u>≁</u> δ
sysinfo2016	-	(Global Scope)		∗ © main(ir	nt argc, char ** argv)	• ution
<pre>sifed _ M_IX66 ==Lif _ M_X64 ==Lif _ M_X64 ==trunce setup: for the sage printlessage printlessage printlessage MEMORYSTATUS statex.duce BOOL success {     f (Isuccess     f (Isuceess     f (Isuceess     f (Isuceess     f (Isuceess</pre>	<pre>i; :Line("Processor Architecture:", p Line("Processor Type", proctype); Line("Number of Processors", nump Line("Processor Revision", procievel itine("Processor Level", proclevel EX states; gth = sizeof(statex); = 6lobalMemoryStatusEx(&amp;statex); ;)</pre>	rocarchmsg); roc(); v); S0mselapsed );				4 Team Explorer
100 % 👻 <			-			- F
Autos		- ù ×	Output			- ų ×
Name	Value	Туре	Show output from: Deb	ug	-   %	: 🖆 🖆 🔛
std::operator<< <std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator< p=""></std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<<<std::operator<>	ł {} ł {}	std::basio std::basio	'sysinfo2016.exe'	(Win32): Loaded	<pre>'C:\deploy\sysinfo2016.exe' 'C:\Windows\System32\ntdll</pre>	. Symbols loader
numproc     procarchmsg	2 0x00007ff77c7a3300 "x64"	unsigned Q + const ch	'sysinfo2016.exe' 'sysinfo2016.exe' 'sysinfo2016.exe'	(Win32): Loaded (Win32): Loaded (Win32): Loaded	<pre>'C:\Windows\System32\Kernel 'C:\Windows\System32\ucrtba 'C:\Windows\System32\ucrtba</pre>	Base.dll'. Canno se.dll'. Cannot 40 dll'. Cannot
Autos Locals Watch1		~	'sysinfo2016.exe' 'sysinfo2016.exe' ∢ Call Stack Breakpoints	(Win32): Loaded (Win32): Loaded Exception Settings (	<pre>'C: \Windows\System32\verunt 'C: \Windows\System32\forwan Command Window Immediate Wini</pre>	dow Output
Ready		142 Col 1		INS		🔶 Publish 🔺 💡

Figure 8 - Visual Studio Remote Debugging



Figure 9 - Application Console Output

## **Processor Directives**

UWP applications are design to run on different processor architectures by simply changing the target processor in Visual Studio. This allows you to create one program that target's different platforms. The same can be done for native applications, but you must address API calls that are unique to each processor architecture.

The processor directives (\_M\_IX86, \_M\_X64, and \_M\_ARM) can be put into your project to select code that is only for a specific processor architecture. For example, if there is code that will run on x86 and x64 but not on ARM, you would structure the processor directives as follows:

If there is code only for ARM, you would structure the processor ARM directive as follows:

#ifdef \_M\_ARM

<<< your code here>>>

#endif

When you select the target processor architecture, the code with the other processor directives will be greyed out.

### **Summary: Going Native**

There are many programming options available for IoT Core, and for the many hardcore C/C++ developers, C/C++ is not going away any time soon. The ability to remote debug native applications allows those familiar with programming in C/C++ the ability to create solutions using IoT Core.

```
Windows is registered trademarks of Microsoft Corporation
All other copyrighted, registered, and trademarked material remains the property of the respective owners.
```