

デスクトップアプリを Windowsストアから配布するための A to Z

日本マイクロソフト株式会社
テクニカルエバンジェリスト

荒井 省三

アジェンダ

- インストーラの振り返り
- デスクトップブリッジとは
- 開発
 - 変換
 - テスト
 - 公開
- Windows 10 S
- アプリを進化させるには

インストーラの振り返り

インストール技術の変遷

MS-DOS (開発者の考え方による)

容易な配布 (XCOPY) やカスタム セットアップ

Windows XP (Windows Installer v2.0、UACへの対応、etc)

Windows インストーラ サービス (MSI)

Windows 8

ユニバーサル アプリ パッケージ

※ OPC - ISO/IEC 29500、ECMA 376

アプリのインストールとサービス

Windows desktop applications > Develop > Desktop technologies ▾

- ▶ Desktop App UI
- ▶ Desktop Environment
- ▼ Application Installation and Servicing
 - ▶ Games Explorer
 - ▶ Isolated Applications and Side-by-side Assemblies
 - ▶ Packaging, deployment, and query of Windows Store apps
 - ▶ Developer licensing
 - ▶ Restart Manager
 - ▶ Windows Installer
- ▶ Audio and Video
- ▶ Data Access and Storage
- ▶ Devices
- ▶ Diagnostics
- ▶ Documents and Printing
- ▶ Graphics and Gaming
- ▶ Networking and Internet
- ▶ Security and Identity

Application Installation and Servicing

Make use of available APIs and services provided by Windows to install, manage, and service your desktop apps.

In this section

Topic	Description
Games Explorer	Describes how to use Games Explorer to display games on Windows.
Isolated Applications and Side-by-side Assemblies	Isolated Applications and Side-by-side Assemblies is a Microsoft Windows solution that reduces versioning conflicts in Windows-client applications.
Packaging, deployment, and query of Windows Store apps	Programmatically create app packages, and install, update, query, and uninstall Windows Store apps.
Restart Manager	The Restart Manager API can eliminate or reduce the number of system restarts that are required to complete an installation or update.
Windows Installer	Microsoft Windows Installer is an installation and configuration service provided with Windows. The installer service enables customers to provide better corporate deployment and provides a standard format for component management. The installer also enables the advertisement of applications and features according to the operating system.
Windows Setup and Migration	This section documents the notifications used to detect and possibly repair an application after a setup or migration has occurred. These notifications can also be used to suspend operations during the volatile setup or migration experience.

Windows Server App (WSA)

アプリとドライバーのインストール

Windows Server App インストーラー

Windows Server App (WSA) インストーラーは、Nano Server 向けの信頼性の高いインストール オプションを提供します。Nano Server では Windows インストーラー (MSI) がサポートされていないため、Microsoft 以外の製品においても WSA が唯一使用可能なインストール テクノロジとなります。WSA では、宣言型のマニフェストを使用してアプリケーションを安全かつ確実にインストールし、サービスを提供するように設計された Windows アプリ パッケージ テクノロジを利用しています。WSA は Windows Server に固有の拡張機能をサポートするように Windows アプリ パッケージ インストーラーを拡張したものですが、ドライバーのインストールがサポートされないという制限があります。

Nano Server で WSA パッケージを作成してインストールするには、パッケージの発行者とコンシューマーの双方で特定の手順を実行する必要があります。

パッケージの発行者は、次の手順を実行する必要があります。

1. **Windows 10 SDK** をインストールします。これには、WSA パッケージを作成するために必要なツール (MakeAppx、MakeCert、Pvk2Pfx、SignTool) が含まれています。
2. マニフェストを宣言します。**WSA マニフェスト拡張スキーマ**に従って、AppxManifest.xml マニフェスト ファイルを作成します。
3. **MakeAppx** ツールを使用して WSA パッケージを作成します。
4. **MakeCert** ツールと **Pvk2Pfx** ツールを使用して証明書を作成し、**SignTool** ツールを使用してパッケージに署名します。

次に、パッケージのコンシューマーは、次の手順を実行する必要があります。

1. certStoreLocation に "Cert:\LocalMachine\TrustedPeople" を指定し、**Import-Certificate** PowerShell コマンドレットを実行して、発行者が上記の手順 4. で作成した証明書を Nano Server にインポートします。以下に例を示します。
`Import-Certificate -FilePath ".\xyz.cer" -CertStoreLocation "Cert:\LocalMachine\TrustedPeople"`
2. **Add-AppxPackage** PowerShell コマンドレットを実行して、WSA パッケージを Nano Server にインストールすることで、アプリを Nano Server にインストールします。以下に例を示します。
`Add-AppxPackage wsaSample.appx`

アプリの作成に関するその他のリソース

WSA は、Windows アプリ パッケージ テクノロジのサーバー拡張機能です (ただし、Windows ストアではホストされません)。WSA を使用したアプリを発行する場合は、アプリ パッケージのパイプラインについて理解するうえで、次のトピックが役に立ちます。

- 基本的なパッケージ マニフェストを作成する方法
- App Packager (MakeAppx.exe) (アプリ パッケージ ツール (MakeAppx.exe))
- How to create an app package signing certificate (アプリ パッケージ 署名証明書を作成する方法)
- SignTool

アプリパッケージ vs Windows Installer

項目	Appx , WSA	MSI
管理者権限	不要	必要
対話型のインストール	×	○ (ウィザード)
インストーラの自由度	×	○
市販ツール	○	◎
ゼロ インパクト インストール	○	×
ユーティリティ	PowerShell Windows ストア	msiexec.exe 市販ツール
コード署名	必須	任意 (事実上は必須)

これからのアプリインストーラ

- 管理者権限を必要としない
 - ・ゼロインパクトインストール
 - ・OSが許可している範囲で動作 = 信頼できるインストーラ
 - ・信頼できる配布先と組み合わせる = 攻撃の機会が激減
- アンインストールでゴミを残さない
 - ・標準のメンテナンスを効果的に活用
 - ・レジストリの肥大化が発生しない
 - ・脆弱性の可能性を削減

デスクトップブリッジ

ユニバーサル Windows プラットフォーム ブリッジ

アプリに新しい魅力を付与する

モダンな配布

タイル、ファイルの拡張子、URI プロトコルを使った起動

アプリ アイデンティティ → UWP API へのアクセス

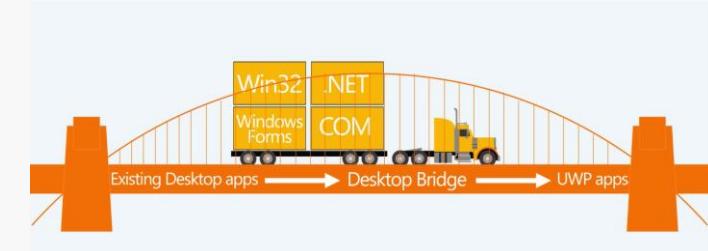
UWP アプリ モデル

何ができるか ?

→ 簡単な答え : UWP ができることができる



デスクトップブリッジ



- **一元管理された配布場所**

- Windows ストア = 信頼されたアプリ配布リポジトリ
- Windows ストアによる監査 = セキュアに保つ

- **一般ユーザー権限の使用**

- 脆弱性を生む機会の削減に貢献
- クリーンなアンインストール
- アプリの安全な実行環境を提供 (App-V)

プロセス実行イメージ



デスクトップ
アプリ

セキュアなプロセス (アプリ)

デスクトップ
ブリッジ
アプリ

仮想化(App-V)

UWP アプリ

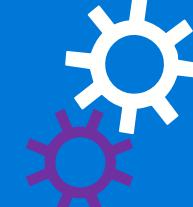
Windows オペレーティング システム

開発フロー

Desktop App Converter

既存のデスクトップ インストーラを変換

UWP マニフェストを持つ
たアプリ パッケージ
(APPX) が生成される



テスト、更新、改善する

Make MSI とアプリ パッケージに含まれる共通コードを作成する

ユニバーサル Windows プラットフォームの利点を生かすようにアプリ パッケージを改善させる

公開と配布

MDM や Windows ストアを使って、サイドローディング用のアプリ パッケージを配布する

変換

Desktop App Converter

既存のデスクトップ インストーラを変換
UWP マニフェストを持つ
たアプリ パッケージ
(APPX) が生成される



テスト、更新、 改善する

Make MSI とアプリ パッケージに含まれる共通コードを作成する

ユニバーサル Windows プラットフォームの利点を生かすようにアプリ パッケージを改善させる

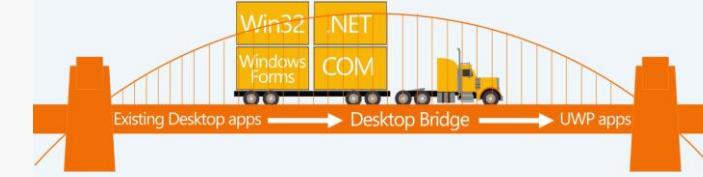


公開と配布

MDM や Windows ストアを使って、サイド ローディング用のアプリ パッケージを配布する

変換のための環境準備

必要なソフトウェア



Windows 10 Version 1607 Professional 以上

Desktop App Converter

<http://aka.ms/converter>

Windows SDK

<https://developer.microsoft.com/windows/downloads/windows-10-sdk>

ベースイメージ

<http://aka.ms/convertimages>

必要なハードウェア



ハードウェアの仮想化機構 (Second Level Address Translation) が必要

Coreinfo ユーティリティで確認が可能

<http://technet.microsoft.com/ja-jp/sysinternals/cc835722>

```
C:¥Sysinternals>coreinfo -v
```

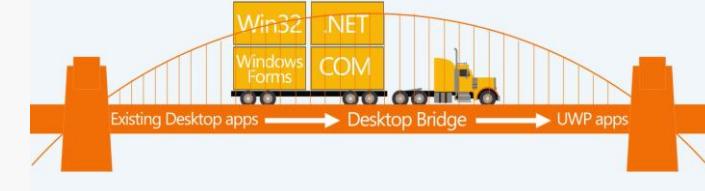
```
Coreinfo v3.4 - Dump information on system CPU and  
省略
```

```
HYPervisor * Hypervisor is present
```

```
VMX - Supports Intel hardware-assisted virtualization
```

```
EPT - Supports Intel extended page tables (SLAT)
```

ベースイメージのインストール



```
C:> DesktopAppConverter.exe -Setup -BaseImage  
BaseImage-15063.wim
```

ホスト OS のビルドと同じイメージを使用
-Setup オプションは 1 度だけ実施すればよい
Anniversary Update はビルド 14393
Creators Update はビルド 15063

変換作業

変換作業

変換パラメータ

```
C:¥> DesktopAppConverter.exe -Installer "[インストーラ]" -InstallerArguments "[サイレントインストール オプション]" -Destination "[出力先]" -PackageName "[パッケージ名]" -Publisher "[CN=開発者]" -Version [バージョン] -MakeAppx -Verify -Verbose
```

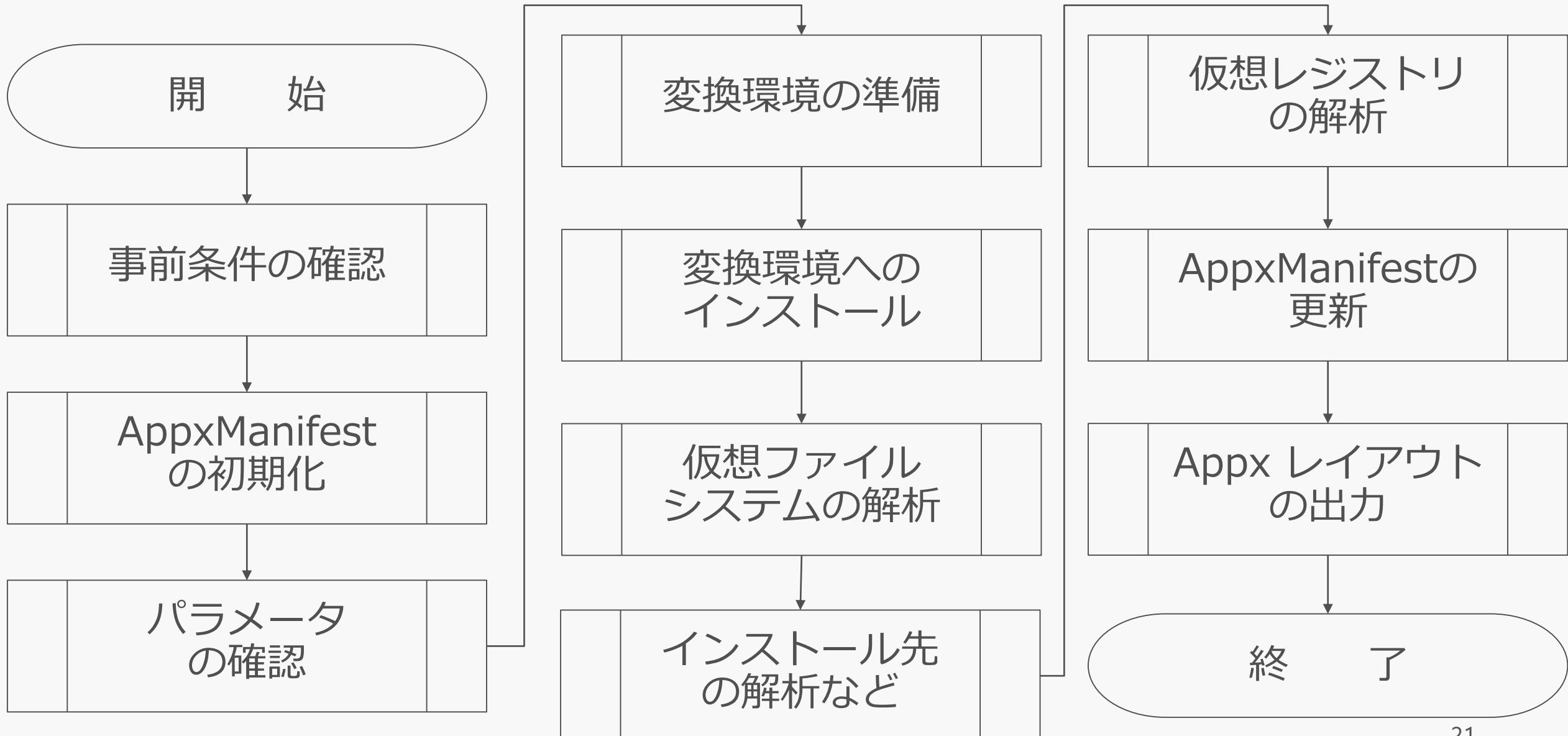


インストーラはサイレントインストールが必須
-Sign パラメータもある

<https://aka.ms/converterdocs>

※最新ドキュメントは英語版を参照

DAC の動作メカニズム



テスト

Desktop App Converter

既存のデスクトップ インス
トーラを変換

UWP マニフェストを持つ
たアプリ パッケージ
(APPX) が生成される



テスト、更新、 改善する

Make MSI とアプリ パッ
ケージに含まれる共通コード
を作成する

ユニバーサル Windows プ
ラットフォームの利点を生か
すようにアプリ パッケージを
改善させる



公開と配布

MDM や Windows ストアを
使って、サイド ローディン
グ用のアプリ パッケージを
配布する

テスト

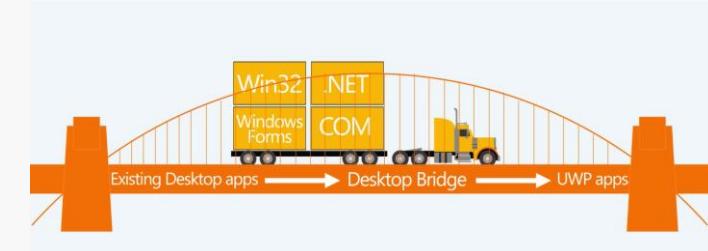
- ・アプリをインストール



```
PS C:\> Add-AppxPackage -Register "[変換した  
AppxManifest.xml]"
```

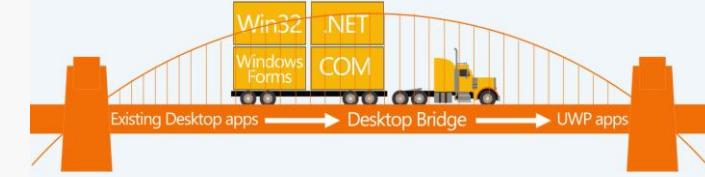
※変換直後は、開発者モードでのテストを推奨

テスト



- **UWP アプリとしての動作確認**
 - 全ての機能が正常に動作するかどうか
 - ストア経由で更新されると仮定して、問題がないかどうか
 - 変換可能なアプリの条件に抵触していないかどうか
- <https://msdn.microsoft.com/ja-jp/windows/uwp/porting/desktop-to-uwp-prepare>
 - (.NET Framework 4.6.1、カーネル モードのドライバー、etc)

更新、改善



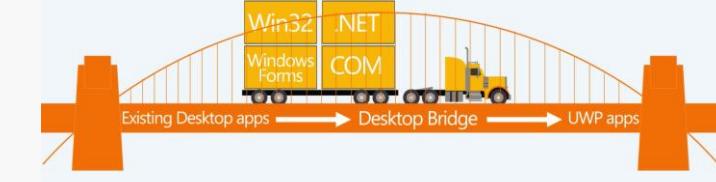
- 問題が見つかれば、プログラムを修正
- 修正したプログラムでテスト

```
PS C:\> Add-AppxPackage -Register "[変換した  
AppxManifest.xml]"
```

- 最後にAppx を再作成

```
C:\> MakeAppx.exe pack /d "変換した Appx レイアウトのフォル  
ダ" /p "作成する Appx のファイル名" /l
```

自己証明書で Appx への署名



- **makecert.exe, etc**

```
C:\> MakeCert.exe -r -h 0 -n "CN=開発者名" -eku  
1.3.6.1.5.5.7.3.3 -pe -sv pvk ファイル cer ファイル  
C:\> pvk2pfx.exe -pvk pvk ファイル -spc cer ファイル -  
pfx pfx ファイル  
C:\> signtool.exe sign -f pfx ファイル -fd SHA256 -v  
appx ファイル
```

-n に指定する開発者は、AppxManifest.xml と一致させる

アプリのインストール



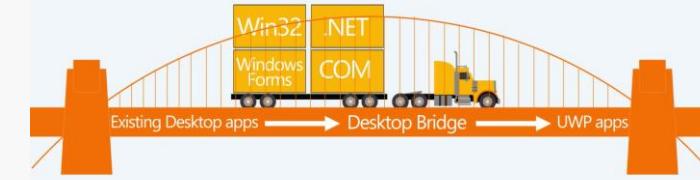
- ・アプリをインストール
署名した Appx をダブルクリック

```
PS C:\> Add-AppxPackage "[作成した Appx ファイル]"
```

インストールできない場合は、Powershell で実行
※一般ユーザー権限で問題無し

※ Appx パッケージをインストールする時は、証明書ファイル (cer) のインストールが必要。必ず、[ローカルコンピュータ]-[信頼されたユーザー]ストアへインストール

最終テスト



- 作成した Appx パッケージでテストします
 - 「Add-AppxPackage –Register AppxManifest.xml」との違いは、ファイルシステムのアクセス権
 - アプリは、インストール フォルダに書き込みができない

※ Appx パッケージをインストールする時は、証明書ファイル (cer) のインストールが必要。必ず、[ローカル コンピュータ]-[信頼されたユーザー] ストアへインストール

公開と配布

Desktop

App

Converter

既存のデスクトップ インストーラを変換

UWP マニフェストを持つ
たアプリ パッケージ
(APPX) が生成される



テスト、更新、
改善する

Make MSI とアプリ パッケージに含まれる共通コードを作成する

ユニバーサル Windows プラットフォームの利点を生かすようにアプリ パッケージを改善させる



公開と配布

MDM や Windows ストアを使って、サイド ローディング用のアプリ パッケージを配布する

変換したアプリを公開する方法



- サイド ローディング

- 企業内専用のアプリ

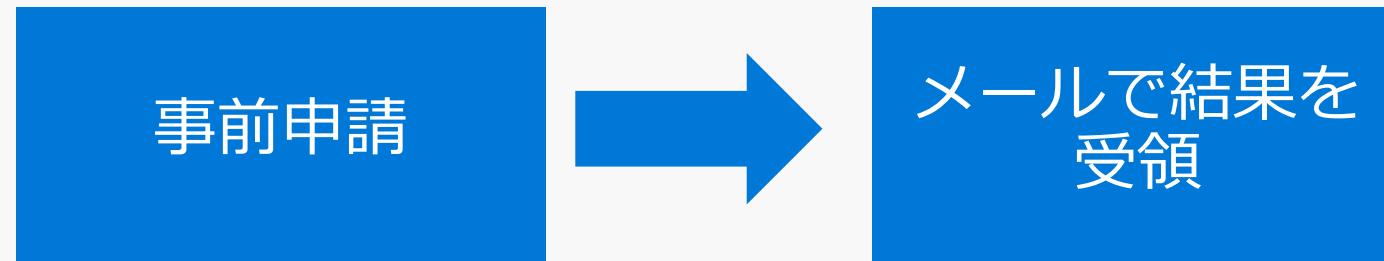
- Windows ストア

- 一般公開するアプリ

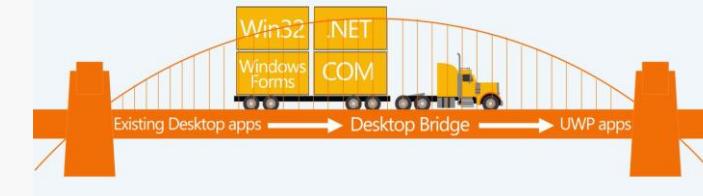
ストアへ提出するには

- 事前申請が必要

<https://developer.microsoft.com/en-us/windows/projects/campaigns/desktop-bridge>



- 提出用の Appx パッケージの作成



事前申請

- 名前
(First Name、Last Name)
- Email Address
- 会社名 (オプション)
- 製品 URL (App URL)
- マネタイズの手法
- 国
- アプリを公開する理由を
詳細に記述

Bring your existing apps and games to the Windows Store with the Desktop Bridge

Convert your apps and games to provides clean installation and updates to your users. You can distribute the app in the Windows Store or use your existing distribution channels.

How to publish Overview Get started

How to publish your desktop app or game to Windows Store

If you're ready to bring your existing desktop app or game to the Windows Store, let us know.

The Desktop Bridge enables developers to bring their existing apps and games to the Universal Windows Platform. If you are using the Desktop Bridge with your existing app or game and are ready to distribute it through the Windows Store, let us know by submitting the form and our team will work closely with you to manage the process of getting your converted app published in the Windows Store.

Please note that you need to be the developer and/or publisher of the app or game to bring it to the Windows Store. As such, make sure your name and e-mail address match with the website you submit as the URL below, so we can validate you are the developer and/or publisher.

Enter your information below:

First Name

Last Name

Email Address

Company Name (optional)

App URL (e.g. website, Download.com, Steam)

- Select your current monetization mod ▾

- Select your country - ▾

Please provide any other relevant information here, such as your current experience converting your desktop

Submit



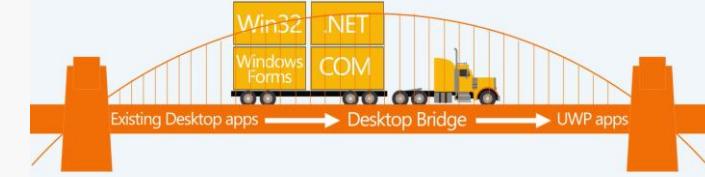
タイルリソースの準備

・タイルリソースの入れ替え

要素名	ファイル名	用途
Logo	AppStoreLogo.png	ストアロゴ(50 x 50 px)
Square150x150Logo	AppMedTile.png	タイルの「中」サイズ
Square44x44Logo	AppList.png	アプリ一覧
Wide310x150Logo	AppWideTile.png	タイルの「横長」サイズ
Square310x310Logo	AppLargeTile.png	タイルの「大」サイズ
Square71x71Logo	AppSmallTile.png	タイルの「小」サイズ

- AppStoreLogo.scale-xxx.png の形式になり、サイズは 100%、125%、150%、200%、400% の 5種類。
- AppList.png は、targetsize-xx と targetsize-xx_altform-unplated も必要で、サイズは 16、24、32、48、256 の 5 種類。合計で 10 種類。

PRI リソースの作成



- タイル リソースのファイル名を変更したり、
タイル リソースを追加した場合は、PRI リソース
が必要

PRI 構成ファイルが必要

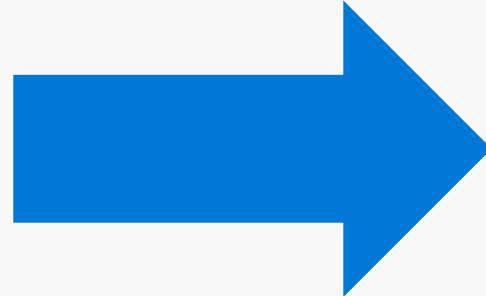
```
C:¥> MakePri.exe new -/pr [Appx レイアウト フォルダ]  
/cf [PRI 構成ファイル] /o
```

PriConfig.xml と layout.resfiles は
DAC v1.0.6 以上の icon_extract より流用

提出用に AppxManifest を編集



Visual Studio でストアと
関連付けたプロジェクトを
準備



関連付けたプロジェクトの
Package.appxmanifest
を参照して
AppxManifest.xml を編集

提出用アプリ パッケージの作成方法

- 手動で作成する方法

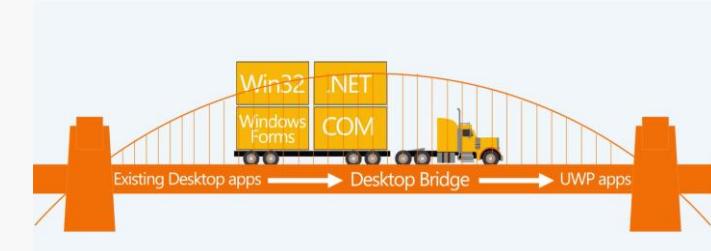
- MakeAppx.exe ユーティリティなどを使用

- Visual Studio で作成する方法

- HTML/JavaScript の アプリプロジェクトを使用
 - Package.appxmanifest 以外をプロジェクトから削除
 - 作成した Appx レイアウトをプロジェクトを追加
 - Package.appxmanifest に AppxManifest.xml をマージ

Package.appxmanifest の抜粋

AppxManifest の修正箇所



- Identityt 要素
 - Name 属性
 - Publisher 属性
 - Version 属性
- DisplayName 要素
- PublisherDisplayName 要素
- uap:VisualElements 要素の DisplayName属性

Appx パッケージの作成



- **MakeAppx.exe**

```
C:> MakeAppx.exe pack /d "変換した Appx レイアウトのフォルダ" /p "作成する Appx のファイル名" /l
```

- /l オプションは、PRI リソースを使用する場合

自己証明書で Appx への署名

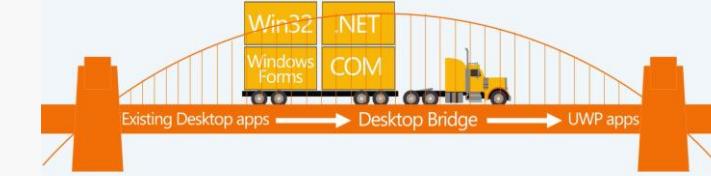


- **makecert.exe, etc**

```
C:\> MakeCert.exe -r -h 0 -n "CN=開発者名" -eku  
1.3.6.1.5.5.7.3.3 -pe -sv pvk ファイル cer ファイル  
C:\> pvk2pfx.exe -pvk pvk ファイル -spc cer ファイル -  
pfx pfx ファイル  
C:\> signtool.exe sign -f pfx ファイル -fd SHA256 -v  
appx ファイル
```

- -n に指定する開発者は、Windows ストアと一致させる

提出前にツールでテスト



- **WACK (Windows App Certification Kit)**

- 最新の Windows SDK で提供
- 署名済みの Appx パッケージをテスト
- Windows ストアへ提出すると、WACK を使った自動テストを実施

※ 証明書ファイル (cer) のインストールが必要

Windows App Certification Kit - Test Results

App name:

Desktop App Converter

App publisher:

Microsoft Corporation

App version:

2.0.2.0

App Architecture:

x64

Kit Version:

10.0.15063.137

OS Version:

Microsoft Windows 10 Pro (10.0.15063.0)

OS Architecture:

x64

Report time:

2017/06/21 11:10:28

Overall result: PASSED

Windows 10 S

Windows 10 S 環境でのテスト



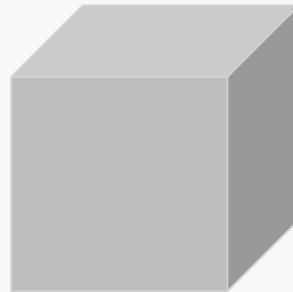
- Windows 10 S と同じロックダウンでの動作確認
 - Windows 10 Pro の仮想マシンに環境を作成
 - <https://docs.microsoft.com/en-us/windows/uwp/porting/desktop-to-uwp-test-windows-s>
 - デバイス ガード ポリシーのインストール
インストールしたポリシー ファイル名のリネーム
 - 同梱された証明書で Appx パッケージへ署名
※ AppxManifest の Publisher を変更
 - 作成した Appx パッケージで動作確認

アプリを進化させるには

アプリを進化させるには

従来のまま

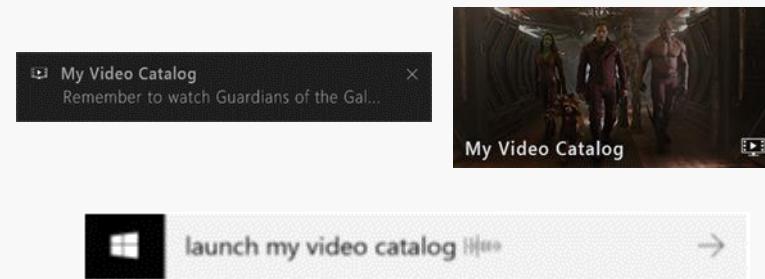
Win32 からのマイグレーション



基本的な APPX パッケージ
ストアでのアプリ公開

拡張

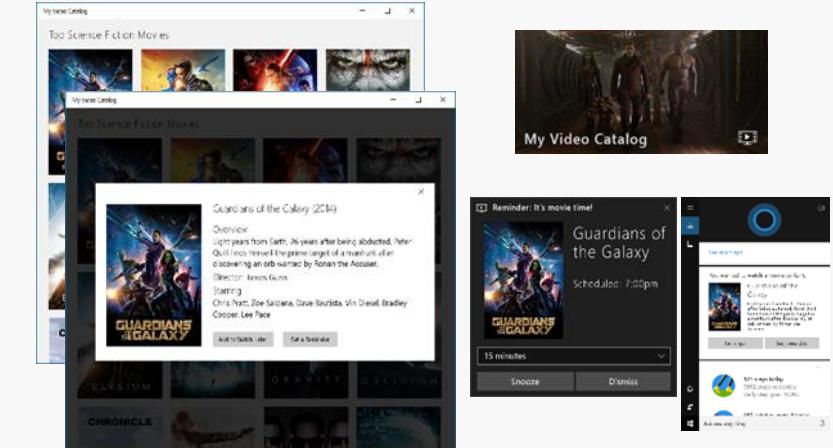
Windows 10 の機能を有効活用



従来の成果に対して
通知
タイル
アプリ起動の基本的な Cortana
サポート
UWP の機能を容易に追加できる

モダナイズ

プラットフォーム向けに最適化



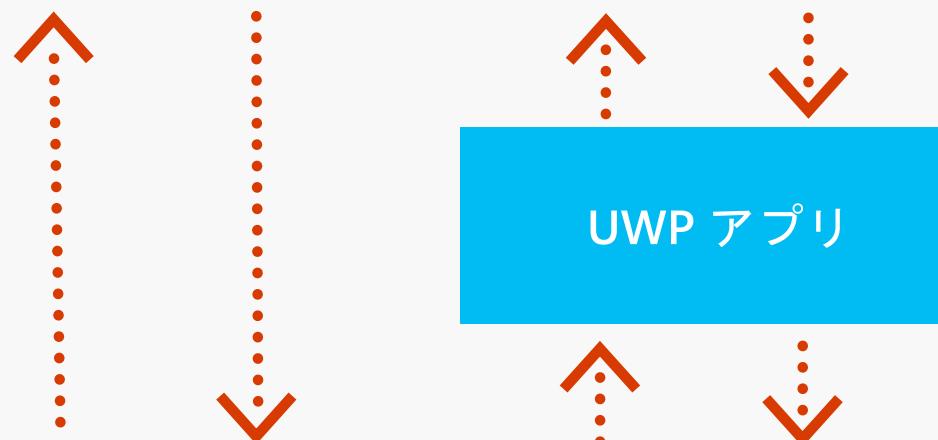
従来の成果に対して
アプリ レイアウトのモダナイズ
UWP コンポーネントの追加
クラウドへの拡張
Cortana サポートとの統合
その他の UWP プラットフォームへの拡大:
モバイル, HoloLens etc…

デスクトップ ブリッジプロセス



ユニバーサル アプリ パッケージ

クラシック Windows アプリ



アプリ コンテナ

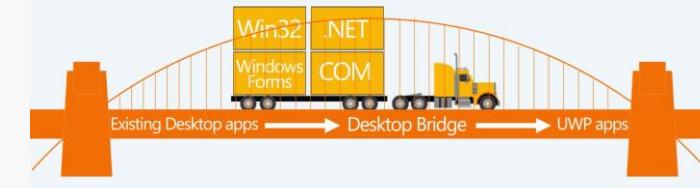
インストーラを変換したアプリ
Windows 10 S で動作する

アプリ コンテナがデスクトップ
アプリと同居するモデル
Windows ストアから配布
部分信頼と完全信頼のハイブリッド
デスクトップ アプリより制限が
存在することに注意

アプリのマイグレーションステップ

ステップ	バイナリ	エントリー ポイント	.NET Native	F5 デバッグ
1 変換	Win32	Win32	N/A	VS 拡張
2 強化	WinMD を参照	Win32	N/A	VS 拡張
3 拡張	Win32 + CoreCLR	Win32	開発者の責任	VS 拡張
4 マイグレート	CoreCLR + Win32	UWP	開発者の責任	VS
5 UWP	CoreCLR	UWP	ストアの責任	VS

Creators Update からの新機能



- **COM サーバーの公開**

- アウトプロセス サーバー
- プレビュー ハンドラ、サムネイル ハンドラ、プロパティ ハンドラ、etc

- **ファイアウォールルールの定義**

- **カスタム フォントの公開**

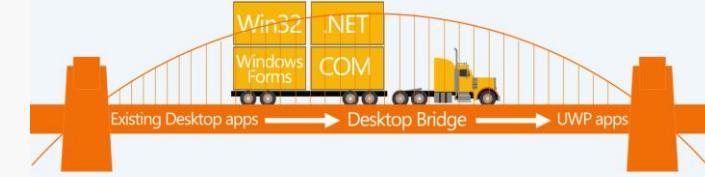
- **アプリの移行期間における対応機能**

- **WACK サポート**

- <https://docs.microsoft.com/en-us/windows/uwp/porting/desktop-to-uwp-run-desktop-app-converter>

まとめ

まとめ



- **ユニバーサルアプリパッケージ**
 - ゼロインパクトインストール
 - 標準化された OPC フォーマット
- **Desktop App Converter** は、インストーラの変換ツール
- **Appx パッケージによるテストが必須**
- **ストアへの提出には、事前申請が必要**
- **ストア提出用にマニフェストの編集が必要**
- **UWP アプリにできないことが可能に**

Appendix

手作業による変換



• Appx レイアウトの作成

1. 作業用フォルダの作成
2. 作業用フォルダにプログラムを配置
たとえば、
C:¥DAC 作業用フォルダ
- ¥Input : ここにプログラムを配置

XCOPY でインストールが可能な場合

手作業による変換

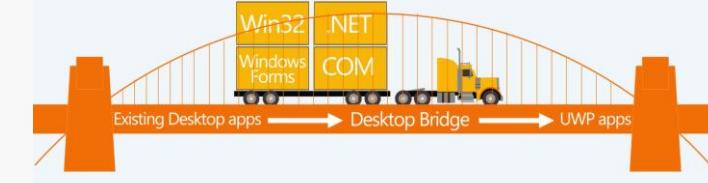


• Appx レイアウトの作成

```
C:> DesktopAppConverter.exe -Installer "[プログラムを  
配置したフォルダ名]" -AppExecutable "[実行ファイル名]" -  
Destination "[出力先]" -PackageName "[パッケージ名]" -  
Publisher "[CN=開発者]" -Version [バージョン] -MakeAppx  
-Verbose
```

Appx パッケージが出力されます
Desktop App Converter v1.0.9 以上

AppxManifest の特徴



```
<Capabilities>
    <rescap:Capability Name="runFullTrust" />
</Capabilities>
```

```
<Application Id="MyDesktopAppStep3" Executable="MyDesktopApp.exe"
EntryPoint="Windows.FullTrustApplication">
```

```
<TargetDeviceFamily Name="Windows.Desktop"
MinVersion="10.0.14393.0" MaxVersionTested="10.0.14393.0" />
```

VC++ ランタイムを組み込む場合

```
<Dependencies>
  <TargetDeviceFamily Name="Windows.Desktop"
    MinVersion="10.0.14316.0" MaxVersionTested="10.0.14316.0" />
  <PackageDependency Name="Microsoft.VCLibs.140.00.UWPDesktop"
    MinVersion="14.0.24217.0"
    Publisher="CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond,
    S=Washington, C=US" />
</Dependencies>
```

VC 14.0 framework packages for Desktop Bridge

<https://blogs.msdn.microsoft.com/vcblog/2016/07/07/using-visual-c-runtime-in-centennial-project>

※テスト環境では、VC ランタイム パッケージのインストールが必要

VC++ ランタイムを組み込む場合

Visual Studio 2015 (VC++ 2015 - 14.0)

```
<PackageDependency Name="Microsoft.VCLibs.140.00.UWPDesktop"  
MinVersion="14.0.24217.0" Publisher="CN=Microsoft Corporation, O=Microsoft  
Corporation, L=Redmond, S=Washington, C=US" />
```

Visual Studio 2013 (VC++ 2013 - 12.0)

```
<PackageDependency Name="Microsoft.VCLibs.120.00.UWPDesktop"  
MinVersion="12.0. 40653.0" Publisher="CN=Microsoft Corporation, O=Microsoft  
Corporation, L=Redmond, S=Washington, C=US" />
```

Visual Studio 2012 (VC++ 2012 - 11.0)

```
<PackageDependency Name="Microsoft.VCLibs.110.00.UWPDesktop" MinVersion="  
11.0. 61135.00" Publisher="CN=Microsoft Corporation, O=Microsoft Corporation,  
L=Redmond, S=Washington, C=US" />
```

Desktop to UWP Packaging Project

Visual Studio 2017 用の拡張パッケージ (VSIX)

<http://go.microsoft.com/fwlink/?LinkId=797871>

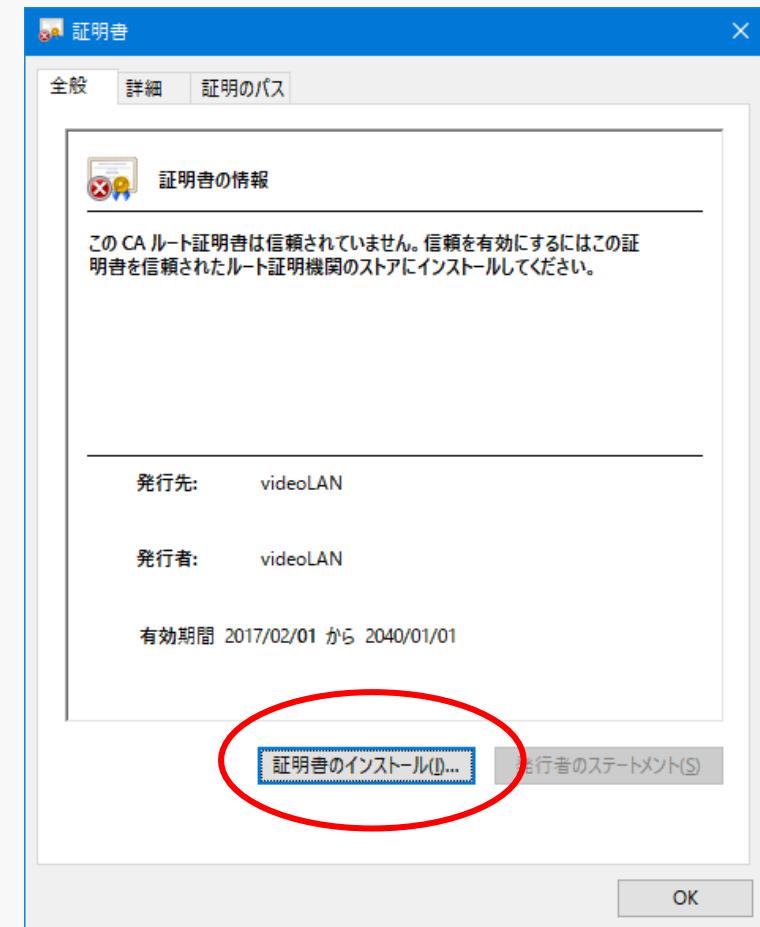
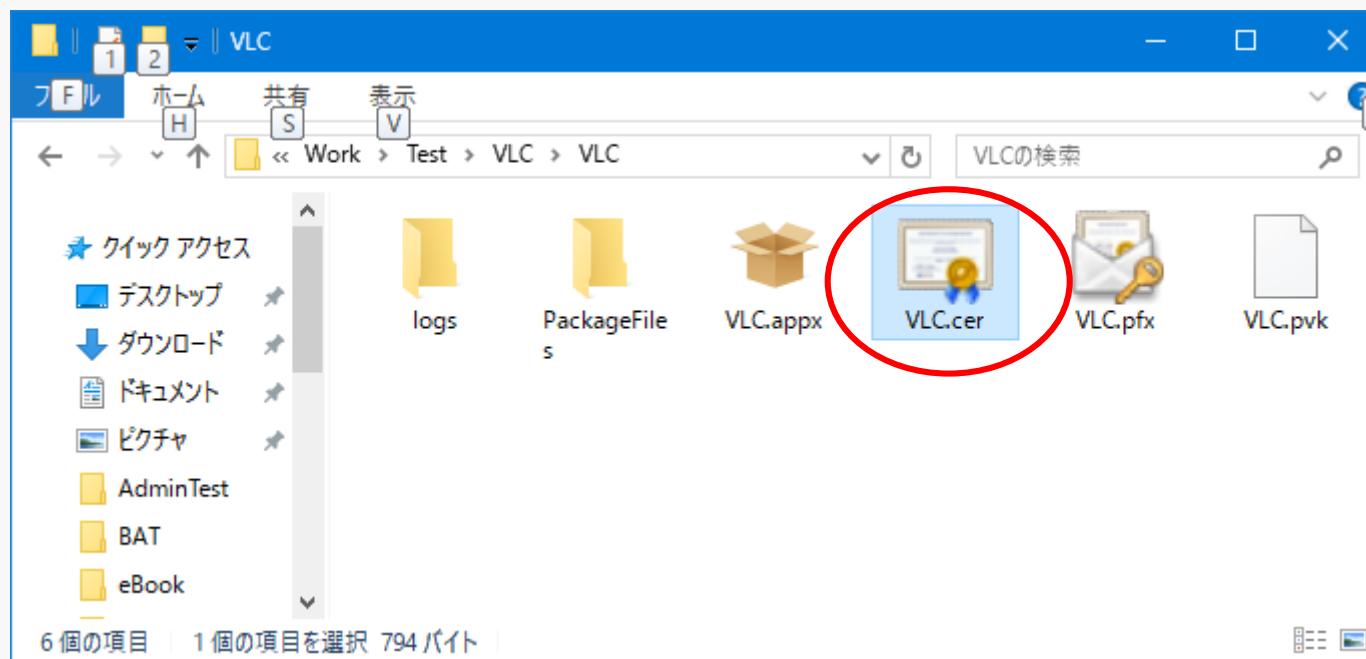
DAC で変換した Appx レイアウトを使用して、
Visual Studio でデバッグや拡張をし易くする

Desktop Bridge を使った拡張サンプル

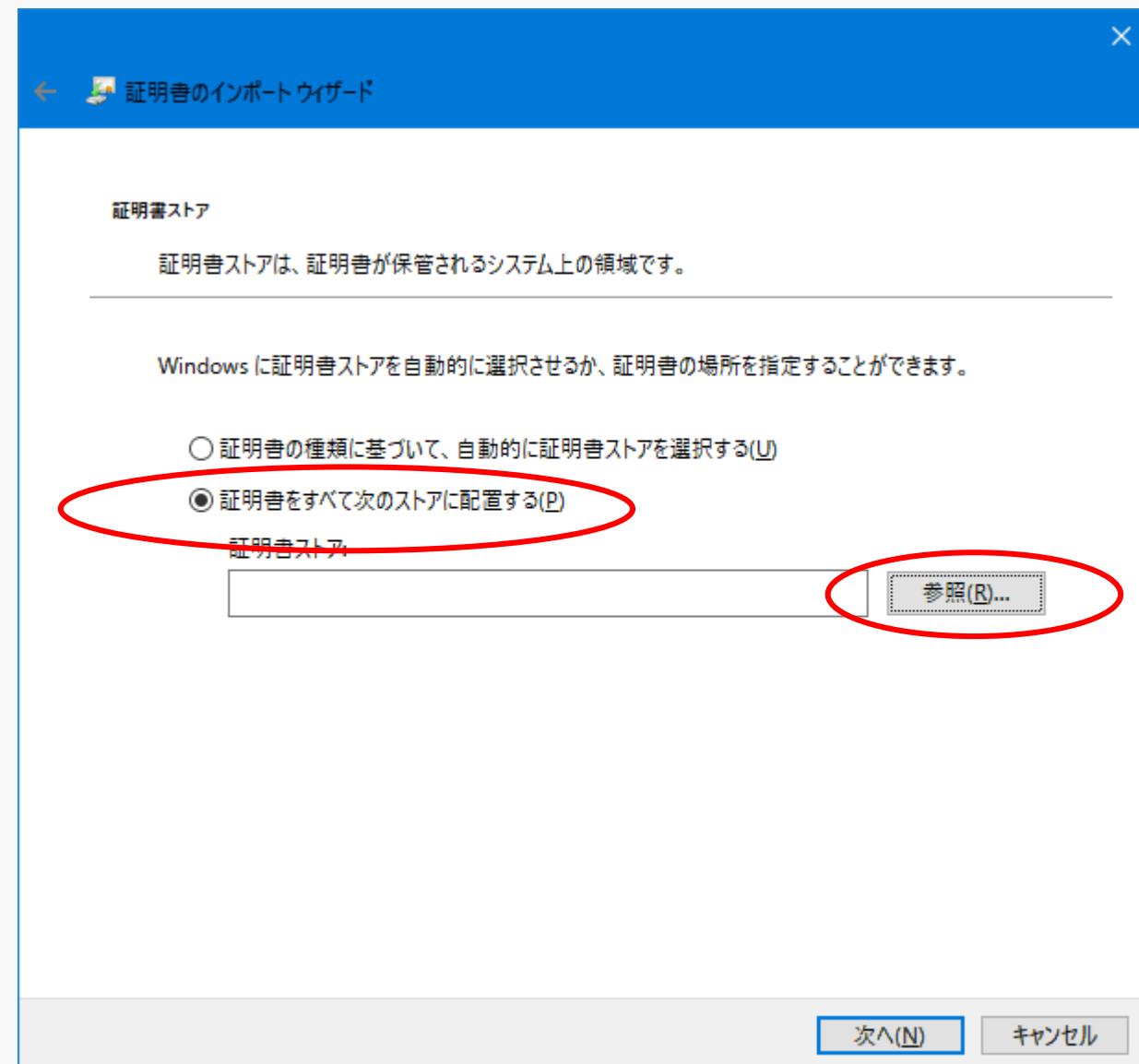
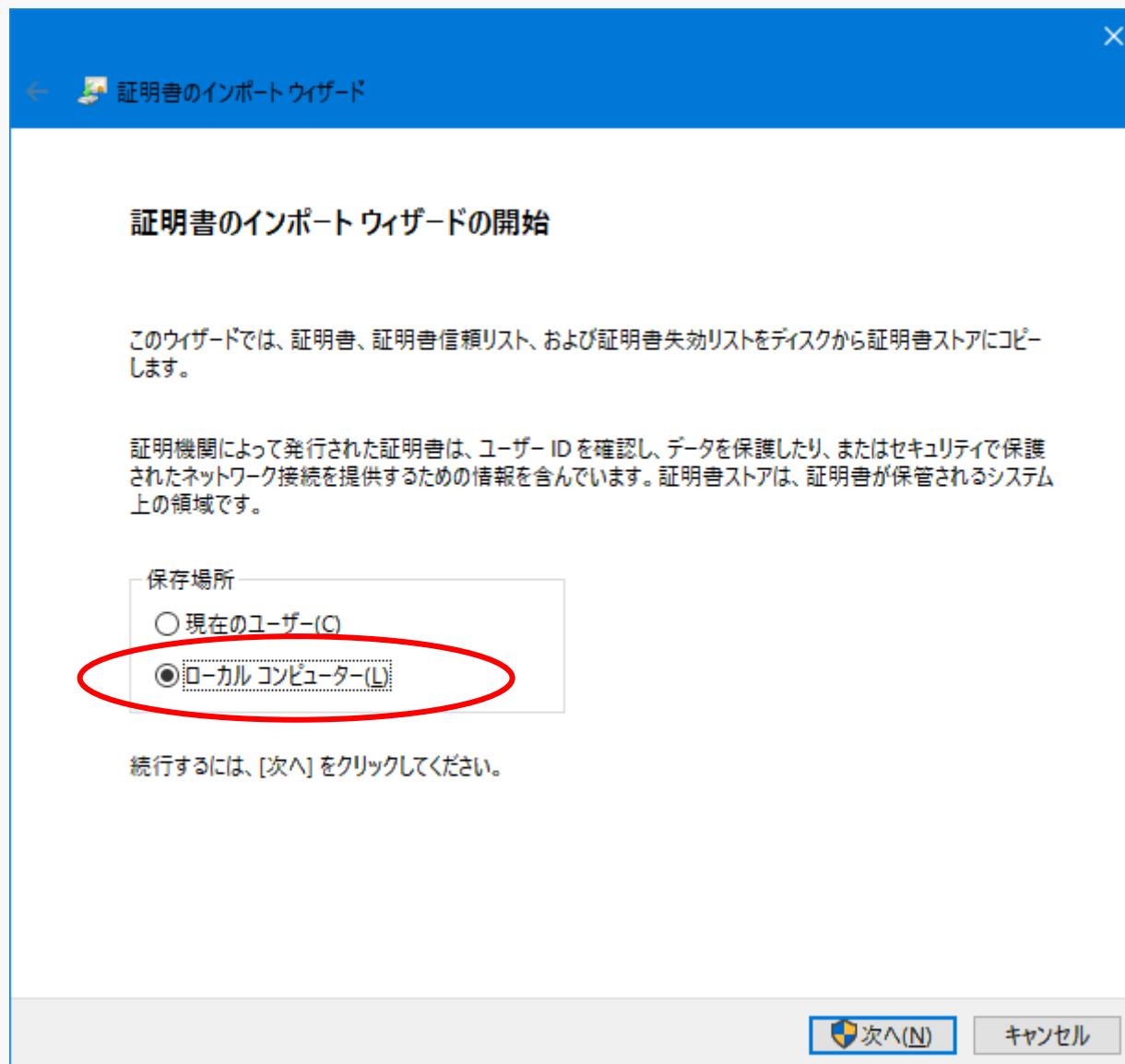
<https://github.com/Microsoft/DesktopBridgeToUWP-Samples>

証明書をインストール 1/4

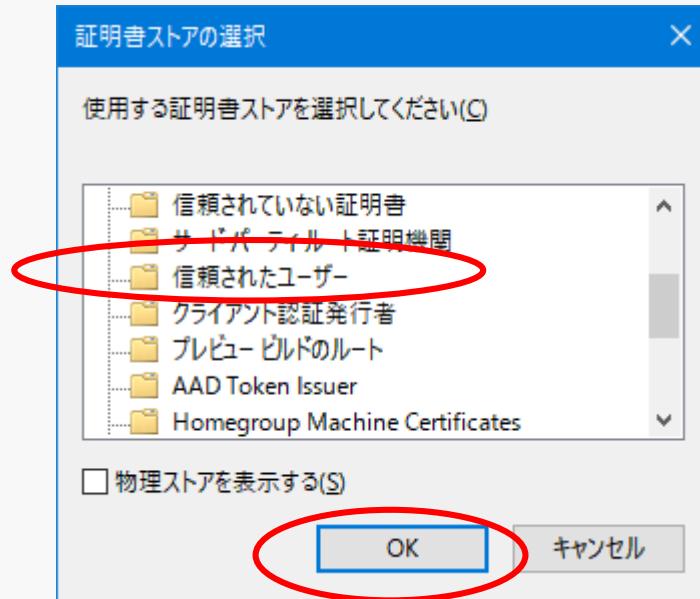
Windows エクスプローラで証明書をダブルクリックします



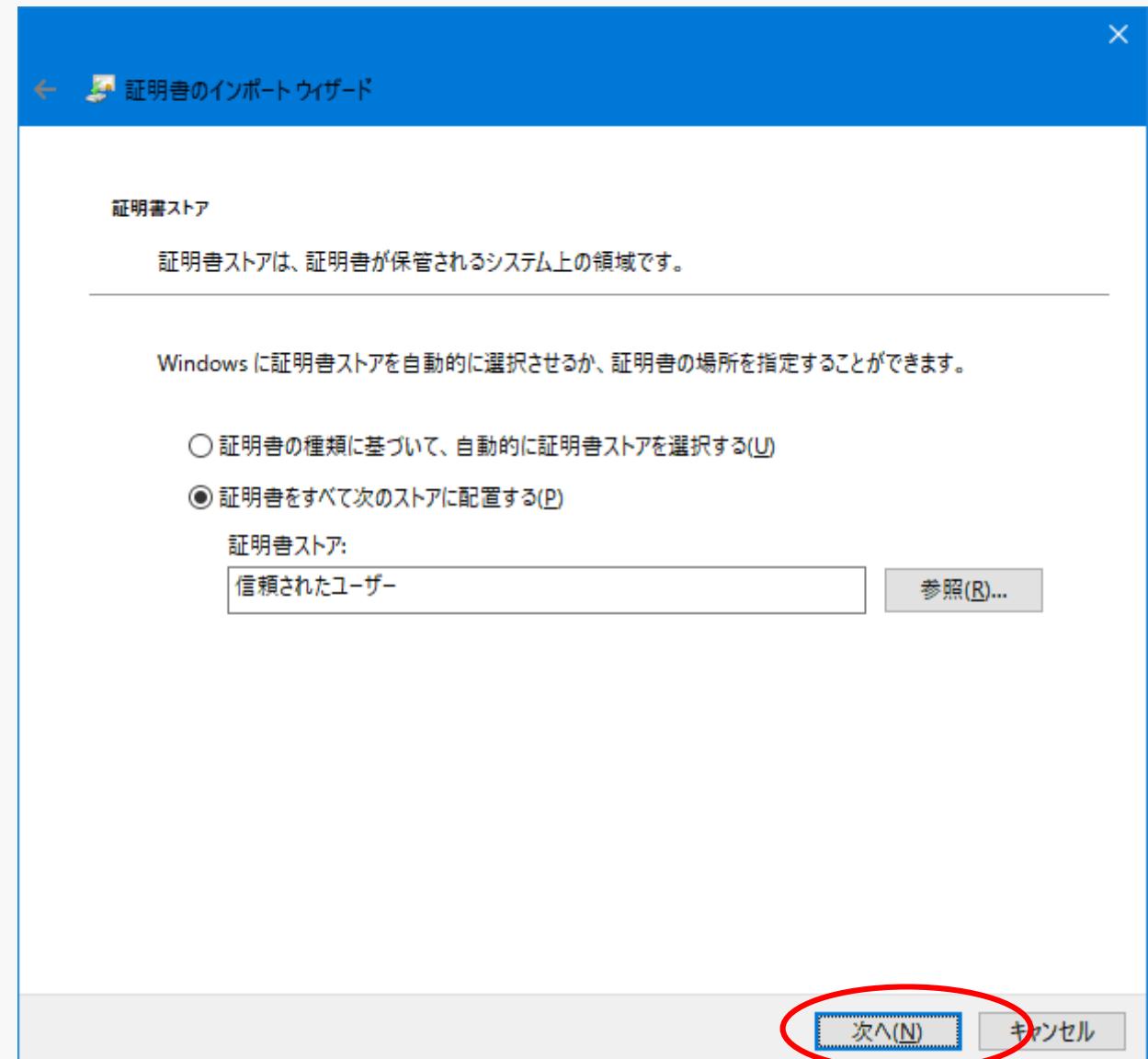
証明書をインストール 2/4



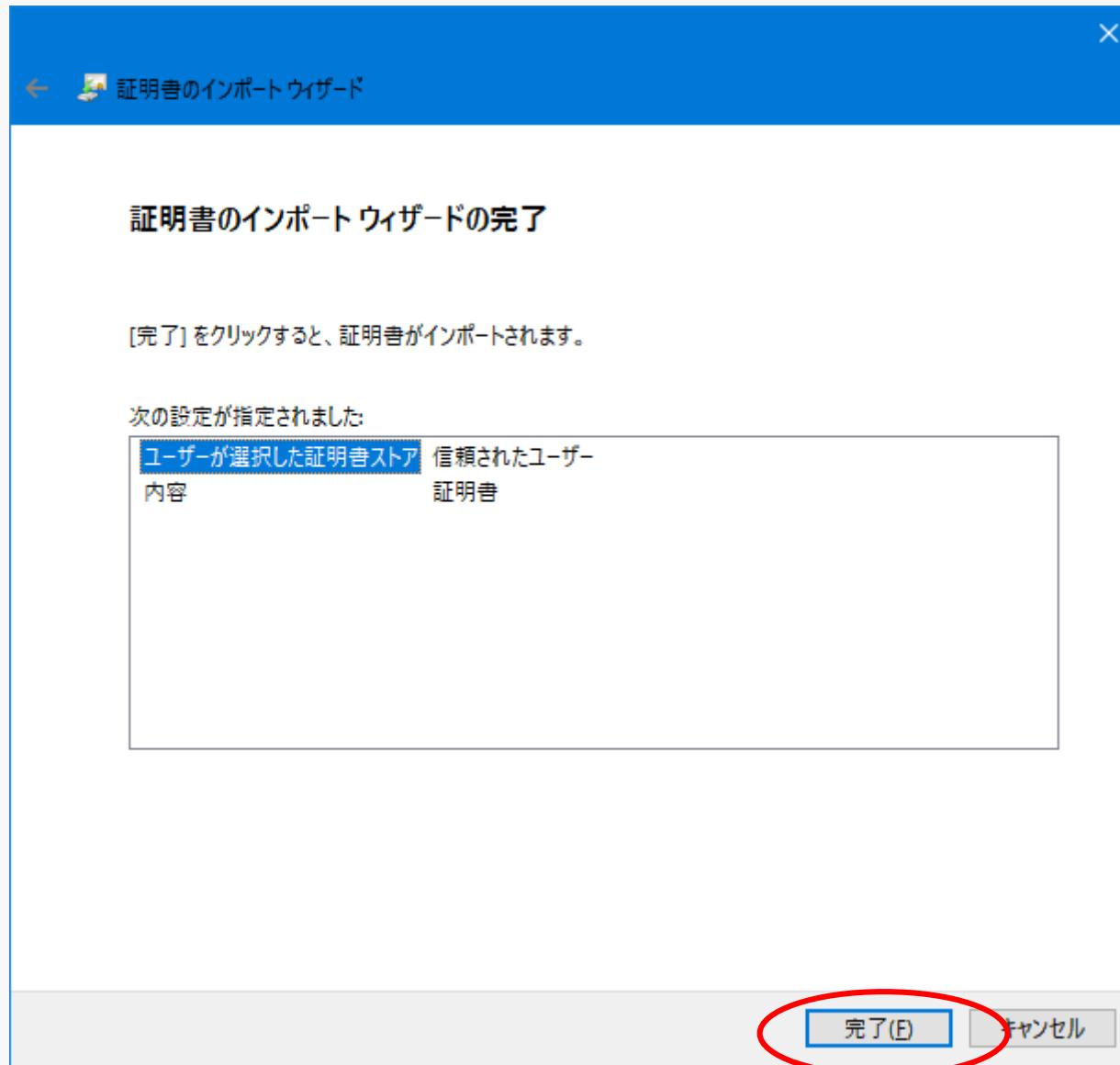
証明書をインストール 3/4



[ローカルコンピュータ]-
[信頼されたユーザー]
ストアを選択します

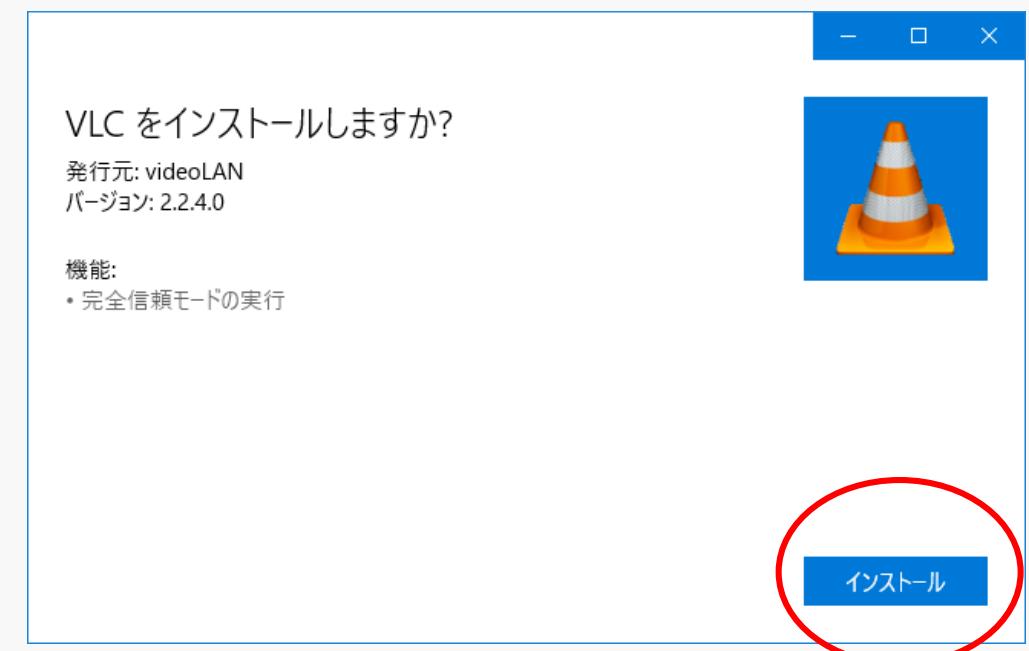
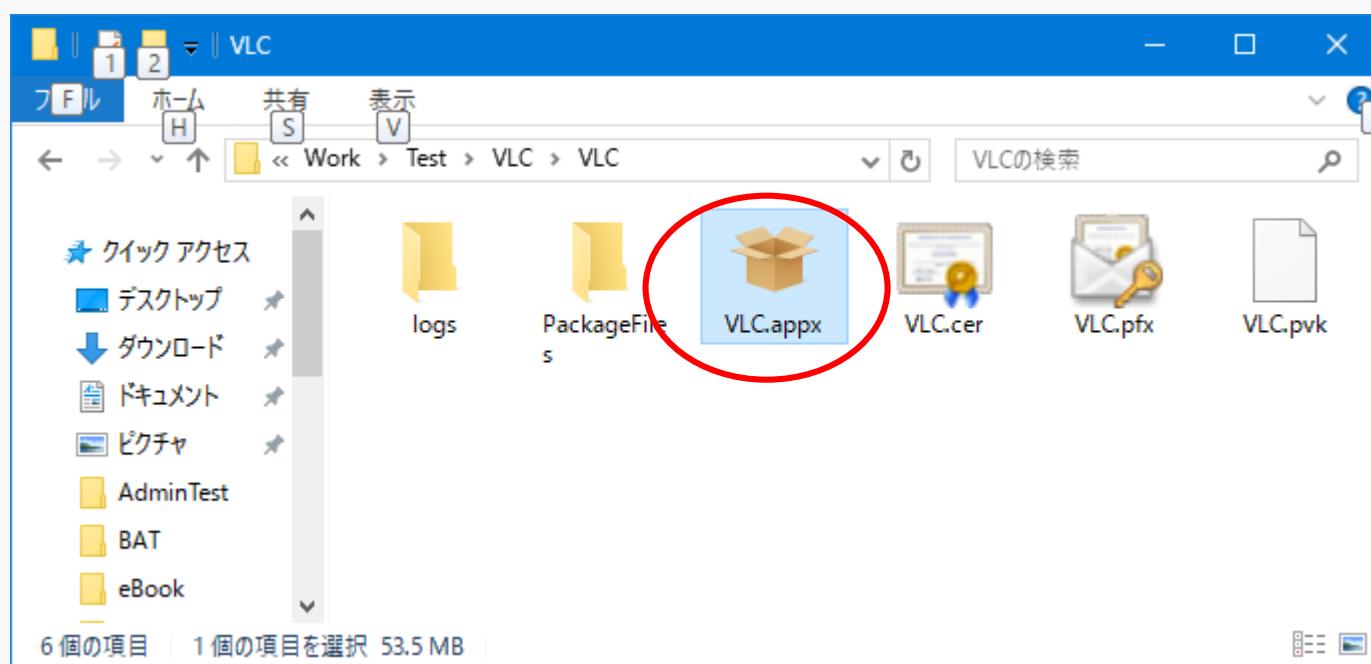


証明書をインストール 4/4



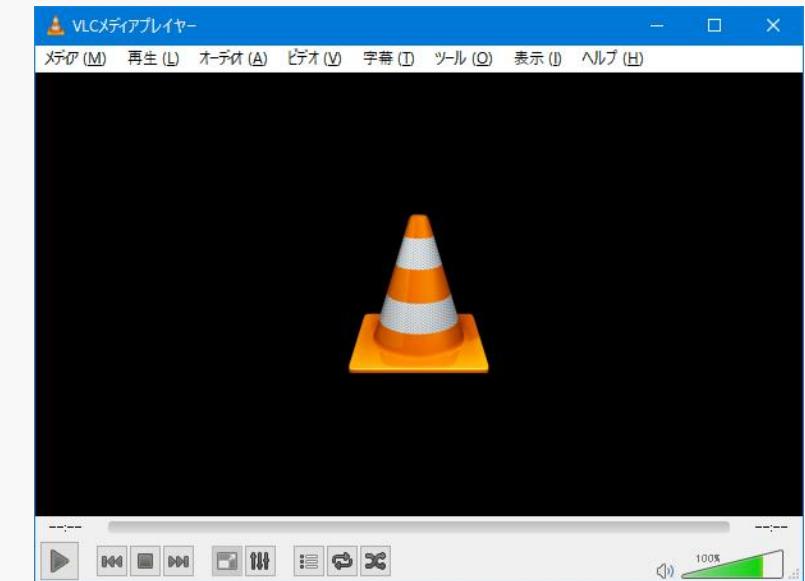
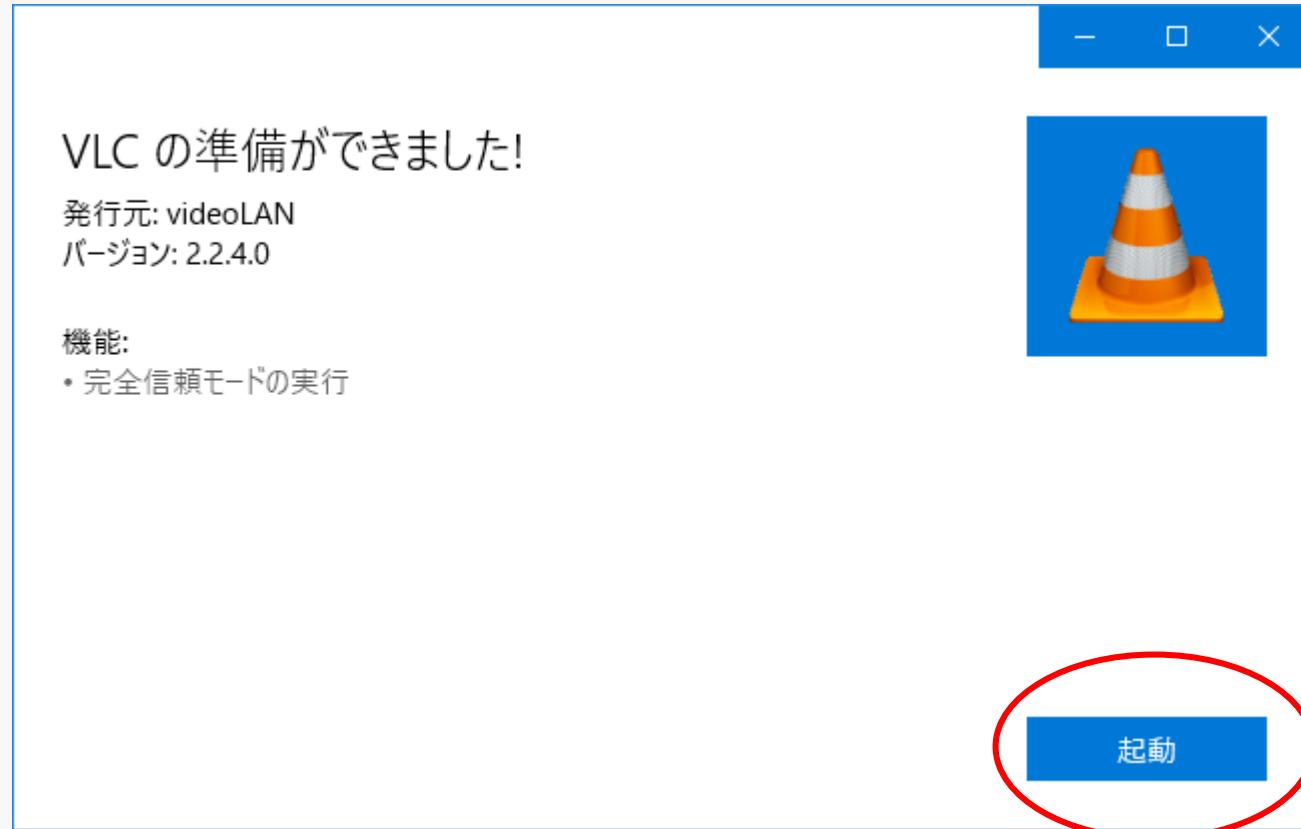
アプリのインストール

Windows エクスプローラで Appx をダブルクリックします



アプリの起動

アプリを起動して、動作を確認します



アプリのインストール (参考)

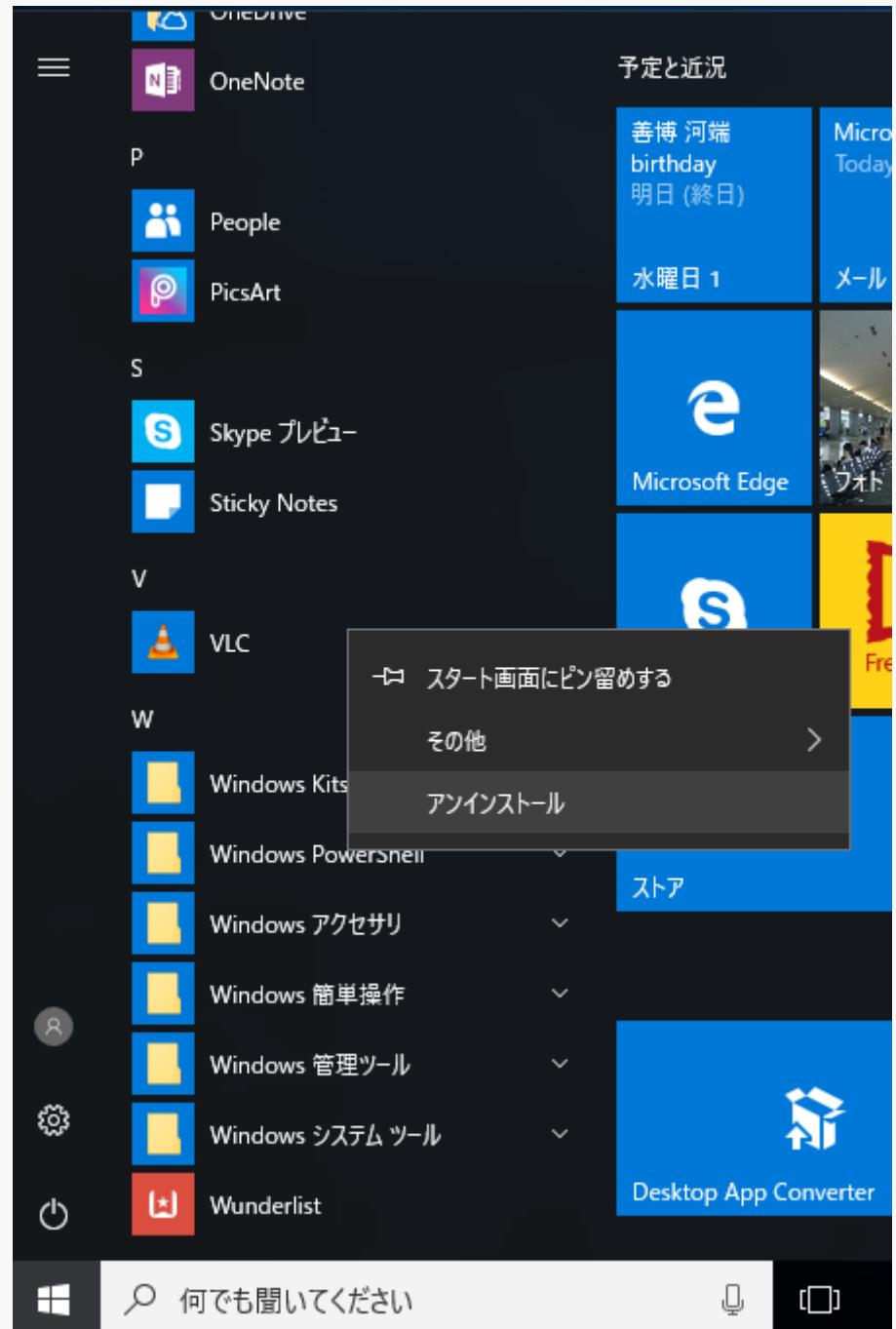
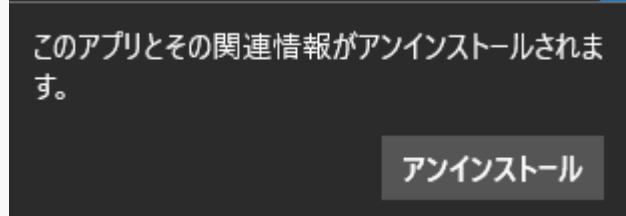
Appx のダブルクリックでインストールできない場合は、PowerShell を使用します

```
C:¥DAC¥VLC> powershell  
Windows PowerShell  
Copyright (C) 2016 Microsoft Corporation. All  
rights reserved.
```

```
PS C:¥DAC¥VLC> Add-AppxPackage .¥VLC.appx
```

アプリのアンインストール

スタートメニューから、VLC を選択して、コンテキストメニューからアンインストールを選択します



環境変数の取り扱いについて

コマンドプロンプト (CMD.EXE)

```
C:¥DAC> CD VLC  
C:¥DAC¥VLC> SET PATH="C:¥Program Files  
(x86)¥Windows Kits¥10¥bin¥10.0.15063.0¥x86";%PATH%
```

Power Shell (PowerShell.exe)

```
PS C:¥DAC> CD VLC  
PS C:¥DAC¥VLC> $Env:Path += ";C:¥Program Files  
(x86)¥Windows Kits¥10¥bin¥10.0.15063.0¥x86"
```

```
PS C:¥DAC¥VLC> $Env:Path = "C:¥Program Files  
(x86)¥Windows Kits¥10¥bin¥10.0.15063.0¥x86;" +  
$Env:Path
```

マイグレーションステップ サンプル

Desktop app bridge to UWP Samples

<https://github.com/Microsoft/DesktopBridgeToUWP-Samples>

JourneyAcrossTheBridge サンプル

ステップ 1 からステップ 5 に対応したサンプルを収録

