MyID PIV

Version 12.11

MyID Client MSIX Installation Guide

Lutterworth Hall, St Mary's Road, Lutterworth, Leicestershire, LE17 4PS, UK
www.intercede.com | info@intercede.com | @intercedemyid | +44 (0)1455 558111

Copyright

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

© You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. ---

Conventions used in this document

- Lists:
  - Numbered lists are used to show the steps involved in completing a task when the order is important.
  - Bulleted lists are used when the order is unimportant or to show alternatives.
- **Bold** is used for menu items and for labels.

  For example:
  - Record a valid email address in **'From' email address**.
  - Select **Save** from the **File** menu.
- *Italic* is used for emphasis:

  For example:
  - Copy the file *before* starting the installation.
  - Do *not* remove the files before you have backed them up.
- ***Bold and italic*** hyperlinks are used to identify the titles of other documents.

  For example: "See the ***Release Notes*** for further information."

  Unless otherwise explicitly stated, all referenced documentation is available on the product installation media.
- A `fixed width` font is used where the identification of spaces is important, including filenames, example SQL queries and any entries made directly into configuration files or the database.
- **Notes** are used to provide further information, including any prerequisites or configuration additional to the standard specifications.

  For example:

  Note: This issue only occurs if updating from a previous version.
- Warnings are used to indicate where failure to follow a particular instruction may result in either loss of data or the need to manually configure elements of the system.

  For example:

  Warning: You must take a backup of your database before making any changes to it.

Contents

1 Introduction

This document contains instructions on using MSIX to install your MyID® client software. MSIX is a Microsoft technology for packaging your apps. It supports the following features:

- Combining all of your client software in a single installation package.

  The MyID Operator Client requires the MyID Client Service, and makes use of MyID Desktop (for administrative operations) and the Self-Service App (for self-service operations). You are recommended to combine all of these applications into a single package for distribution to your MyID Operator Client users.
- Distributing a configuration package containing all required client configuration and customization.

  For example, the location of the MyID server, or files for rebranding.
- Configuring automatic updates.

The installation structure comprises an MSIX installer for the MyID Client Suite, which contains all of the common files for the MyID client software, and MSIX installers for the following client applications:

- MyID Desktop
- Self-Service App
- MyID Client Service

The installers are provided in the following folder in the MyID release image:

```
\MyID Clients\MyID Client Suite
```

Intercede also provides MSIX versions of the following additional software:

- Aware PreFace
- Canon SDK
- SecuGen biometric library
- Fargo Printer Support

These additional installers are distributed separately, but you can combine them into a package for distribution and installation with the rest of the client software.

2 Overview

The MyID MSIX client packages make use of both version-specific optional packages in a related set, and version-independent modification packages. The packages in the related set are tightly-coupled and must be serviced together, whereas the modification packages can be updated independently.

Note: Related set packages require the `MyIDClientSuite.msixbundle` to be used; if you install the `.msix` you will not be able to install the other packages in the related set, as the metadata for the set is only available in the `.msixbundle` package.

The following describes the version-specific optional packages in the related set:

- `MyIDClientSuite`

  This is the main package to which additional packages are applied. Must be installed.

  By default, the Windows start menu entry launches the MyID Client Configurator.

  This package contains the application files for the MyID Client Service, MyID Desktop, and the Self-Service App. However, the package does not add start menu entries for these applications – the start menu entries are added by the additional packages for each application.

  The package does registers protocol handlers to launch the clients without start menu entries; this may be useful for a minimal install where users will be interacting with MyID only though email links, which use these protocols. You can disable the protocol handlers if necessary using Windows configuration.

  Registered protocols are:

  - MyID Desktop: `myiddsk`
  - Self-Service App: `myidssa`
  - MyID Client Service: `myidmcs`
  - MyID Client Configurator: `myidconfig`

  Note: The `MyIDClientSuite` package is provided in the following versions:

  - `MyIDClientSuite_x.x.x.msixbundle`

    Used as part of a related set with the MyID Desktop, Self-Service App, and MyID Client Service packages. Use this version if you intend to install any of these related packages. See section *4.3*, *Related sets*.

  - `MyIDClientSuite_x.x.x.msix`

    Used in the MSIX Packaging Tool when creating a modification package. Installs the protocols for the MyID client applications, but cannot be updated by the related packages to add start menu items and aliases for the client applications. See section *3.4.1*, *MSIX Packaging Tool*.

- `DSKLauncher`

  Adds a start menu entry for MyID Desktop, and registers the `MyIDDesktop.exe` alias.

- `SSALauncher`

  Adds a start menu entry for the Self-Service App, and registers the `MyIDApp.exe` alias.

- `MCSLauncher`

  Adds a start menu entry for the MyID Client Service, and registers the `MyIDClientService.exe` alias.

There are also version-independent modification packages available, not all of which are supported for all editions of MyID:

- `AwarePrefaceForMyID`

  Adds Aware PreFace components, enabling biometric image capture through the MyID Client Service.

- `CanonEDSdkForMyID`

  Adds Canon EDSDK components, enabling Canon camera support for biometric image capture.

- `FargoApiForMyID`

  Adds Fargo SDK components, enabling advanced features for Fargo card printers.

- `SecuGenForMyID`

  Adds SecuGen SDK components, enabling support for SecuGen fingerprint readers.

2.1 Prerequisites

MSIX installation is supported on Microsoft Windows clients. The minimum version of Windows 10 is version 1809. However, note that for versions earlier than 1909 you must enable sideloading for apps not provided through the Microsoft Store; see the following Microsoft article for details:

docs.microsoft.com/en-us/windows/msix/app-installer/troubleshoot-appinstaller-issues#prerequisites

The MSIX installers provided with this release are designed for a system administrator to put together a package for the users across your organization, including an App Installer package that includes all of the required client software in a single package, and a modification package that contains the necessary client configuration (for example, the location of your MyID server).

See section *4*, *Deploying MyID clients with App Installer files* and section *3*, *Customizing and configuring a client installation* for details.

Accordingly, you must have a good knowledge of implementing MSIX software distribution. This guide does not provide a tutorial for MSIX and is not expected to replace the Microsoft MSIX documentation.

If you need to install the MyID client software on a single PC, you are recommended to use the standard .msi installation packages. See the *Installing MyID Desktop* and *Installing the MyID Client Service* sections in the ***Installation and Configuration Guide*** and the *Installing the Self-Service App* section in the ***Self-Service App*** guide for details.

2.2 Differences

The MSIX versions of the MyID client software have some differences from the .msi-installed versions.

2.2.1 Configuration files

The location and structure of the configuration files are different if you install the MSIX versions. The client applications now share a configuration file, which contains only user-configurable configuration options, whereas the .msi-installed clients have individual configuration files that also contain settings that you are not expected to manage.

The configuration file for MyID Desktop, the Self-Service App, and the MyID Client Service is now:

```
%LocalAppData%\Intercede\MyIDClientSuite\config.xml
```

You can also provide an administrator configuration file that overrides any settings in the user configuration file; this file is:

```
%ProgramData%\Intercede\MyIDClientSuite\config.xml
```

See section *3.2*, *Configuration scope* for more information about administrator configuration files.

The MyID Client Suite now also provides a MyID Client Configurator utility that allows you to edit your configuration file without having to use a text editor.

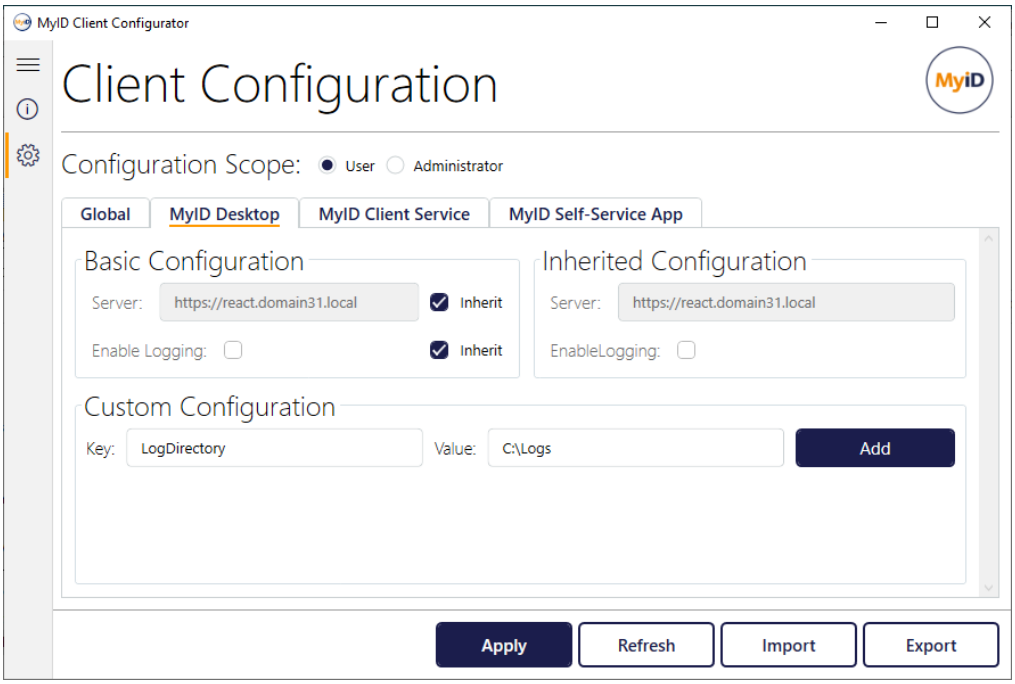
For example, to enable logging in MyID Desktop, the *Windows clients* section in the ***Configuring Logging*** guide contains instructions to add the following line:

```
<add key="LogDirectory" value="C:\Logs"/>
```

to the `MyIDDesktop.exe.config` file in the following folder:

```
C:\Program Files (x86)\Intercede\MyIDDesktop\
```

Instead, use the MyID Client Configurator to add `LogDirectory` as a custom configuration on the **MyID Desktop** tab, and set the value to `C:\Logs`.



See section *3*, *Customizing and configuring a client installation* for more information.

Important: If you have installed a configuration file using a modification package, you cannot edit its content; the file is stored in the virtual file system, not the physical file system. To update configuration that was set up using a modification package, you must apply an updated modification package.

If you need to make a change to the configuration temporarily (for example, to enable logging), you can export the configuration, uninstall the modification package, import the configuration again, and make the configuration changes. When you no longer need the temporary change, reinstall the configuration package to restore your system to its admin-specified configuration.

2.2.2 Installation process

The .msi installation programs prompted for the server location during the installation; the MSIX installation programs do not do this. You can use the MyID Client Configurator after installation to provide the server location; however, the recommended procedure is for your system administrator to put together a modification package that contains any required configuration.

2.2.3 Command-line arguments

MSIX installs do not allow access to the executable file; instead, the installers provide an alias you can use to run the applications from the command line.

For example, you might use the following command line:

```
"C:\Program Files (x86)\Intercede\MyIDApp\Self Service
Application\MyIDApp.exe" /w /un:123 /nopopup
```

For the MSIX version, omit the path, and use the alias instead:

```
MyIDApp.exe /w /un:123 /nopopup
```

The aliases are as follows:

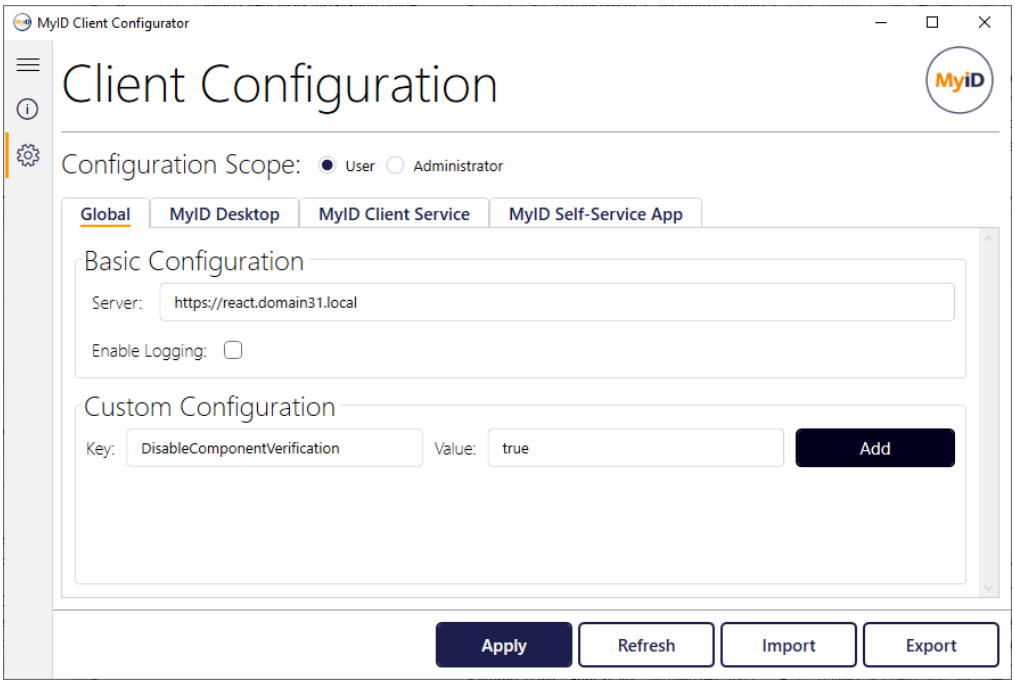
- MyID Desktop: `MyIDDesktop.exe`
- Self-Service App: `MyIDApp.exe`
- MyID Client Service: `MyIDClientService.exe`

2.3 MSIX signature validation

The MyID clients perform signature validation of their components as a form of tamper protection (if a component does not have an expected signature, it is not loaded) – while this provides significant protection, it also adds a small start-up cost and can cause the need for more complex configuration in offline environments that cannot access resources such as public revocation lists over the internet.

When running on Windows 10 2004 or later, MSIX includes tamper protection out-of-the-box, which significantly reduces the attack surface and allows you to relax our own tamper protection safely. MyID clients ship secure by default – this means that Intercede's own tamper protection is performed alongside the MSIX tamper protection unless configured otherwise. To disable the internal signature checks performed by the MyID clients, add a key of `DisableComponentVerification` and a value of `true` to your configuration; for example:

```
<add key="DisableComponentVerification" value="true"/>
```



Note: If you use the `DisableComponentVerification` setting with clients installed using MSI, they will present a prominent and persistent warning about insecure configuration. This warning, by design, cannot be dismissed or disabled. MSIX clients will not present a warning since they are protected by the built-in tamper protection.

Note: Package integrity enforcement was introduced in Windows 10 2004. While there is still improved protection compared to MSI (file-system permissions, and so on) prior to this, package integrity enforcement will not be applied and Windows will not prevent tampered apps from running. As such, it is not recommended to use this configuration if you are using a version of Windows older than Windows 10 2004.

2.4 Upgrading to the MyID Client Suite 1.5.0

When upgrading to the MyID Client Suite 1.5.0, some of the optional packages for MSIX installation (`DSKLauncher`, `MCSLauncher`, and `SSALauncher`) have been updated to version 1.1.0; the MyID Client Suite 1.5.0 is incompatible with the 1.0.0 versions of these packages.

As a result, when upgrading, you must either:

- Upgrade the optional packages with the main package in the same operation.

  You can do this by using an App Installer file, or by using a command; for example the `Add-AppxPackage` in PowerShell.
- Remove the optional packages before upgrading the main package, then install the new versions of the optional packages.

Note: The new optional packages are not backwards compatible.

2.5 Limitations and known issues

The MSIX-installed client applications have the following limitations:

- **Cannot capture an image in Edit Person**

  If you use the MSIX installer to install MyID Desktop, you cannot capture a user image in the **Edit Person** workflow.

  As an alternative, you can use the MyID Operator Client to capture user images.

  Note that you can still use **Edit Person** to upload an image file.
- **Document scanning in MyID Desktop**

  The legacy document scanning feature in MyID Desktop uses the same components as uploading user images in the **Edit Person** workflow; you cannot scan a new image. As a workaround, you can scan the image in an external application, save it to a file, then upload the file in MyID.
- **TWAIN integration with the MyID Document Scanner**

  Due to an incompatibility between the vendor-specific components that are loaded by the TWAIN protocol and the sandboxed virtual file system of MSIX, TWAIN scanning is not currently available in the MyID Document Scanner. You can still use the WIA protocol.

  If you cannot use the WIA protocol, and require TWAIN scanning, you are recommended to use the MSI installers for the MyID applications instead of MSIX.

2.6 Troubleshooting

Microsoft provides general deployment troubleshooting guides:

- ***docs.microsoft.com/en-us/windows/win32/appxpkg/troubleshooting***
- ***docs.microsoft.com/en-us/windows/msix/app-installer/troubleshoot-appinstaller-issues***

The Windows event log contains details of a failure in the deployment logs, usually with remediation suggestions. For example, if you produced an App Installer file with incorrect version numbers, Windows logs a message similar to the following:

```
AppX Deployment operation failed for package
file:///C:/Workspace/msix/updates/MyIDClientSuite.appinstaller with error
0x8008020C. The specific error text for this failure is: The package full
name returned from the AppxManifest (MyIDClientSuite_1.0.0.36_neutral_~_
1yr1geg1kr9qe) does not match the name generated from the AppInstaller
(MyIDClientSuite_1.0.0.35_neutral_~_1yr1geg1kr9qe). Please ensure that the
package attributes specified in the .appinstaller file match the package
attributes referenced in file:///C:/Workspace/msix/updates/MyIDClientSuite_
1.0.0.msixbundle.
```

2.6.1 Resetting MSIX apps

If an application is not behaving as expected, particularly if it previously worked correctly, you can use the built-in repair or reset functionality to restore the applications to their state at first install. A repair will retain any user data, whereas reset will remove it. The impact of removing user data is relatively limited:

- Information such as the last selected printer is forgotten.
- If logs are being written to the virtual profile, they are deleted.
- If user-scope configuration was created and applied using the configuration tool, it is removed.

For more information, see:

- ***docs.microsoft.com/en-us/windows/msix/desktop/managing-your-msix-reset-and-repair***

2.6.2 Troubleshooting specific errors

- **Related set error**

  ```
  A related set cannot be updated because the updated set is invalid. All
  packages in the related set must be updated at the same time.
  (0x80003d17)
  ```

  If you installed from the `.msix` file, you must install the `.msixbundle` file instead; this file contains the metadata for the related set. You can install the `.msixbundle` file over the `.msix` file – click Reinstall on the installation dialog. Note, however, that you cannot install the `.msix` file over the `.msixbundle` file.

  If you installed from the `.msixbundle` and still see this error, it may indicate that you are either using packages that are not in the same set (for example, packages from different

MyID releases), or you are trying to perform an in-place upgrade using the interactive installers, which is not supported.

3 Customizing and configuring a client installation

When the MyID clients are installed from an MSIX package, they use a different configuration file than if they were installed from an MSI. The new configuration file contains only configuration entries, without the additional noise of the MSI configuration files, and allows you to configure all of the clients in one place so that they share configuration where applicable (for example, the MyID server location, or enabling and disabling logging).

This section provides guidance on how to work with the new configuration file, and suggests ways to deploy it.

3.1 Configuration file format

The new configuration file is a much-simplified version of the existing configuration. The following is a simple example:

```
<Configuration>
    <appSettings>
        <add key="Server" value="https://my.server.local" />
        <add key="EnableLogging" value="False" />
    </appSettings>
    <dskSettings>
        <add key="SomeDskSpecificKey" value="SomeValue" />
        <add key="Server" value="https://my.otherserver.local" />
    </dskSettings>
    <ssaSettings>
        <add key="SomeSsaSpecificKey" value="SomeValue" />
    </ssaSettings>
    <mcsSettings>
        <add key="SomeMcsSpecificKey" value="SomeValue" />
    </mcsSettings>
    <suiteSettings>
        <add key="DefaultLaunchApp" value="DSK" />
    </suiteSettings>
</Configuration>
```

The following sections are available:

- `appSettings`

  Global settings to be applied to all clients.

- `dskSettings`

  Settings to be applied to MyID Desktop only.

  Settings that override global settings for MyID Desktop only.

- `ssaSettings`

  Settings to be applied to the Self-Service App only.

  Settings that override global settings for the Self-Service App only.

- `mcsSettings`

  Settings to be applied to the MyID Client Service only.

  Settings that override global settings for the MyID Client Service only.

- `suiteSettings`

  Settings that apply to the MSIX suite environment.

Note: If a key is present in both the `appSettings` node and one of the client-specific nodes, the client-specific value overrides the global setting. In the example above, all the clients use the global `Server` value (`https://my.server.local`) except MyID Desktop, which use the `Server` override (`https://my.otherserver.local`) under the `dskSettings` node.

3.2 Configuration scope

The following scopes are available for configuration: *User* and *Administrator*.

- User scope is configuration that applies only to the current user, and is, by default, allowed to be modified by the user.

  User scope configuration is stored in the following file:

  `%LocalAppData%\Intercede\MyIDClientSuite\config.xml`

- Administrator scope is machine-wide configuration that has been applied by an administrator, and is not intended to be modified by the user.

  Administrator scope configuration is stored in the following file:

  `%ProgramData%\Intercede\MyIDClientSuite\config.xml`

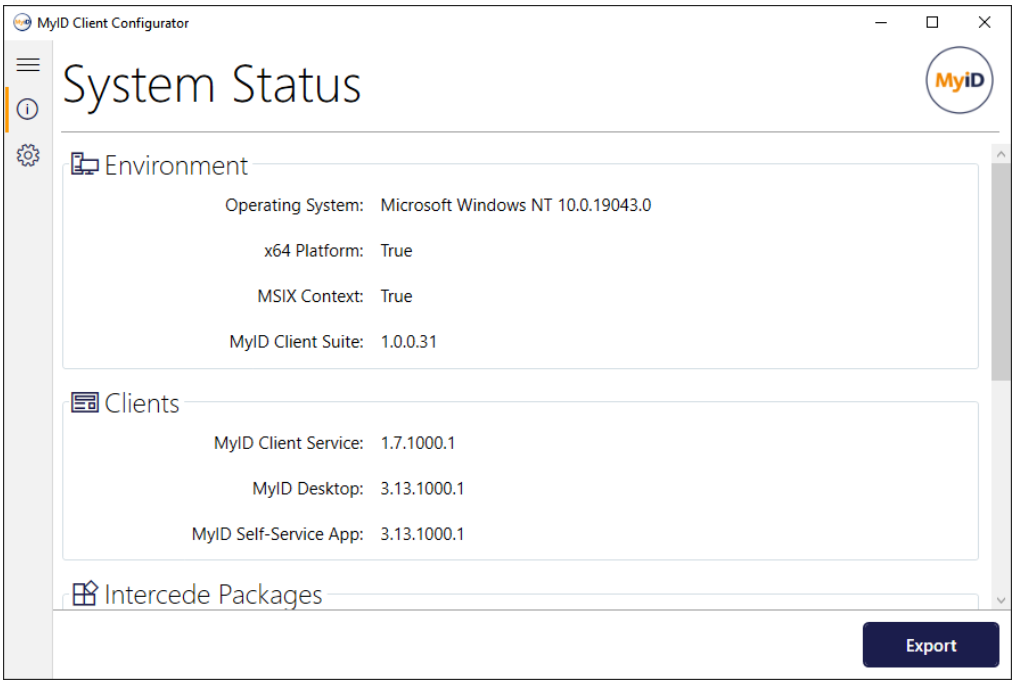
If Administrator scope configuration is present on a machine, the User scope configuration is ignored by MyID. It is recommended that you set appropriate file permissions on the configuration file or directory if you normally allow your users to change files in the `%ProgramData%` folder, but want to prevent changes to the configuration file.

3.3 MyID Client Configurator

The MyID Client Suite MSIX package includes a tool for changing configuration. You can use it to set, import, or export user-scope configuration, view or export admin configuration, and view or export a simple report on the status of the client installation (current platform, installed packages, and so on).

3.3.1 System Status

The default view of the configuration tool provides an overview of the client environment.



3.3.1.1 Environment

The Environment section lists:

- The current operating-system version.
- Whether or not the operating-system is 64-bit.
- Whether or not the tool is correctly running in the context of an MSIX container.
- The version of the main MyID Client Suite package.

3.3.1.2 Clients

The Clients section lists the versions of the MyID clients inside the package. These values match their equivalent MSI versions.

3.3.1.3 Intercede Packages

The Intercede Packages section lists the installed packages that were produced and provided by Intercede.

3.3.1.4 Custom Packages

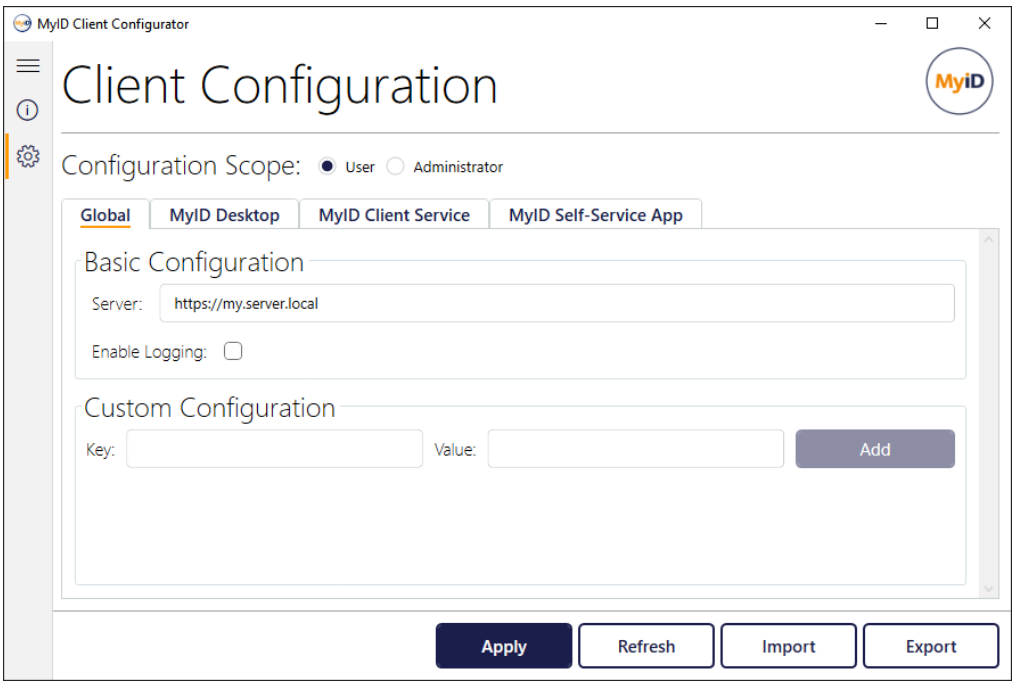
The Custom Packages section lists any installed packages that were not produced by Intercede; for example, if you have applied configuration with a modification package, your package appears in this list.

3.3.1.5 Export

Click **Export** to save the information displayed on the System Status view to a text file. This can be helpful in supplying information about your system to Intercede Support during troubleshooting.

3.3.2 Client Configuration

The Client Configuration view is used to view or edit configuration.



To switch the configuration scope, click the **User** or **Admin** option; see section *3.2*, *Configuration scope* for more information.

- You can view, modify, or export User scope configuration from the tool.
- You can view or export Administrator scope configuration, but cannot modify it.

Changing the selection displays the configuration that corresponds to the selected scope.

3.3.2.1 Using the MyID Client Configurator

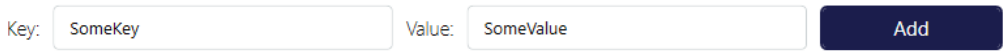
The Client Configuration view includes a **Global** tab, and tabs for the individual clients provided by the MSIX installer. Configuration entered on the **Global** tab is, by default, applied to all clients; for example, if you set the **Server** value on the **Global** tab, it is inherited and displayed on the client-specific tabs.

If you have an existing configuration file that you want to use as the base for a new file, or that you want to apply as-is to the machine you are using, click **Import** and select the file to import.

Each tab in the Client Configuration view contains entries for the commonly-changed configuration keys, and an additional **Custom Configuration** section that allows you to apply additional configuration that does not have dedicated fields. For example, the MyID documentation provides configuration keys in the following format:

```
<add key="SomeKey" value="SomeValue" />
```

Where a dedicated field does not exist, you can add it by entering the key and value then clicking **Add**:



Each client-specific tab contains an **Inherited Configuration** section that displays all the configuration keys and values that are being inherited from the **Global** tab. To override these values on a per-client basis:

- If the client's tab includes a dedicated field for the configuration key, for example **Server**, deselect the **Inherit** option and enter the value you want the selected client to use.
- If the configuration was added as a custom entry, use the **Custom Configuration** section to add an entry with the same key and the new value you want the selected client to use.

Once you are happy with your configuration, and you want to apply it, click the **Apply** button. The resulting configuration file is saved to:

```
%LocalAppData%\Intercede\MyIDClientSuite\config.xml
```

and will be used by clients running from the MSIX container.

Note: If Administrator scope config is present, the user configuration applied by the tool warns you that it will be applied, but will be ignored.

If you want to use the configuration file for a wider deployment, click **Export** to save it to a file. You can then deploy this file to your clients using your chosen mechanism; for example, using a modification package (see section *3.4*, *Deploying configuration as modification packages*) or manually deploying the file (see section *3.5*, *Manually deploying configuration files*).

3.3.3 Setting the default launch app

In environments where you do not want users to load the configuration tool, you can redirect the **MyID Client Suite** shortcut to one of the other applications in the package. If deployed as part of an interactive installation (for example, with App Installer) this also changes what application is launched if the user selects the ***Launch when ready*** option.

Notes:

- This is a manual change to your configuration file; you cannot change it through the configuration tool itself.
- You must apply this change to the *Administrator* scope configuration file; it is ignored if present only in the `%LocalAppData%\Intercede\MyIDClientSuite\config.xml` file. It *must* be placed in the `%ProgramData%\Intercede\MyIDClientSuite\config.xml` file, regardless of how it is deployed.

To configure the default launch app:

1. Open the `config.xml` file in a text-editor.
2. Look for a `suiteSettings` node within the `Configuration` node.

   If this node is not present, add `<suiteSettings></suiteSettings>` as a child of the `Configuration` node.

   If it is present in the format `<suiteSettings/>`, replace it with `<suiteSettings></suiteSettings>`.
3. Add the following as a child of `suiteSettings`:

   `<add key="DefaultLaunchApp" value="">`
4. Set the `value` to one of the following:
   - `DSK` – sets the launch app to MyID Desktop.
   - `SSA` – sets the launch app to the Self-Service App.
   - `MCS` – sets the launch app to the MyID Client Service.

For example, to set the MyID Client Suite shortcut to launch MyID Desktop:

```
<Configuration>
    <appSettings>
        <add key="Server" value="https://my.server.local" />
    </appSettings>
    <dskSettings />
    <sskSettings />
    <ssaSettings/>
    <mcsSettings/>
    <suiteSettings>
        <add key="DefaultLaunchApp" value="DSK">
    </suiteSettings>
</Configuration>
```

3.4 Deploying configuration as modification packages

The recommended way to deploy your configuration file is as part of a Modification Package, a special kind of MSIX package that can be applied as a patch to the MyID Client Suite. This approach brings the following advantages:

- When deployed in this way, the configuration file is placed *inside* the MSIX container in a private virtual file-system that can only be accessed by applications in the same container – this avoids polluting your system with loose files, and ensures they are removed if the applications are uninstalled.
- Allows you to provide configuration files in managed way, including proper versioning.
- Allows you to deploy your configuration file as part of an App Installer file.
- Enables fast and easy deployment or rollback of configuration files, including automatic updates using App Installer.
- Protected by the same trust model as the main MSIX packages; that is, the Modification Package must be signed by a certificate trusted in your organization before it can be applied.

For more information about modification packages, see:

docs.microsoft.com/en-us/windows/msix/modification-packages.

3.4.1 MSIX Packaging Tool

The easiest way to create a modification package is to use Microsoft's MSIX Packaging Tool.

The MSIX Packaging Tool monitors changes to the system, then produces an MSIX package that replicates the detected changes in an MSIX container. This can be installed on an existing system, or run from the built-in Hyper-V environment.

See the following for more information:

- ***docs.microsoft.com/en-us/windows/msix/packaging-tool/tool-overview***
- ***docs.microsoft.com/en-us/windows/msix/packaging-tool/quick-create-vm***

To create a modification package with the MSIX Packaging Tool:

1. Create a staging folder containing:
   - The MyID Client Suite MSIX.

     You can extract this from the `MyIDClientSuite-X.Y.Z.msixbundle` file. Change the file's extension to `.zip`, then open it and extract the contained `.msix` file.
   - The `config.xml` file you want to deploy.

     You can create a config file manually (see section *3.1*, *Configuration file format*) or you can create a user config file using the MyID Client Configurator on a PC that has the MyID Client Suite installed, then copy this file from the following folder:

     `%LocalAppData%\Intercede\MyIDClientSuite\`
   - Any theme files you want to apply (`theme.json`, `UniformLogo.png`, `WideLogo.png`, and so on).

     See the *Customizing the MyID Windows Clients* guide for details of applying local themes and branding.

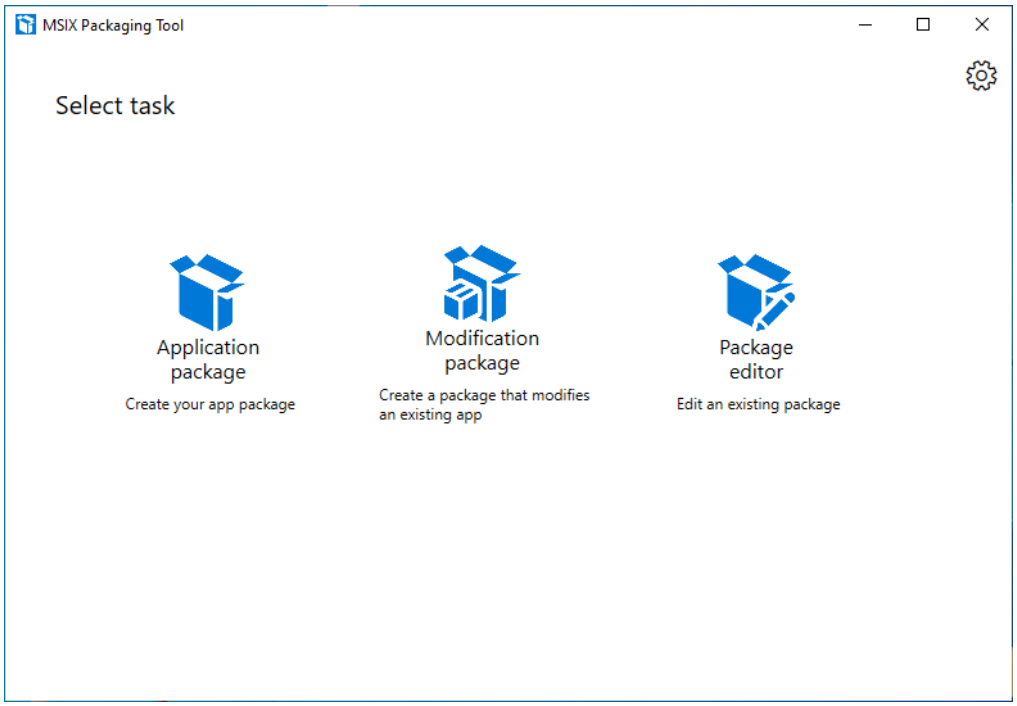
2. Create the folder structure required by the MyID clients:

   - `%ProgramData%\Intercede\MyIDClientSuite\`

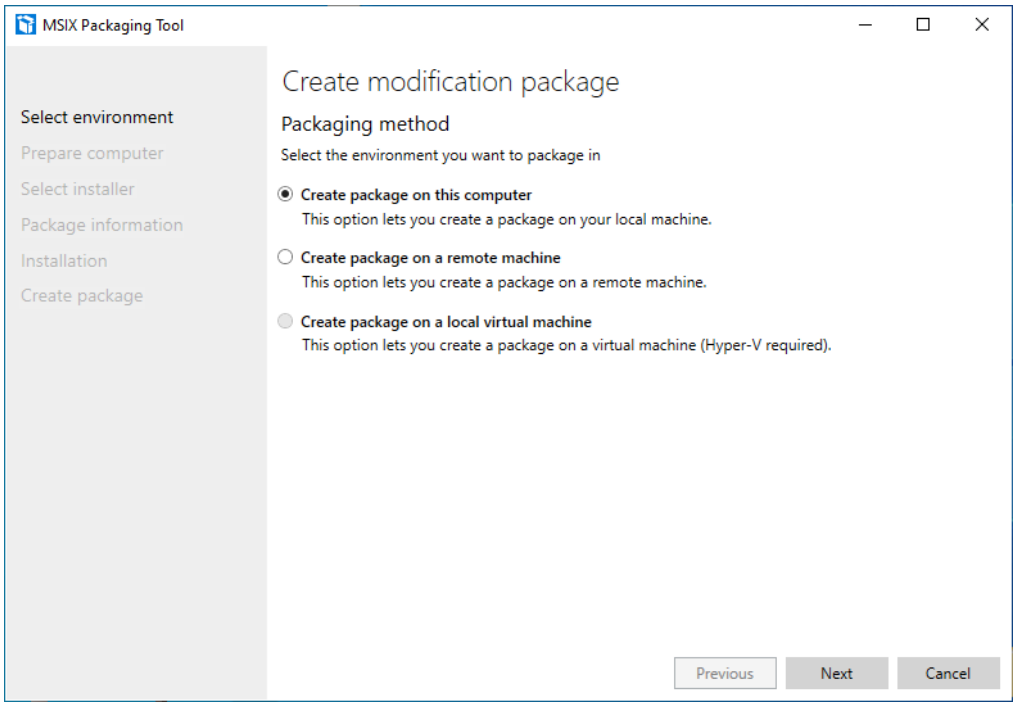
   Optionally, if you intend to apply a theme or branding to the MyID clients with the package, create the following subfolders:

   - `%ProgramData%\Intercede\MyIDClientSuite\MyIDDesktop\`
   - `%ProgramData%\Intercede\MyIDClientSuite\MyIDApp\Self Service Application\`
   - `%ProgramData%\Intercede\MyIDClientSuite\MyIDClientService\`

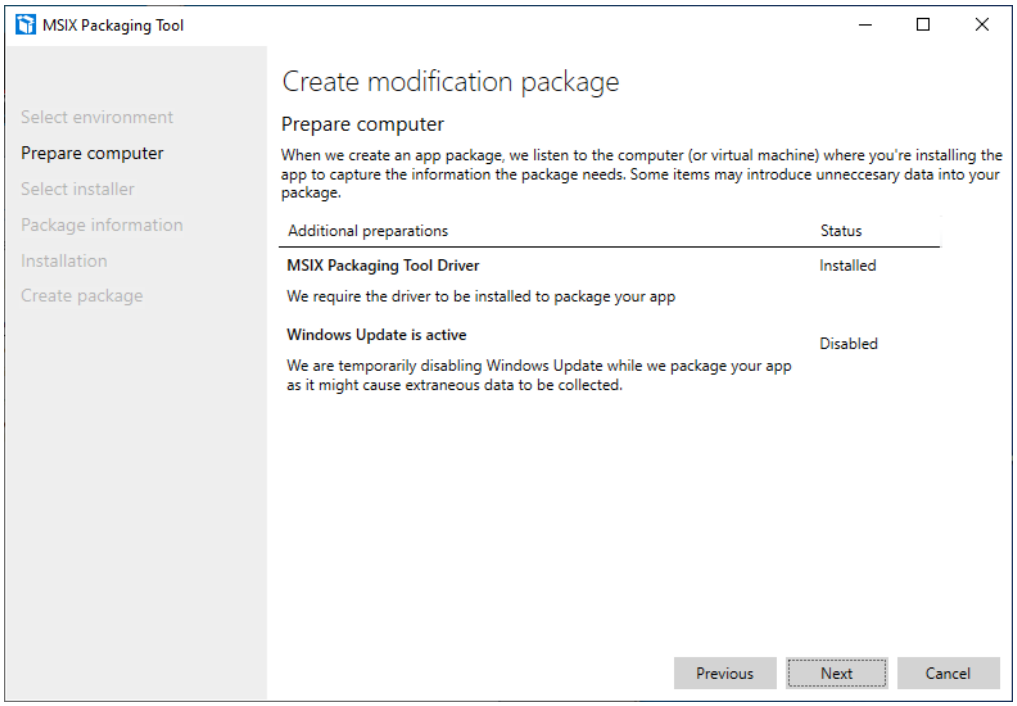
3. Launch the MSIX Packaging Tool and select **Modification Package**.



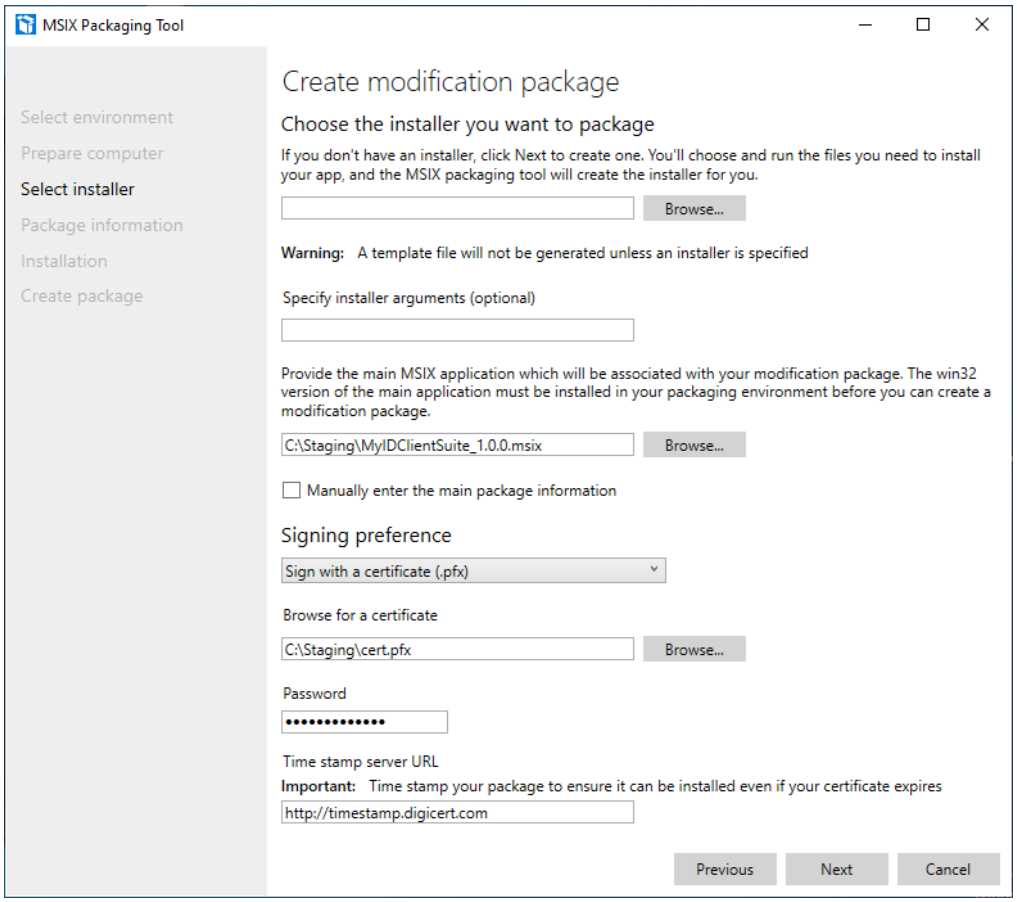
4. Select **Create package on this computer**, then click **Next**.



5. The tool checks for prerequisites, including the presence of the MSIX Packaging Tool Driver. Follow the instructions on this screen, then click **Next** when available.



6. Configure the target package and signing options.



a. As you are not converting an existing installer, leave the first two fields blank.

b. In the third field, provide the path to the MyID Client Suite MSIX in the staging folder.

c. Select how to sign your package. Note that the signature's certificate chain must be trusted within your organization before the package can be installed.

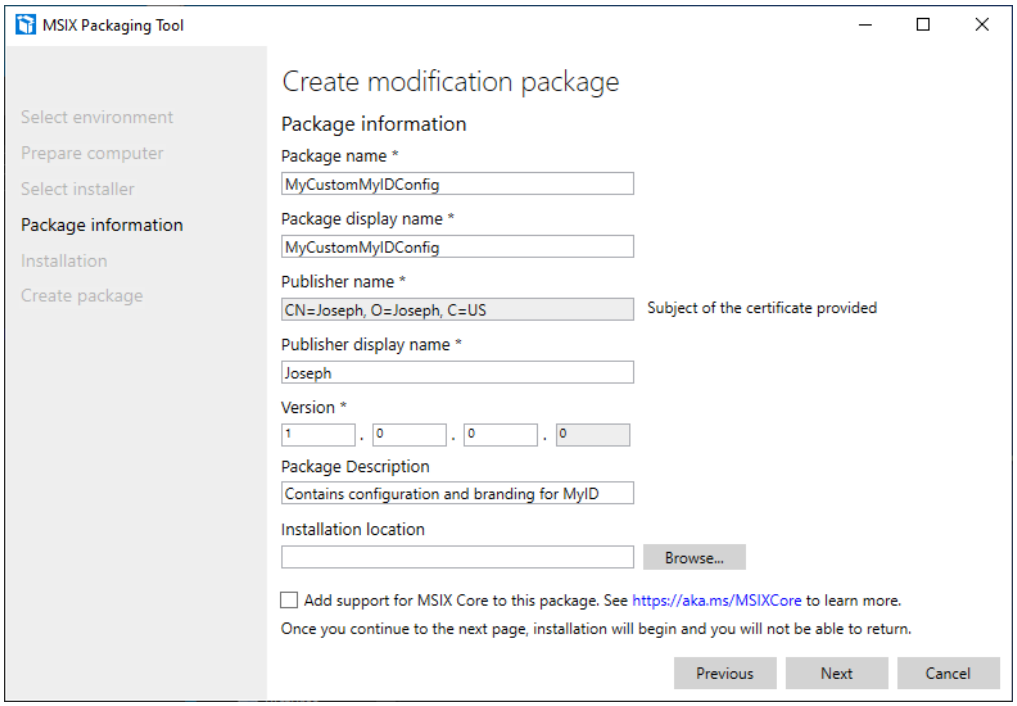
d. Optionally enter a timestamp server address. The MSIX Packaging Tool must be able to communicate with the timestamp server.

   While optional, this step is highly recommended to avoid Windows preventing installation of your package after the signing certificate expires.

   The timestamp certificate chain must be trusted within your organization.

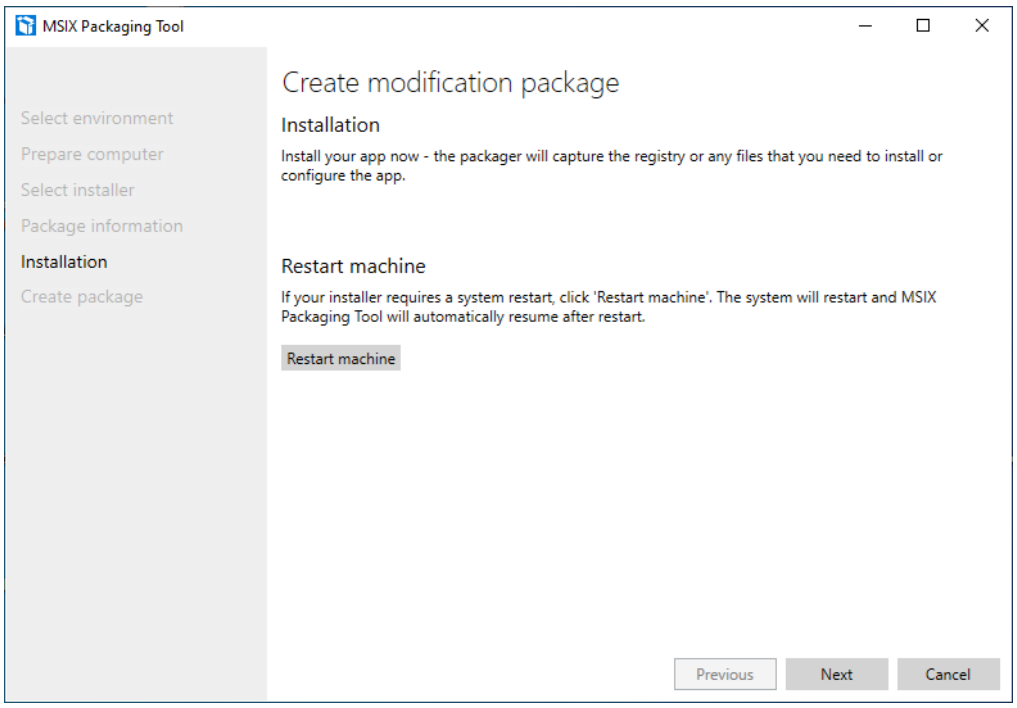
e. Click **Next**.

7. Enter the details for the Modification Package you are about to create, including the name, display name, publisher name, and version.



Once you have entered the information, click **Next**.

8. The MSIX Packaging Tool starts monitoring your system for modifications to be replicated by your modification package.



a. Copy the `config.xml` file from your staging folder into:

   `%ProgramData%\Intercede\MyIDClientSuite\`

   This is the location of the administrator configuration file; see section *3.2*, *Configuration scope* for details.

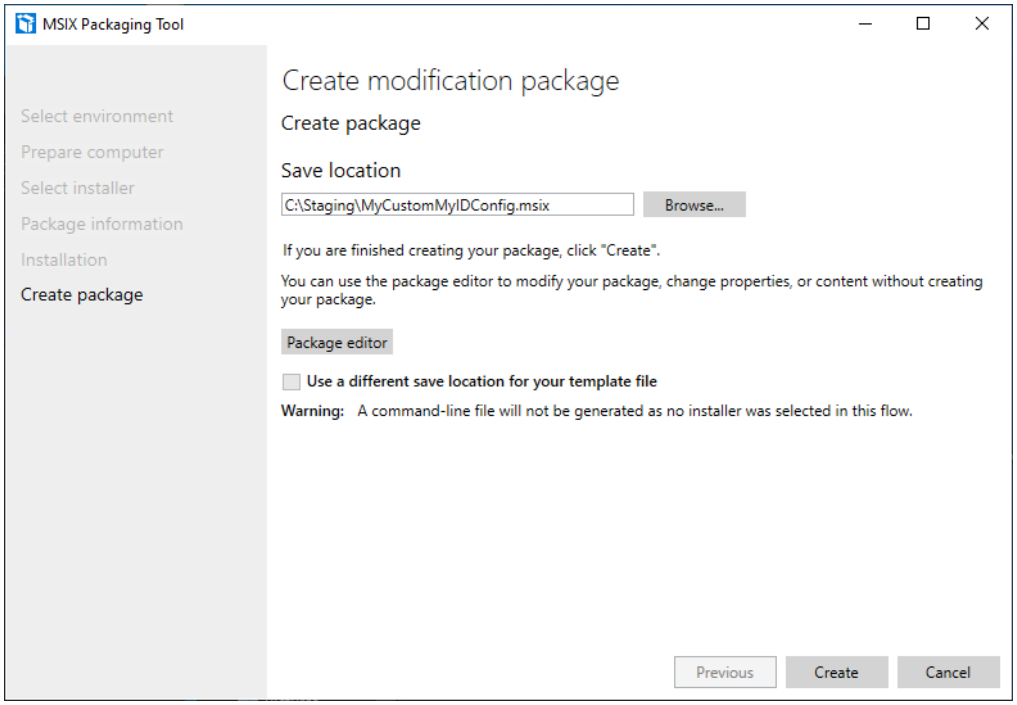
b. Optionally copy `theme.json`, `UniformLogo.png`, and `WideLogo.png` from the staging folder into the subfolders you created earlier corresponding to the clients you want to rebrand.

   For example, to apply your theme to Self-Service App, copy the files into:

   `%ProgramData%\Intercede\MyIDClientSuite\MyIDApp\Self Service Application\`

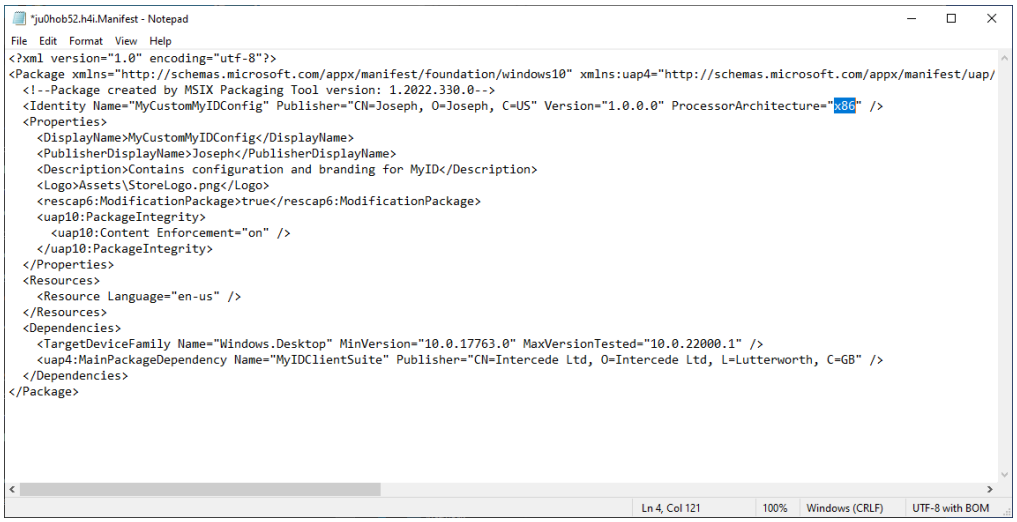
Once you have copied all the customizations, click **Next** then **Yes, move on**.

9. Enter the location you want to save your modification package, then click **Package editor**.



a. Select **Package information**, then scroll down to **Manifest file** and click **Open file**. This opens the package manifest in a text editor.

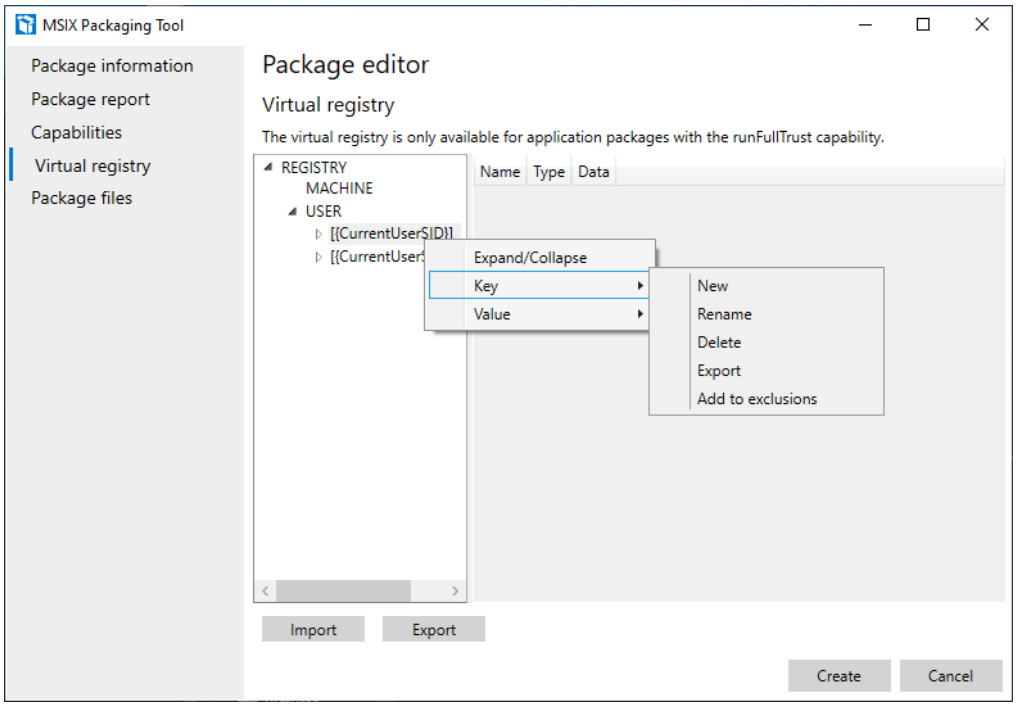
b. Update the `ProcessorArchitecture` attribute to be `x86`.



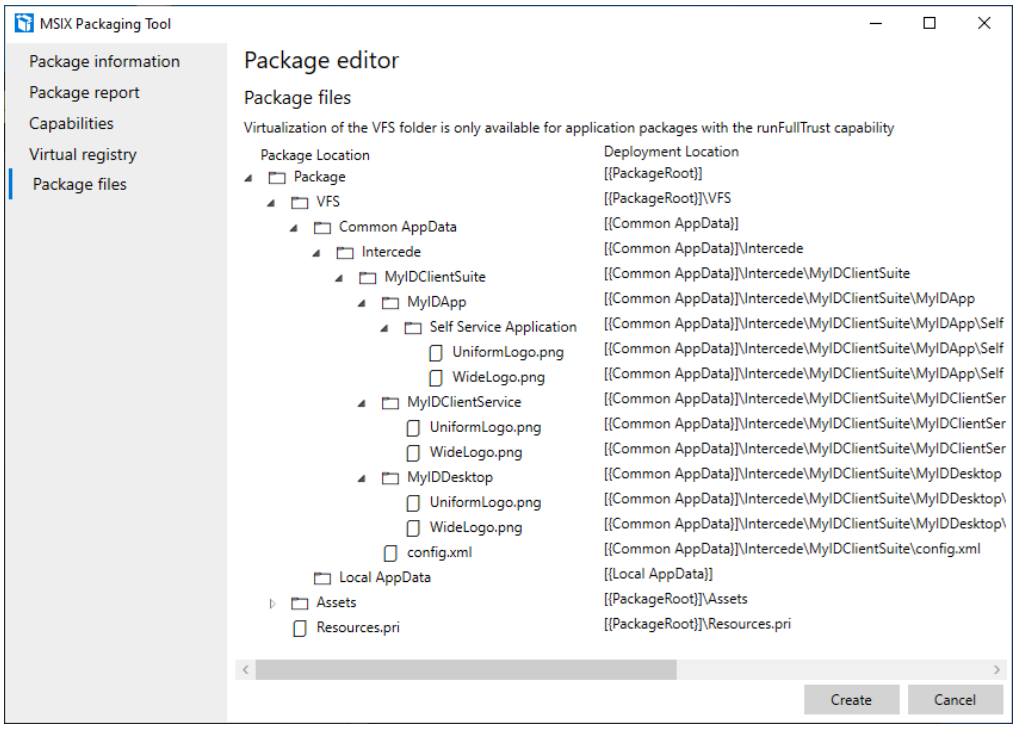
c. Save the file and close the text editor.

d. Select **Virtual registry** and delete any registry entries detected by the tool.

These will have been added by unrelated background tasks.

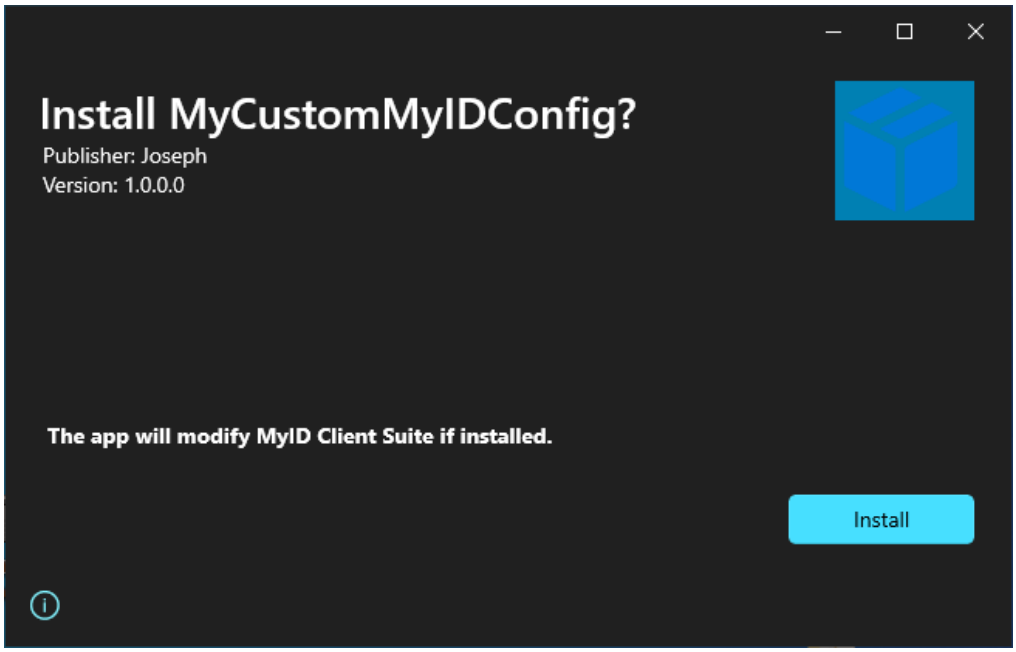


e. Select **Package files** and verify that the files you copied earlier are included in their expected locations.

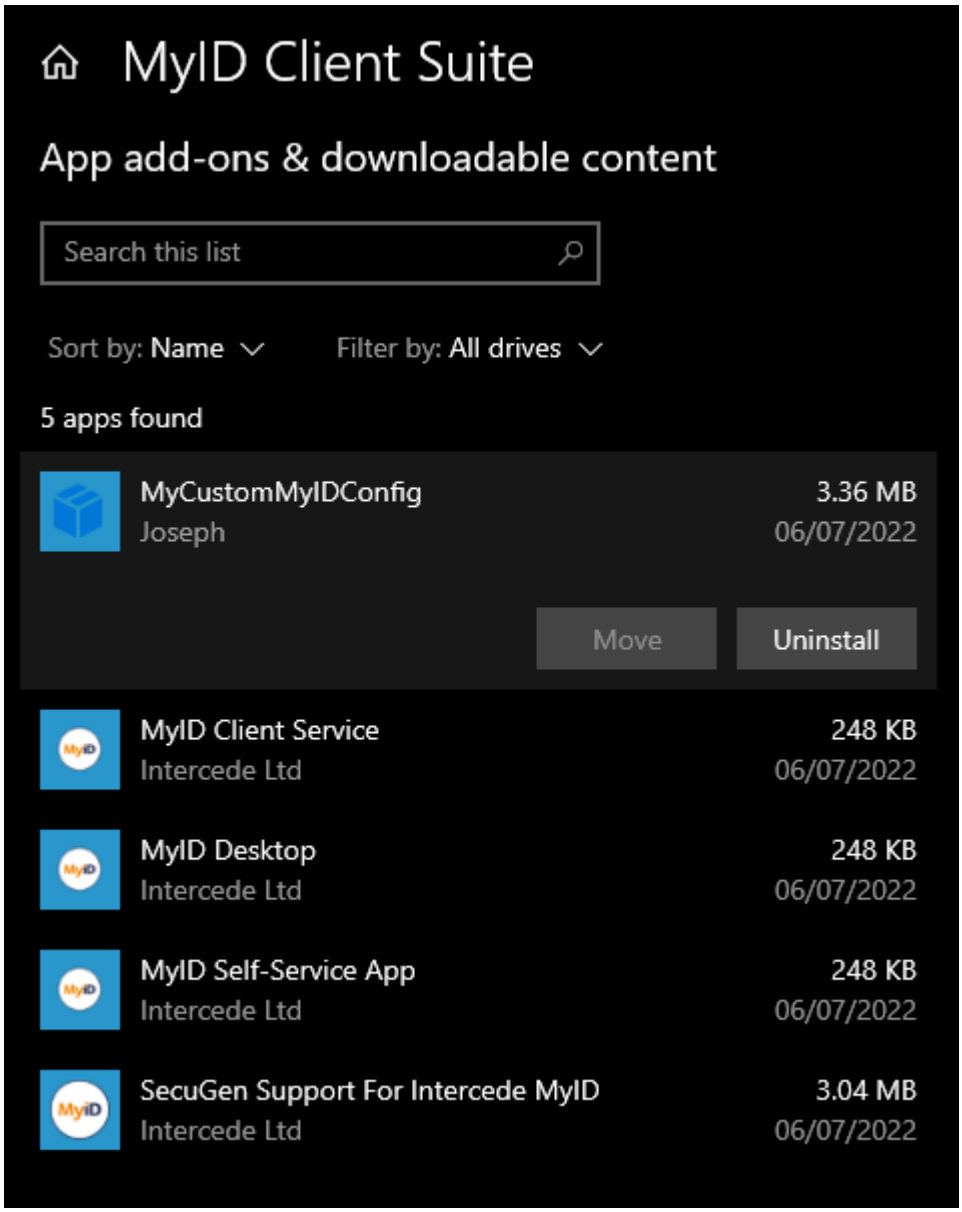


f. Click **Create** to create your package.

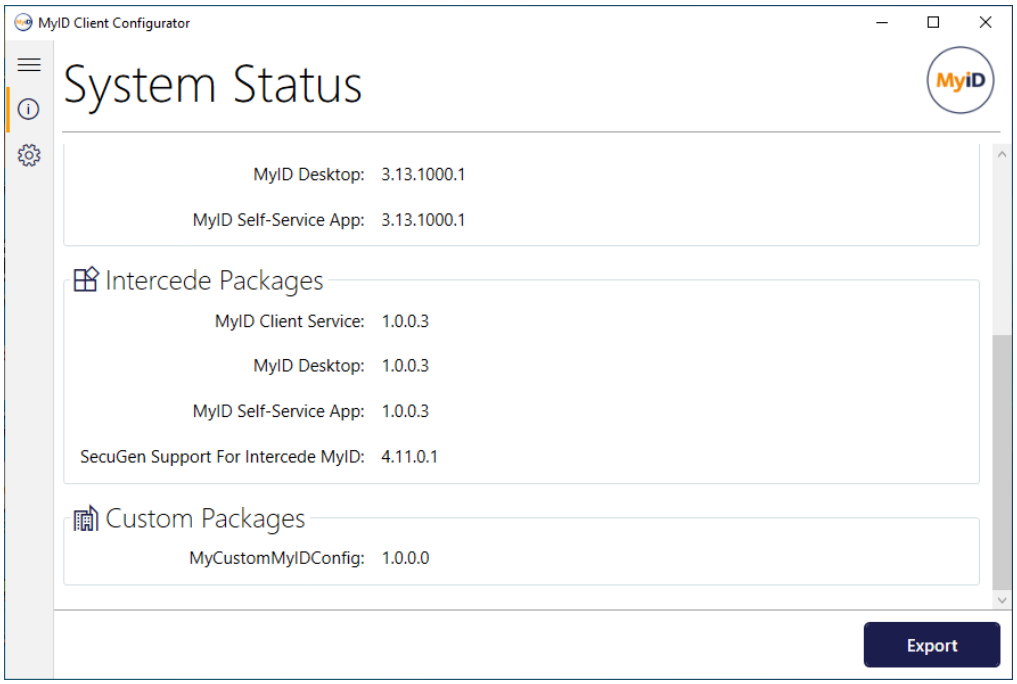
10. You can now install your MSIX Modification Package



Once it is installed, Windows detects it as an add-on to the MyID Client Suite:



The package will be included in the configuration tool's **System Status** view in the **Custom Packages** section; see section *3.3.1*, *System Status*.



11. Delete the `%ProgramData%\Intercede` folder you created.

    Once you have installed the modification package, the configuration is stored in the virtual file system. Deleting the `%ProgramData%\Intercede` folder ensures that the modification package is working as intended, and the application is drawing its configuration from the virtual file system, not the physical file system.

12. Launch the MyID Client Configurator and verify that the configuration is as expected.

    The applied configuration is read-only under the **Administrator** scope. You cannot change administrator config using the MyID Client Configurator; see section *3.2*, *Configuration scope*

13. Finally, launch the MyID clients and verify they work as expected, and include any custom branding you may have applied.

You can now deploy your modification package to your organization using your preferred method; for example, `Add-AppxPackage` (see section *5.1*, *Headless install without App Installer files*), or App Installer (see section *4*, *Deploying MyID clients with App Installer files*). If you are using App Installer files to deploy MyID Client Suite, you can add your modification package as an optional package to be included in your deployment.

3.5 Manually deploying configuration files

If you do not want to use a modification package to deploy your configuration file, themes, or branding, you can deploy them manually. However, the MSIX installers provide no management for this method; for example, if you uninstall MyID Client Suite after deploying your configuration in this way, the configuration is left on the system.

You must restart any running MyID clients to ensure that they detect the configuration changes.

3.5.1 Applying configuration

Copy your `config.xml` file into the directory corresponding to your desired scope.:

- User:

  `%LocalAppData%\Intercede\MyIDClientSuite\config.xml`

- Administrator

  `%ProgramData%\Intercede\MyIDClientSuite\config.xml`

See section *3.2*, *Configuration scope*.

You can create a config file manually (see section *3.1*, *Configuration file format*) or you can create a user config file using the MyID Client Configurator on a PC that has the MyID Client Suite installed, then export the config file.

3.5.2 Applying themes and branding

You can apply the theme or branding for each client independently. When deployed with this manual approach, the theme and branding changes are applied machine-wide for all users.

Copy the `theme.json`, `UniformLogo.png`, or `WideLogo.png` files into the appropriate folder:

- MyID Desktop:

  `%ProgramData%\Intercede\MyIDClientSuite\MyIDDesktop\`

- MyID Client Service:

  `%ProgramData%\Intercede\MyIDClientSuite\MyIDClientService\`

- MyID Self-Service App:

  `%ProgramData%\Intercede\MyIDClientSuite\MyIDApp\Self Service Application\`

See the *Customizing the MyID Operator Client* guide for details of applying local themes and branding.

3.6 Importing configuration files

In environments where the end user regularly changes the configuration themselves, for example, a system administrator who commonly switches between production and pre-production servers, the user can use the MyID Client Configurator to import configuration files.

You can export a configuration file from the MyID Client Configurator (see section *3.3.2*, *Client Configuration*) or create one manually (see section *3.1*, *Configuration file format*).

To import a configuration file:

1. Launch the MyID Client Configurator.

   By default, the MyID Client Suite shortcut is configured to open the configuration tool If your system has been set up to open a different application (see section *3.3.3*, *Setting the default launch app*) you can launch the MyID Client Configurator using its registered protocol; open the Windows Run dialog, and type:

   ```
   myidconfig://
   ```

2. Click the Configuration icon in the navigation pane.

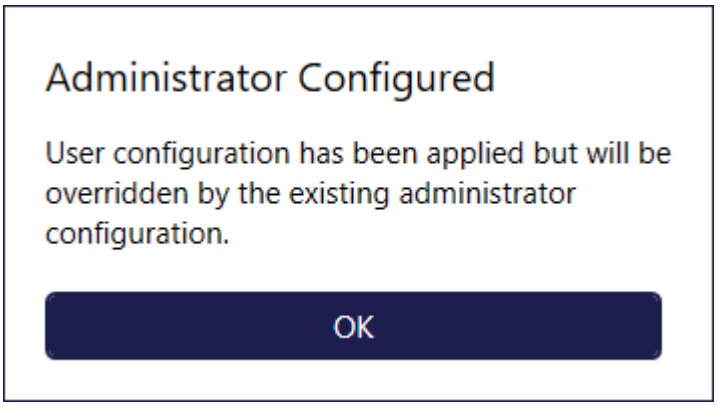
   

3. Select the **User** configuration scope.

   You can export from both the user and administrator configuration, but you can import only into the user configuration.

4. Click **Import** and select the configuration file you want to import.
5. Click **Apply**.
6. Restart any running MyID clients.

If the following warning appears, it means that administrator configuration was detected:



The administrator configuration overrides any user configuration, including the configuration you just imported. You must remove the administrator configuration file from the admin folder:

```
%ProgramData%\Intercede\MyIDClientSuite\config.xml
```

or, if the administrator configuration was applied by a modification package, you must uninstall the modification package. For modification packages the configuration file is stored in the virtual file system, not the physical file system, you can cannot remove the file manually.

4 Deploying MyID clients with App Installer files

App Installer is a Microsoft technology in Windows that is used behind-the-scenes to install MSIX packages. One of the key features is the ability to create App Installer files that define the components that should be deployed, how they should be updated, and so on.

The MyID Client Suite has been designed to work with App Installer files, allowing you to defined custom bundles of MyID components to be deployed in a single install.

4.1 App Installer file

An App Installer file is an XML file with an `.appinstaller` file-extension. It contains details of the packages that should be included in a deployment, along with configuration details, such as automatic update configuration. You can use different App Installer files within your organization to support deployments of different sets of packages to different users.

Microsoft provides an overview of the App Installer file, along with example use cases, in their documentation:

docs.microsoft.com/en-us/windows/msix/app-installer/app-installer-file-overview.

The following is an example of an App Installer file to install MyID Client Suite and the optional launcher packages to add Start Menu entries for **MyID Desktop**, **MyID Client Service**, and **MyID Self-Service App**:

```
<?xml version="1.0" encoding="utf-8"?>
<AppInstaller
  Version="1.0.0.0"
  Uri="\\my.server.local\Installers$\MyIDClientSuite.appinstaller"
  xmlns="http://schemas.microsoft.com/appx/appinstaller/2018">
    <MainBundle
    Name="MyIDClientSuite"
    Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
    Version="1.0.0.33"
    Uri="\\my.server.local\Installers$\MyIDClientSuite_1.0.0.msixbundle" />
    <OptionalPackages>
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\DSKLauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDDesktopLauncher" />
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\MCSLauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDClientServiceLauncher" />
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\SSALauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDSelfServiceAppLauncher" />
    </OptionalPackages>
</AppInstaller>
```

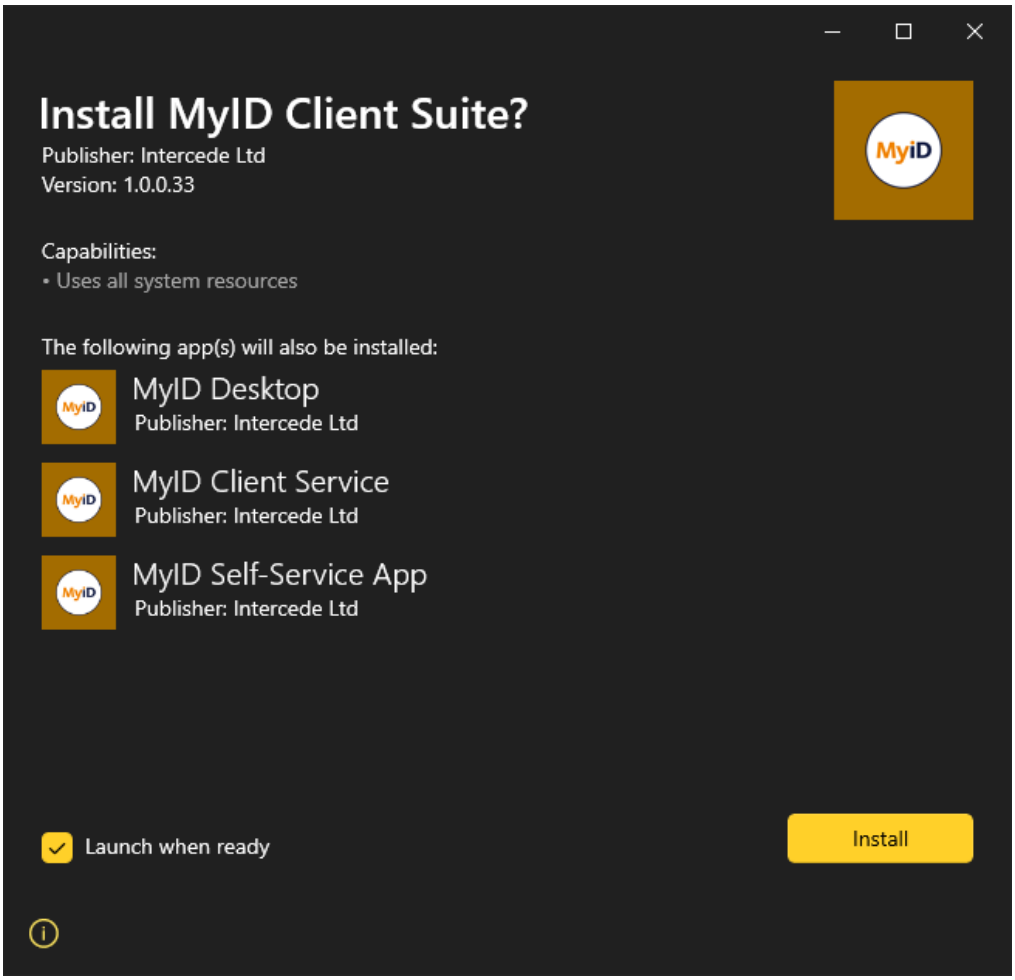
- This example configures a deployment of the main `MyIDClientSuite` package at version 1.0.0.33, and also installs the optional `DSKLauncher 1.0.0.5`, `MCSLauncher 1.0.0.5`, and `SSALauncher 1.0.0.5` packages at the same time.

  Note: Versions in the App Installer file must match to the packages you want to install, and the versions in the example may not correspond to these.

- All of the packages, and the App Installer file itself, include paths to the corresponding file locations (`Uri` attributes) – in this example, they are on a file-share called `Installers` on a server called `my.server.local`.

  Note: You cannot use relative paths. You must include a full local path or UNC path.

When you double-click the App Installer file to launch an interactive installation, you see all the packages to be installed detailed in the App Installer UI:



4.2 Finding metadata for App Installer file

You can find the metadata you need for an App Installer file `Package` entry in the `AppxManifest.xml` file inside the package. To access it, change the package's `.msix` extension to `.zip` and open it – inside, you will find the `AppxManifest.xml` file.

Note: In the case of the main MyID Client Suite `msixbundle` file, the `AppxManifest.xml` file is two levels deep rather than one. If you open the package as above you will find an MSIX package inside it – this is the package that contains the `AppxManifest.xml` file.

The following is an example of the manifest in the `MyIDDesktopLauncher` package:

```
<?xml version="1.0" encoding="utf-8"?>
<Package
    xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10"
    xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10"
    xmlns:uap3="http://schemas.microsoft.com/appx/manifest/uap/windows10/3"
    xmlns:uap4="http://schemas.microsoft.com/appx/manifest/uap/windows10/4"
    xmlns:uap10="http://schemas.microsoft.com/appx/manifest/uap/windows10/10"
    xmlns:desktop="http://schemas.microsoft.com/appx/manifest/desktop/windows10"
    IgnorableNamespaces="uap4 uap10">
  <Identity Name="MyIDDesktopLauncher"
            Version="1.0.0.5"
            Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
            ProcessorArchitecture="x86" />
    <Properties>
       <DisplayName>MyID Desktop</DisplayName>
        <PublisherDisplayName>Intercede Ltd</PublisherDisplayName>
        <Description>MyIDDesktop</Description>
        <Logo>Assets\StoreLogo.png</Logo>
        <uap10:PackageIntegrity>
            <uap10:Content Enforcement="on" />
        </uap10:PackageIntegrity>
    </Properties>
    <Resources>
      <Resource Language="en-us" />
    </Resources>
      <Dependencies>
        <TargetDeviceFamily
Name="Windows.Desktop" MinVersion="10.0.17763.0" MaxVersionTested="10.0.19041.1" />
      <!-- the dependency should refer to the main package's Identity.Name -->
      <uap4:MainPackageDependency Name="MyIDClientSuite" Publisher="CN=Intercede Ltd,
O=Intercede Ltd, L=Lutterworth, C=GB"/>
      </Dependencies>
    <Applications>
      <Application
Id="MyIDDesktop" Executable="VFS\ProgramFilesX86\Intercede\DSKLauncher\DSKLauncher.exe"
                   EntryPoint="Windows.FullTrustApplication">
        <uap:VisualElements  BackgroundColor="transparent"
                             DisplayName="MyID Desktop"
                             Description="MyIDDesktop"
                             Square150x150Logo="Assets\Square150x150Logo.png"
                             Square44x44Logo="Assets\Square44x44Logo.png">
            <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png" >
                    <uap:ShowNameOnTiles>
                        <uap:ShowOn Tile="square150x150Logo"/>
                        <uap:ShowOn Tile="wide310x150Logo"/>
                    </uap:ShowNameOnTiles>
                </uap:DefaultTile>
        </uap:VisualElements>
        <Extensions>
            <uap3:Extension Category="windows.appExecutionAlias"
                            EntryPoint="Windows.FullTrustApplication"

Executable="VFS\ProgramFilesX86\Intercede\DSKLauncher\DSKLauncher.exe">
                <uap3:AppExecutionAlias>
                    <desktop:ExecutionAlias Alias="MyIDDesktop.exe" />
                </uap3:AppExecutionAlias>
            </uap3:Extension>
        </Extensions>
```

```
      </Application>
    </Applications>
  </Package>
```

The information you need is the `Identity` node near the top of the file:

```
<Identity Name="MyIDDesktopLauncher"
          Version="1.0.0.5"
          Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
          ProcessorArchitecture="x86" />
```

This corresponds directly to what you would include in your `Package` entry, with the exception of the `Uri` value, which is specific to your organization:

```
<Package Version="1.0.0.5"
         Uri="\\my.server.local\Installers$\OptionalPackages\DSKLauncher.msix"
         Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
         ProcessorArchitecture="x86"
         Name="MyIDDesktopLauncher" />
```

In scenarios where you want to get the manifest of an installed package, but you do not have the original MSIX that was used to install it, you can use the `Get-AppxPackageManifest` PowerShell cmdlet to extract it then save it to a file; see.:

docs.microsoft.com/en-us/powershell/module/appx/get-appxpackagemanifest?view=windowsserver2019-ps

The following example demonstrates retrieving the manifest for an install of the `MyIDClientSuite`, then saving it to a file:

```
$(Get-AppxPackageManifest -Package $(Get-AppxPackage
MyIDClientSuite).PackageFullName).Save
("C:\Manifests\MyIDClientSuite\AppxManifest.xml")
```

4.3 Related sets

Some of the Intercede packages are created as part of a *related set*, meaning they are tightly coupled and must be updated as a set. For more information about related sets, see:

docs.microsoft.com/en-us/windows/msix/package/optional-packages

The packages that are in a related set with the `MyIDClientSuite` package are:

- `MyIDDesktopLauncher`
- `MyIDClientServiceLauncher`
- `MyIDSelfServiceAppLauncher`

For example, `MyIDClientSuite` 1.0.0.33 is in a related set with version 1.0.0.5 of these optional packages – if these packages are updated to version 1.0.0.6, the `MyIDClientSuite` package must also be updated to a version that knows about the 1.0.0.6 packages.

Intercede always releases the related set packages together, even if they have not changed.

4.4 Custom modification packages

If you are deploying configuration or customization with a Modification Package, you can add it to your App Installer file to be included in your deployment.

See section *3.4*, *Deploying configuration as modification packages* for details of creating modification packages.

For example, if you have created a modification package with the following details:

- Name: `MyCustomMyIDConfig`
- Version: `1.0.0.0`
- Publisher: `CN=MyOrganisation, O=MyOrganisation, L=MyLocation, C=MyCountryCode`
- ProcessorArchitecture: `x86`
- Location: `\\my.server.local\Installers$\OptionalPackages\MyCustomMyIDConfig.msix`

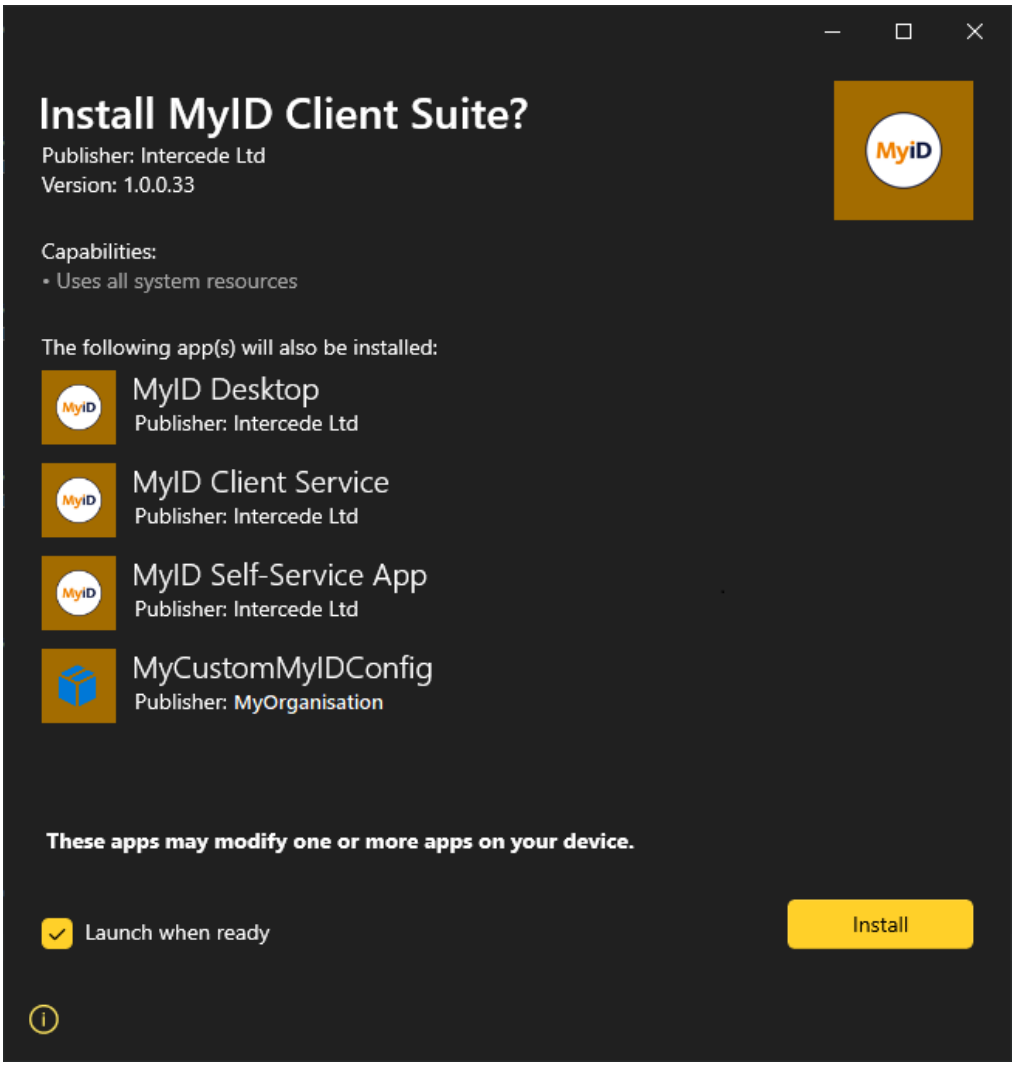
You add the following as a child of the `OptionalPackages` node:

```
<Package
    Version="1.0.0.0"
    Uri="\\my.server.local\Installers$\OptionalPackages\MyCustomMyIDConfig.msix"
    Publisher="CN=MyOrganisation, O=MyOrganisation, L=MyLocation, C=MyCountryCode"
    ProcessorArchitecture="x86"
    Name="MyCustomMyIDConfig" />
```

Adding this to the example produces the following:

```
<?xml version="1.0" encoding="utf-8"?>
<AppInstaller
  Version="1.0.0.0"
  Uri="\\my.server.local\Installers$\MyIDClientSuite.appinstaller"
  xmlns="http://schemas.microsoft.com/appx/appinstaller/2018">
    <MainBundle
    Name="MyIDClientSuite"
    Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
    Version="1.0.0.33"
    Uri="\\my.server.local\Installers$\MyIDClientSuite_1.0.0.msixbundle" />
    <OptionalPackages>
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\DSKLauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDDesktopLauncher" />
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\MCSLauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDClientServiceLauncher" />
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\SSALauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDSelfServiceAppLauncher" />
        <Package
      Version="1.0.0.0"
      Uri="\\my.server.local\Installers$\OptionalPackages\MyCustomMyIDConfig.msix"
      Publisher="CN=MyOrganisation, O=MyOrganisation, L=MyLocation, C=MyCountryCode"
      ProcessorArchitecture="x86"
      Name="MyCustomMyIDConfig" />
    </OptionalPackages>
</AppInstaller>
```

After adding your package, when you double-click the App Installer file to launch an interactive installation, you see your custom package included in the App Installer UI:



4.5 Configuring automatic updates

You can use App Installer files to configure automatic updates. When an update check is performed, Windows checks the App Installer file used to install the packages for changes. You can configure how often this check is performed, how it is performed, and what should happen when an update is found. Automatic updates apply to all packages in the App Installer file, including your own custom Modification Packages. Automatic update configuration applies regardless of how the App Installer file was deployed.

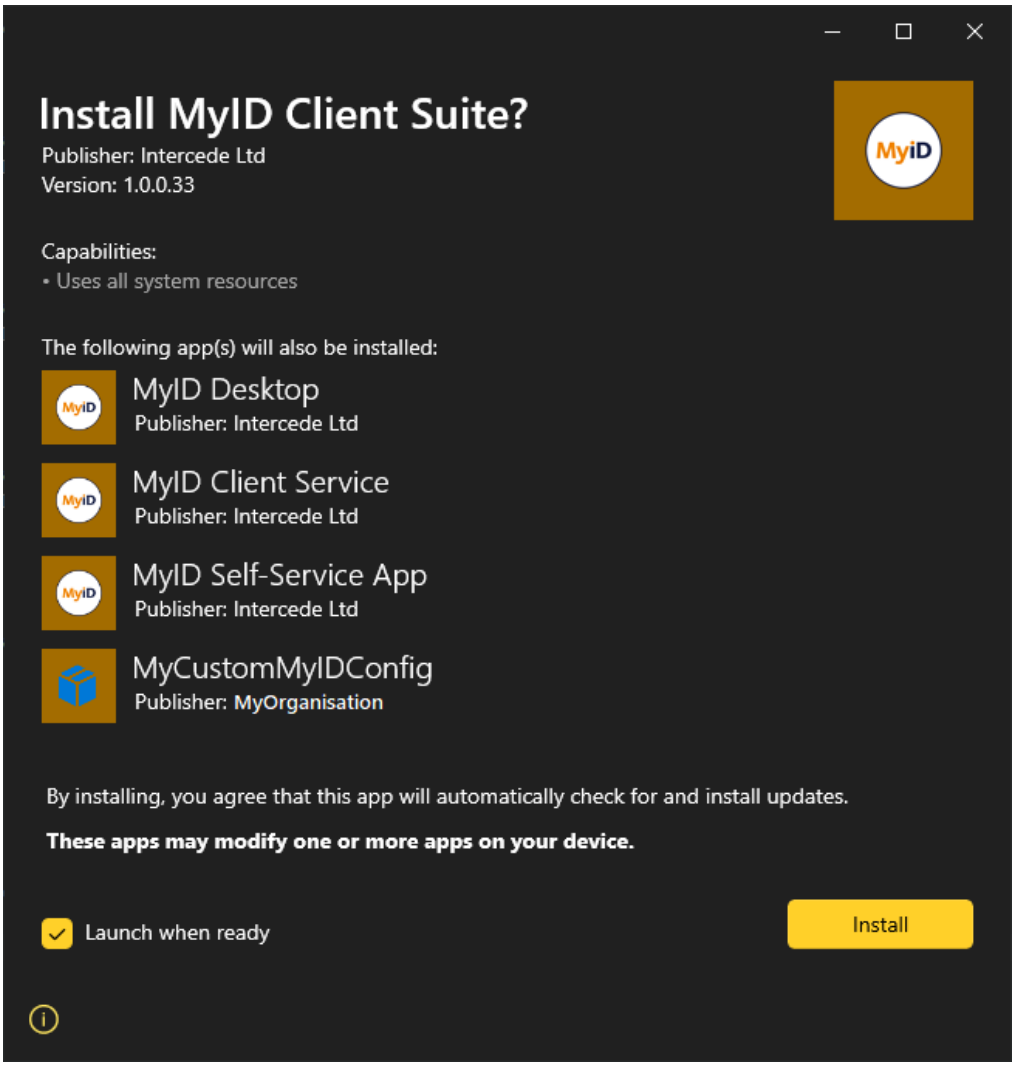
The following is one example of how you might configure updates:

```
<UpdateSettings>
    <OnLaunch HoursBetweenUpdateChecks="8"
              ShowPrompt="true"
              UpdateBlocksActivation="false"/>
    <ForceUpdateFromAnyVersion>true</ForceUpdateFromAnyVersion>
</UpdateSettings>
```

Add this as a child of the `AppInstaller` node. In the example, it would look like:

```
<?xml version="1.0" encoding="utf-8"?>
<AppInstaller
  Version="1.0.0.0"
  Uri="\\my.server.local\Installers$\MyIDClientSuite.appinstaller"
  xmlns="http://schemas.microsoft.com/appx/appinstaller/2018">
    <MainBundle
    Name="MyIDClientSuite"
    Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
    Version="1.0.0.33"
    Uri="\\my.server.local\Installers$\MyIDClientSuite_1.0.0.msixbundle" />
    <OptionalPackages>
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\DSKLauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDDesktopLauncher" />
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\MCSLauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDClientServiceLauncher" />
        <Package
      Version="1.0.0.5"
      Uri="\\my.server.local\Installers$\OptionalPackages\SSALauncher.msix"
      Publisher="CN=Intercede Ltd, O=Intercede Ltd, L=Lutterworth, C=GB"
      ProcessorArchitecture="x86"
      Name="MyIDSelfServiceAppLauncher" />
        <Package
      Version="1.0.0.0"
      Uri="\\my.server.local\Installers$\OptionalPackages\MyCustomMyIDConfig.msix"
      Publisher="CN=MyOrganisation, O=MyOrganisation, L=MyLocation, C=MyCountryCode"
      ProcessorArchitecture="x86"
      Name="MyCustomMyIDConfig" />
    </OptionalPackages>
    <UpdateSettings>
        <OnLaunch HoursBetweenUpdateChecks="8"
                  ShowPrompt="true"
                  UpdateBlocksActivation="false"/>
        <ForceUpdateFromAnyVersion>true</ForceUpdateFromAnyVersion>
    </UpdateSettings>
</AppInstaller>
```

When you double-click the App Installer file to launch an interactive installation, you see that the App Installer UI now indicates that the install will enable automatic updates:



In this example, the configured behavior would be:

- Check for updates on launch, with checks at least 8 hours apart.
- Prompt the user when an update is detected.
- Allow the user to bypass the update.
- Allow updates to perform downgrades as well as upgrades to allow for rollbacks.

Note: The update behavior available to you depends on the version of Windows you are using. For example, some of the configuration in this snippet requires at least Windows 10 1903 (`UpdateBlocksActivation`, `ShowPrompt`, and `ForceUpdateFromAnyVersion`).

For more information on configuring automatic updates, including which features are available on which platforms, see:

docs.microsoft.com/en-us/windows/msix/app-installer/update-settings.

4.6 Headless install with App Installer files

Microsoft provides an `Appx` PowerShell module for managing MSIX installations; see:

docs.microsoft.com/en-us/powershell/module/appx/?view=windowsserver2019-ps

The specific cmdlet you need to use to perform headless deployment of your App Installer file is `Add-AppxPackage`; see:

docs.microsoft.com/en-us/powershell/module/appx/add-appxpackage?view=windowsserver2019-ps

You can run this as a one-time startup task, or integrate as a scripted install; for example, in SCCM.

For example:

```
Add-AppxPackage -AppInstallerFile
"\\my.server.local\Installers$\MyIDClientSuite.appinstaller"
```

Note: Since MSIX installs per user, you must run this command for each user on the machine on which you want to install the packages.

Note: You cannot use the App Installer file to perform an uninstall – for that, you must provide the name of the package to remove. The following is an example of using `Remove-AppxPackage` to remove the `MyIDClientSuite` package and all applied add-ons from a machine:

```
Remove-AppxPackage $(Get-AppxPackage MyIDClientSuite).PackageFullName
```

See the following for more information:

docs.microsoft.com/en-us/powershell/module/appx/remove-appxpackage?view=windowsserver2019-ps

4.7 Interactive install with App Installer files

As the Intercede MSIX packages do not need administrator permissions, you can direct users to perform the install themselves. You can do this in the following ways:

- Give the user the App Installer file itself.

  If the user's machine can reach all of the locations specified inside the App Installer file, you can give them a copy of the file and they can double-click to run.

- Place the App Installer file in a shared location.

  The same as above, but rather than giving the user the file you direct them to its location.

- Publish the App Installer file on a website (internal or otherwise) using the `ms-appinstaller` protocol, and provide the link to users. For more information, see:

  docs.microsoft.com/en-us/windows/msix/app-installer/installing-windows10-apps-web

  Note: The `ms-appinstaller` protocol is disabled by default for security reasons. You must review and consider the implications of enabling it, including applying appropriate security policies.

Note: As with the headless install, you cannot use the App Installer file to perform an uninstall. Either the user must uninstall the app themselves, or you can use the `Remove-AppxPackage` cmdlet; for example:

```
Remove-AppxPackage $(Get-AppxPackage MyIDClientSuite).PackageFullName
```

See the following for more information:

docs.microsoft.com/en-us/powershell/module/appx/remove-appxpackage?view=windowsserver2019-ps

5 Deploying MyID clients without App Installer files

There may be scenarios where App Installer files are not suitable for your organization. In this case, you can use the cmdlets in the following PowerShell modules, depending on your deployment:

- `Appx`

  See the following for more information:

  docs.microsoft.com/en-us/powershell/module/appx/?view=windowsserver2019-ps

- `Dism`

  See the following for more information:

  docs.microsoft.com/en-us/powershell/module/dism/?view=windowsserver2019-ps

5.1 Headless install without App Installer files

To deploy the `MyIDClientSuite` package manually along with its optional packages, use the `Add-AppxPackage` cmdlet.

See the following for more information:

docs.microsoft.com/en-us/powershell/module/appx/add-appxpackage?view=windowsserver2019-ps

This command can be run by the user themselves, configured as a one-time startup task, and so on. The following is an example that installs the same packages as the example App Installer file:

```
Add-AppxPackage -Path "\\my.server.local\Installers$\MyIDClientSuite_
1.0.0.msixbundle" -ExternalPackages
"\\my.server.local\Installers$\OptionalPackages\DSKLauncher.msix","\\my.se
rver.local\Installers$\OptionalPackages\MCSLauncher.msix","\\my.server.loc
al\Installers$\OptionalPackages\SSALauncher.msix","\\my.server.local\Insta
llers$\OptionalPackages\MyCustomMyIDConfig.msix"
```

Note: Since MSIX installs per user, you must run this command for each user on the machine on which you want to install the packages.

To uninstall the clients deployed in this way, use the `Remove-AppxPackage` cmdlet:

```
Remove-AppxPackage $(Get-AppxPackage MyIDClientSuite).PackageFullName
```

See the following for more information:

docs.microsoft.com/en-us/powershell/module/appx/remove-appxpackage?view=windowsserver2019-ps

5.2 Interactive install without App Installer files

While it is possible for users to install the packages interactively by double-clicking the `.msix` files, this approach is not recommended without App Installer files. This is because:

- You must install packages in the correct order; for example, you must install the main MyID Client Suite package before installing the optional packages for MyID Desktop, the Self-Service App, or the MyID Client Service.
- You cannot perform an in-place update using this approach when using the `MyIDDesktopLauncher`, `MyIDClientServiceLauncher`, or `MyIDSelfServiceAppLauncher` packages as you must update all members of a related set at the same time, which is not possible using the interactive installers. To update, you must uninstall the old versions before you install the updated versions.

  See section *4.3*, *Related sets* for more information.

To carry out an interactive installation without App Installer files:

1. Uninstall any existing `MyIDClientSuite` installs.

   Uninstalling the main package also removes all of the optional packages.
2. Install the new main `MyIDClientSuite` package.
3. Install the optional packages one-by-one.

5.3 Provisioning to a Windows image

You can provision the `MyIDClientSuite` package to a Windows image, including an online image, so that it will be installed for all users. To do this, use the `Add-AppxProvisionedPackage` cmdlet.

For more information, see:

docs.microsoft.com/en-us/powershell/module/dism/add-appxprovisionedpackage?view=windowsserver2019-ps

The following is an example that provisions the same packages as our example App Installer file to an online image:

```
Add-AppxProvisionedPackage -Online -PackagePath
"\\my.server.local\Installers$\MyIDClientSuite_1.0.0.msixbundle" -
OptionalPackagePath
"\\my.server.local\Installers$\OptionalPackages\DSKLauncher.msix","\\my.se
rver.local\Installers$\OptionalPackages\MCSLauncher.msix","\\my.server.loc
al\Installers$\OptionalPackages\SSALauncher.msix","\\my.server.local\Insta
llers$\OptionalPackages\MyCustomMyIDConfig.msix" -SkipLicense
```

To remove a provisioned package from an image, use the `Remove-AppxProvisionedPackage` cmdlet. This is slightly more complicated as, unlike `Remove-AppxPackage`, removing the main `MyIDClientSuite` package will not remove Modification Packages, including those produced by Intercede – these must be removed separately.

For more information, see:

docs.microsoft.com/en-us/powershell/module/dism/remove-appxprovisionedpackage?view=windowsserver2019-ps

The following is an example of removing the clients provisioned in the previous example, followed by the `MyCustomMyIDConfig` Modification Package:

```
# Remove the main package and packages in its related set
Remove-AppxProvisionedPackage -Online -AllUsers -PackageName $(Get-
AppxProvisionedPackage -Online | Where-Object {$_.DisplayName -like
'MyIDClientSuite'}).PackageName

# Remove the custom Modification Package
Remove-AppxProvisionedPackage -Online -AllUsers -PackageName $(Get-
AppxProvisionedPackage -Online | Where-Object {$_.DisplayName -like
'MyCustomMyIDConfig'}).PackageName
```

Note: Many of the additional support packages provided by Intercede have been produced as Modification Packages, and so, if provisioned in this way, would have to be removed individually as above.