Recast

How-to

Last Modified on 04.21.26

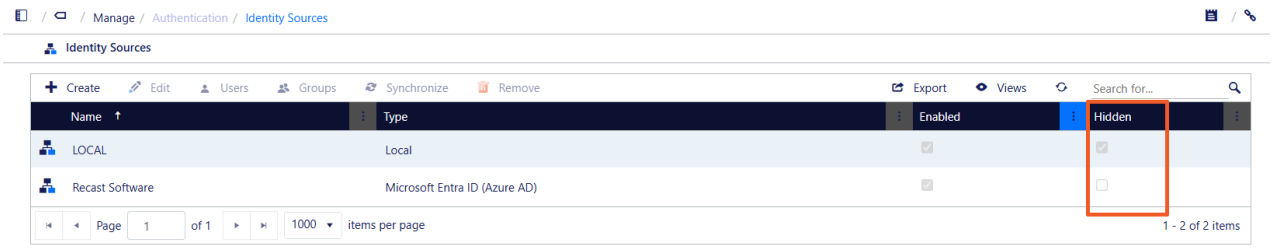
This How-to section includes instructions for popular Application Workspace tasks.

Access a hidden identity source

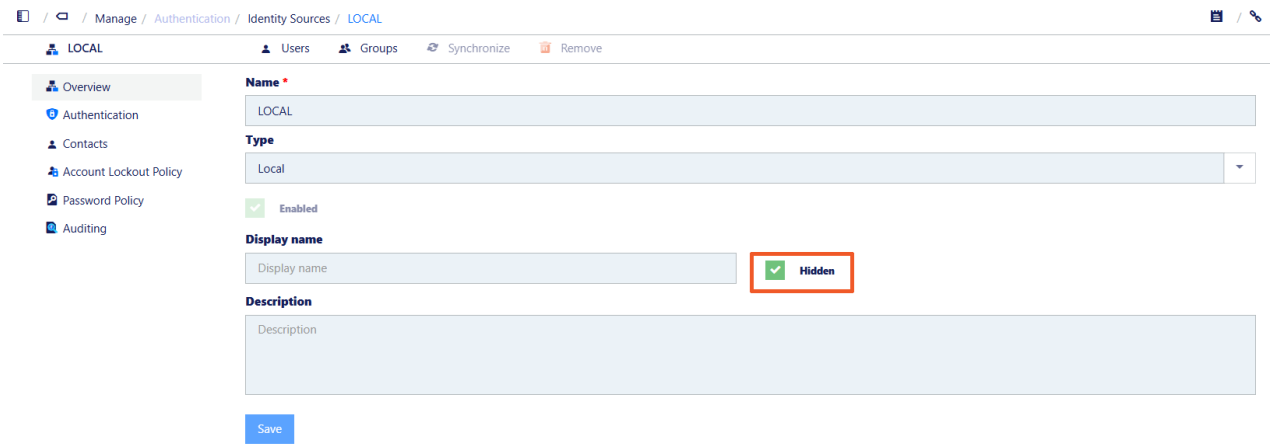
When you have multiple identity sources configured, you have the ability to selectively conceal them if you don't wish to display them to the customer. These concealed identity sources can still be accessed by using specific parameters within the URL of your Application Workspace zone in your web browser.

For instance, consider the following URL:
`https://workspace.recastsoftware.cloud/?sso=0&identitySource=local`

In the following scenario, there are three configured identity sources. To emphasize AzureAD as the primary source, two of them have been marked as hidden.



The **Hidden** option is in the **Overview** screen of the identity source:



In this scenario, you can still access the LDAP identity source by specifying its name in the URL:
`https://workspace.recastsoftware.cloud/?sso=0&identitySource=AD`

Deploy using Microsoft Intune & Autopilot (Windows – Bootstrapper 2.1)

> **Microsoft Windows only**
> The information in this article is applicable only to devices with Windows OS and Bootstrapper 2.1

Recast

Instead of manually converting multiple apps that you need into .intunewin files and then upload and configure them individually in Microsoft Intune, you can just add the Application Workspace Agent Bootstrapper to Microsoft Intune and then deploy multiple apps at once.
Currently, there is no possibility to enrol a machine running the Recast Agent with Microsoft Intune, because you cannot add the Application Workspace configuration file. There are only two ways to integrate Recast with Microsoft Intune: either by adding the Application Workspace Agent Bootstrapper within Microsoft Intune, or creating a Bootstrapper script. The following step-by-step instructions will help you deploy the bootstrapper within Microsoft Intune.

Application Workspace configurations

Create a self-signed certificate

1. In Application Workspace, navigate to **Manage** > **System** > **Device Registrations**.
2. Click on ➕ **Create** in the table toolbar.
3. In the **Create device registration** dialog box that opens, configure the following:
    - In **Type**, select *Certificate** and click **Next**.
    - In **Overview**, write a name, for example *Device enrollment* and click **Next**.
    - In **Settings**, check the **Use a self signed certificate for the device registration** option. This will create a self-signed certificate for you. Click **Next**.
    - In **Self signed**, write a common name and change the days valid and key size if needed. Click **Next**.
    - In **Summary**, uncheck the **Modify device registration after creation** option. Click **Finish**.

Create a device collection

1. In Application Workspace, navigate to **Manage** > **Identities** > **Device Collections**.
2. Click on ➕ **Create** in the table toolbar.
3. In the **Create device collection** dialog box that opens, configure the following:
    - In **Type**, select *Dynamic* and click **Next**.
    - In **Overview**, write a name, for example *MS Intune Devices* and click **Next**.
    - In **Summary**, leave the **Modify device registration after creation** option checked. Click **Finish**.
4. In the detailed view of the new device collection, configure the following:
    - In **Conditions**, click ➕ **Create filter set** in the screen top menu. Then click ➕ **Create filter**.
    - In the **Create filter** dialog box that opens, configure the following:
        - In **Type**, select *Device name*.
        - In **Value**, insert the prefix your organization uses within Microsoft Intune.
        - Click **Confirm**.

- Leave the operator set to AND. Create a new filter in the new filter set, with the following parameters:
  - In **Type**, select **System manufacturer**.
  - In **Value**, insert the system manufacturer your organization uses.
  - Click **Confirm**.
- Create a new filter in the new filter set, with the following parameters:
  - In **Type**, select **System model**.
  - In **Value**, insert the system model your organization uses.
  - Click **Confirm**.
- Click **Save** in the top menu of the **Conditions** screen.

Create a deployment

1. In Application Workspace, navigate to **Manage** > **Automation** > **Deployments**.
2. Click on ➕ **Create** in the table toolbar.
3. In the **Create deployment** dialog box that opens, configure the following:
   - In **Name**, write MS Intune Deployment. Click **Next** and then **Finish**.
4. In the detailed view of the new deployment, configure the following:
   - In **Packages**, click on the browse button ... and select the packages you want to deploy.
   - In **Assignments**, click on the browse button ... and select the MS Intune Devices device collection you previously created. Click **Confirm**.

Configure the Agent required for Microsoft Intune

1. Open Windows File Explorer and navigate to `Local Disk (C:)`.
2. Open the Temp folder. If it's not there, create one. You will use this folder to store all the required files.
3. Create a new folder named Recast.
   - Inside it, create a JSON file named `Agent.json`.
4. Open the JSON file and create your own settings based on the information we provide in Agent Configuration.
5. See Agent Configuration for the Liquit Workspace Agent or Agent Configuration for Application Workspace Agent for more information on creating an Agent configuration file.

It is important to have the following settings configured:

- The device must be configured to register itself with credentials or certificate, depending on the type of the Agent.
- **Context** set to *Device*.
- **Deployment.Autostart.Enabled** set to *true*.

- **Deployment.Enabled** set to *true*.
- **Deployment.Start** set to *false*. The deployment start must be *false* because the bootstrapper and the installer will automatically trigger the deployment.

> Important
> If you run the installer without the bootstrapper remember to set these flags correctly.

Agent.XML example

```
<Register>
   <Type>1</Type>
   <Username><![CDATA[LOCAL\wksimport]]></Username>
   <Password><![CDATA[P@ssw0rd]]></Password>
</Register>
<Deployment>
   <Enabled>True</Enabled>
   <Start>False</Start>
   <Cancel>False</Cancel>
   <Events>False</Events>
   <Context>Device</Context>
   <AutoStart>
      <Enabled>True</Enabled>
      <Timer>0</Timer>
      <Deployment>RecastSoftware</Deployment>
   </AutoStart>
</Deployment>
```

Agent.JSON example

```
{
   "zone":"https://example.recastsoftware.com",
   "registration": {
      "type": "Certificate"
   },
   "log": {
      "level": "Debug"
   },
   "deployment": {
      "enabled": true,
      "start": false,
      "context": "device",
      "cancel": true,
      "triggers": false,
      "autoStart": {
         "enabled": true,
         "deployment": "RecastSoftware",
         "timer": 0
      }
   }
}
```

Microsoft Intune configurations

Download all the files required for Microsoft Intune

1. Navigate to **Manage** > **Device Registration** and open the device registration you created earlier. In the detailed view of the registration navigate to **Settings**. Click on **Download for agent registration**. You will need this certificate later to create an intunewin file for Microsoft Intune.

2. Download the latest version of the bootstrapper in the `C:\Temp\Liquit` folder you created previously.

3. Download the latest version of IntuneWinAppUtil. You should download it in a folder different than `C:\Temp\Liquit` otherwise it will be included in the .intunewin file later on.

Prepare the .intunewin file for Microsoft Intune

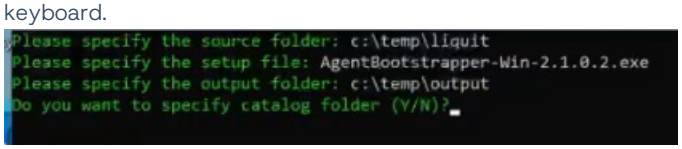
1. Start IntuneWinAppUtil.exe as Administrator

2. Inside the tool, navigate to `C:\Temp\Liquit`. Press the Enter key on your keyboard.

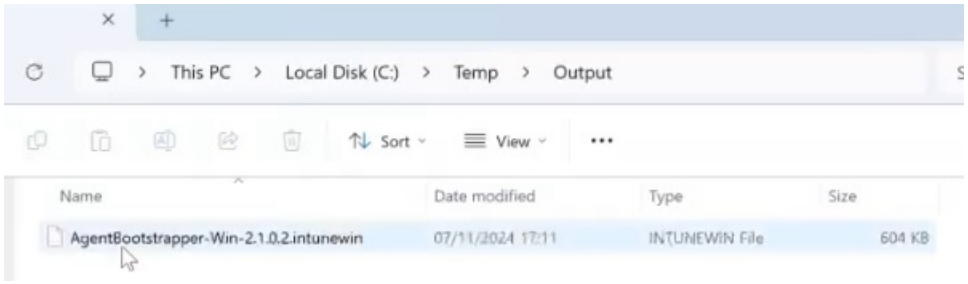
3. Then specify the AgentBootstrapper file. Press the Enter key on your keyboard.

4. Then specify a name for the output folder. For example `C:\Temp\Output`. Press the Enter key on your keyboard.

5. When prompted for the catalog folder, press the N key on your keyboard and then press the Enter key on your keyboard.



6. After the bootstrapper has run, navigate to the output folder to find the .intunewin file.

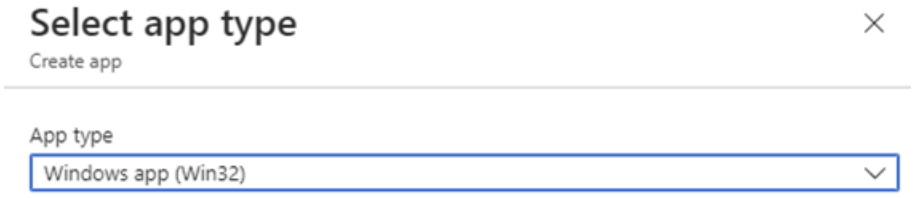


Upload the .intunewin file to Microsoft Intune

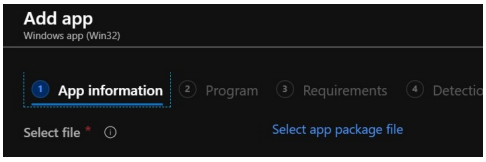
1. Go to Microsoft Intune admin center.

2. Select **Apps** > **By platform** > **Windows** > **+ Create**.

3. On the **Select app type** pane that opens, under the **Other app types**, select **Windows app (Win32)** and then click **Select**.



4. On the **App information** page click **Select app package file**.



5. The **Add package file** page opens where you upload the .intunewin file you previously prepared in Prepare the .intunewin file for Microsoft Intune, step 6. After you finish and click **OK** you can see that Intune already filled in some of the app info. Once you finish filling in all the necessary details, click **Next**.

6. On the **Program** page, configure the following:

    - the install command (see Application Workspace Agent Bootstrapper 2.1.0.2. For example: `AgentBootstrapper-Win-2.1.0.2.exe /startDeployment /waitForDeployment /logPath=C:\Windows\Temp`
      If you're using a certificate to register, specify the name and path. (/certificate="path/to/file.cer"). For more information, see Application Workspace Agent Bootstrapper 2.1.0.2.
      If you want to deploy a Liquit Workspace Agent and not the Application Workspace Agent, specify the `/legacyDownload` parameter. This is for the 3.10 agent or pre-4.0 servers.

    - the uninstall command, for example `AgentBootstrapper-Win-2.1.0.2.exe /uninstall`

    - After you finish specifying the desired command, click **Next**.

7. On the **Requirements** page, Select the 32 and 64-bit operating system architectures and the minimum operating system. Click **Next**.

8. On the **Detection rules** page, add your custom detection script for the Agent. For example:

```
# Define an array of file paths to check
$filePaths = @(
   "C:\Program Files\Liquit Universal Agent\Agent.exe",
   "C:\Program Files (x86)\Liquit Universal Agent\Agent.exe",
   "C:\Program Files (x86)\Liquit Workspace\Agent\Agent.exe"
)

# Loop through each file path and check if it exists
foreach ($path in $filePaths) {
   if (Test-Path $path) {
      Write-Host ("Application Workspace is installed at " + $Path)
      Exit 0
   }
}
Write-Host ("Application Workspace is not installed!")
Exit 1
```

If you signed the script, you can enable the option **Enforce script signature check and run script silently** to run the script silently.

8. On the **Dependencies** and **Supersedence** pages click **Next**.

9. On the **Assignments** page, assign it to devices, users or groups as needed. Click **Next**.

10. On **Review + create** page, review the values and settings that you entered for the app. Click **Create** to add the bootstrapper to Microsoft Intune.

Autopilot Enrollment Status Page

Recast

During the Device setup stage in the Autopilot ESP the Recast Agent will be installed, registered and then the deployment will start automatically,

On our YouTube channel, you can find a video of the entire walkthrough.

Deploy using Microsoft Intune (Windows – Bootstrapper 4.4)

> Microsoft Windows only
> The information in this article is applicable only to devices with Windows OS and Bootstrapper 4.4.
> The command lines and scripts in this article are just an example. When file names are specified, make sure you modify them accordingly before running the command/script.

Instead of manually converting multiple apps that you need into .intunewin files and then uploading and configuring them individually in Microsoft Intune, you can just add the Application Workspace Agent Bootstrapper to Microsoft Intune and then deploy multiple apps at once.

The following step-by-step instructions will help you deploy the Application Workspace Agent Bootstrapper within Microsoft Intune.

Prerequisites

Create an Entra ID identity source. For more information, see Microsoft Entra ID.

Application Workspace configurations

Create a self-signed certificate

1. In Application Workspace, navigate to **Manage** > **System** > **Device Registrations**.
2. Click on ✚ **Create** in the table toolbar.
3. In the **Create device registration** dialog box that opens, configure the following:
    - In **Type**, select *Certificate* and click **Next**.
    - In **Overview**, write a name, for example *Device enrollment* and click **Next**.
    - In **Settings**, check the **Use a self signed certificate for the device registration** option. This will create a self-signed certificate for you. Click **Next**.
    - In **Self signed**, write a common name and change the days valid and key size if needed. Click **Next**.
    - In **Summary**, leave the **Modify device registration after creation** option selected. Click **Finish**.
    - In the newly created certificate that opens, navigate to the **Settings** screen and click **Download for agent registration**. You need to save this file as you will use it when creating the .intunewin file that you will later upload to Intune.

Recast

Create a deployment

1. In Application Workspace, navigate to **Manage** > **Automation** > **Deployments**.
2. Click on ✚ **Create** in the table toolbar.
3. In the **Create deployment** dialog box that opens, configure the following:
    - Enter the desired name and description and click **Next**.
    - In **Summary**, leave the *Modify deployment after creation* selected.
4. In the detailed view of the new deployment, configure the following:
    - In **Packages**, use the lookup field or the browse button ... to select the packages you want to deploy. Be sure to specify the *Install* action.
    - In **Assignments**, use the lookup field or the browse button ... and select an existing device collection for a targeted deployment or the *All devices* predefined collection to deploy to all devices.

Create a Temp folder for storing files necessary for Microsoft Intune

1. Create a folder on `Local Disk (C:)` to store all the files required to create an .intunewin file for Microsoft Intune:
    - The self-signed certificate you downloaded before from Application Workspace.
    - The latest version of the bootstrapper.
    - (Optional) If there are specific settings that cannot be handled by the bootstrapper, you should create an Agent.json file and add it to this folder. If you don't, the bootstrapper creates one for you during deployment. The objects that you can configure in the command line are Zone, Registration and Deployments (except for cancel, trigger, zoneTimeout). You can specify command lines for the bootstrapper in Intune, as explained later in this article.

Agent.json file example

Make sure that the Agent.json is formatted as UTF-8.

```
{
 "zone": "https://company.liquit.com/ ",
 "promptZone": "Disabled",
 "registration": { "type": "Certificate" },
 "log": { "level": "Info" },
 "deployment": {
 "enabled": true,
 "start": true,
 "context": "device",
 "cancel": true,
 "triggers": false,
 "autoStart": {
 "enabled": true,
 "deployment": "Deployment",
 "timer": 0
 }
 },
 "login": {
 "enabled": true,
 "sso": true,
 "identitySource": "Microsoft Entra ID",
 "timeout": 4
 },
 "icon": {
 "enabled": true,
 "exit": false,
 "timeout": 30
 },
 "launcher": {
 "enabled": true,
 "state": "Default",
 "start": "Auto",
 "tiles": true,
 "minimal": false,
 "contextMenu": true,
 "sideMenu": "Tags",
 "close": true
 },
 "nativeIcons": {
 "enabled": true,
 "primary": true,
 "startMenuPath": "${Programs}\\Liquit"
 },
 "restrictZones": true,
 "trustedZones": ["company.liquit.com "]
}
```

Microsoft Intune configurations

Download IntuneWinAppUtil

Download the latest version of IntuneWinAppUtil to a folder different than `C:\Temp\Liquit`, otherwise it will be included in the .intunewin file later on.

Prepare the .intunewin file for Microsoft Intune

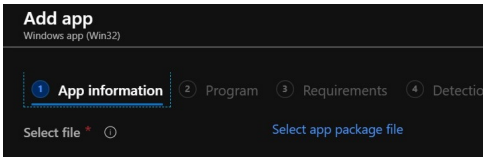
1. Start IntuneWinAppUtil.exe as Administrator.

2. Inside the tool, navigate to `C:\Temp\Liquit`. Press the Enter key on your keyboard.
3. Then specify the AgentBootstrapper file. Press the Enter key on your keyboard.
4. Then specify a name for the output folder. For example `C:\Temp\Output`. Press the Enter key on your keyboard.
5. When prompted for the catalog folder, press the N key and then the Enter key on your keyboard.
6. After the bootstrapper has run, navigate to the output folder to find the .intunewin file.

Upload the .intunewin file to Microsoft Intune

1. Go to Microsoft Intune admin center.
2. Select **Apps** > **By platform** > **Windows** > **+ Create**.
3. On the **Select app type** pane that opens, under the **Other app types**, select **Windows app (Win32)** and then click **Select**.

4. On the **App information** page click **Select app package file**.



5. The **Add package file** page opens where you upload the .intunewin file you previously prepared in Prepare the .intunewin file for Microsoft Intune, step 6. After you finish and click **OK**, you can see that Intune already filled in some of the app info. Once you finish filling in all the necessary details, click **Next**.
6. On the **Program** page, configure the following:
   - the install command (see Application Workspace Agent Bootstrapper 4.4). For example:

```
Bootstrapper.exe --zone "https://my.zone.com/" --startDeployment --deviceDeployment --registrationType Certificate --certificate "AgentRegistration.cer" --wait
```

or

```
Bootstrapper.exe --zone "https://my.zone.com/" --startDeployment --deviceDeployment --registrationType Credentials --registrationUsername "exampleUser" --registrationPassword "examplePass"``` --wait
```

   - the uninstall command, for example `Bootstrapper.exe --uninstall`
   - the log path, for example `--logPath "C:\Windows\Temp "`, to easily identify any errors in the process. If you do not specify a path, the current working directory will be used.
   - After you finish specifying the desired command, click **Next**.

7. On the **Requirements** page, select the 32 and 64-bit operating system architectures and the minimum operating system. Click **Next**.
8. On the **Detection rules** page, add your custom detection script for the Agent. For example:

```
# Define an array of file paths to check
$filePaths = @(
 "C:\Program Files\Liquit Universal Agent\Agent.exe",
 "C:\Program Files (x86)\Liquit Universal Agent\Agent.exe",
 "C:\Program Files (x86)\Liquit Workspace\Agent\Agent.exe"
)

# Loop through each file path and check if it exists
foreach ($path in $filePaths) {
 if (Test-Path $path) {
 Write-Host ("Application Workspace is installed at " + $Path)
 Exit 0
 }
}
Write-Host ("Application Workspace is not installed!")
Exit 1
```

If you signed the script, you can enable the option **Enforce script signature check and run script silently** to run the script silently.

9. On the **Dependencies** and **Supersedence** pages click **Next**.
10. On the **Assignments** page, assign it to devices, users or groups as needed. Click **Next**.
11. On **Review + create** page, review the values and settings that you entered for the app. Click **Create** to add the bootstrapper to Microsoft Intune.

Reboot

The device requires a reboot after the initial configuration. Until the deployment becomes inactive, the Application Workspace Agent service is disabled, and the Application Workspace Launcher is not available.

Deploy using Microsoft Intune (macOS – Bootstrapper 4.4)

macOS only
The information in this article is applicable only to devices with macOS and Bootstrapper 4.4.
The command lines and scripts in this article are just an example. When file names are specified, make sure you modify them accordingly before running the command/script.
This guide does not cover how to create a .PKG file.

Prerequisites

Create an Entra ID identity source. For more information, see Microsoft Entra ID.

Application Workspace configurations

Create a self-signed certificate

1. In Application Workspace, navigate to **Manage** > **System** > **Device Registrations**.

2. Click on ➕ **Create** in the table toolbar.
3. In the **Create device registration** dialog box that opens, configure the following:
    - As the **Type**, select *Certificate* and click **Next**.
    - In **Overview**, write a name, for example *Device enrollment* and click **Next**.
    - In **Settings**, check the **Use a self signed certificate for the device registration** option. This will create a self-signed certificate for you. Click **Next**.
    - In **Self signed**, write a common name and change the days valid and key size if needed. Click **Next**.
    - In **Summary**, leave the **Modify device registration after creation** option selected. Click **Finish**.
    - In the newly created certificate that opens, navigate to the **Settings** screen and click **Download for agent registration**. You need to save this file, as you will use it when creating the .PKG file that you will later upload to Intune.

> **Certificate trust for self-hosted environments**
> If you are self-hosting Application Workspace, client devices must trust the Application Workspace server. This typically requires deploying one or more certificates to your devices.
> If your server uses a certificate issued by a public Certificate Authority (CA), no additional configuration is required.
> If you are using an internal CA or the self-signed certificate generated during the Application Workspace installation, you must export the required certificates and deploy them to client devices. This is usually done through Microsoft Intune or your preferred MDM solution.
> Depending on your setup, you may need to deploy:
>
> The internal CA root certificate
> The Application Workspace self-signed certificate
>
> Ensure these certificates are installed in the Trusted Root Certification Authorities store on the client devices to establish a valid trust chain between the clients and the Application Workspace server.
> You can create one or more Intune certificate profiles as needed, based on your environment and configuration. For guidance, see: Microsoft documentation.

Create a deployment

1. In Application Workspace, navigate to **Manage** > **Automation** > **Deployments**.
2. Click on ➕ **Create** in the table toolbar.
3. In the **Create deployment** dialog box that opens, enter the desired name and description.
    - In **Name**, write MS Intune Deployment. Click **Next** and then **Finish**.
    - In **Summary**, leave the *Modify deployment after creation* selected.
4. In the detailed view of the new deployment, configure the following:
    - In **Packages**, use the lookup field or the browse button ... to select the packages you want to deploy. Be sure to specify the *Install* action.
    - In **Assignments**, use the lookup field or the browse button ... and select an existing device collection for a targeted deployment or the *All devices* predefined collection to deploy to all devices.

Recast

Build custom .PKG

Create a folder where you will store all files that will be included in the .PKG file, namely:

- The agent bootstrapper for macOS from our downloads page https://download.liquit.com
- (Optional) The certificate is for certificate-based device registration.
- (Optional) If there are specific settings that cannot be handled by the bootstrapper, you should create an Agent.json file and add it to this folder. If you don't, the bootstrapper creates one for you during deployment. The objects that you can configure in the command line are Zone, Registration and Deployments (except for cancel, trigger, zoneTimeout). You can specify command lines for the bootstrapper in Intune, as explained later in this article.

Important notes
Make sure the files in this folder have the right permissions and attributes. Use the terminal and navigate to this folder, and make sure the bootstrapper and post-install script files are executable using the chmod +x command.
You can create the .PKG file using tools like Package Builder from Araelium https://www.araelium.com/packagebuilder
There is no need to include a post-install script. The Bootstrapper handles the log file and launch agent, including all the required post-install actions. On macOS, your only action is to run the Bootstrapper executable with the correct command-line arguments.

Agent.json file example

Make sure that the Agent.json is formatted as UTF-8.

```
{
 "zone": "https://fqdn.yoursiteserver.com",
 "promptZone": "Disabled",
 "registration": { "type": "Certificate" },
 "deployment": {
 "enabled": true,
 "start": false,
 "context": "device",
 "cancel": false,
 "triggers": false,
 "autoStart": {
 "enabled": true,
 "deployment": "Your Deployment Name",
 "timer": 0
 }
 },
 "log": { "level": "Info" },
 "icon": { "enabled": true, "exit": true, "timeout": 30 },
 "launcher": {
 "enabled": true,
 "state": "Default",
 "start": "Auto",
 "tiles": true,
 "minimal": false,
 "contextMenu": true,
 "sideMenu": "Tags",
 "close": true
 },
 "restrictZones": true,
 "trustedZones": ["FQDN.yoursiteserver.com"]
}
```

Microsoft Intune configurations

Deploying the Liquit Root CA Certificate via Microsoft Intune

1. Download the Liquit Root CA certificate from our download page . If the Application Workspace Universal Agent is already installed, you can find the Liquit Root CA certificate on macOS at `/Applications/Liquit/Contents/Resources`, named Agent.pfx or download it from our download page . This certificate is part of the self-signed certificate chain and therefore needs to be trusted as well.

2. Go to the Microsoft Intune admin center.

3. Select **Devices** > **Platform** > **macOS** > **Configuration** > **Policies** > **+ Create** > **+New policy**.

4. In the Create a Profile type page that opens, under Profile type, select Template.

5. Select Trusted certificate, then click Create.

6. In the Trusted certificate page that opens, configure the following:
    - In **Basics**, fill in the necessary fields. Click Next.
    - In **Configuration settings**, set Deployment Channel to Device Channel, and in Certificate file upload the certificate from step 1.
    - In **Assignments**, click Add groups and add a group of devices that should receive this certificate.

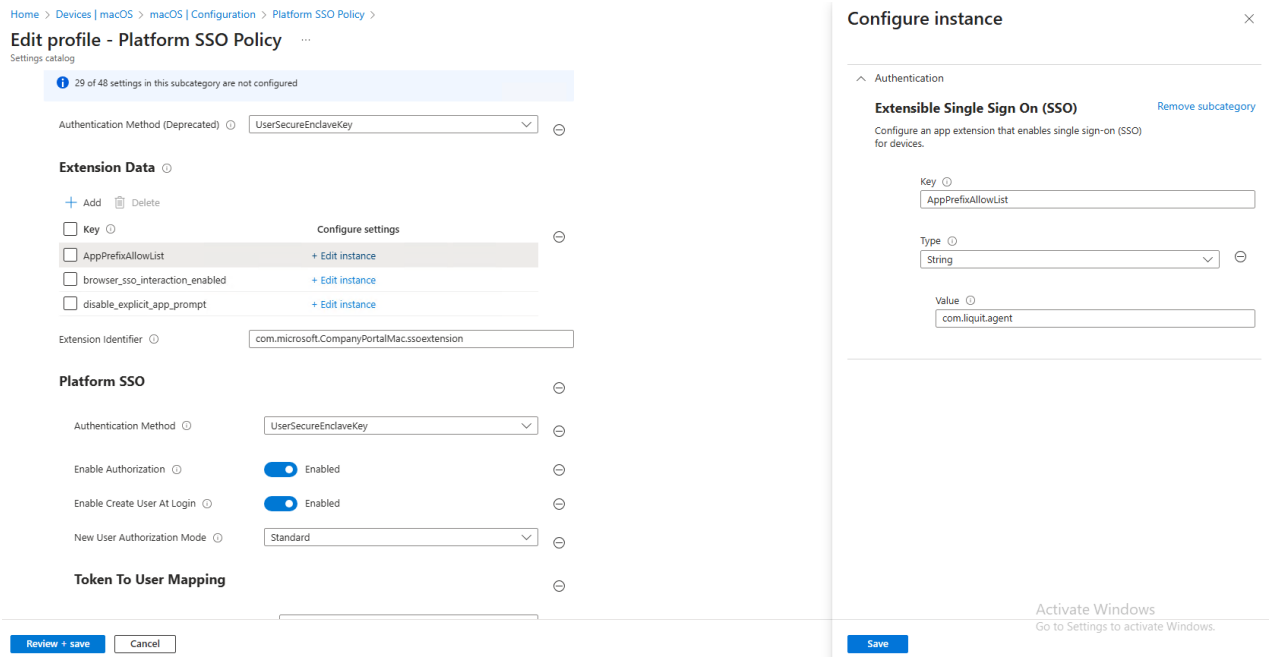
- On the **Review + create** page, review the values and settings that you entered. After you click **Create**, Intune will install the Application Workspace certificate on the group of macOS devices.

Create a macOS app in Intune

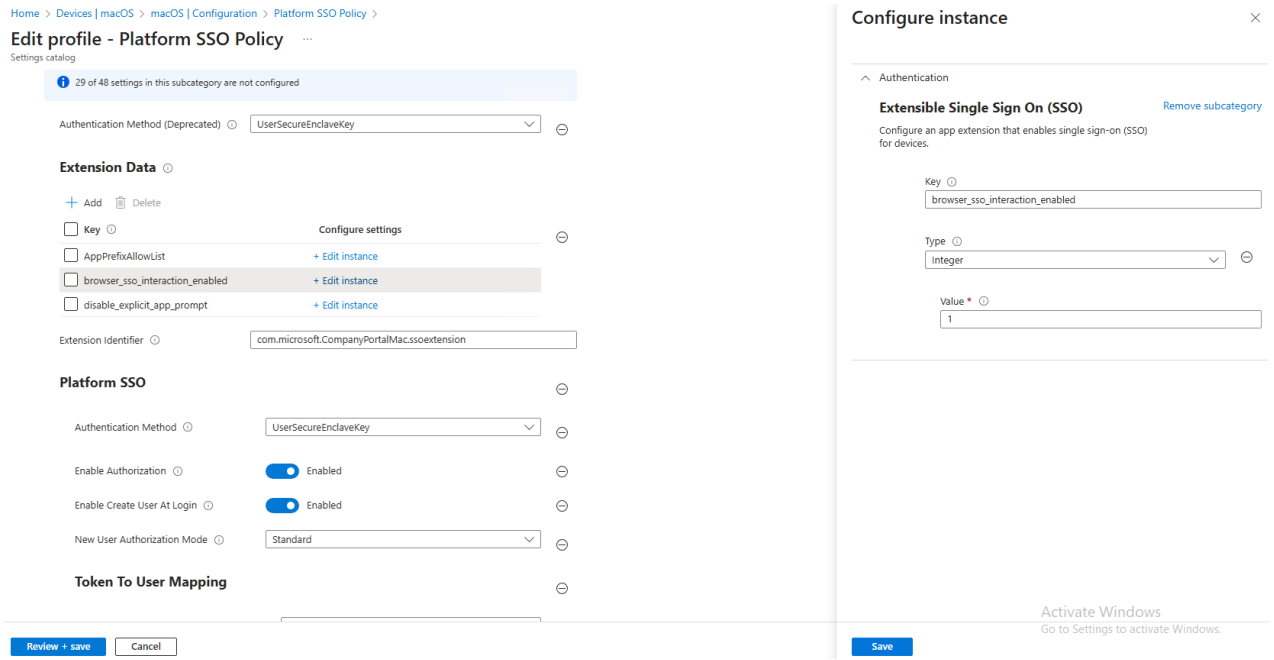
1. Go to the Microsoft Intune admin center.
2. Select **Apps** > **Platform** > **macOS** > **+ Create**.
3. On the Select app type pane that opens, under **App type**, select 'macOS app (PKG)' and then click **Select**.
4. The Add App page opens, where you configure the following:
   - On the **App information** page, click **Select app package file**.
   - On the **Add package file** page that opens, upload the .PKG file you previously prepared. Once you finish filling in all the necessary details, click **Next**.
   - On the **Program file** page, you can add the post-install script, if you did not already add it in the .pkg file itself.
   - On the **Requirements** page, choose the latest supported version of macOS.
   - On the **Detection rules** page, modify the bundle version to match the Application Workspace Agent version you intend to deploy.
   - On the **Review + create** page, review the values and settings that you entered for the app. Click **Create** to add the .pkg file to Microsoft Intune.

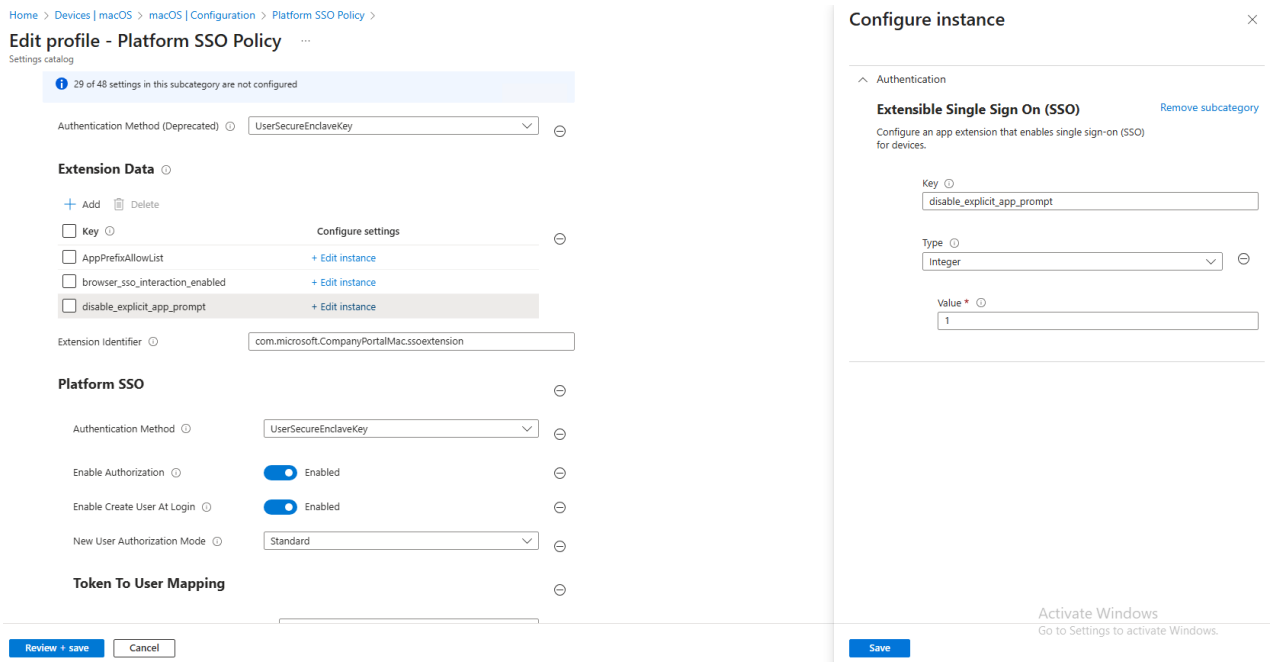
SSO configuration

Follow Microsoft's guide to create a Platform SSO policy in Intune.
Then apply the following Application Workspace – specific settings:

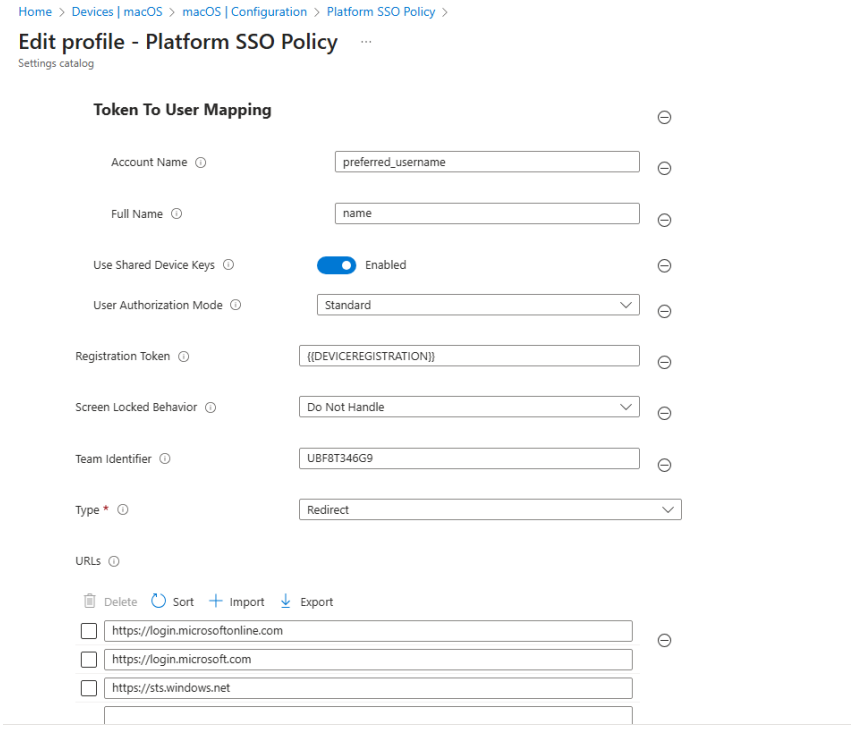


Recast


Home > Devices | macOS > macOS | Configuration > Platform SSO Policy >
Edit profile - Platform SSO Policy
Settings catalog
29 of 48 settings in this subcategory are not configured
Authentication Method (Deprecated)
UserSecureEnclaveKey
Extension Data
Add
Delete
Key
Configure settings
AppPrefixAllowList
+ Edit instance
browser_sso_interaction_enabled
+ Edit instance
disable_explicit_app_prompt
+ Edit instance
Extension Identifier
com.microsoft.CompanyPortalMac.ssoextension
Platform SSO
Authentication Method
UserSecureEnclaveKey
Enable Authorization
Enabled
Enable Create User At Login
Enabled
New User Authorization Mode
Standard
Token To User Mapping
Review + save
Cancel
Configure instance
Authentication
Extensible Single Sign On (SSO)
Remove subcategory
Configure an app extension that enables single sign-on (SSO) for devices.
Key
browser_sso_interaction_enabled
Type
Integer
Value *
1
Save
Activate Windows
Go to Settings to activate Windows.


Home > Devices | macOS > macOS | Configuration > Platform SSO Policy >
Edit profile - Platform SSO Policy
Settings catalog
29 of 48 settings in this subcategory are not configured
Authentication Method (Deprecated)
UserSecureEnclaveKey
Extension Data
Add
Delete
Key
Configure settings
AppPrefixAllowList
+ Edit instance
browser_sso_interaction_enabled
+ Edit instance
disable_explicit_app_prompt
+ Edit instance
Extension Identifier
com.microsoft.CompanyPortalMac.ssoextension
Platform SSO
Authentication Method
UserSecureEnclaveKey
Enable Authorization
Enabled
Enable Create User At Login
Enabled
New User Authorization Mode
Standard
Token To User Mapping
Review + save
Cancel
Configure instance
Authentication
Extensible Single Sign On (SSO)
Remove subcategory
Configure an app extension that enables single sign-on (SSO) for devices.
Key
disable_explicit_app_prompt
Type
Integer
Value *
1
Save
Activate Windows
Go to Settings to activate Windows.

Recast



Reboot

The device requires a reboot after the initial configuration. Until the deployment becomes inactive, the Application Workspace Agent service is disabled, and the Application Workspace Launcher is not available.

Further reading

Manage MacOS with Intune, including Apple Business Manager, Defender Enrollment, Platform SSO, and much more – The Complete Guide Part 1

Configure an Application Workspace connector

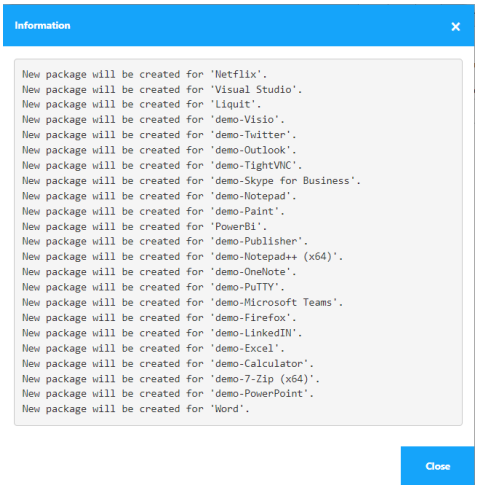
This example shows how to configure an Application Workspace connector using the LOCAL identity source.

1. In the source Application Workspace zone, create a new user in the LOCAL identity source called *Test*.
2. Assign a *Connector* type access policy to the newly created *Test* user.
3. Entitle the *Test* user to all the packages that you want to export later. You can do that in **Manage** > **Users** > *Test* user > **Packages** screen or in **Manage** > **Workspace** > **Packages** > desired package > **Entitlements** screen.
4. In the destination Application Workspace zone, create an Application Workspace type connector with the following parameters:
    - In the **Overview** tab:
      Direction *Pull*
      Synchronization method *Synchronize*
      Stage *Development*
      Package name prefix *RecastSoftware*
    - In the **Settings** tab:
      URL the URL of the source Application Workspace zone,
      Username insert "LOCAL\Test" (the username of the user configured at steps 1–2)

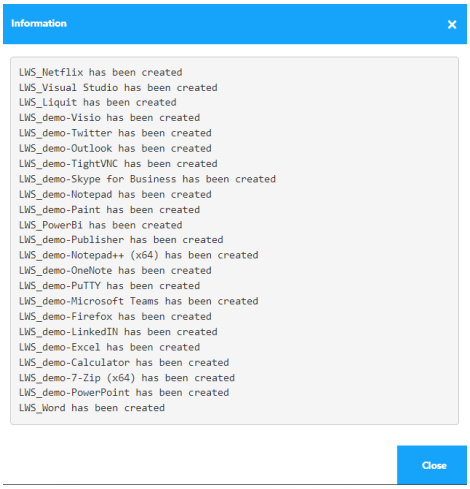
Password insert the password of the *Test* user (the user configured at steps 1–2)

- In the **Summary** tab, leave the **Modify connector after creation** selected

5. In the connector detailed view that opens, in the **Overview** screen, click **Synchronize**.
6. In the **Synchronize** dialog box that opens, select *Check for updates*. A log will appear, containing all the packages that will be created in the target Application Workspace zone



7. Close the log, click **Synchronize** again and this time select *Download and apply updates*. After it finishes a log will appear listing all the newly created packages.

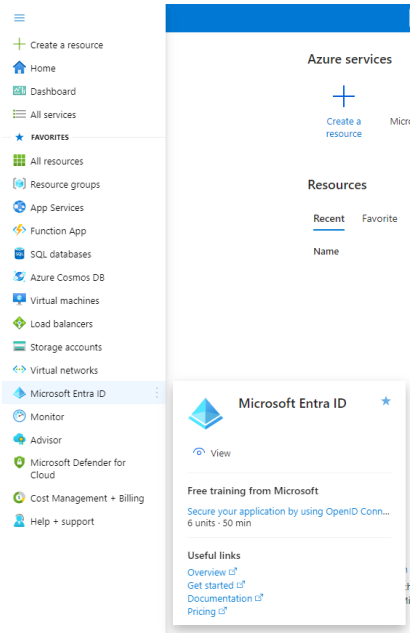


8. Close the log and go to the **Managed packages** screen to access all the packages you imported.

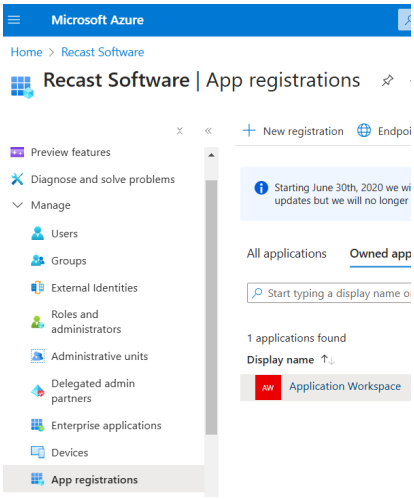
Configure the Microsoft Graph mail server

Register an application in Azure Portal

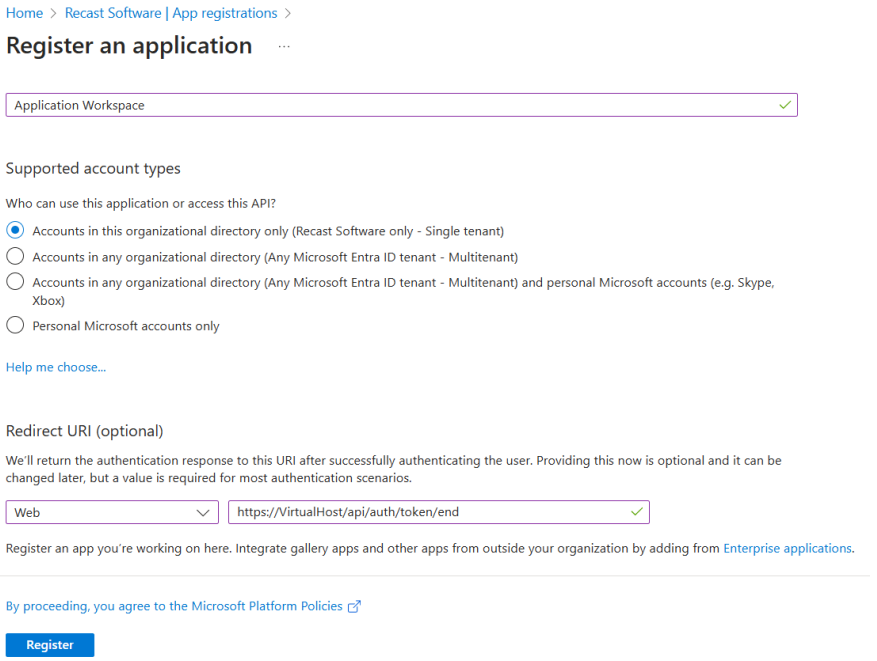
1. Log into the Azure Portal.
2. In the Azure Portal menu, navigate to **Microsoft Entra ID**.



3. In the left pane, navigate to **Manage** > **App registrations**.



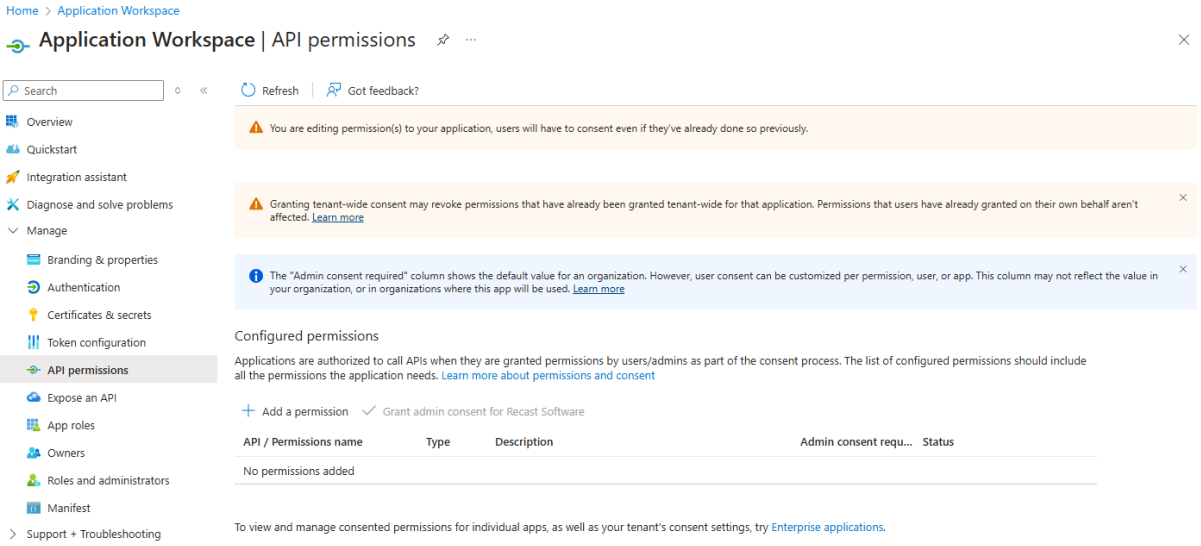
4. Click **+ New registration** on the top toolbar.



5. In the **Register an application** window that opens, configure the following:
    - In the **Supported account types** section select *Accounts in this organizational directory only (tenant only – Single tenant)*. For more information, see Microsoft documentation.
    - In the **Redirect URI (optional)** section select *Web* and in the value field insert the FQDN of the Application Workspace Zone you want to add, with the `/api/auth/token/end` suffix.

6. Click **Register** on the bottom left, to complete the initial app registration.

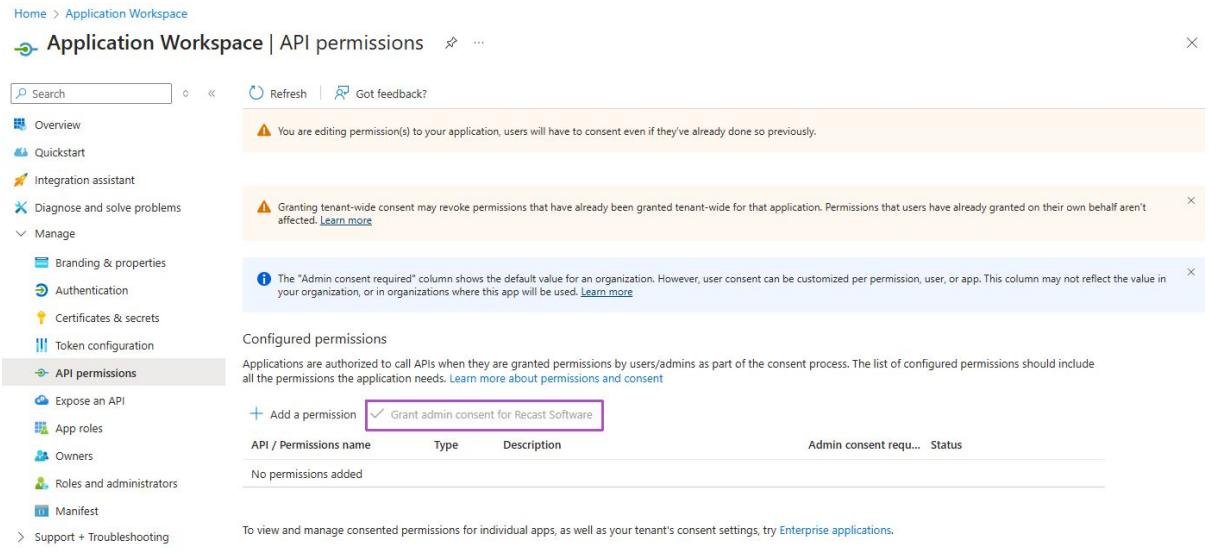
7. You need to generate a client secret that facilitates communication between Application Workspace and Microsoft Entra ID (Azure AD). In the newly created app registration, navigate to **Manage** > **Certificates & secrets** > **Client secrets** > **New client secret**.



8. Add a description and an expiration date for your client secret and then click **Add**. Note down your client secret after you create it because there is no way of retrieving the value after you leave this screen.

9. You need to add permission to your app registration. With the new app open, navigate to **Manage** > **API permissions** and add the *Mail.Send* permission.

10. After you add the permission, click **Grant admin consent for {your tenant}** above the permission list. It can take up to an hour before these settings take effect in Microsoft Entra ID (Azure AD).



Example:

```
https://<Virtual Host>/api/auth/token/end
```

Application Workspace configuration

1. In Application Workspace, navigate to **Manage** > **System** > **Mail Settings**.

2. Click **+ Create**. The **Create mail server** dialog box opens.

3. On the **Type** screen, select *Microsoft Graph*.

4. On the **Overview** screen enter the desired name, description and priority. After you finish inserting all necessary information, click **Next**.

5. On the **Settings** screen, configure the following:

    - Enter the client ID and tenant ID of the application you previously registered in Register an application in Azure Portal . You can find them in the Overview screen within Microsoft Entra ID (Azure AD).

    - The client secret you generated earlier in Register an application in Azure Portal at step 7.

    - In the **From** field insert the mail address used to send the emails.

6. On the **Summary** screen, click **Finish**.

Create a category and assign packages to it

Before creating a category, make sure the desired packages are created.

1. Go to **Manage** > **Workspace** > **Categories**.
2. Click **Create** in the table toolbar.
3. In the **Create category** dialog box that opens, in **Overview**, fill in the necessary info. Click **Next**.
4. In **Summary**, leave the **Modify category after creation** selected. Click **Next**.
5. In the detailed view screen that opens, in **Packages**, search for the necessary package and add it.
6. Double-click the entry of the package to open it.
7. Go to **Entitlements** and add the necessary identities. This step makes the category available for the user in the Side Menu of **Workspace** or **Catalog**, depending on where you configured the package to be published.

Create a license and assign packages to it

Before creating a license, make sure the desired packages are created.

1. Go to **Manage** > **Workspace** > **Licenses**.
2. Click **Create** in the table toolbar.
3. In the **Create license** dialog box that opens, in **Overview**, fill in the necessary info. Click **Next**.
4. In **License** fill in the necessary info and click **Next**.
5. In **Summary**, leave the **Modify license after creation** selected. Click **Next**.
6. In the detailed view screen that opens, in **Overview**, insert the license key.
7. In **Packages**, search for the necessary package and add it.
8. Double-click the entry of the package to open it.
9. Go to **Entitlements** and add the necessary identities and select the Catalog publish stage for them. This step makes the category available for the user in the Side Menu of **Catalog**.

Create a managed package using Application Workspace Setup Store connector

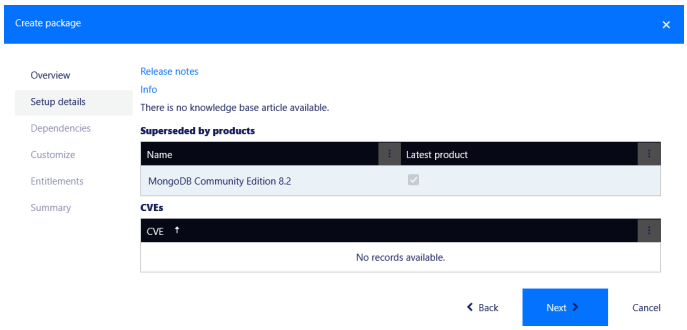
In our example, we will use the MongoDB Community Edition 6.0 product and go through all the steps of creating a managed package in Application Workspace via the Application Workspace Setup Store.

1. In your Application Workspace Setup Store connector, navigate to **Overview** > **Resources**, and search for MongoDB Community Edition 6.0.
2. Double-click on it or select it and then click **Create package** in the table toolbar.
3. In the **Create package** dialog box that opens, under **Overview**:
    - Enter a name for your new package or keep the default name.
    - As the **Package type**, select 'Managed'. Managed means that every time MongoDB releases a new version for this product, the Application Workspace Setup Store connector updates the package with the latest version.

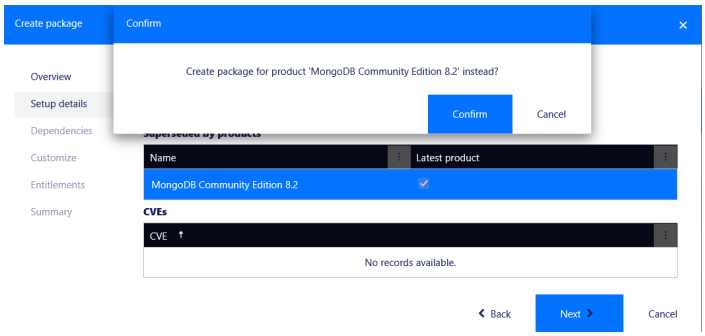
For more details about package types, see Connectors.

- In **Publish**, select the stage to which you want to publish the new package. For more details, see Packages.
- Leave the **Update existing packages** unchecked. If this option is selected, it will overwrite an existing package. Note that, if a package already exists and this option is not selected, it will trigger an error as it cannot create a duplicate.

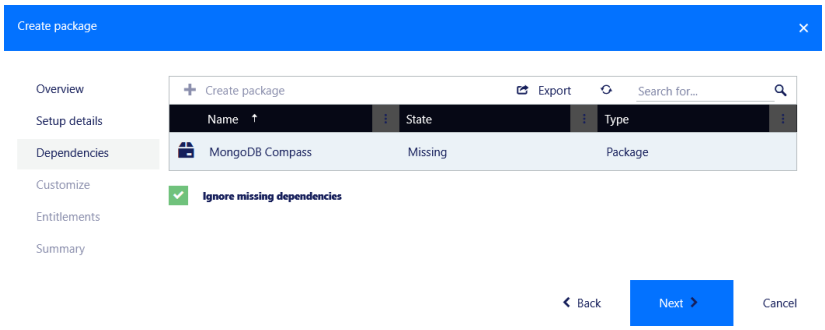
4. In **Setup details**, you can access links to the release notes and product description on the developer's official website, links to Recast knowledge base articles, virus scan scores and CVE reports in case any vulnerabilities and exposures exist for this resource. Also, in the **Superseded by products** table, you can view all the product versions within Application Workspace Setup Store that supersede the currently selected version. Note that the supercedence info is displayed only in the **Create package** dialog box, of a new package created from the Application Workspace Setup Store connector.



If you decide that you want to install one of the versions marked as superseded in this table, just double-click on it, and then click **Confirm**. It will open a new **Create package** dialog box for this version.

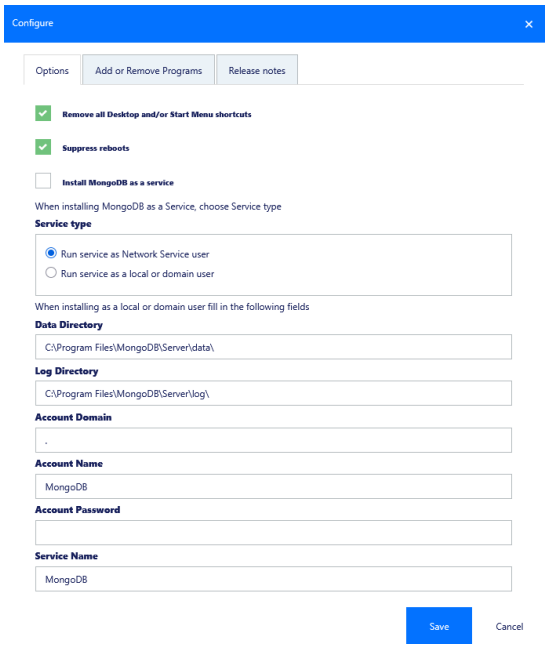


5. In **Dependencies**, Application Workspace lists all prerequisite software components and automatically detects if they are installed or not on your local system. You can easily create a package for each uninstalled component right from the dialog box by double-clicking it or by selecting it and pressing **Create package** in the table toolbar. It opens a new Create package dialog box where you configure the new package.
Select **Ignore missing dependencies** if you do not want to create managed packages for connected dependencies.



6. In **Customize**, you can access the customization wizard that helps you browse through additional settings offered by

the developer to choose the ones that suit your needs.



7. In **Entitlements**, add the identity (like contexts, device collections, devices, groups, users and user collections) to which you wish to publish the newly created package.

8. In **Summary**, leave the **Modify package after creation** selected if you want to open the newly created package and further configure it.

> Catalog information prefilled
> If you configure the managed package to be published in the user's catalog, the following sections come prefilled: excerpt, description, media, details and website. For more information see Packages.

Further reading

CVE, Then and Now: Overview and Practical Integration with Application Workspace

Deploy a package with hidden entitlement

Scenario

You have a script which synchronizes to OneDrive the SharePoint sites the user can access. The script also removes the SharePoint sites the user is not entitled to anymore, but which are still synced to OneDrive.

You need this script to run automatically on the user devices without relying on user interaction.

Steps

1. Add a new entitlement to the package.

2. In the **Add entitlement** dialog box that opens,
    - In the **Overview** tab, **Publish** field, select *Hidden*.
    - In the **Events** tab add a 'Application Workspace Logon' type event, with a 'Launch' type action.

3. Click **Confirm**.

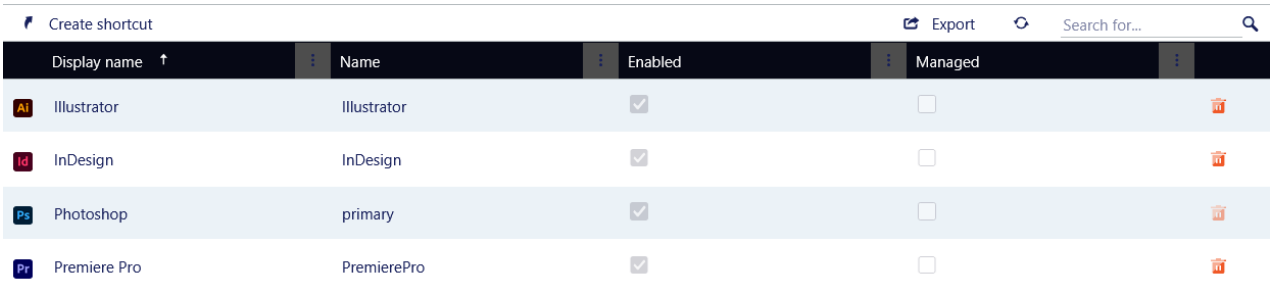
The package will start when the entitled user logs in to Application Workspace, without requiring manual user interaction. Now the package will not be visible in the Application Workspace, since no user interaction is needed.

Deploy a package with multiple shortcuts

How to deploy multiple shortcuts for a single package

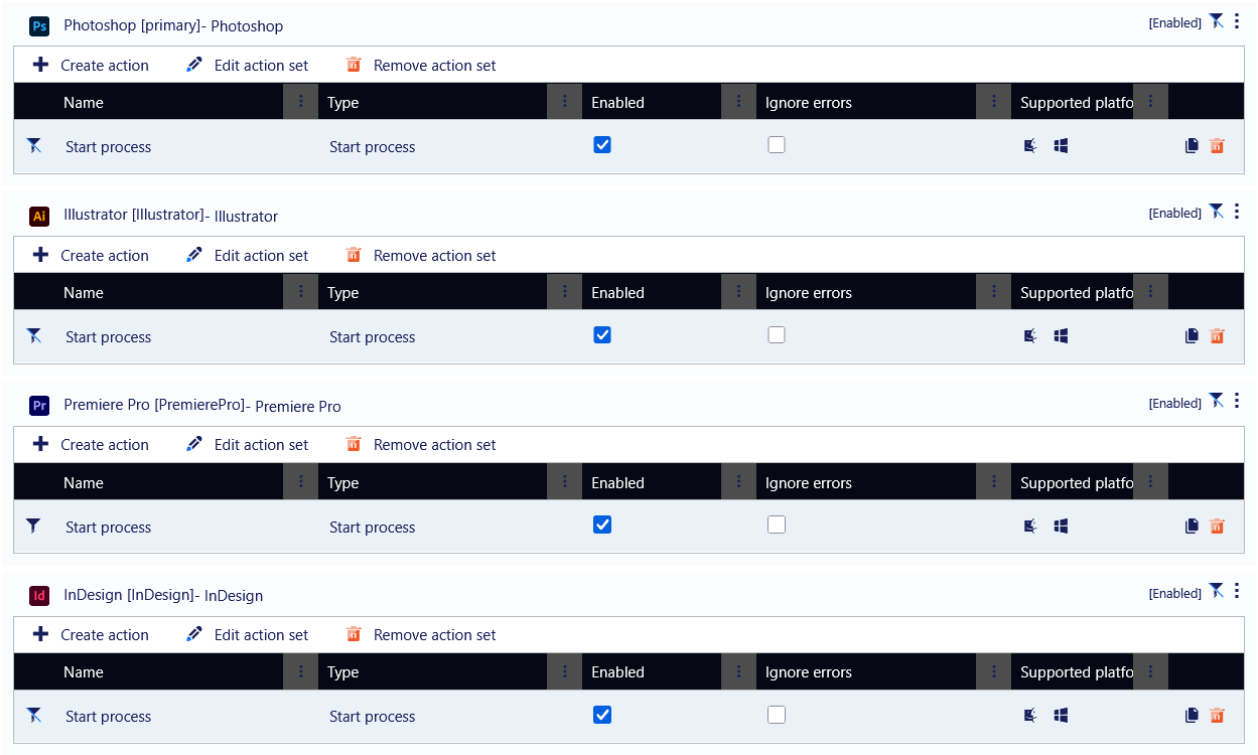
Let's say you create an Adobe Creativity and Design suite package and you want to make several applications available tc your users as individual smart icons: Photoshop, Premiere Pro, Illustrator, InDesign.

1. Go to **Manage** > **Workspace** > **Packages**.
2. Create a custom package called *Adobe Creativity and Design suite* and select an icon for it.
3. Go to **Entitlements** and assign your desired identities.
4. Go to **Release** > **Shortcuts**.
5. Open the primary shortcut and name it *Photoshop* and select an icon for it. Create additional shortcuts for Premiere Pro, Illustrator, InDesign and set icons for them.



6. Go to **Releases** > **Actions**.
7. Create 4 *Launch* type action sets for each application and select the corresponding shortcuts created at steps 4–5.

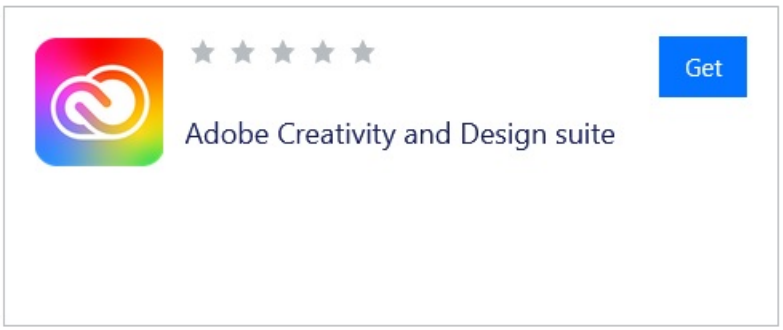
Launch



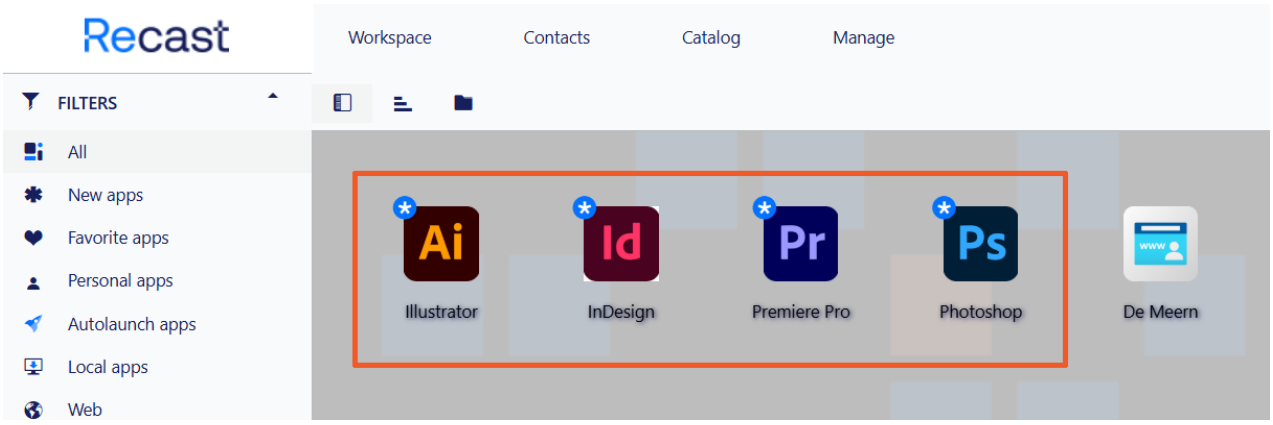
8. Inside each action set create a *Start process* action for all the 4 applications.

9. Move the *Development* stage to *Production*, to make the package available to your users.

On the user's device, in the **Catalog** tab of Application Workspace, the Adobe Creativity and Design suite package will be displayed as a single item.



After the user gets the package from the **Catalog** tab to the **Workspace** tab, 4 new smart icons will be displayed in the **Workspace** tab and each one will open a different app.



Shortcuts can be set only in *Launch* type actions/action sets.

Deploy custom fonts

There are multiple ways in which you can deploy fonts across devices. For example, you can use theRegisterFonts action inside an MSI. This is the official Microsoft-supported way of installing a particular font, which not only copies the file to the `C:Windows\Fonts` folder but also takes care of the registration process.

But using the RegisterFonts action would mean creating an MSI package with one of your favourite MSI packaging tools. An alternative is to use PowerShell.

Within Application Workspace it's more simple.

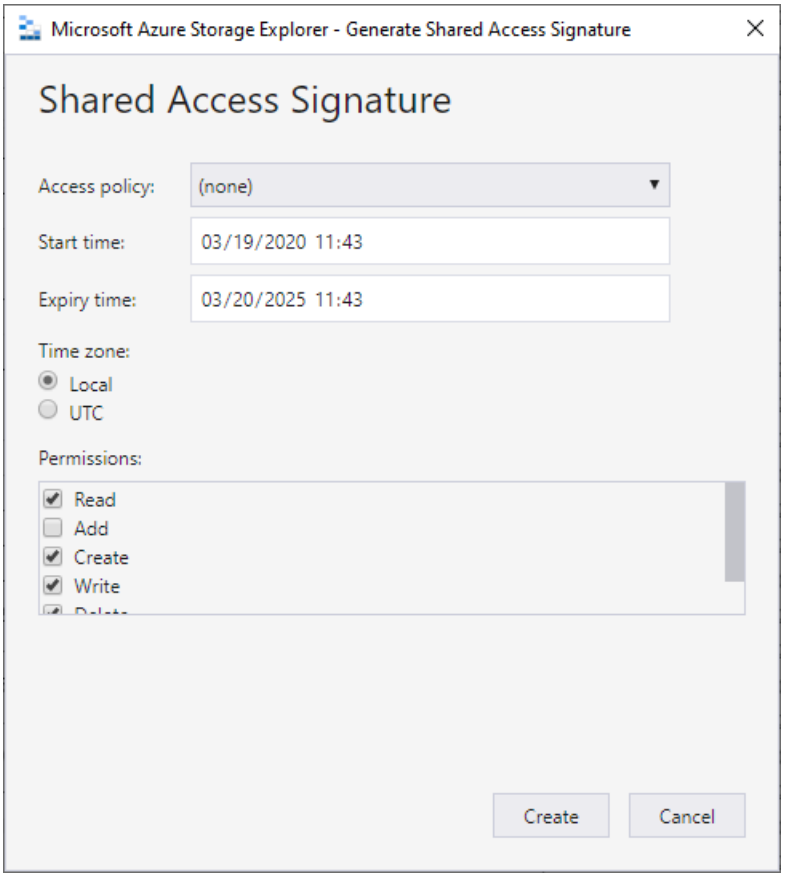
In an install type action set you can use theInstall font or Install uploaded font.

- Install font can be used when you want to fetch the .TTF file of the selected font from a local drive.
- Install uploaded font can be used to upload the font to the Content Store and install it on devices from the content repository.

Generate SAS token

We recommend you use an access policy for a SAS token.
The SAS token can be easily generated with the Microsoft Azure Storage Explorer

1. In the Microsoft Azure Storage Explorer, browse to the desired blob container.
2. Right click on the container and select on "Get Shared Access Signature...".
3. Include the following permissions for: *Read, Create, Write, Delete, List*



For the Direct Access option, you must include only the *Read* permission. Other permissions are not needed for Direct Access and could cause security issues when included.

Launch Universal Windows Platform apps with Application Workspace

Overview

We get many questions about how to launch provisioned Universal Windows Platform apps from within Application Workspace. In this article, we will use a common app: Snip & Sketch.

UWP apps are designed to work across platforms and can be installed on multiple ones including Windows client, Windows Phone, and Xbox. All UWP apps are also Windows apps, but not all Windows apps are UWP apps.

Appx is Microsoft's file format used to distribute and install these apps on the Universal Windows Platform. Appx is a .ZIP–based file containing the app's payload files plus info needed to validate, deploy, manage, and update the app.

Some Windows apps are already provisioned in Windows by default. Read Overview of apps on Windows client devices to find out how you can use Windows PowerShell to find out the name of all these provisioned apps.

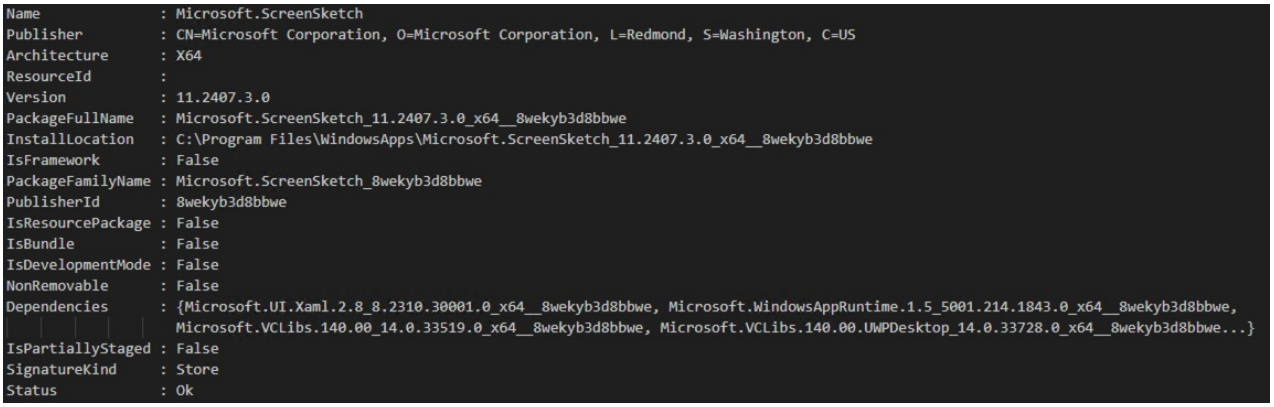
Retrieve PackageFamilyName and App ID using PowerShell

Next, we will use PowerShell to find about Snip & Sketch all the details needed to create a Custom type package in the Application Workspace.

We use the `Get-AppxPackage` cmdlet to list information about the Appx package only:

```
Get-AppxPackage | Where {$_.Name -match 'ScreenSketch'}
```

We're going to have a look at three properties here, Name, InstallLocation and PackageFamilyName:



Name – is something we need to make certain, that we are looking at the details of the correct app; the 'Snip & Sketch' app.

InstallLocation – is where we can find the `AppxManifest.xml` file which contains the info the system needs to deploy, display, or update this Windows app. Within it, we can find the ID of the app which in this case is 'App'. But we can also retrieve it by running:

```
(Get-AppxPackage | Where {$_.Name -match 'ScreenSketch'} | Get-AppxPackageManifest).package.applications.application.id
```

PackageFamilyName – is what we need to identify the app. The combination of the PackageFamility Name and ID is needed to launch Snip & Sketch successfully. The ID value can be found in the `AppxManifest.xml`.

Create an Application Workspace package

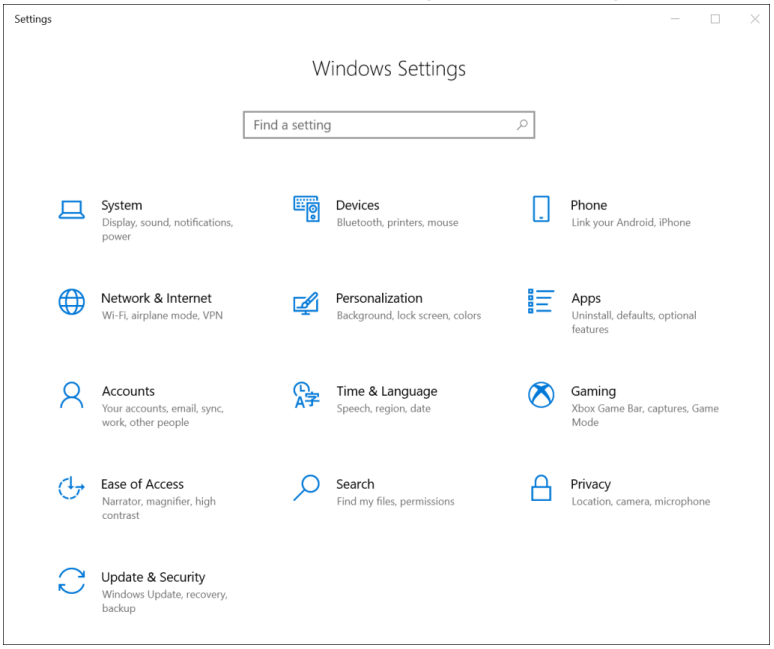
We can now bring this all together and create a package in the Application Workspace:

1. Navigate to **Manage** > **Workspace** > **Packages**.
2. Click ➕ **Create** in the tabel toolbar.
3. In the **Create package** dialog box that opens in **Type**, select *Custom package*. Click **Next**.
4. In **Overview**:
    - In the **Name** field write "ScreenSketch"
    - In the **Icon** field select a desired icon from the `C:\Program Files\WindowsApps\Microsoft.ScreenSketch_8wekyb3d8bbwe\Assets` path. We recommend using the highest available resolution.
    - Click **Next**.
5. In **Entitlements**, add the identities to which you want to assign this package.
6. Click ➕ **Create action set** in the tabel toolbar.
7. In the **Create action set** dialog box, select Type *Launch*. Click **Confirm**.
8. Click ➕ **Create action** in the action set toolbar.

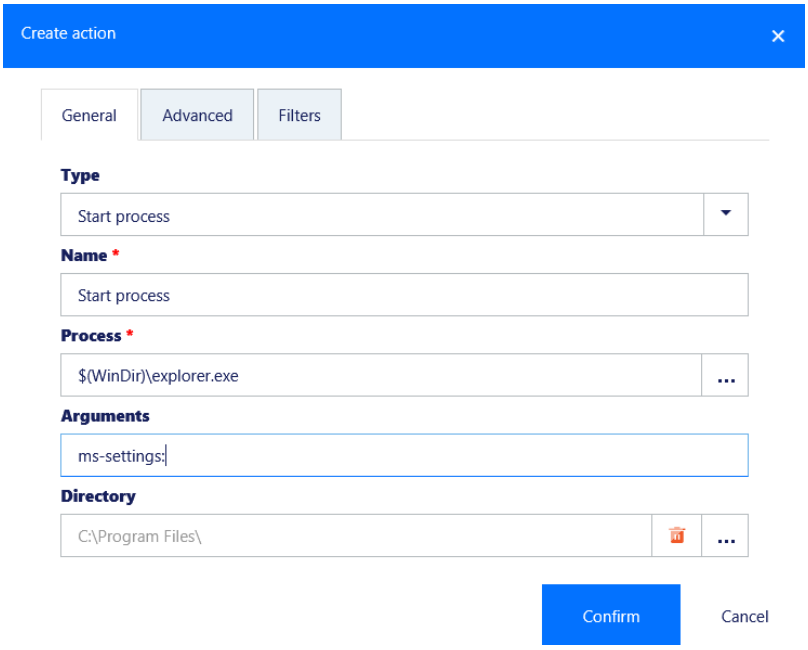
9. In the **Create action** dialog box that opens in **Type**, select *Start Windows App* and configure the following:
    - Change the Name if desired.
    - Enter the **Package Family Name**, previously discovered with the Powershell statement.
    - Enter the **Application Id**, previously discovered with the Powershell statement.

Launch Windows Settings from Application Workspace

You can start any of the Windows Settings UWP apps using explorer.exe and an URI parameter.



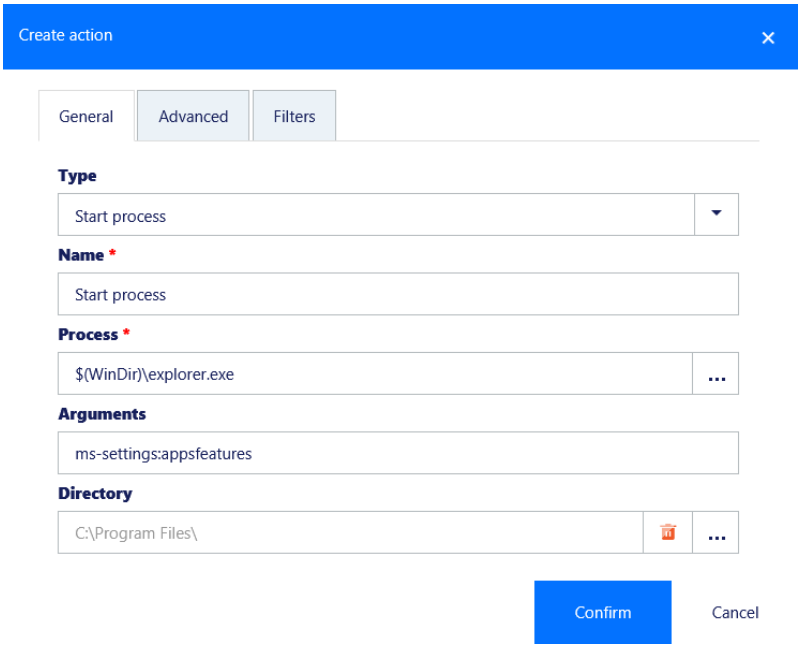
In this article we give you some examples of such settings which are started using `ms-settings:`



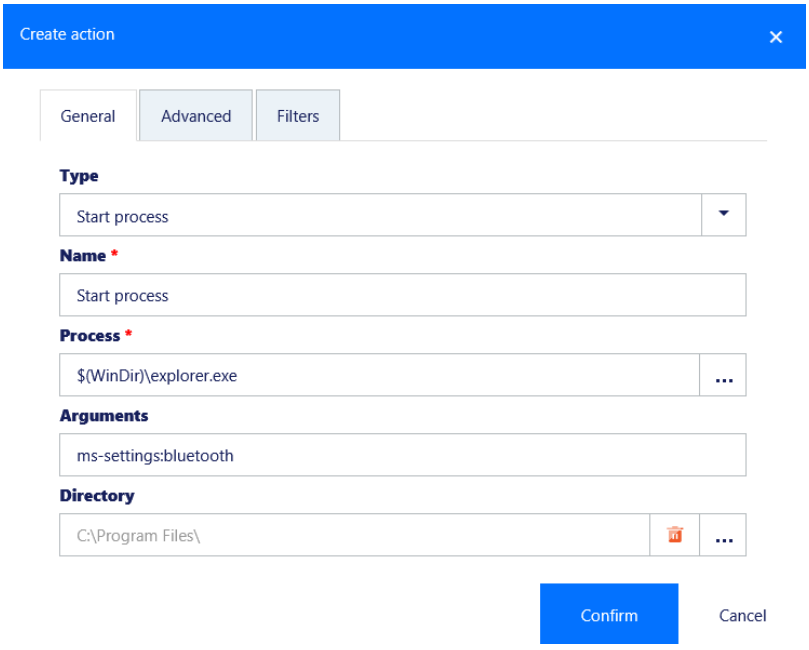
Recast

Examples

Apps & Features Settings can be started using `ms-settings:appsfeatures`



Bluetooth Settings can be started using `ms-settings:bluetooth`



Further reading

Complete list of ms-settings URIs used to open various pages of the Settings app (Microsoft documentation)

Move the Content Store to a new local location for a single server setup

1. Stop the Application Workspace Server service.

2. Move the content of `C:\ProgramData\Liquit Workspace\Server\Content` to the new location.
3. Ensure the Application Workspace Server service has read and write permissions for the new location.
4. Modify the `Server.json` file located in `C:\Program Files (x86)\Liquit Workspace\Server`
5. By default, the `Content` object in the `Server.json` file contains the `path` key with the value `Content`. Replace this value with a full path to the new location.
6. Start the Application Workspace Server service.

> **Backslashes**
> All backslashes need to be escaped when entering a full path. For example, if the content needs to be placed in `D:\Contents`, the path would be `D:\\\\Contents` within the JSON file.

Move the Content Store to a new location for a multi-server setup

1. Stop the Application Workspace Server service.
2. Configure the new storage location by changing the Storage Settings on the server and/or updating the `Content` object in the `Server.json` file for local storage.
3. Start the Application Workspace Server service.
4. Monitor the content replication from the content screen on the reconfigured server.
5. After the replication has finished, stop the other Application Workspace Server service(s).
6. Reconfigure the other Application Workspace Servers the same way as the first server. This can also be done globally.
7. Start the reconfigured Application Workspace Server service(s).

Move the Content Store to the Azure Blob service for a single server setup

1. Stop the Application Workspace Server service.
2. Make a backup of the local Content Store. By default, its location is `C:\ProgramData\Liquit Workspace\Server\Content`.
3. Remove all ".tmp" files from the Content Store.
4. Remove all ".dat" extensions of the files within the Content Store. For example, from the command line run: "rename *.dat *."
5. Upload the content files to the desired Azure Storage blob container. You can use the Microsoft Azure Storage Explorer for this, for example. The container should look like in the following image, where the Access Tier can be different:

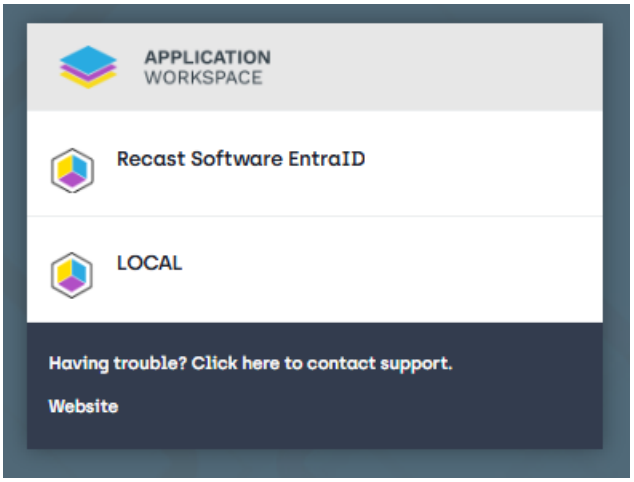
Recast

Name	Access Tier	Access Tier Last Modified	Last Modified	Blob Type	Content Type	Size	Status
000bb1a5-2e80-465a-0b2c-35e4e3fde595	Cool	1/7/2020, 5:38:06 PM	1/7/2020, 5:38:06 PM	Block Blob	application/octet-stream	29.2 MB	Active
00128c6b-cbaa-f830-0828-35ef35e38c50	Cool	2/29/2020, 6:23:51 AM	2/29/2020, 6:23:51 AM	Block Blob	application/octet-stream	10.8 MB	Active
001a41c8-0c09-9de4-51a2-35e2bd7ca4ef	Cool	1/7/2020, 4:34:25 PM	1/7/2020, 4:34:25 PM	Block Blob	application/octet-stream	1.1 MB	Active
002da3ff-598d-cf02-2e78-35ee590b3001	Cool	2/24/2020, 9:01:00 PM	2/24/2020, 9:01:00 PM	Block Blob	application/octet-stream	302.0 MB	Active
002e598c-a35d-1f15-16fe-35e9d47ec8a0	Cool	2/1/2020, 9:05:55 PM	2/1/2020, 9:05:55 PM	Block Blob	application/octet-stream	54.5 MB	Active
0043e89f-2fd1-a981-9d45-35ed2b522c63	Cool	2/18/2020, 9:01:10 PM	2/18/2020, 9:01:10 PM	Block Blob	application/octet-stream	74.0 MB	Active
005b0e0c-e2f4-45f7-5d3b-35eb023302d8	Cool	2/7/2020, 9:03:29 PM	2/7/2020, 9:03:29 PM	Block Blob	application/octet-stream	18.6 MB	Active
0061de82-ab66-8a2c-57f0-35e2c35c7a76	Cool	1/7/2020, 5:10:57 PM	1/7/2020, 5:10:57 PM	Block Blob	application/octet-stream	46.2 MB	Active
007db171-db42-2880-5bc1-35e2c529c4e3	Cool	1/7/2020, 5:23:43 PM	1/7/2020, 5:23:43 PM	Block Blob	application/octet-stream	59.3 MB	Active
00884a37-88f9-ddb4-3d08-35f19bf12073	Cool	3/12/2020, 11:26:04 AM	3/12/2020, 11:26:04 AM	Block Blob	application/octet-stream	5.3 MB	Active

6. Start the Application Workspace Server service.
7. Log in as an administrator and navigate to **Manage** > **System** > **Storage Settings**.
8. Configure the Storage Settings to use the Azure Blob service.

Pair a zone

1. In the **Zones** screen, click **Pair zone**.
2. Enter the URL of the zone to pair with and its name. Enable the **Content**, **Managed** or **Connectors** options according to your needs.
3. After clicking **Confirm**, you are prompted to log in to the zone to which you want to pair the local Application Workspace Satellite Server. Use administrator credentials that have the "Create servers" privilege.



Receive email notifications for Application Workspace Setup Store managed package updates

This article describes how to create scheduled tasks to receive e-mail notifications when updates are available for the managed application you've created through the Application Workspace Setup Store.

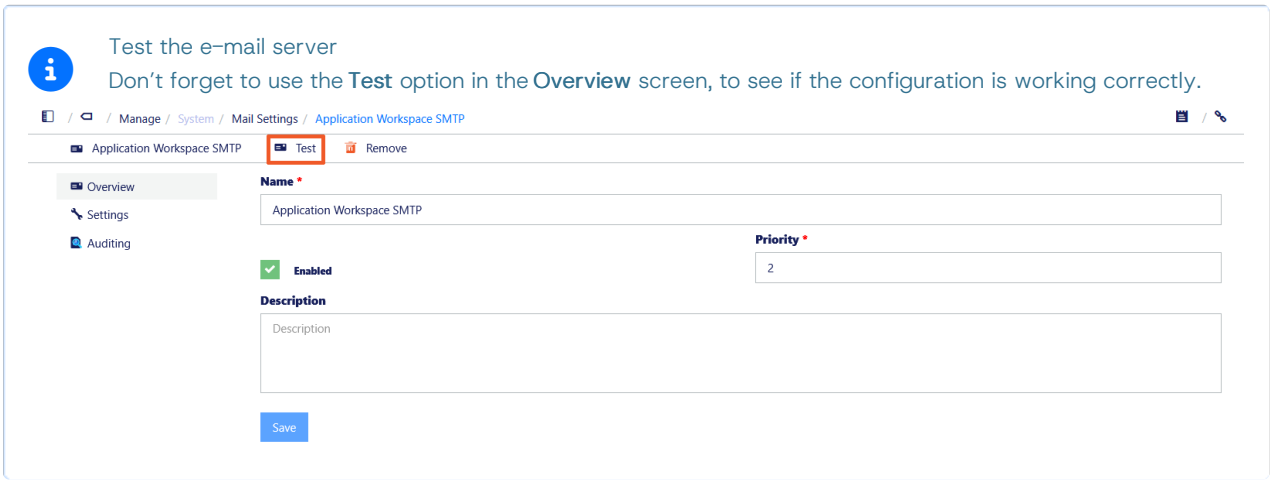
Configure the email settings

To receive email notifications from within Application Workspace, you first have to add an email server in **Manage** > **Mail Settings**:

Click **Create** in the table toolbar and configure all the necessary settings for this email server.

Recast

This email server is used by the scheduled tasks to send email messages. For more information, see Mail Settings.

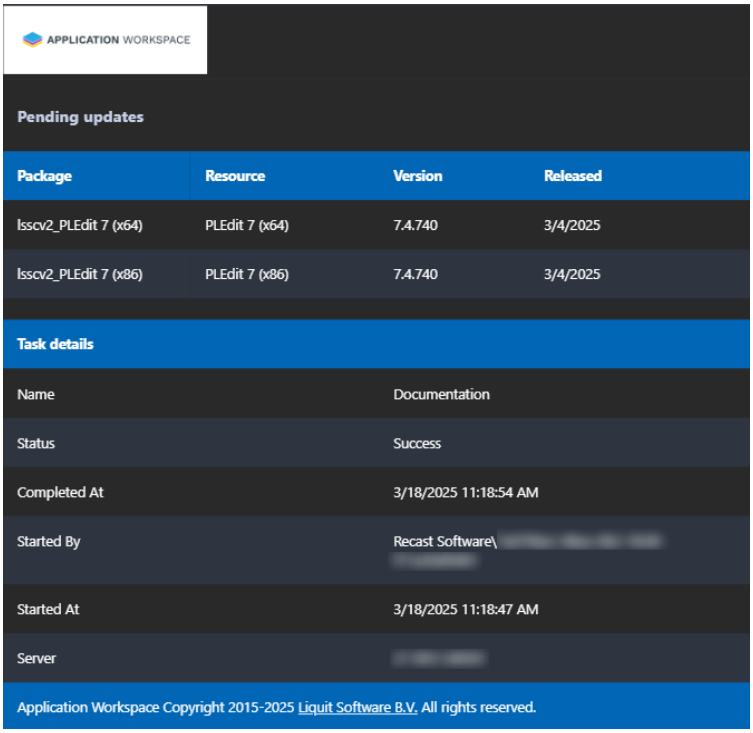


Create a scheduled task

1. Navigate to **Manage** > **Automation** > **Scheduled Tasks**.
2. Click **+ Create**.
3. In the **Create scheduled task** that opens, select the *Synchronize connector* type.
4. In **Settings**:
    - Select an existing Application Workspace Setup Store.
    - Select **Check for updates** action.
5. In **Summary**, leave the **Modify package after creation** selected and click **Finish**.
6. Navigate to **Notifications** and enable the notification and add one or more email addresses where Application Workspace should send detailed emails per managed package.
7. Navigate to **Schedule** and add a new schedule where you specify the desired frequency and time.

This is an example of how an update email notification looks like:

Recast



Set up and use remote control

Overview

The remote control functionality requires the Agent to run, regardless of the type of entitlement, device, or user.
The **Remote Control** button can be found in the following locations:

- **Manage** > **Identities** > **Users**
- **Manage** > **Identities** > **Users** > **Overview**
- **Manage** > **Identities** > **Devices**
- **Manage** > **Identities** > **Devices** > **Overview**

Before you can use the remote control, a few configurations need to be completed. This document outlines those steps in detail.

Choose a remote control software

First, select the remote control software you wish to use.

Examples:

- TeamViewer
- Bomgar (now BeyondTrust)
- TightVNC
- UltraVNC

For the following examples, we are using TeamViewer and TightVNC.

TeamViewer example

Create an Application Workspace Setup Store Managed Package for TeamViewer

TeamViewer installers (MSI files) for both 32-bit and 64-bit versions of Windows are available in the Application

Workspace Setup Store.

1. In your Application Workspace Setup Store connector, navigate to **Overview** > **Resources**, and search for TeamViewer.
2. Double-click on it or select it and then click **Create package** in the table toolbar.
3. In the **Create package** dialog box that opens, in **Overview**:
    - Enter a name for your new package or keep the default name.
    - In **Package type**, select *Managed*.
    - In **Publish**, select *Production*.
    - In **Setup details**, click **Next**.
    - In **Dependencies**, click **Next**.
    - In **Customize**, configure TeamViewer using enterprise best practice settings, otherwise click **Next**.
    - In **Entitlements**, add the user/device you want to control remotely. In the **Add entitlement** dialog box select **Stage** to *Production* and **Publish** to *Hidden*.
    - In **Summary**, uncheck the **Modify package after creation** box and click **Finish**.

Create a Remote Control Package for TeamViewer

Once the managed package is created, follow these steps to create a remote control type of package for TeamViewer:

1. Navigate to **Manage** > **Packages**.
2. Click **Add** and select **Remote control** from the package type list.
3. In **Launch** configure the following:
    - In **Process**, write `${ProgramFiles}\TeamViewer\TeamViewer.exe`. For this type of package, Application Workspace automatically creates a filter based on the value you insert in this field. Any later change to the actions does not impact the filter. So in case of any typo, for example, you have to change the filter manually.
    - Select the **Use local file icon** checkbox.
    - Click **Next**.
4. In **Overview**, enter the desired package name and click **Next**.
5. In **Entitlements**, add the user(s) who should have permission to remote control devices. In the **Add entitlement** dialog box, set **Stage** to **Production** and **Publish** to **Hidden**.
6. In **Summary**, uncheck the **Modify package after creation** box and click **Finish**.

TightVNC example

Create an Application Workspace Setup Store Managed Package for TightVNC

TightVNC installers (MSI files) for both 32-bit and 64-bit versions of Windows are available in the Application

Workspace Setup Store.

1. In your Application Workspace Setup Store connector, navigate to **Overview** > **Resources**, and search for TightVNC.
2. Double-click on it or select it and then click **Create package** in the table toolbar.
3. In the **Create package** dialog box that opens, in **Overview**:
    - Enter a name for your new package or keep the default name.
    - In **Package type**, select *Managed*.
    - In **Publish**, select *Production*.
    - In **Setup details**, click **Next**.
    - In **Dependencies**, click **Next**.
    - In **Customize**, configure TightVNC using enterprise best practice settings. The **Customization Wizard** allows you to set options such as the password for remote control access. For further details on installing TightVNC, consult the official TightVNC documentation .
    - In **Entitlements**, add the user/device you want to control remotely. In the **Add entitlement** dialog box select **Stage** to *Production* and **Publish** to *Hidden*.
    - In **Summary**, uncheck the **Modify package after creation** box and click **Finish**.

Create a Remote Control Package for TightVNC

Once the managed package is created, follow these steps to create a **Remote Package** for TightVNC:

1. Navigate to **Manage** > **Packages**.
2. Click **Create** in the table toolbar
3. In the dialog box than opens, in **Type** select **Remote control** from the package type list.
4. In **Launch**, configure the following:
    - In **Process**, write `${ProgramFiles}\TightVNC\tvnviewer.exe` . For this type of package, Application Workspace automatically creates a filter based on the value you insert in this field. Any later change to the actions does not impact the filter. So in case of any typo, for example, you must change the filter manually.
    - Select the **Use local file icon** checkbox.
    - Click **Next**.
5. In **Overview**, enter the desired package name and click **Next**.
6. In **Entitlements**, add the user(s) who should have permission to remote control devices. In the **Add entitlement** dialog box, set **Stage** to **Production** and **Publish** to **Hidden**.
7. In **Summary**, uncheck the **Modify package after creation** box and click **Finish**.

> The `admin` username is granted remote control privileges by default, but you can grant any user the same privileges. See Access Policies for more information.

Using Remote Control

Once the package is set up, navigate to the user or device you wish to remote control and follow the below instructions:

1. Click the **Remote Control** button.
2. In the the dialog box that opens:
    - In the **Package** field, select the remote control type of package you previously configured.
    - Click **Confirm** to proceed.

The remote control software will launch.

Update a managed package from Application Workspace Setup Store

There are two ways in which you can update a managed package, manually and automatically.

In this article, we take as an example Mozilla Firefox, a managed package created from the Application Workspace Setup Store. Application Workspace Setup Store is configured with he on-demand synchronization method, the only one supported.

Manual update

1. Open the Mozilla Firefox package and navigate to **Overview** > **Updates**.
2. Select the new version of Mozilla Firefox and click **Apply update** in the table toolbar.

Automatic update

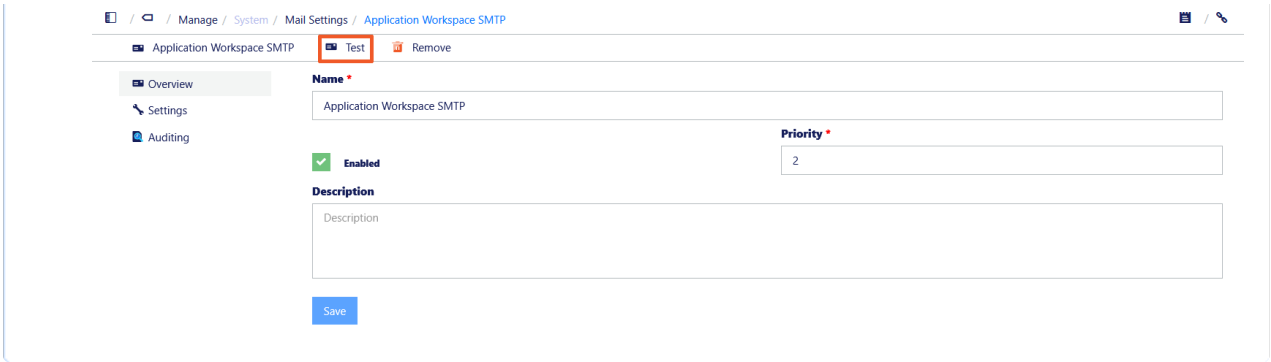
Configure the email settings

To receive e-mail notifications from within Application Workspace, you first have to add an email server in **Manage** > **Mail Settings**:

Click **Create** in the table toolbar and configure all the necessary settings for this email server.
This email server is used by the scheduled tasks to send email messages. For more information, see Mail Settings.

> Test the e-mail server
> Don't forget to use the **Test** option in the **Overview** screen, to see if the configuration is working correctly.

Recast

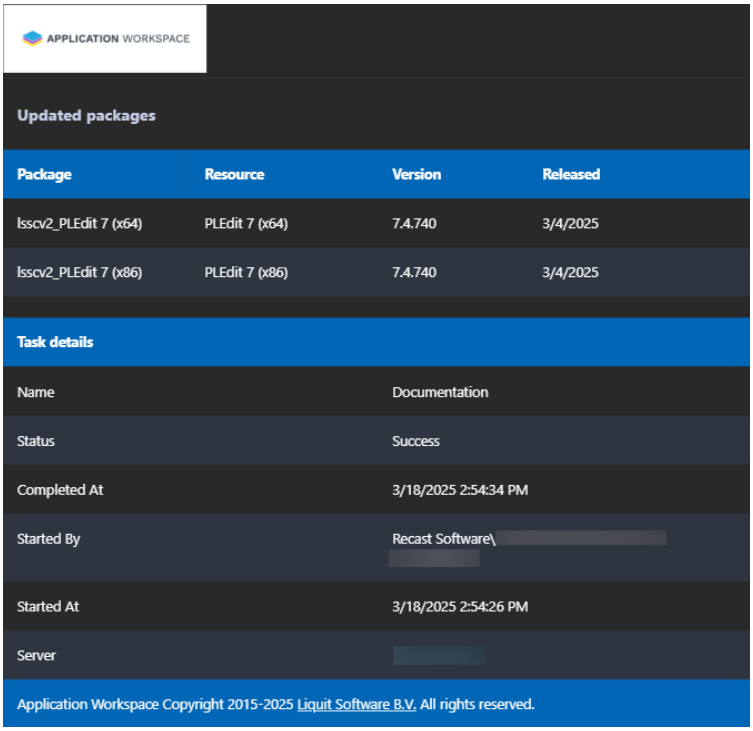


Create a scheduled task

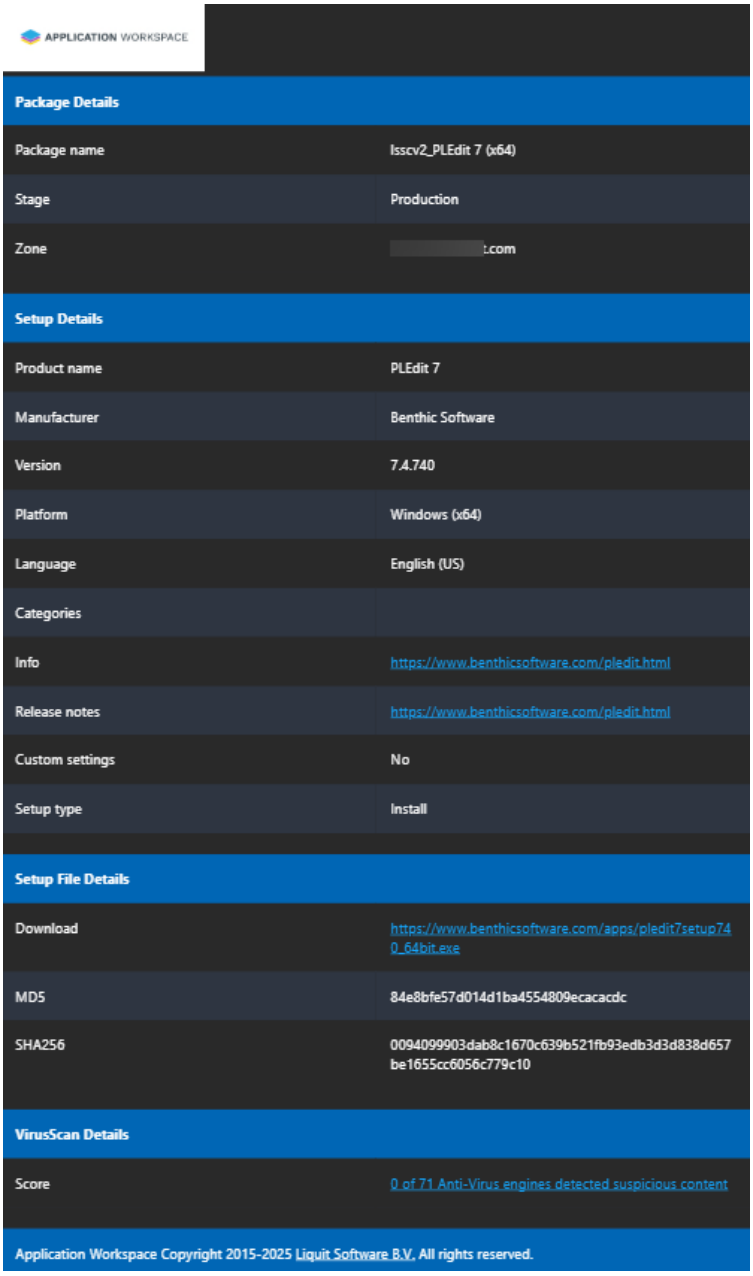
1. Navigate to **Manage** > **Automation** > **Scheduled Tasks**.
2. Click ✚ **Create**.
3. In the **Create scheduled task** that opens, select the *Synchronize connector* type.
4. In **Settings**:
    - Select an existing Application Workspace Setup Store.
    - Select **Download and apply updates** action.
    - Enable **Send detailed per package notifications**. By using this option you will receive, per managed package, a very detailed email message containing information such as release notes, CVE, CVSS information (if applicable), and the virus scan score. This information is important for risk mitigation.
    - Select to which DTAP stage you want to deploy the updated package. Note that the default option represents the stage configured in the **Releases** screen of the connector.
5. In **Summary**, leave the **Modify package after creation** selected and click **Finish**.
6. Navigate to **Notifications** and enable the notification and add one or more email addresses where Application Workspace should send detailed emails per managed package.
7. Navigate to **Schedule** and add a new schedule where you specify the desired frequency and time.

This is an example of how the email notification looks like when an automatic update is applied:

Recast


APPLICATION WORKSPACE
Updated packages
Package
Resource
Version
Released
lsscv2_PLEdit 7 (x64)
PLEdit 7 (x64)
7.4.740
3/4/2025
lsscv2_PLEdit 7 (x86)
PLEdit 7 (x86)
7.4.740
3/4/2025
Task details
Name
Documentation
Status
Success
Completed At
3/18/2025 2:54:34 PM
Started By
Recast Software\
Started At
3/18/2025 2:54:26 PM
Server
Application Workspace Copyright 2015-2025 Liquit Software B.V. All rights reserved.

Recast



APPLICATION WORKSPACE

Package Details	
Package name	lsscv2_PLEdit 7 (x64)
Stage	Production
Zone	[redacted]t.com

Setup Details	
Product name	PLEdit 7
Manufacturer	Benthic Software
Version	7.4.740
Platform	Windows (x64)
Language	English (US)
Categories	
Info	https://www.benthicsoftware.com/pledit.html
Release notes	https://www.benthicsoftware.com/pledit.html
Custom settings	No
Setup type	Install

Setup File Details	
Download	https://www.benthicsoftware.com/apps/pledit7setup740_64bit.exe
MD5	84e8bfe57d014d1ba4554809ecacacdc
SHA256	0094099903dab8c1670c639b521fb93edb3d3d838d657be1655cc6056c779c10

VirusScan Details	
Score	0 of 71 Anti-Virus engines detected suspicious content

Application Workspace Copyright 2015-2025 Liquit Software B.V. All rights reserved.

Upload your own config.xml within a Setup Store managed app

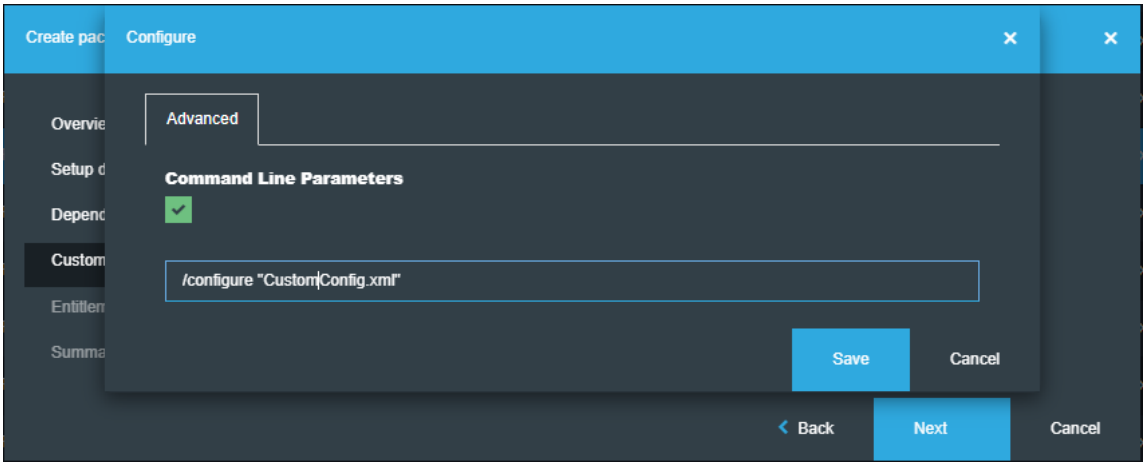
Scenario: within a Microsoft Office managed package you need a pre-install action which installs a custom config.xml on the user device. Then you can edit the install string of the clicktorun.exe process to point to that xml instead.

Currently, the only way to include your own custom configuration .xml file within a managed package is to add an extra action set that copies
the file you created with the ODT webtool at Microsoft 365 Apps admin center, to the the desired location, for example ${PackageTempDir}.

Steps

1. Open the Application Workspace Setup Store connector and navigate to **Overview** > **Resources**.
2. Select a Microsoft 365 Apps product and click **Create package**.
3. In the **Create package** dialog box that opens

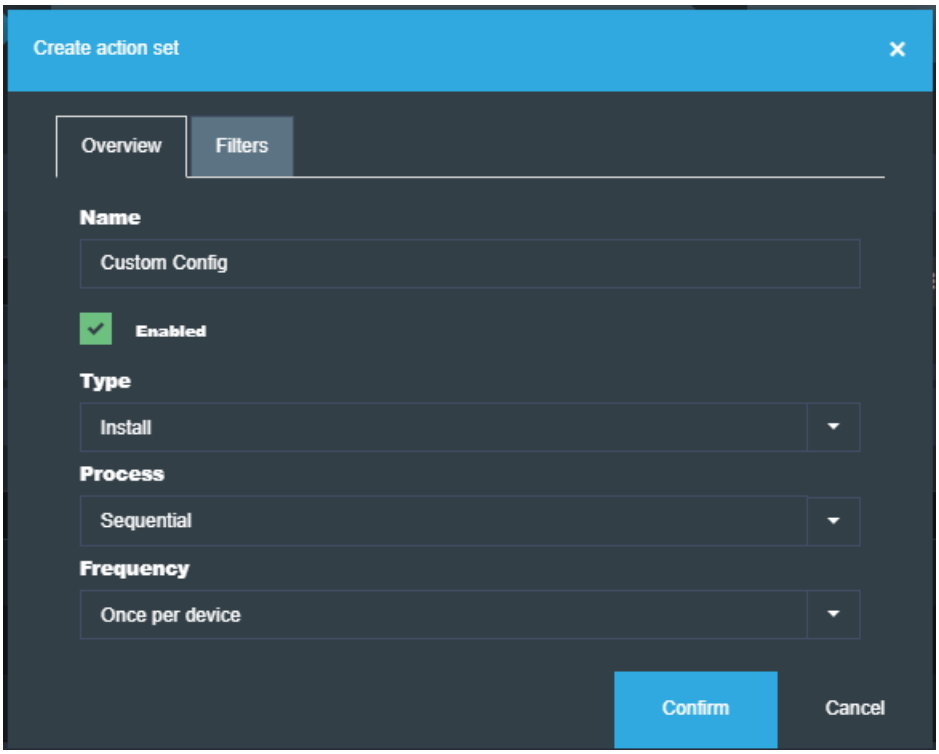
- in the **Customize** screen, modify the command line parameters to `/configure "CustomConfig.xml"` where you replace CustomConfig.xml with the name of your file. Note that when no variable is used, it uses the working directory specified in the package, which is ${PackageTempDir}



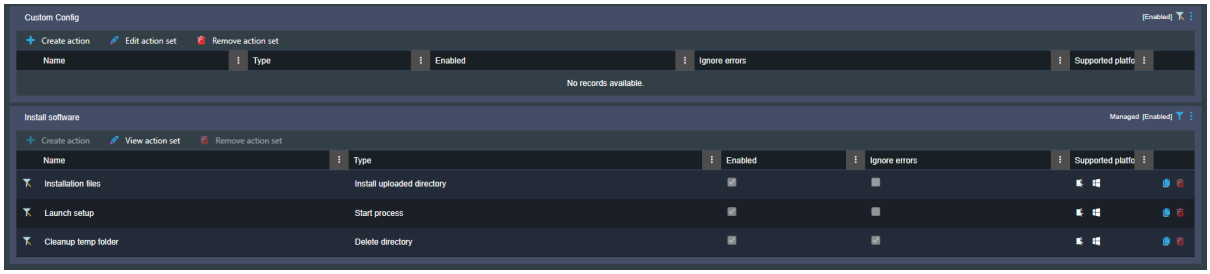
> In the **Summary** screen, leave the "Modify package after creation" option selected, so that the newly created package opens automatically after you click **Finish**.

4. Within the package, navigate to **Releases** > **Actions**.

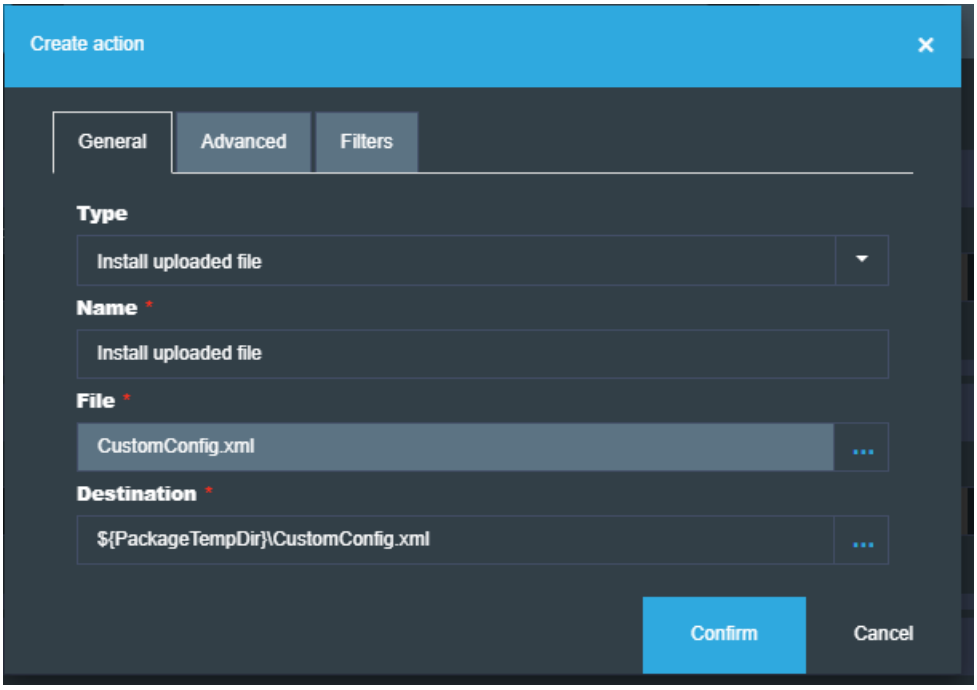
5. Create a new Install type action set named Custom Config.



6. Move the *Custom Config* action set above the *Install software* action set, by clicking on the title bar of the action set and dragging and dropping it in the new location.



7. In the table toolbar of the action set, click **+ Create action**, and an *Install uploaded file* type of action. Browse for the file you want to upload and specify the destination path where you want the file to be placed.



Note that in the case of managed packages, existing action sets cannot be edited or deleted. You can still add new action sets to the package.
To convert a package from managed to unmanaged you must by click on **Modify package** at the top of the **Overview** screen.

By becoming unmanaged, future Application Workspace Setup Store updates will not be applied to the app.

Use a license key as a dynamic variable

Creating a dynamic variable for a license key can help you save time instead of wasting it searching for a key each time you install an application. Then you use that variable inside an Install action as a placeholder for a dynamically value obtained by Application Workspace.
In the example below we will create a dynamic variable for the license key of FolderSizes, a disk space analysis software.

1. Go to **Manage** > **Workspace** > **Licenses**.

2. Create a new license with the following parameters:

    - **Name** *FolderSizes License*

- **License count** *2*
- **License cost** *20*
- **License key** *123-123-123-123*

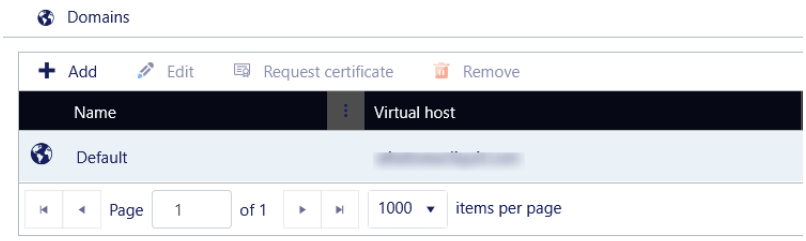
3. Go to **Manage** > **Workspace** > **Variables**.
4. Create a new variable with the following parameters:
   - **Type** *Dynamic*
   - **Name** *FolderSizes_License*
   - **Entity type** *License*
   - **Entity** select the FolderSizes License created at step 2
   - **Property** *License key*
5. Go to **Manage** > **Workspace** > **Packages**.
6. Create a new package with the following parameters:
   - **Type** *Custom package*
   - **Name** *File Size Reporting*
     Leave the **Modify package after creation** selected. Click **Finish**.
7. In the detailed view screen that opens, go to **Releases** > **Actions**.
8. Click ✚ **Create action set** in the screen toolbar. Name the action set *Install* and select the **Type** *Install*.
9. Inside the action set click ✚ **Create action** in the table toolbar. Insert the following parameters:
   - **Type** *Start process*
   - **Name** *Start fs9-setup.exe*
   - **Process** select the fs9-setup.exe file
   - **Arguments** */qn LICENSECODE=$(FolderSizesLicense)*

The installer is run silently and it gets the license key from your variable. Now you only have to think how you want to make the app available to your users and create actions accordingly.

Use your own company domain name as virtual host

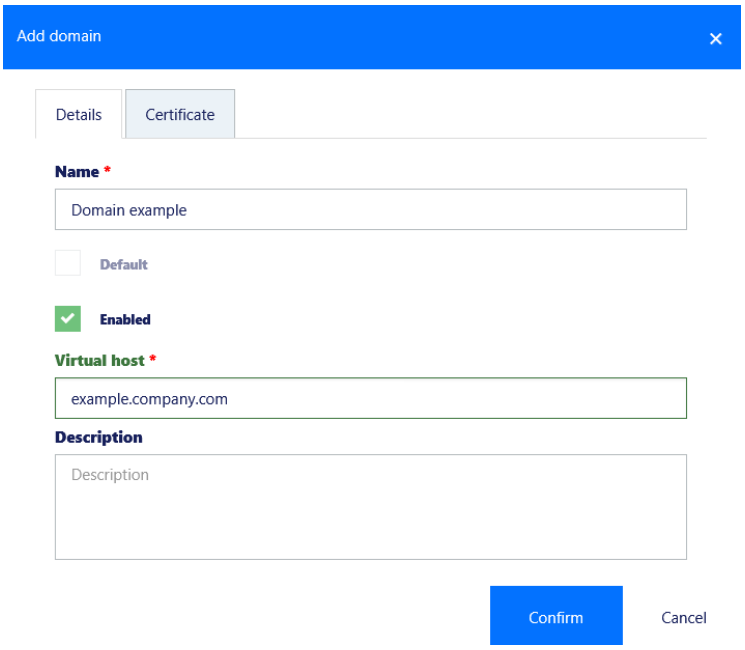
Application Workspace delivers a standard domain (virtual host) to its partners. Some partners use this default virtual hostname to identify and access the specific Application Workspace Systems or services. Most of the partners want to use their domain. In this article, we describe how to create your virtual host name inside the domain section of Application Workspace.

1. Navigate to **Manage** > **System** > **Domains**. Here you will find the default virtual hostname.

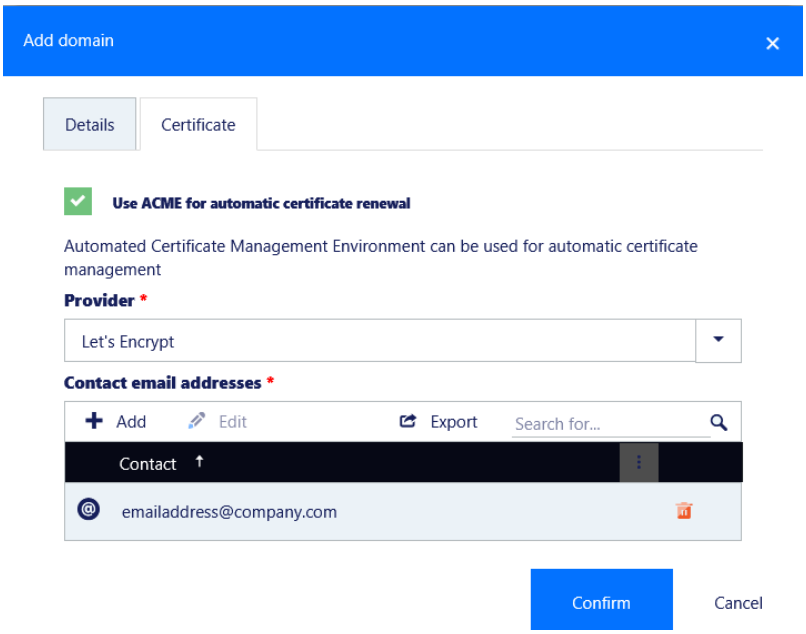


2. Click on ✚ **Add**.

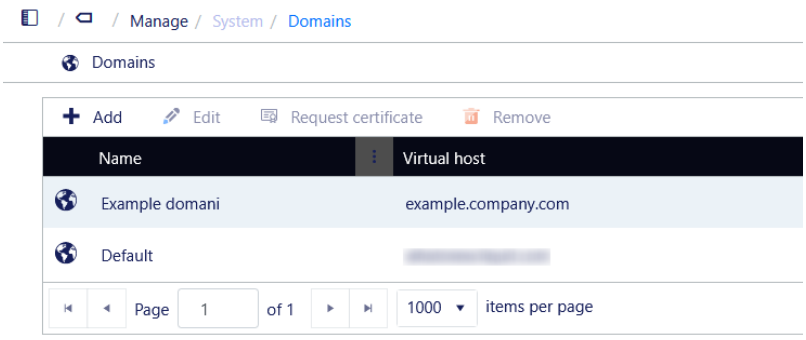
3. In the **Details** tab fill in the needed information.



4. In the **Certificate** tab upload your company certificate or use the ACME service.



5. Click **Confirm**.



6. Create a CNAME (canonical name) DNS record from `example.company.com` to `beta.liquit.cloud`.

7. Test if you can resolve the newly added domain within your browser or terminal (cmd: NSLOOKUP example.company.com). It should resolve `beta.recastsoftware.com`.

Important information

- Please note that you must maintain a certificate (with ACME or a company certificate).
- You can never create another `<example>.recastsoftware.cloud` domain.