liquidware

ProfileUnity™ with FlexApp™ Technology

FlexApp One™ Applications

Release 6.8.x

February 28, 2024

This guide has been authored by experts at Liquidware in order to provide information and guidance concerning the use of FlexApp One™ applications.

Liquidware Labs, Inc.

3600 Mansell Road
Suite 2000
Alpharetta, Georgia 30022
U.S.A.

Phone: 678-397-0450

Web: www.liquidware.com

Table of Contents

What's New for FlexApp One?

Version 6.8.6.8805 R1- Released February 28, 2024

What's New

- Enhanced Security with OAuth Integration now supports multiuser operating systems.
    - We are excited to introduce OAuth integration with Microsoft Entra ID (formerly Azure AD), bringing an additional layer of security to your FlexApp One experience. Before activating a FlexApp One package, users are required to authenticate, embracing a Zero-Trust approach for all your Win32 applications. This integration ensures that only authorized personnel have access to your applications, aligning with the industry's best practices for cybersecurity.

Component Versions

Component	6.8.6 R1
VirtFS	6.8.6.8805
FPC	6.8.6.8805
Bundler	6.8.6.8805
Engine	6.8.6.8805
Installer	6.8.6.8805

Version 6.8.6.8711 - Released November 9, 2023

What's New

- Enhanced Security with OAuth Integration:
    - We are excited to introduce OAuth integration with Microsoft Entra ID (formerly Azure AD), bringing an additional layer of security to your FlexApp One experience. Before activating a FlexApp One package, users are required to authenticate, embracing a Zero-Trust approach for all your Win32 applications. This integration ensures that only authorized personnel have access to your applications, aligning with the industry's best practices for cybersecurity.
- Session Isolation for Multi-User Support:
    - FlexApp One now offers session isolation, a pivotal feature for environments with multiple users sharing the same system. This enhancement allows different users to access distinct

applications simultaneously, fostering a collaborative and efficient workspace. Moreover, this feature extends to on-boot applications, ensuring they are only activated when the user has the necessary entitlements. This level of control and customization ensures a tailored user experience.

- Superior Performance with Memory Caching:
  - We have leveraged memory caching to significantly boost the performance of your FlexApp One packaged applications. This optimization ensures quicker application launches and smoother operation, enhancing productivity and user satisfaction. Furthermore, administrators have the flexibility to adjust settings to optimize application performance based on specific needs.

- Support for Larger Applications:
  - FlexApp One now proudly supports larger application packages, exceeding 4GB in size. For applications that expand beyond this size post-compression, FlexApp One intelligently creates a separate executable (.exe) and a FA1 file containing all necessary data. This development addresses a critical Windows limitation, ensuring seamless operation and compatibility regardless of application size.

- Simplified License Management:
  - FlexApp One now supports machine-based FlexApp_One.lic files. This means that subscription customers don't need to re-bundle, or "migrate", existing package EXEs and can instead deploy their new FlexApp_One.lic file into C:\Program Files\ProfileUnity\FlexApp\ContainerService\x64\Licenses on endpoint machines. In addition, customers utilizing ProfileUnity Management Console and Client Tools for FlexApp One deployments will benefit from the license check being deferred to the existing ProfileUnity License Server implementation, bypassing the embedded license in the package EXEs.

Issues Resolved

- Fixed an issue where --Remove command was not removing the FlexApp One package in certain circumstances.
- Fixed an issue where --Stop command was not terminating all respective FlexApp One PIDs.
- Fixed an issue where FlexApp One PIDs created with --Sync command do not terminate after successful sync completion.
- Fixed an issue where possibility of profile corruption with FlexApp One packages that contain SYSTEM environment variables.
- Fixed an issue where FlexApp One packages occasionally fail to activate/playback when SentinelOne is installed and monitoring.

- Fixed an issue where CTL shortcuts were not created that exist in a subdirectory of the Start Menu 'Programs' folder.
- Fixed an issue where your unable to cleanup (--stop --clean) a FlexApp One package where the associated PID has already been terminated.
- Fixed an issue where FlexApp One packages activated by SYSTEM with --Ctl and/or -AddToStart do not create/remove shortcuts in respective PUBLIC and PROGRAMDATA locations.
- Fixed an issue where FlexApp One packages activated by SYSTEM with --Reg and/or --Startup do not create registry entries in respective HKLM keys.
- Fixed an issue where a broken logic flow of license detection when leveraging ProfileUnity to manage FlexApp One.
- Fixed an issue where the lwl_proc_info system driver is not removed during uninstallation.

Known Limitations:

- OAuth integration is limited to single-user desktops/sessions with package activation at logon. Mutli-user environments (RDSH, XenDesktop, etc.) and/or package activation at boot are not supported.

Component Versions

Component	6.8.6
VirtFS	6.8.6.8711
FPC	6.8.6.8711
Bundler	6.8.6.8711
Engine	6.8.6.8711
Installer	6.8.6.8711

Version 6.8.5.8507 - Released April 19, 2023

What's New

- Updated build number to stay in sync with core FlexApp and ProfileUnity.

Component Versions

Component	6.8.5 R1
VirtFS	6.8.5.8507
FPC	6.8.5.8507
Bundler	6.8.5.8507
Engine	6.8.5.8507
Installer	6.8.5.8507

Version 6.8.5.8490 - Released April 5, 2023

Issues Resolved

- Fixed an issue where packages could not be stopped if their PID had ended.
- Fixed an issue where FlexApp One applications failed to execute when run in the same millisecond.
- Fixed an issue where Click-To-Layer shortcuts failed to be created when they were in a Start Menu Programs sub-folder.

Component Versions

Component	6.8.5 R1
VirtFS	6.8.5.8490
FPC	6.8.5.8490
Bundler	6.8.5.8490
Engine	6.8.5.8490
Installer	6.8.5.8490

Version 6.8.5.8454 - Released March 15, 2023

What's New

- Packages executed as SYSTEM automatically hide the tray app since system does not have a shell to load a tray app.

- When migrating a FlexApp One engine from one version of FlexApp One to another the existing package icon is migrated as well.
- FlexApp One logs now include the OS version, UAC status, elevation status and Version info with every log header.
- The FlexApp One command line argument --clean will now cleanup write VHDX files.
- FlexApp One now supports pre/post scripts. If you create and import a script called system_ myscript.cmd or system_myscript.PS1 into the FlexApp Packaging Console, the script will run as system elevated. Any scripts that start with "system_" will run as system elevated. Any other script name will run as the user without elevation.
- The FlexApp One Installer and Engine now have optional reboot flags.
    - engine: --upgrade --reboot, if anything was upgraded, it will reboot.
    - installer: --install --reboot, if anything was upgraded, it will reboot. A fresh install will not reboot.
- FlexApp One now supports physical registry values being written to registry keys that are virtualized by FlexApp.
- The FlexApp One Bundler option -Migrate will now perform a test for a valid package file, license file, flexapp.zip and engine.exe before execution to ensure success.
- FlexApp One Engine --startup --index 999 will NOT automatically add --CTL to the HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run key.
    - Using --startup --CTL --index 999 will still add CTL to the Run key.
- When FlexApp One is managed by ProfileUnity, no license file is needed. The license logic is handled by the ProfileUnity Client and the ProfileUnity Management Console.
- The FlexApp One standalone license checks at the packagename.exe level can be overridden by placing your FlexApp_One.lic in the following path "C:\Program Files\ProfileUnity\FlexApp\ContainerService*\Licenses".

Issues Resolved

- Fixed an issue where an expired perpetual license allowed applications to still be bundled.
- Fixed an issue where packages did not install or upgrade the engine.
- Fixed an issue where users were not able to modify files contained within a package activated as SYSTEM despite having permissions.
- Fixed an issue where addtostart and CTL shortcuts created as system using --system were not usable by regular users.

- Fixed an issue where the FlexApp One tray apps were mounted as system using -system. The FlexApp One tray can only stop one app.
- Fixed an issue where the container service still tried to load lwl_proc_info driver when it was unnecessary.
- Fixed an issue where packages started and stopped as SYSTEM failed to remove CTL shortcuts.
- Fixed an issue where running installer.exe --install created a literal folder called $TEMP.
- Fixed an issue where If you ran the same packagename.exe more than once then you ended up with the same number of tray icons showing.
- Fixed an issue where packages failed to parse the icon during execution and failed back to using the default Liquidware droplet icon.
- Fixed an issue where CTL shortcuts failed when the Desktop shell folder was redirected to a UNC path.
- Fixed an issue where call to home was not disabled when --calltohome false was specified.
- Fixed an issue where the Engine Start In shortcut field was ignored for CTL and index 0.
- Fixed an issue where long fully qualified mount point paths encountered the path length limit of 260 characters.
- Fixed an issue where micro isolation was occurring when a single FlexApp was played back.
- Fixed an issue where Flexapp One did not always start at startup.
- Fixed an issue where packagename.exe --uninstall and installer.exe --uninstall did not work.
- Fixed an issue where packagename.exe --uninstall did not uninstall while played back. Fixed an issue where Packages that fail to mount continue running.
- Fixed an issue where FlexApp One applications used ProgramData and the users public profile when running as SYSTEM account for shortcuts.
- Fixed an issue where command line arguments were case-sensitive.
- Fixed an issue where container service files failed to be removed during -uninstall.

Version 6.8.5.63276 - Released August 18, 2021

What's New

The first version of FlexApp One is released to market. FlexApp One makes offline distribution and consumption of applications possible.

Known Issues/Limitations

- Installing or uninstalling applications into the operating system that could be intermingled with active FlexApp One applications could present challenges. Note that FlexApp One applications might have to be stopped in order to install or uninstall other software on the end user device. Support for installing or uninstalling applications into the operating system with active FlexApp One applications is best effort on well-managed devices.
- FlexApp One applications are incompatible with the simultaneous use of ProfileUnity on an end user device. To get around this limitation, read the following knowledge base article, How to Use FlexApp One and ProfileUnity Version 6.8.4.

FlexApp Technology with FlexApp One

ProfileUnity's FlexApp technology provides a way to consume applications without installing them within each workspace. FlexApp allows IT administrators to assign applications to groups of end users. By making use of virtual disks for application storage, FlexApp enables organizations to create a single base image for all users and layer in extra applications as needed, thus reducing storage requirements to maintain multiple base images for different departments or segments of users. FlexApp delivers applications that do not install into the workspace, thereby keeping the workspaces clean. Each FlexApp will be the same when deployed onto varied machines. Because no applications are installed with FlexApp, there is typically no need for complex uninstallation scripts to forcefully remove files, settings, or services.

Up until the release of FlexApp One, the ProfileUnity Management Console was required to run FlexApp packages and the user's workspace needed to be well connected to the corporate domain. With FlexApp One, IT admins now have an easy way to consume user applications on a variety of endpoint or desktop types. The FlexApp One bundler takes an existing FlexApp package and creates an executable that can be distributed in a variety of ways, including via your Microsoft Corporate Portal, SCCM, Microsoft Intune, a flash drive, cloud storage like Dropbox, or OneDrive. ProfileUnity 6.8.5, and newer, also includes a FlexApp One Module that can be used to easily sync packages to client machines, eliminating the need for custom scripting where desired.

Using FlexApp One applications allows remote desktop or laptop users to run applications when disconnected from the corporate network. IT admins can deliver FlexApp One applications to end users on clean, well-managed machines ahead of time. Then end users can run the applications locally when needed.

FlexApp One applications run on demand and act as if they are fully installed on the local machine. By default, when the workspace is restarted, the application is dropped and the machine remains as if the application had never been installed, leaving a clean base image or laptop.

IT admins can use both FlexApp One applications and traditional FlexApp packages for application delivery. FlexApp One applications fulfill the basic need to quickly distribute applications to end users. For advanced use cases, FlexApp packages can rely on the power of ProfileUnity for additional delivery options including cloud storage options, context-aware filters, privilege elevation, application restrictions, and registry fixups.

FlexApp Benefits

The following list highlights the core benefits of using FlexApp for application delivery:

- No file system tattooing (no Windows rot)
- No installation variances
- No complex uninstallations
- One-to-many centrally managed applications
- A simplified application update process
- A drastic reduction in image proliferation and management

Users of FlexApp One enjoy the following additional benefits:

- Support for offline desktop & laptops
- Support for remote workers
- Support for cloud desktops
- A portable executable that makes distribution simple
- Seamless support for Microsoft Intune
- Zero-Trust model enforces per-application authorization via Microsoft Entra ID (formerly Azure AD)
- An easy way to try FlexApp

FlexApp One Software Requirements

The software download for FlexApp One can be found within the ProfileUnity Document Repository under the *Tools & Resources* section. This ZIP file contains the following:

- **FlexApp One Bundler**–Bundles traditional FlexApp packages into distributable executables.
- **FlexApp One Documentation**
- **FlexApp Packaging Console Installer**–Used to create a traditional FlexApp packages as well as FlexApp One packages. For more information, refer to the *FlexApp Packaging Console Guide*. If the FlexApp Packaging Console will be used without being connected to the ProfileUnity Management Console, you need to create your FlexApp packages offline. Refer to the knowledge base article entitled "How to use the FlexApp Packaging Console in Offline Packaging Mode" for more information on logging in to the FlexApp Packaging Console in offline mode.
- **FlexApp Run-time Engine Installer**–Installs prerequisite software on end user device to run FlexApp One Applications. Refer to the Prerequisites to Run FlexApp One Applications section below.

FlexApp One Bundler

The FlexApp One bundler requires the following for operation:

Component	Requirements
OS Platforms Supported	Windows 10/11 or Windows Server 2016/2019/2022. Only 64-bit versions where applicable are supported. Both physical and virtual instances are also supported.
CPU	2 CPUs minimum
Memory	4 GB minimum
Storage Space	Enough storage to hold the resulting FlexApp One executable. The resulting FlexApp One application will be approximately 25-50% smaller than the initial FlexApp package.

FlexApp One Applications

The FlexApp One engine is required to run FlexApp One Applications on end user devices. The FlexApp One engine requires the following:

Component	Requirements
OS Platforms Supported	Windows 10/11 or Windows Server 2016/2019/2022. Only 64-bit versions where applicable are supported. Both physical and virtual instances are also supported.
CPU	2 CPUs minimum, 4 CPUs preferred
Memory	4 GB minimum, 6 GB preferred
Storage Space	Enough storage to hold each used application, SSD or better preferred

Prerequisites to Run FlexApp One Applications

To run FlexApp One applications, the FlexApp run-time engine must be installed on the end user's machine. The run-time engine can be included in the base image or installed one time from a FlexApp One application. The FlexApp One bundler will include the necessary files in the resulting EXE.

To install the run-time engine from a FlexApp One application:
(This requires privilege elevation.)

- Run any FlexApp One EXE to install the FlexApp service and engine.

  OR

- Run the Installer.exe using the `--install` command line argument. Note that all command line arguments use two hyphens.

  ```
  Installer.exe --install
  ```

Well-Managed Devices

FlexApp One is designed to play back applications in a user's environment from a self-contained executable. These applications run as if they are natively installed in the user's Windows environment without modifying the base image. Windows system files and the Registry remain unchanged, keeping the base image clean, unlike what would happen when a traditional application is installed or uninstalled. Having a clean, well-managed device provides a repeatable, consistent playback for each FlexApp One application. However, in order to maintain that experience, these end user devices need to start out as clean, well-managed devices. Playback of FlexApp One executables on devices that have been modified with the addition of other software might give unexpected results. The following are recommended:

- End-user devices should have a base image installed without additional applications.
- End-users should not be allowed to install non-managed software.
- Packaging and bundling of software should follow packaging best practices and occur on virtual machines that can be rolled back to a clean state after each run.

Licensing Considerations

FlexApp One is available to customers who have purchased ProfileUnity with FlexApp or FlexApp Only licenses. To request your FlexApp One license, email your Liquidware Sales Representative or contact us. Customers current on support can continue to create FlexApp One applications from FlexApp packages. When support expires, the FlexApp One bundler can only be used on versions of the bundler that were released before the support expiration date. Existing FlexApp One applications will stop working when evaluation trials and subscription contracts end, but will continue to run for perpetual software customers.

About FlexApp One Applications

Naming

Make the final FlexApp One application a simple static name. The user's registry will be instructed to start an application with a specific path (including filename) when you use --startup or **'Activate at Login'**. The user's system tray will show the original FlexApp package given name before it was bundled as a FlexApp One application. This can help with version identification.

Storing

FlexApp One applications should reside somewhere permanently since each EXE contains the application read-only data bits. Do not delete the FlexApp One EXE after you launch it. Put your FlexApp One EXE on the fastest storage. It is recommended you store your EXE on local storage which allows for faster and more reliable operation. If your EXE resides on a remote network location, you may be subject to SMB file handle reconnect issues should your connection drop for any reason.

Cloud Syncing

The ProfileUnity Console version 6.8.5 and newer makes deploying and syncing FlexApp One applications easy with no scripting required to maintain a local cache of packages or to update packages. You can copy your FlexApp One EXE's to your cloud storage manually or use the FlexApp Packaging Console to capture and upload your FlexApp One package and EXE to cloud storage, directly.

Alternatively, an example process on an end-user machine for cloud syncing updated packages might look like the following.

1. Copy the updated application to cloud storage using a different name.
2. Rename the current, in-use EXE to `*.old`.
3. Rename the new updated application to the simple name of the now `.old` EXE.

These operations can be staggered to allow endpoints to sync down. When the user logs out or reboots, any binaries in use will close and the renames can occur.

You can also use the --replace argument to replace the application if space is a concern, but the client might have to wait for the binary to completely sync down prior to seeing changes to a large application. Rebooting can ensure processes are closed for cloud storage to process changes.

Large Applications

To overcome a limitation present when using a large FlexApp One executable (4GB or more), rather than creating a single executable, the package is broken into two components: a smaller stub file (.exe) and a larger backing file (.fa1). Both files are automatically created during FlexApp One bundling when the FlexApp One exceeds the 4GB limit. When created, these components are identically named, except for the file extension. If the package name needs to change, both files are required to be renamed. Both files are also required for the package to function - e.g. the stub executable will not launch without the respective backing file present.

Sample Use Case

For this use case, we have an organization with laptop users who are frequently offline or working outside of the corporate network. These users need to be able to run applications no matter where they are or whether they are online or offline. FlexApp One allows us to create modular applications that can be delivered using your preferred software distribution system. Successful deployment of FlexApp One Applications depends on adherence to application packaging best practices. End user devices should be clean machines with a base image loaded.

FlexApp One Workflow

Here is the main workflow using FlexApp One:

1. Create a traditional FlexApp package. This automatically creates a new FlexApp One EXE, as well.
2. Deliver the bundled FlexApp One executable on your favorite software distribution system.
3. End users can double-click the shortcuts provided in the FlexApp package (preferred) when deployed using the `--CTL` option, or by double-clicking the FlexApp One application executable, directly.

Creating and Importing FlexApp Packages

Creating and importing FlexApp packages is done with the FlexApp Packaging Console. The FlexApp Packaging Console is well integrated with ProfileUnity for optimal application delivery. The FlexApp Packaging Console is available for download from both the ProfileUnity Management Console and from the FlexApp One Applications documents page.

When run with the ProfileUnity Console, the FlexApp Packaging Console will authenticate with the ProfileUnity Console Service. If you will not be running a ProfileUnity Console, you will need to create your FlexApp packages in “Offline Packaging Mode”. Refer to the knowledge base article entitled “How to use the FlexApp Packaging Console in Offline Packaging Mode“ for more information.

Please refer to the "Creating a FlexApp Package from an Application Installer" section of the *FlexApp Packaging Console Guide* for help creating your first FlexApp package or the "Importing FlexApp Packages" section for help importing your existing packages. Additionally, review these best practices on how to optimize the FlexApp Packaging Console operating system.

During capture or any time after the FlexApp packages have been created, or during import, they can be bundled into FlexApp One application executables. NOTE – While importing existing packages will help you more easily maintain your FlexApp package inventory, it is not a requirement for bulk-creating FlexApp One applications from existing packages. Just use the Create FlexApp One function in the red Create button drop-down menu in the FPC instead of Import FlexApp.

Creating FlexApp One Applications from Existing FlexApp Packages without Importing

FlexApp One applications start out as traditional FlexApp packages which are then bundled with the FlexApp run-time engine to become distributable executable files. To bundle existing FlexApp packages into FlexApp One applications, without importing them into the inventory, complete the following steps:

1. Before beginning, ensure that you have copied your **FlexApp_One.lic** file into the **C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Console\FlexApp One Bundler** folder on your **FlexApp Packaging Console** machine.
2. Open the **FlexApp Packaging Console** and login or select **Offline Packaging Mode**.
3. Click the **Create** dropdown menu and select **Create FlexApp One**.
4. Click the **magnifying glass button** to browse for the **folder containing the FlexApp Packages** you want to create FlexApp One EXE's from, leaving the **Include Subfolders** checked-on, and click **OK**.
5. Review the list of FlexApp Packages from which FlexApp One EXE's will be created and click **Create FlexApp One**.
6. Once the process is complete, click **OK** and you can then find your FlexApp One EXE's in each of the original FlexApp Package's folder.

Sample Bundler Command Line Usage

Alternatively, you can run the `Bundle.exe` with command line options and specify the FlexApp application folder to bundle into an EXE.

```
bundler.exe bundle -p "\\Path\To\The\FlexApp\PackageFolder" -f "X:\FA1Bundler\Prod\flexapp.zip" -o "\\Path\To\The\FlexAppOne\OutputFolder" -t "X:\FA1Bundler\Prod\engine.exe" " --licfile ".\flexapp_one.lic" --accepteula --seticon
```

FlexApp One Bundler Command Line Options

This section covers two sets of command line arguments; one for bundling and other other for migrating.

Command Line Arguments when Bundling

The following table shows the command line arguments that are available when bundling existing FlexApp packages into FlexApp One applications.

These arguments are for the `Bundler.exe bundle` command.

Option	Description
-p <pack-agefolder>	Specifies the folder for the FlexApp package that should be bundled into a FlexApp One. (*Required*)
-f <flexapp.zip>	Specifies the flexapp.zip to be included in the FlexApp One. (*Required*)
-t <engine.exe>	Specifies the engine.exe to be included in the FlexApp One. (*Required*)
-o <outputfolder>	Specifies the folder in which the FlexApp One application EXE will be created (*Required*).
--licfile <path\lic-file,lic>	Specifies the location to the **FlexApp_One.lic** file. **Default**: `.\` (Current working directory).
--seticon	>Will set the output file to the package default icon.
--i <path\file.ico>	Allows you to specify the icon file that will be used when setting the icon for the output file. Requires `--seticon`
--accepteula	Accept the EULA. (*Required*)
--nocallhome	Disables usage data collection.
--plainconsole	Use alternate console text output format.
-e <yyyy-mm-dd>	Causes FlexApp One application EXE to expire on the specified date so that it can no longer be executed.
--certfile <cert.pfx>	Digitally sign the FlexApp One with the specified PFX certificate. Certificate must be for Digital Signing and be a Code Signing certificate. Requires saving the PFX password using: `Bundler.exe setsigningcredential -p "<pfx password> "`
--oauth	Require Microsoft Entra ID (Azure AD) authentication via OAuth to authorize application access. Requires `--appid`.
--appid <app-id>	Application ID (client_ID) of registered app in Azure Portal. Required for `--oauth`.
--allowedtenants <domain.name>	Define the Azure AD domain to use during authentication. Optional with `--oauth`. **Example**: `--allowedtenants abc.com --allowedtenants xyz.com`

Command Line Arguments when Migrating

The following table shows the command line arguments that are available when migrating existing FlexApp One packages to newer FlexApp.zip, Engine.exe, or FlexApp_One.lic files.

These arguments are for the `Bundler.exe migrate` command.

Option	Description
-p <packagefolder1> -p <packagefolder2> -p <packagefolder3>	Specifies the folder for the FlexApp package(s) that should be bundled into FlexApp One. (*At least one is Required*)
-f <flexapp.zip>	Specifies the flexapp.zip to be included in the FlexApp One. (*Required*)
-t <engine.exe>	Specifies the engine.exe to be included in the FlexApp One. (*Required*)
--licfile <path\licfile,lic>	Specifies the location to the FlexApp_One.lic file. **Default**: `.\` (Current working directory).
--seticon	Will set the output file to the package default icon.
-i <path\file.ico>	Allows you to specify the icon file that will be used when setting the icon for the output file. Requires `--seticon`
--nocallhome	Disables usage data collection.
--plainconsole	Use alternate console text output format.
-e <yyyy-mm-dd>	Causes FlexApp One application EXE to expire on the specified date so that it can no longer be executed.
--certfile <cert.pfx>	Digitally sign the FlexApp One with the specified PFX certificate. Certificate must be for Digital Signing and be a Code Signing certificate. Requires saving the PFX password using: `Bundle.exe setsigningcredential -p <pfx password>`
--oauth	Require Microsoft Entra ID (Azure AD) authentication via OAuth to authorize application access. Requires `--appid`.
--appid <app-id>	Application ID (client_ID) of registered app in Azure Portal. Required for `--oauth`.
--allowedtenants <domain.name>	Define the Azure AD domain to use during authentication. Optional with --oauth. Example: `--allowedtenants abc.com --allowedtenants xyz.com`

FlexApp One Application Command Line Arguments

The following section describes the available command line arguments for FlexApp One applications. Where applicable, command arguments can be combined. For example, you can combine `--ctl --index 999` to both create the shortcut and replay the application – but not open it immediately. In some command line arguments below, bracketed placeholders indicate information to be supplied based on your use case. Neither of the actual characters such as < and > should be typed in on the command line. This merely represents the information that should be inserted at that point in the command line.

Important: Use **two** hyphens with all command line arguments.

Argument	Description
--admin	Shows additional options in the system tray icon for this FlexApp One application.
--install	Installs the FlexApp service (LWLContainerService) as an elevated process. Use as the only argument, elevated.
--upgrade	Upgrades older FlexApp service only and works the same as the `--install` option if the FlexApp service does not exist.
--uninstall	Stops and removes the FlexApp service. Use as the only argument, elevated.
--startup	The FlexApp One application will be added to the HKCU\Run and launched each logon with any subsequent options that may also be provided after this one.
--index <#>	Activates this layer and starts the shortcut at index number #, based on the FlexApp Package's XML file <Links> section. Use 999 to skip launching shortcuts but still mount and be ready for usage. Useful for file associations and context-menus.
--ctl	Causes a Click-to-Layer shortcut to be created on the Desktop and Start Menu.
--addtostart	Adds the FlexApp One to the **Start Menu\Programs\FlexApp One** folder. This option provides an additional way to always access the application.
--stop	Stops the application layer and unmounts it from the OS.
--replace <path to old flexapp.exe>	Replaces and restarts the application if it was running. Run the new package.exe, specifying the old package in the argument. The new application will have the old application name when finished as this flag REPLACES the previous EXE. Use simple names as your final application name that the user will run. Continue to use versions in the package name when you package. The tray will show the package's given name, which can identify the version running in the workspace. **Example**: `NewUpdated.exe --stop --replace C:\LocalCopy\OldApp.exe`

Argument	Description
--reg	Creates a registration key at: `HKCU\Software\FlexApp\App.exe\%username%` This can be used in a variety of ways, for example, as a user Microsoft Intune detection rule.
--clean	Removes the shortcuts and registry key. **Must use with or after** `--stop`.
--sync <target path>	Block level copy of the file to an existing folder. **Example**: `<RemotePath>\App.exe --sync C:\FlexApps\App.exe --ctl --index 999`
--system	Creates shortcuts in the Public areas versus User areas and will create registry tracker in HKLM if using the `--reg` option. Packages executed as LOCAL SYSTEM will automatically use this functionality.
--remove	Removes the FlexApp One application EXE. Use with or after `--stop --clean`
--skipactivation	Will carry out the runtime commands, but not activate the package. Use with `--ctl` or `--addtostart` to create shortcuts only.

--assoc "<.ext|index#|args>"
 	
Index is the shortcut index of the package.
	Set file association(s) for applications within the FlexApp One package. Use with `--skipactivation`.
	Examples:
	`Atom.exe --assoc ".txt" --skipactivation`
	`Paint.Net.exe --assoc ".png|0" --assoc ".jpg|0" --skipactivation`
	`CustomerApp.exe --assoc ".cust|1|-launchflag" --skipactivation`
	`Office.exe --assoc ".doc|0" --assoc ".xls|1" --skipactivation`
	`Chrome.exe --assoc "browser|0" --skipactivation`

--extract <target path>	Extracts the VHDX, ICO, and XML files, which can be imported into the FlexApp Packaging Console and edited, cloned, etc.
--skipdisk	Skips the VHDX during extraction. Must be used with `--extract`.
--skipico	Skips the ICO file during extraction. Must be used with `--extract`.
--skipxml	Skips the XML file during extraction. Must be used with `--extract`.
--mapicons	Extracts the ICO files for the shortcuts referenced in each <link> section of the package XML. Must be used with `--extract`.
--debug	Runs the engine in debug mode. Can be used with `--console`.

Argument	Description
--console	Runs the engine outputting the log to the console.
--persist	Used to persist changes to the application's files.
--sessionisolate	Enable session isolation for the packaged application.
--hidetray	Hide the FlexApp One System Tray icon for this package.

Configuring FlexApp One Zero Trust Application Access

The integration of OAuth with Azure AD in FlexApp One signifies a monumental shift towards bolstering security. Users are now mandated to undergo authentication before activating a FlexApp One package, embracing a Zero Trust approach for all Win32 applications. This enhancement ensures that only verified individuals have the capability to access and activate FlexApp One packages, upholding the highest standards of cybersecurity.

Integrating FlexApp One into a Zero Trust security model is not only possible, but extremely straightforward and requires completing the following two processes:

- Create a registered application in Azure.
- Pass the OAuth argument when creating the FlexApp One package, providing the respective application and/or tenant domain details.

Register the application in Azure Portal

1. Login to the Azure Portal.
2. Navigate to the **App registrations** service page.
3. Click **New Registration** from the banner at the top-left of the page.
4. Define a name for the registered app (e.g., LWL FlexApp One).
5. Under **Supported account types**, select **Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)**.
6. Under **Redirect URI**, select **Public client/native (mobile & desktop)** for the platform and provide the following in the URI field:

   ```
   http://localhost/flexappone
   ```

7. Click **Register** at the bottom of the page to complete the app registration process.
8. Make note of the value present in the **Application (client) ID** column as it will be required during packaging.

Limit access to specific Azure AD users and/or groups

1. Navigate to the **Enterprise applications** service page.
2. Click on the newly created registered app from the list of available apps shown in **All applications** (e.g., LWL FlexApp One).
3. Under the **Manage** section on the left of the page, click **Properties**.
4. In the **Properties** page, set the **Assignment required** setting to **Yes**.
5. Click **Save** in the ribbon at the top of the page.
6. Under the **Manage** section on the left of the page, click **Users and groups**.
7. On the **Add Assignment** page, click **None Selected** to display a search pane.
8. From the search pane, search for and check the box next to the accounts and/or groups requiring access to the application.
9. Once all desired accounts and/or groups are selected, click **Select** and then click **Assign**.
10. Under the **Manage** section on the left of the page, click **Permissions**.
11. Click **Grant admin consent for {TenantName}**.

> **Note:** Cloud Application Administrator, Application Administrator, or Global Administrator role is required to grant consent.

Create a FlexApp One package with OAuth enabled

Option A

Create a package using `Bundler.exe`, the required arguments, and any additional arguments typically used, including the following arguments as well:

1. Oauth -Enables OAuth functionality.
2. AppId -Recorded value from step 8 of Register the application in Azure Portal.
3. AllowedTenants (Optional) -Restrict access based on domain name (e.g., company.com).

> **Note:** Multiple AllowedTenants arguments can be defined to accommodate different allowed domains.

Example command:

```
".\bundler.exe" bundle -p "\\Server\FlexApps\PackageFolder" -f ".\Flexapp.zip" -
o "\\Server\FlexApps\PackageFolder" -t ".\engine.exe" --seticon --oauth --appid
<RegisteredAppId> --allowedtenants <FQDN>
```

Option B

Define the FA1 CLI arguments via FlexApp Packaging Console.

1. Launch the FlexApp Packaging Console.
2. Navigate to the console **Settings** page (gear icon at top-right).
3. Define the Oauth, AppId, and AllowedTenants (optional) CLI arguments in the **Create FlexApp One Arguments** field. For example:

```
--oauth --appid <RegisteredAppId> --allowedtenants <FQDN>
```

4. Click **OK** to apply the changes.

> **Note:** Any FlexApp One packages created via FPC will not contain these additional parameters.

Option C

Define the FA1 CLI arguments via FlexApp Packaging Automation by adding it to the json file specified in the `/PackagesFile` FPA argument or passing directly via CLI by including the FPA argument below:

```
/FlexAppOneCliOverride "--oauth --appid <RegisteredAppId> --allowedtenants <FQDN>"
```

Microsoft Intune Deployments

FlexApp One applications can be distributed to end users using Microsoft Intune. For more information see FlexApp One Configuration Guide for Microsoft Intune

Digitally Signing FlexApp One Application Packages

Digitally Signing FlexApp One Application Packages

FlexApp One Applications can be digitally signed for enhanced security and verification, ensuring that your applications have not been tampered with.

This feature causes the certificate information to show in the Digital Signature tab of the FlexApp One EXE Properties window. This information can then be used to allow the FlexApp One EXE to be verified, or allowed by ProfileUnity's Application Restriction Module, AppLocker, or other application-control solutions, resulting in an enhanced level of security.

Obtaining a Code Signing Certificate

The FlexApp One Bundler accepts password protected PFX files created with most normal certificate request procedures, whether self-signed, from an Active Directory CA or third-party CA. The certificate must be set for Digital Signing and specifically be a Code Signing certificate. Other certificate types are not supported.

Microsoft offers an example for creating a self-signed certificate at: Create a certificate for package signing

Creating Signed FlexApp One Application Packages

To create signed FlexApp One Applications, you will need to supply some information to the FlexApp One Bundler. This is done differently depending on your method of creating FlexApp One Application packages.

Using the FlexApp Packaging Console

Creating signed FlexApp One Applications with the FlexApp Packaging Console requires the following steps:

1. On the FPC machine, already reverted to a clean state, copy your PFX Code Signing certificate to the following folder:

   ```
   "C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Console\FlexApp One Bundler"
   ```

2. Open an elevated `cmd.exe` prompt and run the following command using the password for the PFX file:

```
"C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Console\FlexApp One
Bundler\bundler.exe" setsigningcredential -p "<PFX password>"
```

3. Open and login to the FlexApp Packaging Console software and open the **Settings** screen using the "gears" button at the top-right of the window. Scroll down to **Create FlexApp One Arguments** and enter the following using your PFX filename:

```
--certfile "C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging
Console\FlexApp One Bundler\<filename>.pfx"
```

4. Click the **OK** button at the bottom to save the settings.
5. Proceed to take a new clean-state snapshot to be used for automatically signing new FlexApp One Applications from future FlexApp package captures.

Using FlexApp Packaging Automation

Creating signed FlexApp One Applications with FlexApp Packaging Automation requires the following steps:

1. On the FPA Packaging Agent machine(s), already reverted to a clean state, copy your PFX Code Signing certificate to the following folder:

```
"C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Automation\FlexApp One
Bundler"
```

2. Open an elevated `cmd.exe` prompt and run the following command using the password for the PFX file:

```
"C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Automation\FlexApp One
Bundler\bundler.exe" setsigningcredential -p "<PFX password>"
```

3. Edit your existing PackagesFile, DefaultsJSON, or existing CLI(s) and add the following to the "FlexAppOneCliOverride" or `/FlexAppOneCliOverride` setting:

```
"--certfile \"C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging
Console\FlexApp One Bundler\<filename>.pfx\""
```

4. Proceed to take a new clean-state snapshot to be used for automatically signing new FlexApp One Applications from future FlexApp package captures.

Using the FlexApp One Bundler.exe

Creating signed FlexApp One Applications with `Bundler.exe`, directly, requires the following steps:

1. Open an elevated `cmd.exe` prompt and run the following command using the password for the PFX file: `bundler.exe setsigningcredential -p "<PFX password>"`
2. Append the following argument to your existing `Bundler.exe` bundle CLI(s) using your PFX file-name: `--certfile "<filename>.pfx"`
   Example bundle command:

```
"C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Automation\FlexApp One
Bundler\bundler.exe" bundle -p "\\server\Packages\FlexAppName_20240207114326" -f
".\flexapp.zip" -t ".\engine.exe" --accepteula --plainconsole --certfile
".\FlexAppOneCertificate.pfx" -o "\\server\Packages\FlexAppName_20240207114326"
--licfile ".\flexapp_one.lic" --seticon
```

Microsoft AVD, Citrix, Other Remote Desktop Distribution

To deploy FlexApp One applications in remote desktop environments, three steps are typically required. All three steps are required for Microsoft AVD and Citrix Virtual Apps environments.

1. Install the prerequisite FlexApp service and run-time engine. This can be done running one of the following:
    a. Run the installer in system context:

    ```
    Installer.exe --install
    ```

    OR

    b. Run any FlexApp One application (this application might optionally contain prerequisites):

    ```
    FlexAppOne.exe --install
    ```

2. Sync the file to a local folder, activate the layer, and do this at the system level using one of the following:
    a. If the app is stored on a UNC, run the FlexApp One in system context with:

    ```
    \\server\share\FlexApp\Appname\Appname.exe --sync C:\Pro-
    gramData\FlexApp\Appname\Appname.exe --index 999 --system
    ```

    OR

    b. If the app is local, run the FlexApp One in system context with:

    ```
    C:\ProgramData\FlexApp\Appname\Appname.exe --index 999 --system
    ```

3. Finish the app deployment by running the local FlexApp One in system context with the following to add the application to the Start Menu, and create a shortcut in the public area without launching the app using the following:

    ```
    C:\ProgramData\FlexApp\Appname\Appname.exe --ctl --addtostart --skipactivation
    --system
    ```

Using Session Isolation on multi-user OSes

FlexApp One can be deployed in a way such that activated application folders and files are hidden from users that haven't, or can't, launch the FlexApp One package EXE. Normal operation does not hide the activated application's files from users. If a user can't access the FlexApp One EXE, they would normally still be able to find the application's files in Program Files and launch it directly. Enabling Session Isolation prevents a user from being able to see or launch the files.

1. Add the --SessionIsolate option to existing package activation commands, whether on-boot or at on-demand using Click-to-Layer (--ctl --sessionisolate). This prevents users without access to the app from looking for the files within Program Files and launching it directly.
2. Restrict access to the FlexApp One package EXE only to those that should have access. If anyone can access the FlexApp One EXE and launch it with --SessionIsolate, they can unhide the files within Program Files, so an ACL to restrict access is a necessary part of this feature.

OneDrive Deployments

You can distribute your FlexApp One applications to end users using whatever method you prefer. The following outlines a sample process using OneDrive.

1. Share your FlexApp One application on OneDrive with your distribution group.
2. Run a login script to register the apps in the OneDrive folder:

   ```
   --startup --addtostart --ctl
   ```

Configure OneDrive GPO Administrative Settings

1. Locate your OneDrive build number:
   a. Click the **OneDrive** icon in your system tray.
   b. Click **Help & Settings** and select ***Settings*** from the menu.
   c. Go to the **About** tab and write down your build number.
2. Browse to your OneDrive root folder and correct build number. Example path for source ADMX and ADML files:

   ```
   C:\Users\xx\AppData\Local\Microsoft\OneDrive\21.052.0314.0001\adm
   ```

For English:

1. Take the OneDrive ADMX and place it in a `PolicyDefinitions` folder in:

   ```
   \\dc\sysvol\domain\policies
   ```

2. Place the ADMX in the `PolicyDefinitions` folder and the ADML in an `en-US` folder under `PolicyDefinitions`.
   a. Make sure the language folder exists (`en-US`) and place the ADML file in there.
   b. Make `PolicyDefinitions` and `PolicyDefinitions\en-US` folders if they do not exist.
   c. Example path for destination ADMX and ADML files:

   ```
   \\dc\sysvol\domain.local\policies\PolicyDefinitions

   \\dc\sysvol\domain.local\policies\PolicyDefinitions\en-US
   ```

Configure the OneDrive GPO

1. With your OneDrive GPO, browse to **Computer Configuration > Policies > Administrative Templates > OneDrive**.
2. Get the **Tenant ID** and **Sync Path** to be used when configuring the synced site library.
   a. Log in to your OneDrive, locate the folder to be shared and click ***Sync***.
   b. Click **Copy ID** near the bottom of the large OneDrive white information dialog box.
   c. Paste this information into the GPO.

Sample Command Line Usage

Launching and activating FlexApp One applications can be done in many ways using a variety of command line options.

For the best single user experience, instruct each FlexApp to create any provided Desktop and Start Menu Shortcuts for the user, to place a shortcut to itself in the Start Menu FlexApp group and to do this each time the user logs in. The command line options for this are --CTL --AddToStart --Startup.

For multi-user systems, run in the system context and call each FlexApp with options --System --Index 999 to mount the application. You can also use --Sync to copy the FlexApp One local and then launch it locally when the sync completes. Including the --SessionIsolate option when initially activating the package will enable session isolation, preventing users from directly running the package executable and hiding the files of the played back application unless the user also passes the --SessionIsolate option when launching the application. After the initial launch, call each application again with *--System --CTL --AddToStart --SkipActivation* to create any Desktop and Start Menu Shortcuts, add a shortcut to the executable in the FlexApp Start Menu group, and skip activating the application since it was already mounted in the prior step. In order to ensure those without access to the application can't also run the same shortcuts, an ACL will need to be used on the FlexApp One package EXE so that only authorized users or groups can execute it. The --SessionIsolate option alone only hides the files, like in Program Files, from users that haven't activated the package within their session.

> **Note:** FlexApp One applications are non-persistent and revert back each time you restart your desktop, unless you use the --persist command line option to persist the applications virtual disk changes.

The following are commonly used command line options.

System Context Options

Variations and combinations often used for multiuser workspaces depending on distribution solution or use case:

--system --addtostart --index 999

--sync C:\ProgramData\FlexApp\App\App.exe --system --startup --addtostart
--index 999
--sync C:\ProgramData\FlexApp\App\App.exe --system --index 999 --sessionisolate

User Context Options

Variations and combinations often used for single user workspaces depending on distribution solution or use case:

--startup --addtostart --ctl --skipactivation

--sync %appdata%\FlexApp\App\App.exe --startup --addtostart --index 999

--sync %appdata%\FlexApp\App\App.exe --startup --addtostart --ctl --skipactivation

--sync %appdata%\FlexApp\App\App.exe --startup --addtostart --index 0

--addtostart --ctl --skipactivation --sessionisolate

Using Login Scripts

Variations and combinations often used for single user workspaces:

A login script can also be used to execute each FlexApp One executable with command line options. The following is a single user system example:

'app.exe --startup --addtostart --ctl'

This will add the FlexApp binary to the USERS registry RUN key, add a shortcut to the binary to a FlexApp start menu folder, and add Click-to-Layer shortcuts for the applications executable (if provided in the original package) to the Start Menu and Desktop locations.

Add --system to add to the MACHINES registry RUN key.

Support Logs

Support log files can be found in the following locations:

%temp%\ProfileUnity

%windir%\Temp\ProfileUnity

%programdata%\Microsoft\IntuneManagementExtension\Logs (for Intune Deployments)

In addition, the ProfileUnity and FlexApp Diagnostic Tool is included at:

```
C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging
Automation\punc\LwL.ProfileUnity.Client.Diagnostic.exe
```

When double-clicked, it will create a ZIP in %TEMP% with some logs and other machine-related information that is useful when submitting a support case.

Getting Help

If you have questions or run into issues while using our software, Liquidware is here to help. Our goal is to provide you with the knowledge, tools, and support you need.

Using Online Resources

Liquidware maintains various kinds of helpful resources on our Customer Support Portal. If you have questions about your product, use these online resources. The Support Portal includes product forums and a searchable knowledge base, as well as the ability to submit a case to the Liquidware Support system on the Liquidware Customer Support Portal. For product documentation, refer to our Liquidware Document Repository.

Troubleshooting with the Software

ProfileUnity and FlexApp provide full logging capabilities to track activities. In addition, Liquidware provides several avenues for self-service, which includes the Liquidware Customer Support Portal, Knowledgebase, and Online Training site. You may find answers to many of your questions, along with Liquidware Self Service options.

Contacting Support

If you need to contact our Support staff for technical assistance, log a request on the Liquidware Customer Support Portal. Prior to logging a case you should review these helpful tips:

- Check the Product Documentation included with your Liquidware Product.
- Try to see if the problem is reproducible.
- Check to see if the problem is isolated to one machine or more.
- Note any recent changes to your system and environment.
- Note the version of your Liquidware product and environment details such as operating system, virtualization platform version, etc.