ProfileUnity™
with FlexApp™ Technology

FlexApp Packaging Automation

v1.0.25

Introduction

This guide has been authored by experts at Liquidware in order to provide information and guidance concerning the ProfileUnity with FlexApp's Packaging Automation framework.

Liquidware Labs, Inc.
3600 Mansell Road
Suite 200
Alpharetta, Georgia 30022
U.S.A.
Phone: 678-397-0450
www.liquidware.com

Contents

What's New for FlexApp Packaging Automation?

Version 1.0.25 – Released August 18TH, 2021

The first version of FlexApp Packaging Automation is released to market. This framework allows for multiple and concurrent unattended FlexApp Package captures.

Overview - FlexApp Packaging Automation

In some cases, one-off captures of applications for dynamic layering to end-users can be repetitive and time-consuming. This applies to any application layering or virtualization technology – call it capturing, recording, or sequencing – packages or layers must be created and tested prior to being deployed to production.

Some situations call for a more automated packaging process. Consider scenarios like migrating from another application deployment tool or method to FlexApp, compliance-mandated scheduled application updates or even integrating FlexApp package creation as a component of a larger DevOps system to reduce time and intervention in getting new software builds deployed to end-users.

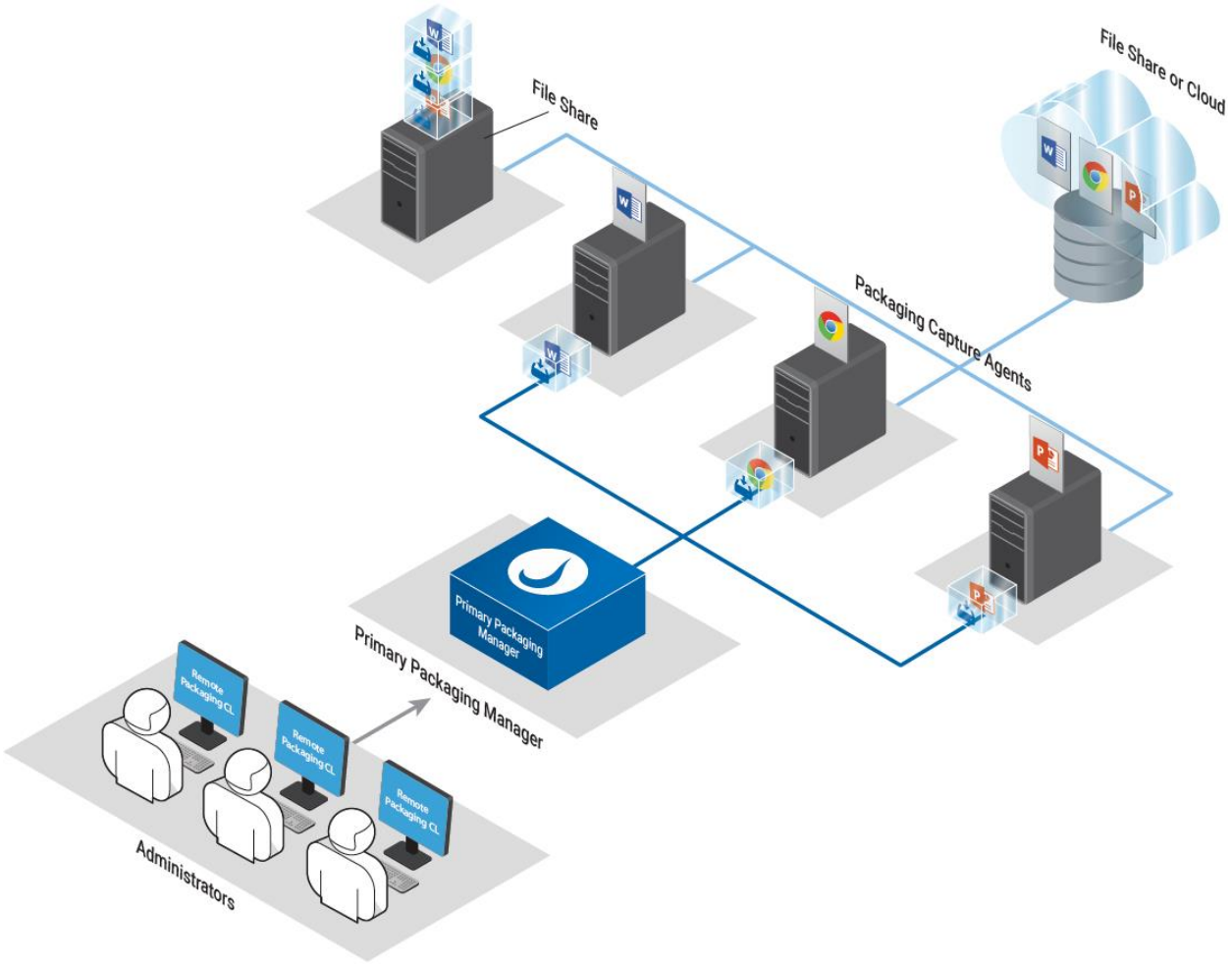
To address requests of this nature, Liquidware has created the FlexApp Packaging Automation framework. FlexApp Packaging Automation, or FPA, uses a Primary Packaging Manager to delegate packaging jobs to a network of Packaging Capture Agents that execute the silent install commands, capture the software being installed and create FlexApp packages on a network share – immediately ready to be assigned to end-users.

Going deeper into these few example scenarios,

1. If an organization is using a tool to push out silent installs to end-users like SCCM, Intune, MEM, etc., then migrating your application deployments to FlexApp can be accelerated using FlexApp Packaging Automation by building packaging jobs using the installers and silent install commands already contained in your current software deployment tool.
2. If an internal policy requires some or all your end-user software applications to be updated on a recurring schedule, maybe for security or industry-compliance reasons, then FlexApp Packaging Automation can be used to accelerate the repetitive re-packaging of the new versions to shorten the time-to-live for version updates and reduce manual IT intervention.
3. If an organization relies on an application that is developed in-house, FlexApp Packaging Automation can be called by the software build server as the last step of a DevOps process to create a FlexApp package from the resultant binaries or installer. Once complete, that package then shows in the ProfileUnity Console ready to be assigned to end-users.

Architecture and Multi-Admin Usage

There are various use-cases one can come up with around the use of FlexApp Packaging Automation. Here is one example of a common architecture that can help you plan your installation of FPA.



This architecture would allow packaging jobs to be submitted from the Primary Packaging Manager or remotely by one or more admins using the Remote Packaging CLI. In addition, you can conserve resources and use your existing ProfileUnity Management Console machine as the Primary Packaging Manager.

Software Requirements

The FlexApp Packaging Automation framework is made up of a Primary Packaging Manager and one or more Packaging Capture Agents that run the packaging jobs and place the resulting packages in the specified storage locations.

Primary Packaging Manager

Component	Minimum Requirements
Virtual Machines	1 VM to be used as the Primary Packaging Manager. The ProfileUnity Management Console can be used as the Primary Packaging Manager or a new VM can be created.
VM OS Support	Windows 10 or Windows Server 2016/2019
VM vCPU	2 vCPU
VM Memory	4 GB
VM Storage	1 GB free on C:
Prerequisites	Microsoft® .NET 5.0 is automatically installed
Network and Firewall	Primary Packaging Manager connects to TCP/9074 on Packaging Capture Agents

Packaging Capture Agents

Component	Minimum Requirements
Virtual Machines	1+ VM(s) to be used as Packaging Capture Agents **Liquidware recommends 3+ VMs for large batch jobs and 2+ for linear/DEVOPS/one-off processing to maintain redundancy.*
VM OS Support	Windows 10 or Windows Server 2016/2019
VM vCPU	2+ vCPU
VM Memory	6+ GB
VM Storage	1 GB free on C:
VM Free Space	Free disk space is needed locally, usually C:, on each Packaging Capture Agent VM for capturing application installations. It is recommended to have at least double what you think you will ever need for your largest capture.
Prerequisites	Microsoft® .NET 5.0 is automatically installed
Network and Firewall	Packaging Capture Agents connect to TCP/9075 on Primary Packaging Manager

Network or Cloud Storage

A network file share or a cloud location used to store new packages is required. In addition, storage for the installer EXEs and MSIs that will be accessed by the Capture Agents will be needed.

Silent Install Requirement

Applications are required to install silently due to the headless nature of FPA. Most application installers require the use of flags on the CLI to indicate a silent install. Test each installer's silent install commands manually prior to using in a packaging job. There must be no user-interaction involved during install. Prompts will hang capture until the timeout.

Multi-Administrator Package Creation

When multiple package administrators are sending packaging jobs to the same Primary Packaging Manager it is recommended that any `PathUsername` or `InstallerUsername` be a service account created for this purpose and stored in the DefaultsJSON file. As a result, the service account will handle all file operations on the network share preventing the need for each admin to use their own credentials.

Installation - FPA-Installer.exe Command Line Arguments

The following describes the available commands for `FPA-Installer.exe`. Be sure to wrap all complex passwords in double-quotes like "&()[]{}^=;!'+,`~ my c0mpl3x PW". As a result of this requirement, double quotes cannot be used in passwords or AES secrets. Paths that include a space also require double quoting.

Command	Description
Help	Displays the command line options Usage: `fpa-installer.exe Help`
Eula	Extracts the embedded End-User License Agreement (EULA) to disk for review Usage: `fpa-installer.exe Eula [/Path "<folder>"]`
Install Agent	Installs FPA Packaging Capture Agent Usage: `fpa-installer.exe Install Agent /AgentUsername "fpa_services" /AgentPassword "<pass>" [/Path "<folder>"] [/Temp "<folder>"] [/ServiceUsername "<domain\user>" /ServicePassword "<pass>"] [/ServiceLogPath "<folder>"] [/ServiceLogLevel <level>] [/Port <port>] [/CertificateFile "<path\cert.pfx>" [/CertPassword "<pass>"]] /AcceptEula`
Install Primary	Installs FPA Primary Packaging Manager Usage: `fpa-installer.exe Install Primary /PrimaryUsername "fpa_services" /PrimaryPassword "<pass>" [/Path "<folder>"] [/Temp "<folder>"] [/ServiceUsername "<domain\user>" /ServicePassword "<pass>"] [/ServiceLogPath "<folder>"] [/ServiceLogLevel <level>] [/Port <port>] [/CertificateFile "<path\cert.pfx>" [/CertPassword "<pass>"]] /AcceptEula`
Install	Installs FPA Remote Packaging CLI Usage: `fpa-installer.exe Install [/Path "<folder>"] [/Temp "<folder>"] /AcceptEula`
Uninstall	Removes services and installed files but leaves behind .NET 5 and ProgramData Usage: `fpa-installer.exe Uninstall [/Path "<folder>"]`
DEFAULTS	**By default, the primary-service and agent-service both run as Local System.*
Eula [/Path]	"%USERPROFILE%\Desktop"
Install * [/Path]	"C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Automation"
Install * [/Temp]	"%TEMP%" *(Used for installation only)*
Install * [/ServiceLogPath]	"C:\Windows\Temp\fpa"
Install * [/Port]	*Primary*=9075, *Agent*=9074
Uninstall [/Path]	*(Path used during installation)*

Setting up FlexApp Packaging Automation

Download the FlexApp Packaging Automation framework from the Liquidware Support Portal.

> ***FPA requires a set of internal credentials to secure the API, like prou_services. They are specified during installation and consumed during package creation. They are not used to access any resources or linked to Active Directory.***

Preparing the Packaging Capture Agents

**Liquidware recommends required frameworks and runtimes be natively installed on end-user and Packaging Capture Agent VMs for reduced package overhead. See this KB article for more information.*

1. Create a new VM and optimize the operating system to reduce unnecessary background activity. There are various options and opinions on how best to optimize a packaging capture VM. Liquidware, VMware and Citrix all offer VM optimization tools. More information here and here.

    a. An alternative method for creating Packaging Capture Agents is to clone your existing, already-optimized FlexApp Packaging Console VM and then uninstall the FPC software, reboot and that clone is ready to become an FPA Packaging Capture Agent.

2. From an elevated Command Prompt, install the Packaging Capture Agent:
```
fpa-installer.exe install agent /AgentUsername "fpa_services"
/AgentPassword "<pass>" /AcceptEULA
```

3. Ensure you have a "clean state" snapshot taken after installing the Packaging Capture Agent to which you can revert, if needed.

Preparing the Primary Packaging Manager

1. From an elevated Command Prompt, install the Primary Packaging Manager:
```
fpa-installer.exe install primary /PrimaryUsername "fpa_services"
/PrimaryPassword "<pass>" /AcceptEULA
```

2. If desired, add `"C:\Program Files (x86) \Liquidware Labs\FlexApp Packaging Automation"` to your `PATH` env variable and open a new `cmd.exe`, otherwise `cd` there now.

3. Add your Packaging Capture Agents to your Primary Packaging Manager:
```
primary-client.exe add agent /PrimaryUsername "fpa_services"
/PrimaryPassword "<pass>" /AgentAddress
https://<AgentMachineNameOrIp>:9074 /AgentUsername "fpa_services"
/AgentPassword "<pass>"
```

4. Repeat `Step 3` for each Packaging Capture Agent VM you've created.

5. Confirm your agents are available and ready to accept packaging jobs:
```
primary-client.exe list agent /PrimaryUsername "fpa_services"
/PrimaryPassword "<pass>"
```

(Optional) Installing the Remote Packaging CLI

1. If you want to run commands from another machine instead of needing to run them from the Primary Packaging Manager, you can install the Remote Packaging CLI on another workstation: `fpa-installer.exe install /AcceptEULA`
2. If desired, add "`C:\Program Files (x86)\Liquidware Labs\FlexApp Packaging Automation`" to your `PATH` environment variable.

Testing Package Creation

Testing scenario

In this example, we'll setup files that allow a packaging admin to save their credentials and only need to provide the `CryptoKey` on the CLI when creating packages. This makes it more convenient to share PackagesFiles with other admins since it wouldn't need to contain login credentials and you also wouldn't need to specify them on the command line each time you run the `Create Packages` command.

Creating a test PackagesFile and DefaultsJSON

1. From your Primary Packaging Manager or the Remote Packaging CLI, create your PackagesFile:

   a. `primary-client.exe Add PackagesFile /PackagesFile "%USERPROFILE%\Desktop\PackagesFile.json" /Name "Notepad++ v8.1.1" /Path "\\fileserver\FlexApp\Packages" /Installer "\\fileserver\FlexApp\Installers\npp.8.1.1.Installer.exe" /InstallerArgs "/S"`

   b. `primary-client.exe Add PackagesFile /PackagesFile "%USERPROFILE%\Desktop\PackagesFile.json" /Name "PuTTY Utils v0.76" /Path "\\fileserver\FlexApp\Packages" /Installer "C:\Windows\System32\msiexec.exe" /InstallerArgs "/i \\fileserver\FlexApp\Installers\putty-0.76-installer.msi /qn"`

2. Next, create the DefaultsJSON:

```
primary-client.exe Create DefaultsJSON /DefaultsJSON
"%USERPROFILE%\Desktop\defaults.json" /PrimaryUsername "fpa_services"
/PrimaryPassword "<pass>" /PathUsername "<domain\user>" /PathPassword
"<pass>" /PuAddress <https://ProuServerAddress:8000> /PuUsername
"<domain\user>" /PuPassword "<pass>"
```

Testing your packaging job by creating the specified packages

- **Method 1** - Create Packages using PackagesFile and DefaultsJSON, wait for the job to finish:

```
primary-client.exe Create Packages /PackagesFile
"%USERPROFILE%\Desktop\PackagesFile.json" /DefaultsJSON
"%USERPROFILE%\Desktop\defaults.json" /WaitForDone
```

***Example console status when complete:**

```
------------------------------------------------------------
Batch Job Report
b7bd 8/1/2021 2:48:58 PM 8/1/2021 2:50:09 PM Successful
Total:2 Pending:0 Running:0 Successful:2 Failed:0 Cancelled:0
  Notepad++ v8.1.1 df52 8/1/2021 2:48:58 PM 8/1/2021 2:49:29 PM Successful 0 Agent1
    Notepad++ v8.1.1 3d18 8/1/2021 2:48:58 PM 8/1/2021 2:49:27 PM Successful 0
  PuTTY Utils v0.76 e4ee 8/1/2021 2:48:59 PM 8/1/2021 2:49:31 PM Successful 0 Agent2
    PuTTY Utils v0.76 ae64 8/1/2021 2:48:59 PM 8/1/2021 2:49:28 PM Successful 0
------------------------------------------------------------
```

- **Method 2** - Create Packages using PackagesFile and DefaultsJSON in background, email results:
  `primary-client.exe create Packages /PackagesFile "%USERPROFILE%\Desktop\PackagesFile.json" /DefaultsJSON "%USERPROFILE%\Desktop\defaults.json" /MailServer <smtp.something.com> /MailFrom <x@y.com> /MailTo <a@b.com>`
  ***Example email status when complete:** *(condensed)*

Job Name	Status	Agent	MM:SS	Installer	Installer Args
Notepad++ v8.1.1	Successful	Agent1	0:30	npp.8.1.1.Installer.exe	/S
PuTTY Utils v0.76	Successful	Agent2	0:31	msiexec.exe	/i putty-0.76-installer.msi /qn

PackagesFile and DefaultsJSON File Contents

- Our example file contents will look like this (excluding the unspecified parameters).
  `packagesfile.json`:

```
"Name": "Notepad\u002B\u002B v8.1.1"
"Path": "\\\\fileserver\\FlexApp\\Packages"
"Installer": "\\\\fileserver\\FlexApp\\Installers\\npp.8.1.1.Installer.exe"
"InstallerArgs": "/S"
"Name": "PuTTY Utils v0.76"
"Path": "\\\\fileserver\\FlexApp\\Packages"
"Installer": "C:\\Windows\\System32\\msiexec.exe"
"InstallerArgs": "/i \\\\fileserver\\FlexApp\\Installers\\putty-0.76-installer.msi /qn"
```

  `defaults.json`:

```
"PrimaryUsername": "537F6AA867FB405lookatmeimencrypted3u9Mt3sXxGcaigPxIngPqyw=="
"PrimaryPassword": "537F6AA867FBlookatmeimencryptedz1MjK/h/ZdYpPBuMyg\u002BQ=="
"PathUsername": "537F6AA867FB4lookatmeimencryptedA8\u002BycoKQXBOgZ5g=="
"PathPassword": "537F6AA867FB40598lookatmeimencryptedVysei\u002BXyj1EJakAQ=="
"PuAddress": "https://MyProuServerAddress:8000"
"PuUsername": "537F6AA867FB405989836E5EDlookatmeimencrypted02BycoKQXBOgZ5g=="
"PuPassword": "537F6AA867FB4059898lookatmeimencryptedj1EJakAQ=="
```

- Editing your `PackagesFile` can be done with a text editor or by using `primary-client.exe Add PackagesFile`, `Remove PackagesFile`, `List PackagesFile`, `Clear PackagesFile`.

- The `DefaultsJSON` works a bit differently than the `PackagesFile` in that the `primary-client.exe Create DefaultsJSON` command is used to completely overwrite the file with a new file containing only the newly specified parameters. Therefore, a text editor is likely the easiest method to adjust any non-encrypted values.

- When adding information into a `PackagesFile` or `DefaultsJSON`, you have two encryption options for user/pass fields. Users can never see the credentials stored in these files. See Notes about encryption for more info.

Available Packaging Job Parameters

The following list of parameters are available for use on the CLI and in the `PackagesFile` or `DefaultsJSON`. CLI usage requires a preceding / (ie, `/Name "Some Application v1.0"`) and usage in a JSON file requires the parameter to be in double quotes (ie, `"Name":"Some Application v1.0"`). Be sure to wrap all complex passwords in double-quotes like "&()[]{}^=;!'+,`~ my c0mpl3x PW". As a result of this requirement, double quotes cannot be used in passwords or AES secrets. Paths that include a space also require double quoting.

All parameters are optional except `Name`, `Path` and `Installer`.

Job Parameter	Description
Name "<package-name>"	Package Name **(Required)* *Should be unique within a given packaging batch job
Path "<path>"	Target folder where the package(s) will be created **(Required)* *`Path` ending in `.vhdx` will perform an Extend of existing package *Cloud paths are supported – `s3://` `az://` `gs://`
Installer "<path>"	Path of the installer exe to be executed and captured **(Required)* *CIFS path like `\\server\share\folder\installer.exe` *Use `C:\Windows\System32\msiexec.exe` for MSI installs
InstallerArgs "<args>"	Installer-specific silent install flags required for proper operation *Args must be wrapped in quotes (ie, `"/S"` or `"/i \\path\installer.msi /qn"` (for msiexec)
SizeMb <integer>	Package VHDX size Default: 20000 *(20gb)*
Fixed	Use Fixed VHDX *(allocate all space now)* Default: Expandable *(Grow as needed)*
Test	Plays back the package after save and takes screenshots Default: Don't Test
PathUsername "<domain\user>"	Username used to access the package's path *Cloud paths use s3=Access Key, az=Account Name, gs=<omit>
PathPassword "<password>"	Password used to access the package's path *Cloud use s3=Secret Key, az=Account Key, gs=<credential.json>
InstallerUsername "<domain\user>"	Username used to access the installer's path
InstallerPassword "<password>"	Password used to access the installer's path
InstallerExitCode <integer>	Expected installer SUCCESS exit code Default: 0
InstallerTimeoutMs <integer>	How long to wait for an installer to finish before failing the capture Default: 3600000 *(1hr)*

Job Parameter	Description
PuAddress <https://ProuServerNameOrIP:8000>	The ProfileUnity Console where new packages will be imported
PuUsername "<domain\user>"	Username used to access the ProfileUnity Console
PuPassword "<password>"	Password used to access the ProfileUnity Console
NoSystemRestore	Do not perform a System Restore rollback after capture/extend ***REQUIRED for Server 2016 and 2019-based Capture Agents!** Default: False (Use System Restore to roll back)
AltRestoreCmd "<pathToScript>"	Instead of System Restore, use a rollback script after capture/extend *Runs from the Agent VM and implies `NoSystemRestore=True` Default: Rollback determined by `NoSystemRestore` parameter
AltRestoreCmdArgs "<args>"	Args (if any) needed for the `AltRestoreCmd` *Args must be wrapped in quotes (ie, `"/S"`)
WaitAfterInstallerExitsMs <integer>	How long to wait after installation ends before saving the capture Default: 0
DontCopyInstallerLocal	Run the installer directly from the `Installer` path Default: Copy installers locally before executing ("C:\Windows\Temp\fpainstaller" or "%TEMP%\fpainstaller")
Installer2 "<path>"	Path of an additional installer to be executed during the capture
InstallerArgs2 "<args>"	Installer-specific silent install flags required for proper operation *Args must be wrapped in quotes (ie, `"/S"`)
InstallerExitCode2 <integer>	Expected installer exit code used to determine successful installation Default: 0
Installer3 "<path>"	Path of an additional installer to be executed during the capture
InstallerArgs3 "<args>"	Installer-specific silent install flags required for proper operation *Args must be wrapped in quotes (ie, `"/S"`)
InstallerExitCode3 <integer>	Expected installer exit code used to determine successful installation Default: 0
PreActivationScript "<path>"	CMD file to include in package and execute before playback
PostActivationScript "<path>"	CMD file to include in package and execute after playback
PreDeactivationScript "<path>"	CMD file to include in package and execute before stopping playback
PostDeactivationScript "<path>"	CMD file to include in package and execute after stopping playback
NoCallToHome	Don't send job `Installer` and `InstallerArgs` stats to Liquidware Default: False (Always send this data to Liquidware)

Job Parameter	Description
LogPath "<path>"	Folder in which to store logs Default: "%TEMP%\fpa"
LogLevel <level>	Logging level – Debug, Info, Warn, Error, Fatal (Advanced Logging Levels: Console, Always) Default: Debug
MailServer <smtp.company.com> *(DefaultsJSON and CLI only)*	SMTP server to use as a relay for job-related emails Default: No emails are sent unless specified
MailPort <integer> *(DefaultsJSON and CLI only)*	SMTP port used for `MailServer` Default: 25
MailSsl *(DefaultsJSON and CLI only)*	Use SSL/TLS when connecting to `MailServer` Default: False (No SSL/TLS)
MailUsername "<username>" *(DefaultsJSON and CLI only)*	Username to use for relaying emails through `Mailserver`
MailPassword "<password>" *(DefaultsJSON and CLI only)*	Password to use for relaying emails through `Mailserver`
MailTo <SomeGuy@company.com> *(DefaultsJSON and CLI only)*	Email address that should receive job-related emails *Required for `/MailServer`
MailFrom <noreply@company.com> *(DefaultsJSON and CLI only)*	Email address to show as sender of job-related emails *Required for `/MailServer`
PrimaryAddress <https://PrimaryNameOrIp:9075> *(DefaultsJSON and CLI only)*	Primary Packaging Manager address Default: https://localhost:9075
PrimaryUsername "fpa_services" *(DefaultsJSON and CLI only)*	Primary Packaging Manager username
PrimaryPassword "<password>" *(DefaultsJSON and CLI only)*	Primary Packaging Manager password
/WaitForDone *(CLI only)*	Wait for job to finish and allow job logs to be copied locally Default: Return to cmd prompt after submitting packaging job
/Crypto Aes *(CLI only)*	Encrypt the user/pass fields contained in the command line w/Aes *See Notes about encryption for more info! Default: Encrypt all user/pass fields using `/Crypto DpApi`
/CryptoKey "<aes-secret-key>" *(CLI only)*	AES Encryption Passphrase / Key *Required for `/Crypto Aes`

Notes About Encryption and Log Paths

When adding information into a `PackagesFile` or `DefaultsJSON`, you have two encryption options for user/pass fields. In any case, users can never see the credentials stored in these files – they can only use them to process `primary-client` commands. If you don't specify `/Crypt`, the default of `DpApi` is used.

- `/Crypto DpApi` is a self-contained, machine-based encryption using Windows Data Protection API. Anyone with access to the machine can utilize (but not see) the creds without needing an encryption key. Files encrypted with this method cannot be moved to another machine. **THIS IS DEFAULT!**

- `/Crypto Aes` is an AES-based encryption method utilizing a secret key and requires the key to be passed on the command line with `/CryptoKey` to utilize (but not see) the creds. Files encrypted with this method can be shared between different machines as long as the secret key is known.

- `/LogPath` and `/LogLevel` used on the CLI apply only to the `primary-client.exe` log for the current command. `"LogPath"` and `"LogLevel"` used within a JSON file apply only to the capture job log on the capture agent. There is no automatic authentication for a UNC-based `LogPath` so the capture agent's machine account or service account will need "unprompted access" to the `LogPath`.

Primary-Client.exe Commands

The following describes the available commands for `primary-client.exe`. Be sure to wrap all complex passwords in double-quotes like "&()[]{}^=;!'+,`~ my c0mpl3x PW". As a result of this requirement, double quotes cannot be used in passwords or AES secrets. Paths that include a space also require double quoting.

Command	Description
Help	Displays the command line options Usage: `primary-client.exe Help`
Add PackagesFile *(omit all params to get an empty template file)*	Adds new information to an existing PackagesFile or creates a new file Usage: `primary-client.exe Add PackagesFile /PackagesFile <path/packagesfile.json> [/Crypto Aes /CryptoKey "<my-aes-secret-key>"] /Name <package name> /Path <path> [/PathUsername <domain\user> /PathPassword "<pass>"] /Installer <path\installer.exe> [/InstallerArgs "<args>"] [/InstallerUsername <domain\user> /InstallerPassword "<pass>"] [/SizeMb <MB>] [/Fixed] [/Test] [/PuAddress <https://pu.server:8000> /PuUsername <domain\user> /PuPassword "<pass>"] [/InstallerExitCode <integer>] [/InstallerTimeoutMs <ms>] [/WaitAfterInstallerExitsMs <ms>] [/DontCopyInstallerLocal] [/NoSystemRestore] [/AltRestoreCmd /AltRestoreCmdArgs "<args>"]] [/Installer2 <path\installer.exe> /InstallerArgs2 "<args>"] /InstallerExitCode2 <integer>]] [/Installer3 <path\installer.exe> /InstallerArgs3 "<args>"] /InstallerExitCode3 <integer>]] [/PreActivationScript <path\file>] [/PostActivationScript <path\file>] [/PreDeactivationScript <path\file>] [/PostDeactivationScript <path\file>] [/NoCallToHome] [/LogPath <path>] [/LogLevel <level>]`
List PackagesFile *(or view the file by hand)*	List package information from an existing PackagesFile Usage: `primary-client.exe List PackagesFile /PackagesFile <path\packagesfile.json>`
Remove PackagesFile *(or edit the file by hand)*	Removes <package name> from an existing PackagesFile Usage: `primary-client.exe Remove PackagesFile /PackagesFile <path\packagesfile.json> /Name <package name>`

Command	Description
Clear PackagesFile *(or edit the file by hand)*	Removes ALL entries from an existing packages file Usage: `primary-client.exe Clear PackagesFile /PackagesFile <path\packagesfile.json>`
Create DefaultsJSON *(Optional, omit all params to get an empty template file)*	Creates optional DefaultsJSON with default params used by packaging jobs Usage: `primary-client.exe Create DefaultsJSON /DefaultsJSON <path\defaults.json> [/Crypto Aes /CryptoKey "<my-aes-secret-key>"] [/MailServer <smtp.something.com> /MailTo <a@b.com> /MailFrom <x@y.com> [/MailPort <integer>] [/MailSsl] [/MailUsername <domain\user> /MailPassword "<pass>"]] [--All 'Add PackagesFile' parameters are supported here--]`
List DefaultsJSON *(or view the file by hand)*	List parameters from a DefaultsJSON file Usage: `primary-client.exe List DefaultsJSON /DefaultsJSON <path\defaults.json>`
Create Packages	Creates the packages listed on the CLI or in the specified file(s) Usage: `primary-client.exe Create Packages [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>" /PackagesFile <path\packagesfile.json> [/DefaultsJSON <path\defaults.json>] [/Crypto Aes /CryptoKey "<my-aes-secret-key>"] [/MailServer <smtp.something.com> /MailTo <a@b.com> /MailFrom <x@y.com> [/MailPort <integer>] [/MailSsl] [/MailUsername <domain\user> /MailPassword "<pass>"]] [/WaitForExit]`
Status Packages	Retrieves the status of Create Packages batch jobs Usage: `primary-client.exe Status Packages [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>" [/Id <guid>]`
Wait Packages	Waits for a currently running create packages batch job to complete Usage: `primary-client.exe Wait Packages [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>" /Id <guid>`
Add Agent	Adds a Capture Agent to the Primary Packaging Manager Usage: `primary-client.exe Add Agent [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>" /AgentAddress <https://server:9074> /AgentUsername <user> /AgentPassword "<pass>"`
Remove Agent	Removes a Capture Agent from the Primary Packaging Manager Usage: `primary-client.exe Remove Agent [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>" /AgentAddress <https://server:9074>`

Command	Description
List Agent	Lists all Capture Agents currently added to the Primary Packaging Manager Usage: `primary-client.exe List Agent [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>"`
Clear Agent	Deletes ALL Capture Agents from the Primary Packaging Manager Usage: `primary-client.exe Clear Agent [/PrimaryAddress <https://server:9075>] /PrimaryUsername <user> /PrimaryPassword "<pass>"`

Getting Help with FlexApp Packaging Automation

If you have questions or run into issues while configuring or running FlexApp Packaging Automation, you can contact Liquidware for help if your ProfileUnity or FlexApp support contract is valid. Our goal is to provide you with the knowledge, tools, and support you need to be productive.

Using Online Resources

Liquidware maintains various kinds of helpful resources on our Customer Support Portal. If you have questions about your product, please use these online resources to your full advantage. The Support Portal includes product forums, a searchable Knowledge Base, documentation, and best practices among other items. You can visit our website at https://www.liquidware.com/support.

Contacting Support

If you wish to contact our Support staff for technical assistance, please either log a request on the Liquidware Customer Support Portal or give us a call. Prior to Logging a Case you may want to 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.

To speak directly with Support, please use the following numbers:

Main Line: 1-678-397-0460

Toll Free in US & Canada: 1-866-914-9665

Europe/Middle East/Africa: +44 800 014 8097

Toll Free in Europe

UK: 0800 014 8097

Netherlands: 0800 022 5973

Switzerland: 0800 561 271