



Developing Integration Test with WSL APIs

WSLHUB

JUNG HYUN, NAM

MICROSOFT AZURE MVP

Contents

Motivation for starting the project

WSL SDK in Action

Problems solved while developing the SDK


Missing Pieces

Future

Motivation for starting the project



Windows Server Install

05/12/2020 • 2 minutes to read •  +8

The Windows Subsystem for Linux is available for installation on Windows Server (version 1709) and later. This guide will walk through the steps to install WSL on your machine.

Enable the Windows Subsystem for Linux

Before you can run Linux distros on Windows, you must enable the "Windows Subsystem for Linux" optional feature and reboot.

Open PowerShell as Administrator and run:

```
PowerShell

Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

What I learned while installing WSL for Windows Server

The APPX/MSIX package you receive from the Microsoft Store is in ZIP file format.

On the Windows Server, unzip the APPX package and install it manually.

The file called Install.tar.gz is the root file system image!

Either way, you can manually install WSL on Windows 10 or Windows Server.

So, wouldn't it be possible to customize the WSL installation to my needs?

Wslapi.h header - Win32 apps | x +

https://docs.microsoft.com/en-us/windows/win32/api/wslapi.h/header/wslapi.h

AWS Login Haru Keycloak Admin Co... Install-Module (Po... Ne

Microsoft | Docs Documentation Learn Q&A Co

Windows Developer Explore ▾ Platforms ▾ Downloads San

Windows / Apps / Win32 / API / Wslapi.h / Overview

Filter by title

- Overview
- WSL_DISTRIBUTION_FLAGS enumeration
- WslConfigureDistribution function
- WslGetDistributionConfiguration function
- WslsDistributionRegistered function
- WslLaunch function
- WslLaunchInteractive function
- WslRegisterDistribution function

Download PDF

wslapi.h header

01/11/2019 • 2 minutes to read

This header is used by Windows Subsystem for Linux (WSL).

- Windows Subsystem for Linux (WSL)

wslapi.h contains the following functions:

Functions

[WslConfigureDistribution](#)

Modifies the behavior of a distribution (WSL).

[WslGetDistributionConfiguration](#)

Curiosity about WSL

WSL is also a feature of Windows, so of course, there is an API.

The WSL API that I found in that way is mainly optimized for designing new distributions for WSL.

If so?

1

- It is possible to create an image containing the Linux OS at will.

2

- You are free to register and unregister WSL distributions.

3

- You can send commands from the Windows side and receive the results.

Then,

Wouldn't it be possible to implement integrated test automation that is easier to use than a virtual machine?

Wouldn't it be helpful to increase productivity by actively customizing the distribution for WSL?

That's how I started the project.

Introducing the WSL SDK project

[GITHUB.COM/WSLHUB/WSL-SDK-COM](https://github.com/wslhub/wsl-sdk-com)

main 1 branch 0 tags

46f52c4 20 hours ago 60 commits

Fix stdout blocking issue partially	20 hours ago
Change and tune build configurations	6 days ago
Enhance and add readme and license stuff	12 days ago
Update README.md	11 days ago

Out-of-process COM server style WSL SDK

This project contains the out-of-process style COM server-based WSL APIs, which overcome the ColnitializeSecurity issues and still maintain ease of code management with the .NET Framework. You can use this OOP-style COM server to query and run WSL commands via the Windows PowerShell, LINQPad, and all COM-supported clients. Once registered, every time you call the COM interface, the executable file automatically called and launched on-demand. If the reference count reaches zero, the process is automatically closed.

How to build and test

1. You can start building this project with the .NET Framework SDK v4.7.2 or higher and the Windows 10, at least 1903 or higher release.
2. Build a release of the WslSdk project, and register the OOP COM server via elevated permission with the

About

WSL SDK COM Module (Out-of-Process Type)

Readme

MIT License

Releases

No releases published

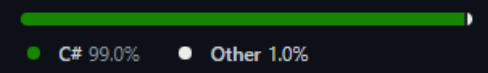
[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages



```
WslSdk - IWslService.cs
IWslService.cs
WslSdk
7 [ComVisible(true)]
8 [Guid("{62BD3105-260E-45AF-834B-E6C790F986D0}")]
9 public interface IWslService
10 {
11     bool IsDistroRegistered(string distroName);
12
13     DistroRegistryInfo GetDefaultDistro();
14
15     string[] GetDistroList();
16
17     string RunWslCommand(string distroName, string commandLine);
18
19     DistroRegistryInfo GetDistroInfo(string distroName);
20
21     string GetDefaultDistroName();
22
23     DistroInfo QueryDistroInfo(string distroName);
24
25     void SetDefaultUid(string distroName, int defaultUid);
26
27     void SetDistroFlags(string distroName, DistroFlags distroFlags);
28
29     string GenerateRandomName(bool addNumberPostfix);
30
31     void RegisterDistro(string newDistroName, string tarGzipFilePath, s
32
33     void UnregisterDistro(string existingDistroName);
34
35     string GetWslWindowsPath(string distroName);
36
37     string TranslateToWindowsPath(string distroName, string linuxAbsolu
38
39     string TranslateToLinuxPath(string distroName, string windowsAbsolu
40
41     bool TestLinuxPath(string distroName, string linuxAbsolutePath);
42
43 }
```

What can I do with this project?

With Any language and environment running on Windows, you can do:

- Registering, Modifying, and Deleting WSL Distributions
- Query WSL distribution information
- Convert File Paths (between WSL and Windows)
- Generate random names (borrowing code from Moby/Docker)

You can do any of the above with no hassle.

WSL SDK in Action



Implement an integrated test environment

Initializing
WSL SDK
Service Objects

Generate
random names

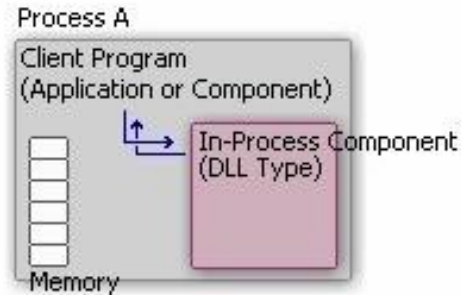
Download
Root File System

Distro installation

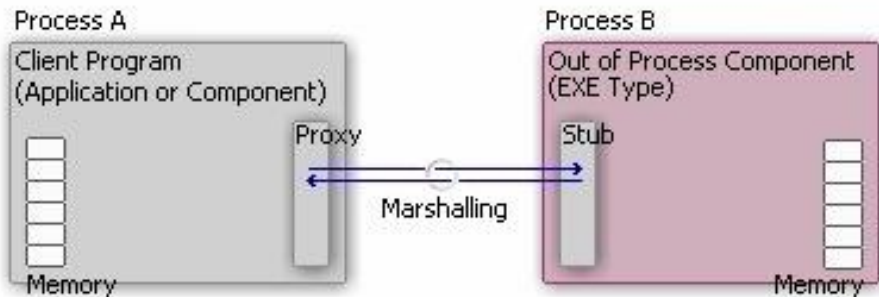
Passing/executing
commands
to the distribution

Uninstall a distro

Initializing WSL SDK Service Objects



In-Process Component Model



Out of Process Component Model

The WSL Win32 API just can't be used.

It must be initialized with the `CoInitializeSecurity` API, but...

Depending on the environment, this API has already been called and may not be available for use.

So I choose a separate out-of-process COM server model to overcome hassles.

```
// Youyou Tu - Chinese pharmaceutical chemist and  
"tu",  
  
// Alan Turing was a founding father of computer  
"turing",  
  
// Varahamihira - Ancient Indian mathematician w  
"varahamihira",  
  
// Dorothy Vaughan was a NASA mathematician and  
"vaughan",  
  
// Sir Mokshagundam Visvesvaraya - is a notable  
"visvesvaraya",  
  
// Christiane Nüsslein-Volhard - German biologis  
"volhard",  
  
// Cédric Villani - French mathematician, won Fi  
"villani",  
  
// Marlyn Wescoff - one of the original programm  
"wescoff",  
  
// Sylvia B. Wilbur - British computer scientist
```

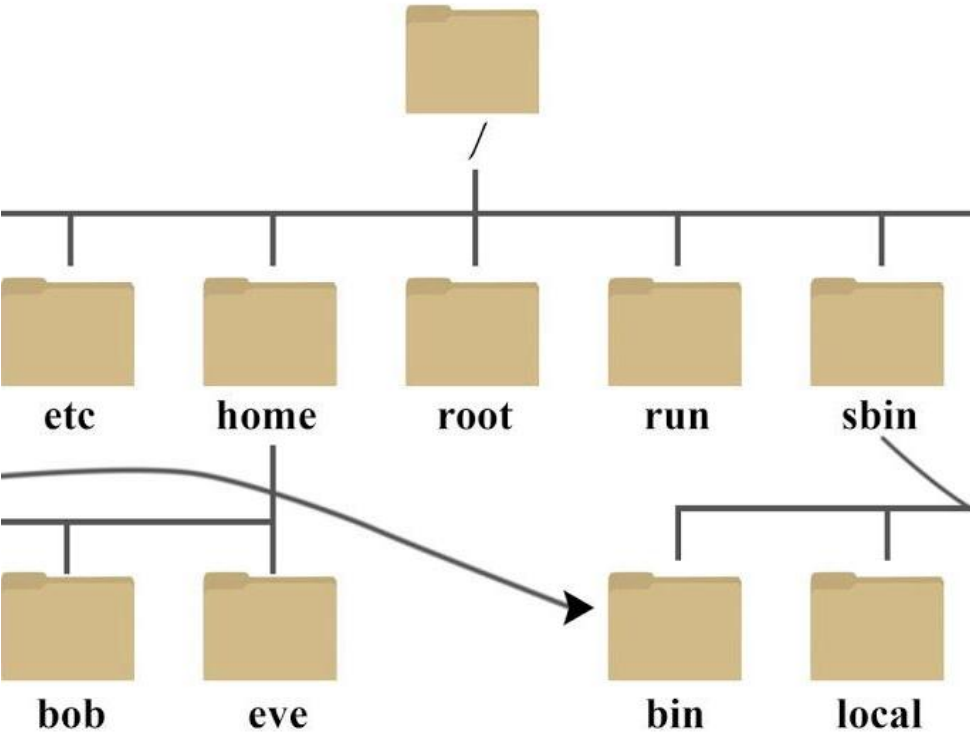
Generating Random Names (feat. Docker/Moby)

Used to allow WSL distributions to be dynamically registered as needed (but human-friendly naming)

Add code by porting Docker's source code into C#

<https://github.com/moby/moby/blob/master/pkg/namesgenerator/names-generator.go>

Getting the Linux root filesystem easy



You can create it manually as described in the article below, but...

- <https://link.dotnetdev.kr/3qy68RT>

It is recommended to use a pre-made root file system that is easier and more convenient for everyday use.

- <http://cdimage.ubuntu.com/ubuntu-base/releases/>

Furthermore, Busybox or Alpine can be used

- <https://github.com/0xbadfca11/miniwsl>
- <https://github.com/yuk7/AlpineWSL>

Microsoft Windows [Version 10.0.19043.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rkttu>



Utilizing the Linux Root File System

Basically, the root file system contains the necessary elements required to run a Linux system.

Examples you can use:

- Ways to export/import existing Docker container image
- Ways of registering/exporting/importing distribution images for WSL 1 and 2

Passing/executing commands to the distribution

When SDK run the WslLaunch API,

- Create Anonymous Pipe with CreatePipe
- Execute CreateProcess internally
- Sharing STDIN, STDOUT, STDERR with Linux processes in WSL
- STDOUT, STDERR capture as a string
- Returns the captured content as a string as a return parameter of the COM API



Demonstration

EXCEL·POWERSHELL·PYTHON/JUPYTER
& VISUAL STUDIO 2022 UNIT TEST

A1



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		
19																		
20																		
21																		
22																		
23																		
24																		
25																		
26																		

Run WSL Commands

Clear Output Cell

Command Execution Result



Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>PS C:\Users\rkttu> cd .\Desktop\
PS C:\Users\rkttu\Desktop> dir

디렉터리: C:\Users\rkttu\Desktop

Mode	LastWriteTime	Length	Name
-a----	2021-06-29 오후 11:31	6910	CreatePipe.txt
-a----	2021-07-01 오전 10:19	6507924	Developing Integration Test with WSL APIs.pptx
-a----	2021-06-27 오후 11:30	1041	Developing Integration Test with WSL APIs.txt
-a----	2021-04-01 오후 9:11	3681	docker-hyperv-builder.txt
-a----	2021-06-05 오후 5:55	1790543	learn-wsl-figures.pptx
-a----	2021-07-01 오전 12:40	121	MinGW - Minimalist GNU for Windows - Thread- [Mingw-users] redirected stdout-stderr problems in Windows program.url
-a----	2021-06-29 오후 5:36	13768	Sample.ipynb
-a----	2021-06-29 오후 4:27	2495	Sample.ps1
-a----	2021-07-01 오전 11:28	27355	WslSdkSample.xlsb
-a----	2021-07-01 오전 12:53	5933869	WslSdkSample.zip
-a----	2021-05-16 오후 8:06	490	파산 신청 관련 상담.txt

PS C:\Users\rkttu\Desktop> |

WSL SDK 파이썬 샘플 노트북

이 노트북에는 WSL SDK 프로젝트를 이용해서 Alpine 리눅스 배포판을 동적으로 생성하고, 통합 테스트 환경을 API 방식으로 생성/관리하는 방법을 설명하는 예제 코드를 담고 있습니다.

이 샘플 노트북을 실행하려면 먼저 <https://github.com/wslhub/wsl-sdk-com> 에서 최신 버전의 소스 코드를 받아 regasm.exe 유틸리티로 Out-of-Process COM 서버를 등록해야 합니다.

Test Explorer

Search Test Explorer

Test	Duration	Traits
WslSdkTest (16)	1.8 sec	
WslSdkTest (16)	1.8 sec	
DistroEnumTest (6)		
Test_GetDefaultDistro		
Test_GetDefaultDistroN...		
Test_GetDistroList		
Test_IsDistroRegistered		
Test_QueryDistroInfo		
Test_RunWslCommand		
DistroFileSystemTest (5)		
Test_LinuxToWindowsPa...		
Test_LinuxToWindowsPa...		
Test_PathExistence		
Test_WindowsToLinuxPa...		
Test_WindowsToLinuxPa...		
DistroManipTest (3)		
Test_DistroConfiguratio...		
Test_DistroRegisterUnre...		
Test_GenerateRandomN...		
DistroScenarioTest (2)	1.8 sec	
Test_CurlLargeContent	915 ms	
Test_CurlSimple	879 ms	

Group Summary

WslSdkTest

Tests in group: 16

Total Duration: 1.8 sec


Outcomes

- 14 Not Run
- 2 Passed

```

404     writePipeHandle = writePipeHandleTemp;
405     return true;
406 }
407
408 0 references | Jung Hyun Nam, 19 hours ago | 1 author, 3 changes
409 private static unsafe void Main(string[] args)
410 {
411     var result = CoInitializeSecurity(
412         IntPtr.Zero,
413         (-1),
414         IntPtr.Zero,
415         IntPtr.Zero,
416         RpcAuthnLevel.Default,
417         RpcImpLevel.Impersonate,
418         IntPtr.Zero,
419         EoAuthnCap.StaticCloaking,
420         IntPtr.Zero);
421
422     if (result != 0)
423         throw new COMException("Cannot complete CoInitializeSecurity.", result);
424
425     // Test Arguments
426     int bufferSize = 1024;
427     string distroName = "Ubuntu-20.04";
428     string commandLine = "cat /etc/os-release";
429
430     var attributes = new SECURITY_ATTRIBUTES
431     {
432         lpSecurityDescriptor = IntPtr.Zero,
433         bInheritHandle = true,
434     };
435     attributes.nLength = Marshal.SizeOf(attributes);
436
437     IntPtr
438     bufferPointer = IntPtr.Zero,
439     hTemp = IntPtr.Zero,
440     childStdoutReadPipe = IntPtr.Zero,
441     childStdoutWritePipe = IntPtr.Zero,
442     child = IntPtr.Zero;
443
444     var memStream = new MemoryStream();
445
446     try
447     {
448         if (!CreatePipe(out hTemp, out childStdoutWritePipe, attributes, 0))

```

A hand holding a Rubik's cube in a dimly lit room. The background is dark and out of focus, showing what appears to be a desk with a laptop and some papers. The text is overlaid on the image in a large, white, sans-serif font.

Problems solved while developing the SDK

ColnitializeSecurity Problem

To use WSL's API, the caller's credentials who calls the API must be passed to the WSL service.

For this, an Impersonation request is required while calling ColnitializeSecurity.

However, depending on the environment, there are cases where the ColnitializeSecurity API has already been called and cannot be called again (e.g., Windows PowerShell, LINQPAD, etc.)

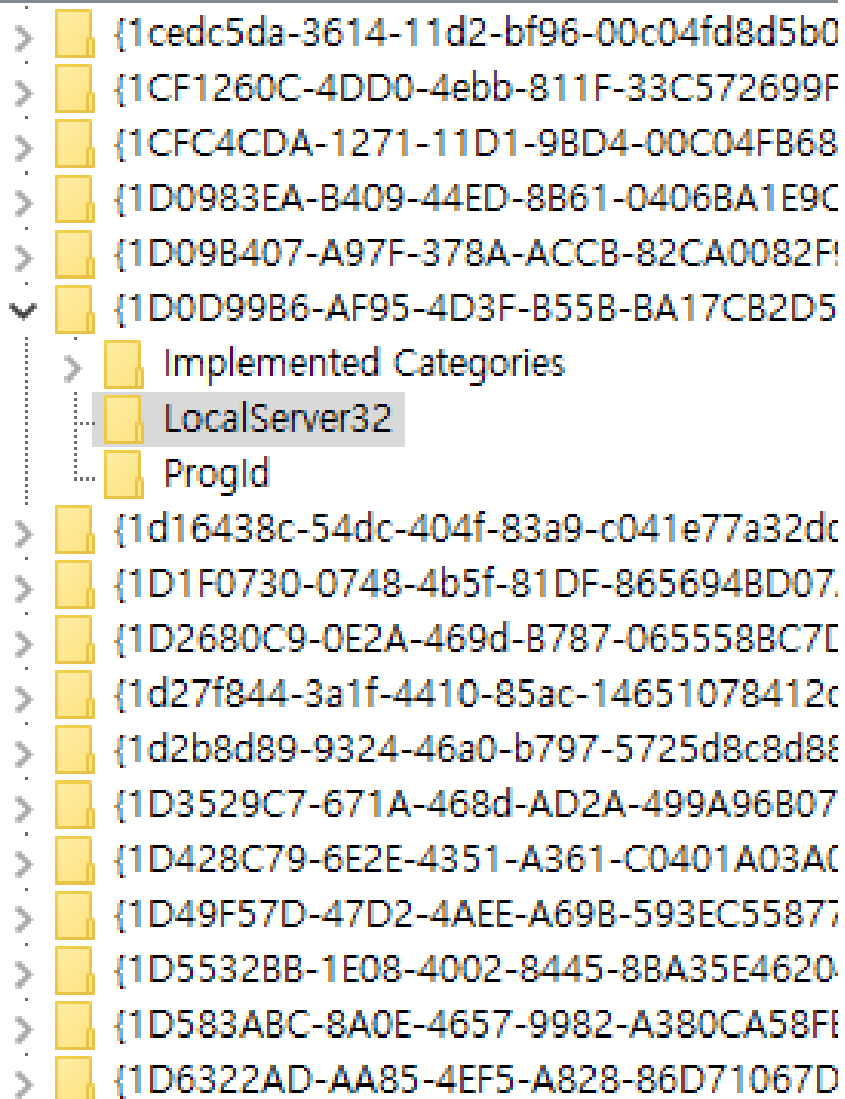
You need to run the WSL API in a separate process to avoid this problem.

Out-of-Process COM Server

Separation of processes is mandatory, but you want to meet the requirements below

- A Windows process should be a singleton that runs only once per user
- No effort is required to consider the path to the executable or create a process
- Do not rely on any network resources at all
- Should be available locally in any language without a separate wrapper

An out-of-process COM server that satisfies all these conditions, so I chose it.



CodePlex Archive will be shut down

CodePlex was Microsoft's free, open source project hosting site, which ran from 2006 through 2019, for over 13 years. We now encourage customers to use [Github](#) for their open source project hosting needs.

CodePlex will continue as an archive until next July (2021), at which point it will be shut down. Projects, documentation, issues, and discussions which were posted before the site went offline will be preserved.

For questions or comments please contact [CodePlex Archive support](#).

CodePlex Archive Open Source Project Archive

Project '1code' was not found

© 2006-2018 Microsoft [Shutdown Announcement](#) [Support](#)

Thankfully

Found content on Microsoft's old collection of example code, the All-In-One Code Framework

Exactly an out-of-process COM server implementation in C#.

However, CodePlex, the original repository of AIO Code Framework, is now closed.

So, I decided to look it up on GitHub.

Why still use the .NET Framework?

For APIs made with .NET to be provided through COM, the creation of type libraries must be automated, but...

Only .NET Framework can automate this part; .NET Core or .NET 5 does not have this feature yet.

- Windows SDK and Visual C++ Compiler required to replace this feature

.NET Framework is still advantageous to simplify installation and execution

WslRegisterDistribution

Perhaps among the WSL APIs, the WslRegisterDistribution API finds the path to the EXE file of the process that calls it.

Create a file system for WSL directly under the directory path containing the EXE file that calls this function

To freely register WSL distributions, I made a custom launcher.

encies (x64)

Options Help

Windows\System32\wslapi.dll

PI	Ordinal	Hint	Function
<input checked="" type="checkbox"/>		N/A	16 (0x00000010) PathCchRemoveFileSpec
<input checked="" type="checkbox"/>		N/A	6 (0x00000006) CreateFileW
<input checked="" type="checkbox"/>		N/A	12 (0x0000000c) GetStdHandle
<input checked="" type="checkbox"/>		N/A	0 (0x00000000) GetProcessHeap
<input checked="" type="checkbox"/>		N/A	2 (0x00000002) HeapAlloc
<input checked="" type="checkbox"/>		N/A	5 (0x00000005) HeapFree
<input checked="" type="checkbox"/>		N/A	20 (0x00000014) QueryFullProcessImageNameW
<input checked="" type="checkbox"/>		N/A	1 (0x00000001) RoGetActivationFactory
<input checked="" type="checkbox"/>		N/A	3 (0x00000003) RoInitialize
<input checked="" type="checkbox"/>		N/A	7 (0x00000007) RoUninitialize
<input checked="" type="checkbox"/>		N/A	12 (0x0000000c) WindowsDeleteString

E	Ordinal	Hint	Function
<input checked="" type="checkbox"/>	1 (0x0001)	0 (0x00000000)	WslConfigureDistribution
<input checked="" type="checkbox"/>	2 (0x0002)	1 (0x00000001)	WslGetDistributionConfiguration
<input checked="" type="checkbox"/>	3 (0x0003)	2 (0x00000002)	WslIsDistributionRegistered
<input checked="" type="checkbox"/>	4 (0x0004)	3 (0x00000003)	WslLaunch
<input checked="" type="checkbox"/>	5 (0x0005)	4 (0x00000004)	WslLaunchInteractive
<input checked="" type="checkbox"/>	6 (0x0006)	5 (0x00000005)	WslRegisterDistribution
<input checked="" type="checkbox"/>	7 (0x0007)	6 (0x00000006)	WslUnregisterDistribution

Module	Machine	Type	File Size	Image Base	Virtual Size	Entry point
i-ms-win-crt-runtime-11-1-0.dll -> C:\Windows\System32\wslapi.dll	AMD64	Dll; Executable	0x000ff198	0x180000000	0x00100000	0x00016110
i-ms-win-crt-private-11-1-0.dll -> C:\Windows\System32\wslapi.dll	AMD64	Dll; Executable	0x000ff198	0x180000000	0x00100000	0x00016110
i-ms-win-crt-string-11-1-0.dll -> C:\Windows\System32\wslapi.dll	AMD64	Dll; Executable	0x000ff198	0x180000000	0x00100000	0x00016110
i-ms-win-core-libraryloader-11-2-0.dll	AMD64	Dll; Executable	0x002c8b80	0x180000000	0x002c8000	0x00010650
i-ms-win-core-debug-11-1-0.dll -> C:\Windows\System32\wslapi.dll	AMD64	Dll; Executable	0x002c8b80	0x180000000	0x002c8000	0x00010650

le "C:\Windows\System32\wslapi.dll" successful.

Missing Pieces



The Hidden LXSS Service Manager

Information in the registry vs. LXSS Service Manager

- Information that is permanently stored is primarily stored in the registry
- The LXSS service manager is responsible for executing WSL 2's built-in Linux kernel and various services while at the same time controlling the status of current WSL distributions.
- What's available right now is the registry and basic C-style APIs are the best!
- Surprisingly, neither the LXSS service manager API nor the COM interface is revealed until today.

ILxssUserSession Interface

As mentioned earlier, the current WSL API is only provided for distribution creators.

The various functions provided by WSL.EXE must be contained in the ILxssUserSession interface!

However, the interface type information was not disclosed anywhere.

<https://github.com/Biswa96/WslReverse>

Conclusion



Future

I'm trying to develop WSL SDK v2 without elevation of privilege using ASP.NET Core and gRPC, and AF_UNIX socket support introduced since Windows 10 1803.

This has many benefits, such as MSIX packages, ARM64 native support, and support for a broader range of languages and tools.

Future (Cont.)

GitHub Action support for Windows Server 2019 and later

- Build environment support for WSL distribution developers
- Automation of WSL image creation that pre-packages familiar developer tools used by the team

Future (Cont.)

Added support for multiple programming languages

- When the API of the WSL SDK is stabilized, and the first version can be released, it will be materialized.

Future (Cont.)

Integration with Docker and Docker Hub

- Using the API provided by Docker Engine
- Build container image for WSL distribution Export to Root FS
- WSL distribution Root FS creation
Import with Docker

Takeaway

WSL API exposed in MS Docs has few features

- If you're interested in building a Linux distribution for WSL, I recommend taking a look.
- It can be made only with Root FS or released as a package for MS Store.

WSL SDK is currently under development

- Looks like I can release version 0.1 in the not-too-distant future

WSL distributions are up to you to make them.

- Not only the distributions in the store, but you can also customize them to your liking.

About Our Community

WSLHUB - Korean WSL User Group

- <https://fb.com/groups/wslhub>
- <https://github.com/wslhub>

Thank you!

RKTTU@RKTTU.COM



thanks!