ProfileUnity™
with FlexApp™ Technology

FlexApp Packaging Console
Guidance and Best Practices

Introduction

This guide has been authored by experts at Liquidware Labs in order to provide best practices and guidance concerning the ProfileUnity FlexApp Packaging Console.

Information in this document is subject to change without notice. No part of this publication may be reproduced in whole or in part, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any external use by any person or entity without the express prior written consent of Liquidware Labs.

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

©2016 Liquidware Labs Inc. All rights reserved. Stratusphere, ProfileUnity, FlexApp, FlexDisk, ProfileDisk and Flex-IO are trademarks of Liquidware Labs. All other products are trademarks of their respective owners. 16-1021

Contents

General Best Practices

FlexApp Layer packaging best practices are built upon the same guidance leveraged by many typical standard application packaging platforms. Here are some common concepts to establish a baseline:

- Conduct initial target application analysis. Consider leveraging Application Assessment data from Stratusphere FIT, prior to the FlexApp packaging process.
- Maintain a centralized, reusable virtual machine (VM).
    - Virtualization technologies have evolved into a critical core component for any enterprise application packaging process.
    - Establish a range of FlexApp Packaging virtual machines representing the various operating system versions within the enterprise.
- Maintain a FlexApp Layer repository.
    - Leverage a centralized, highly available network share for FlexApp Package/Layer storage.
    - Additionally, where possible, leverage storage technology or best practices to ensure redundancy.
    - Ultimately this becomes critical with respect to any FlexApp Layer updates that need to occur within the production environment
- Perform routine backups with respect to the FlexApp Layer repository.
- Perform preliminary application test installs as needed and document the results.
- Consider leveraging tools like Citrix AppDNA with respect to OS compatibility analysis as part of an overall application lifecycle.

FlexApp Packaging Prerequisites

This section will discuss the benefits of preparation and planning with respect to the FlexApp Layer creation process. Often, one of the most overlooked aspects of this process is the staging required for the basic application installation, as part of the packaging process. What if the application has a heavy memory requirement? What if the disk footprint of the install is larger than most apps? These are questions that need to be answered before you start the packaging process, or else you run the risk of multiple failures. Here is a basic list of some of the items to be aware of:

- Packaging Operating System (OS)
    - Determine which OS versions are required – Windows 7, 8.x, or 10
        - The OS of the packaging console should match the OS of the target machines to which application layers will be deployed.
    - Determine whether 32- or 64-bit architecture versions are required
        - The bit-level of the packaging console should match the bit-level of the target machines to which application layers will be deployed.
- Review the FlexApp Packaging Console Manual for additional system requirements for your FlexApp Packaging VM.
- VM preferences
    - Establish hardware settings within the corresponding VMs that match or resemble production target application requirements.
- Identify prerequisite software that must be added to the FlexApp Package workspace.
- Identify application dependencies.
- Identify basic packaging steps based on outcome launch characteristics.

- Eliminate pop-ups.
- Eliminate license demand.
    - Identify application licensing requirements.
- Plan for any user customizations that will be required during the packaging process.
- For now, Google Chrome is recommended to access either the FlexApp Packaging Console or the ProfileUnity Management Console
- Make sure .NET version 4.5.x at a minimum is installed on the FlexApp Packaging Console.
- Make sure .NET version 4.5.x at a minimum is installed on the UAT environment.
- Disable UAC on the FlexApp Packaging Console VM. (Optional)
- Disable Anti-Virus on the FlexApp Packaging Console VM.
- Block GPO inheritance if the FlexApp Packaging Console is member of domain.
- Compare this list to additional Knowledge Base articles for FlexApp Best Practices on the Liquidware Labs Support Portal.

Building FlexApp Packaging Console Virtual Machines Checklist

There are many considerations that come into play, when establishing a FlexApp Packaging Console VM. Below is a list of items for consideration:

	Task	Status ✓
1.	**Determine which operating system versions are required.** *The OS of the packaging console should match the OS of the target machines to which application layers will be deployed. FlexApp requires Windows 7 or higher.*	
2.	**Determine whether 32- or 64-bit architecture versions are required.** *The bit-level of the packaging console should match the bit-level of the target machines to which application layers will be deployed.*	
3.	**Maintain a minimum of one (1) VM with snapshots per OS version.** *Typically, applications should be packaged using the same OS version as the desktop to which packaged layers will be deployed, meaning you may need multiple packaging VMs.*	
4.	**FlexApp Packaging Console desktops can be either physical or virtual.** Typically, a VM would provide a more robust efficient use case.	
5.	**Review FlexApp Packaging Console System Requirements.** *Visit the Liquidware Labs Support Portal and download the FlexApp Packaging Console Manual from the ProfileUnity Documentation page.*	
6.	**Configure Virtual Machine Preferences.** *Establish hardware settings within the corresponding VMs that match or resemble production target application requirements.*	
7.	**Disable Anti-Virus on the FlexApp Packaging Console VM.**	
8.	**Block GPO inheritance if the FlexApp Packaging Console is member of domain.**	
9.	**Install the FlexApp Packaging Console on your VM.** *Follow the directions in the FlexApp Packaging Console Manual for downloading and installing the software. The installer will check to see if any prerequisite software is needed including Microsoft .NET Framework. Do NOT install the FlexApp Packaging Console on the same machine as the ProfileUnity Management Console.*	
10.	**Make sure .NET version 4.5.x at a minimum is installed on the FlexApp Packaging Console.**	
11.	**Make sure .NET version 4.5.x at a minimum is installed on the UAT environment.**	
12.	**Consider leveraging the VMware OS Optimization Tool.** *Download the guide here.*	
13.	**Compare this list to additional Knowledge Base articles for FlexApp Best Practices on the** Liquidware Labs Support Portal**.**	

FlexApp Packaging Optimization Checklist

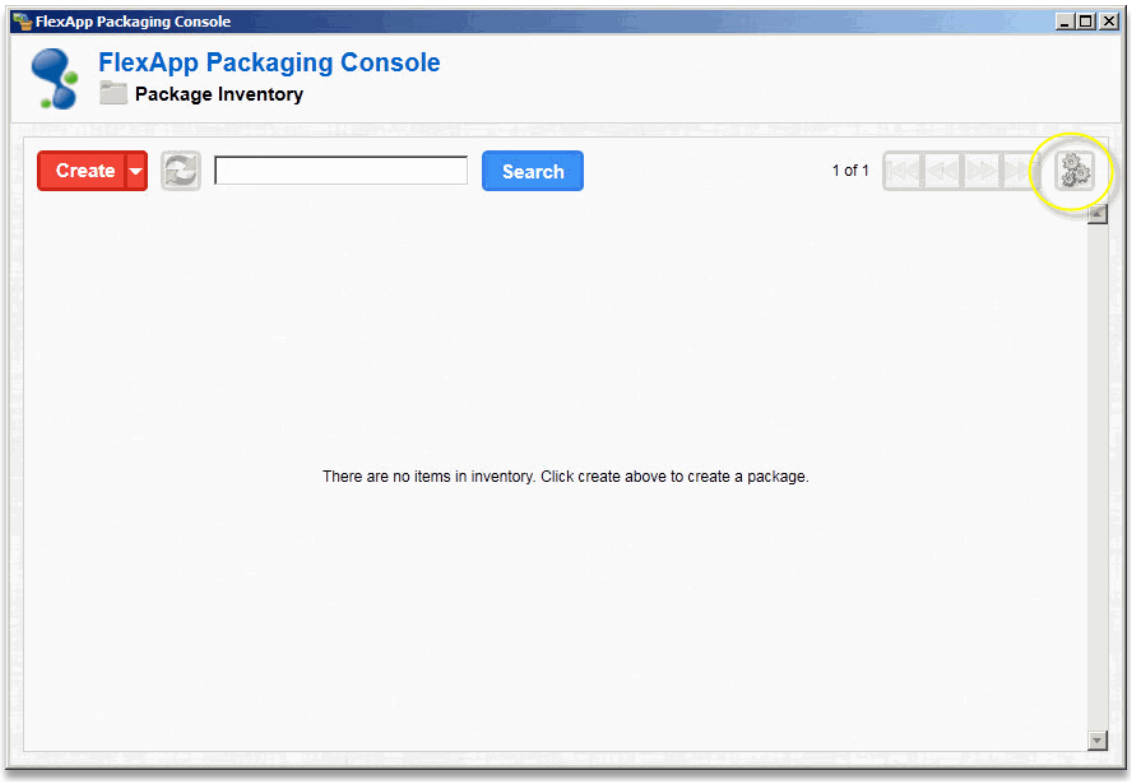
There are also opportunities to optimize the FlexApp Packaging process and enhance the results. Preparation and planning with respect to the FlexApp Layer creation process are key. Often, one of the most overlooked aspects of this process is the staging required for the basic application installation, as part of the packaging process. What if the application has a heavy memory requirement? What if the disk footprint of the install is larger than most apps? These are questions that need to be answered before you start the packaging process, or else you run the risk of multiple failures. Below are some items to consider with respect to FlexApp Packaging optimization:

	Task	Status ✓
1.	**Preferred browser.** *For now, Google Chrome, Firefox or recent versions of IE are recommended to access either the FlexApp Packaging Console or the ProfileUnity Management Console*	
2.	**Eliminate browser pop-ups.**	
3.	**Identify application dependencies.** *If an application being packaged requires other software prerequisites, one package can be created to hold both the application and its prerequisites. Start by packaging the prerequisite software first. Then extend the package to include the original application to be packaged.*	
4.	**Identify application licensing requirements.** *Eliminate license demand. Application licensing is done on the FlexApp Packaging Console. If your application binds itself to any information tying it back to the FlexApp Packaging Console, your application's license will fail once the application is opened on your target OSs. To work around this scenario, you would move to a centralized licensing model where the application license can be staged into the base image of your OS and the FlexApp Packaging Console. The other option if your application can silently auto-register licensing on startup would be to not disrupt the user each time with licensing.*	
5.	**Make sure .NET version 4.5.x at a minimum is installed on the FlexApp Capture Console.**	
6.	**Make sure .NET version 4.5.x at a minimum is installed on the UAT environment.**	
7.	**Identify basic packaging steps based on outcome launch characteristics.** *Plan for any user customizations that will be required during the packaging process.*	
8.	**Consider performing preliminary application test installs and documenting the results.**	
9.	**Compare this list to additional Knowledge Base articles for FlexApp Best Practices on the** Liquidware Labs Support Portal**.**	

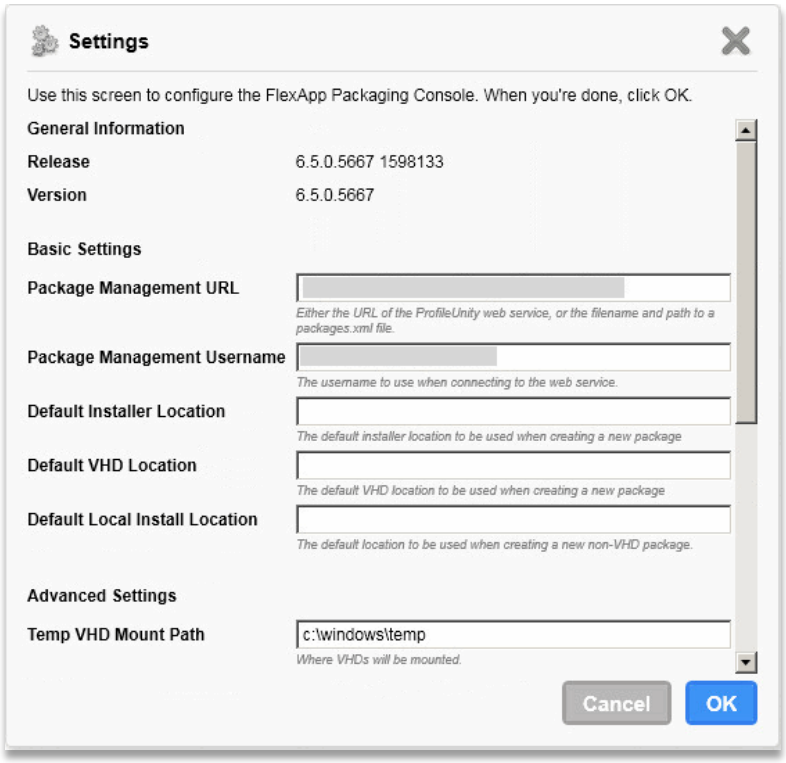
Optimizing the FlexApp Packaging Console

1. From the FlexApp Packaging Console, open a web browser and connect to the corresponding address of your ProfileUnity Management Console, typically in the form of HTTPS://servername:8000. Login to the ProfileUnity Management Console with your admin credentials.

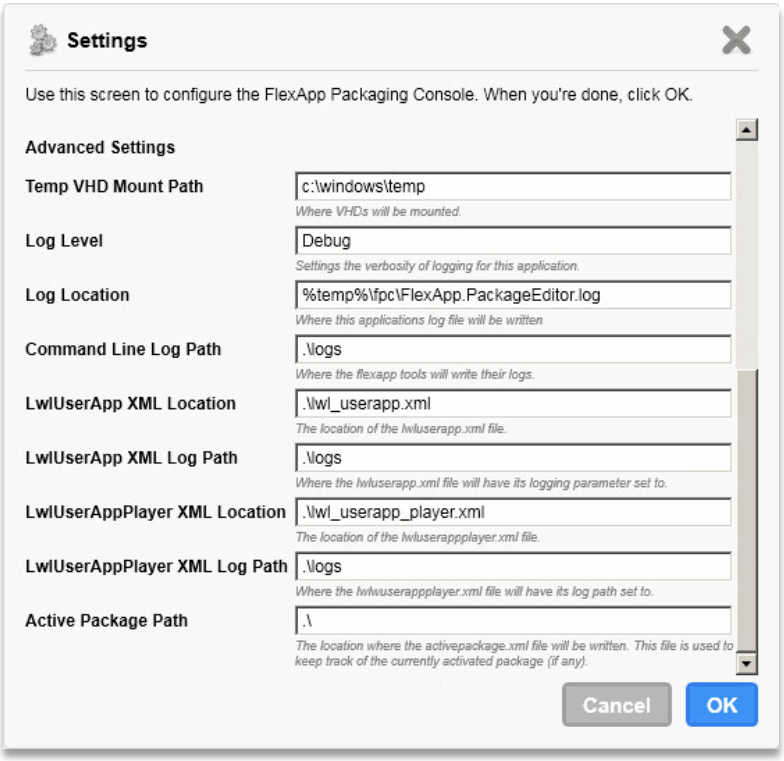
2. Once logged into the FlexApp Packaging Console, click on the settings button.



3. Enter information to streamline the FlexApp Layer packaging process.
    a. Consider entering the **Default VHD Location** to save time. A highly available network storage location is recommended.



b. Configure any additional options.



4. Click **OK** when finished updating settings

Repeatability Options

There are a few concepts within the virtualization space that allow for efficient repeatability of the application packaging process. Any of these options allow for increased efficiency within the FlexApp packaging process. There are no wrong choices when it comes to which repeatability approach chosen.

- Virtual Machine Snapshots
    - Embedded technology in most virtualization platforms that allows for the creation of a "photocopy" of the entire OS.
    - You can essentially revert or rollback the operating system to that "photocopy".
    - You can easily maintain many different snapshots in various states to maximize efficiency.
    - Taking a snapshot after the FlexApp Packaging Console is configured would provide for an efficient repeatable process.
    - Additional snapshots could be leveraged for either large applications or with respect to troubleshooting opportunities.
- Non-persistent Desktops
    - Embedded technology in most virtualization platforms that allows for a settings change within the corresponding virtual disk.
    - You can essentially set the virtual disk to a "non-persistent" state, after which any changes made within virtual machine would be discarded when you power off the VM.
    - Once the FlexApp Packaging Console has been configured, setting the virtual disk too non-persistent would provide for an efficient repeatable process.
- VDI infrastructure can be leveraged with the FlexApp packaging process when available
    - In addition to the options listed above, Administrators have the option to simply destroy the corresponding FlexApp Packaging Console at logoff once the FlexApp layer has been created.
    - The FlexApp packaging VMs can then be recreated and assigned to the FlexApp administrators.

FlexApp Layer Storage Options

FlexApp Layers can essentially be stored on any available location that users have access to. One of the great features of FlexApp is its ability to seamlessly integrate with any infrastructure. Leveraging any existing storage platform is a perfect example of this. Below is some basic guidance to get started:

- How to determine storage requirements
    - Once the FlexApp candidate list has been identified, calculate the basic install footprint required for each application.
    - These calculations can be tabulated to come up with the total disk footprint required for the corresponding FlexApp layers.
    - At which point, an N + 1 share can be created for redundancy.
        - ***Stratusphere FIT can conduct an application assessment of existing applications and desktops to establish the basic install footprint applications.***
- Simple
    - Leverage existing VDI best practices. Most VDI platforms call for a specific storage footprint during installation and configuration.
    - Creating shares off of this existing infrastructure is a very straight forward approach.
    - Most VDI platforms call for the creation of a Windows Server VM that acts as a storage appliance, hosting the various shares required.
    - Follow existing user permission guidance for the most simplistic approach.
- Advanced
    - Storage vendors like Nutanix, Simplivity, NetApp, etc. can be leveraged to host the FlexApp Layer repository.
    - Follow vendor specific installation and configuration guidance when creating the various file shares.
    - FlexApp will seamlessly integrate with and leverage these new file shares on the advanced hyper-converged platform.
- Disaster Recovery concepts
    - FlexApp Layer VHD or VMDK files can be replicated across storage locations to achieve a basic DR strategy.
    - There are many options for replication including Microsoft DFS. Most storage vendors have replication software included with the solution and can be leveraged to achieve results.

FlexApp Layer Storage Replication Options

FlexApp Layer replication for enterprise scalability has become a very important question recently. Here are a few examples of how to achieve results with FlexApp leveraging known platforms available to the enterprise.

Microsoft DFS considerations

- Microsoft DFS is a platform geared towards storage replication, redundancy and distribution of content within the enterprise environment.
- Domain Namespaces – Namespaces allow administrators to create single, logical reference points for folders that contain content. The namespaces are then replicated across enterprise AD infrastructure establishing unique application fault tolerance for the enterprise.
- DFS Targets and Replication – These Namespaces can encompass multiple targets or locations. Creating multiple targets can provide greater redundancy for enterprise users with respect to application availability.
- FlexApp Layers can be organized within DFS infrastructure and replicated across sites using unique namespaces and multiple targets.

3rd Party storage replication technology

Most Storage vendors, whether it be hyper-converged offerings from Nutanix or Simplivity, or proprietary offerings from EMC or Netapp, include software designed to provide content replication benefits. FlexApp Layers can easily be stored on files shares hosted within these various storage platforms, and then replicated across sites using the included software.

FlexApp Layer Testing

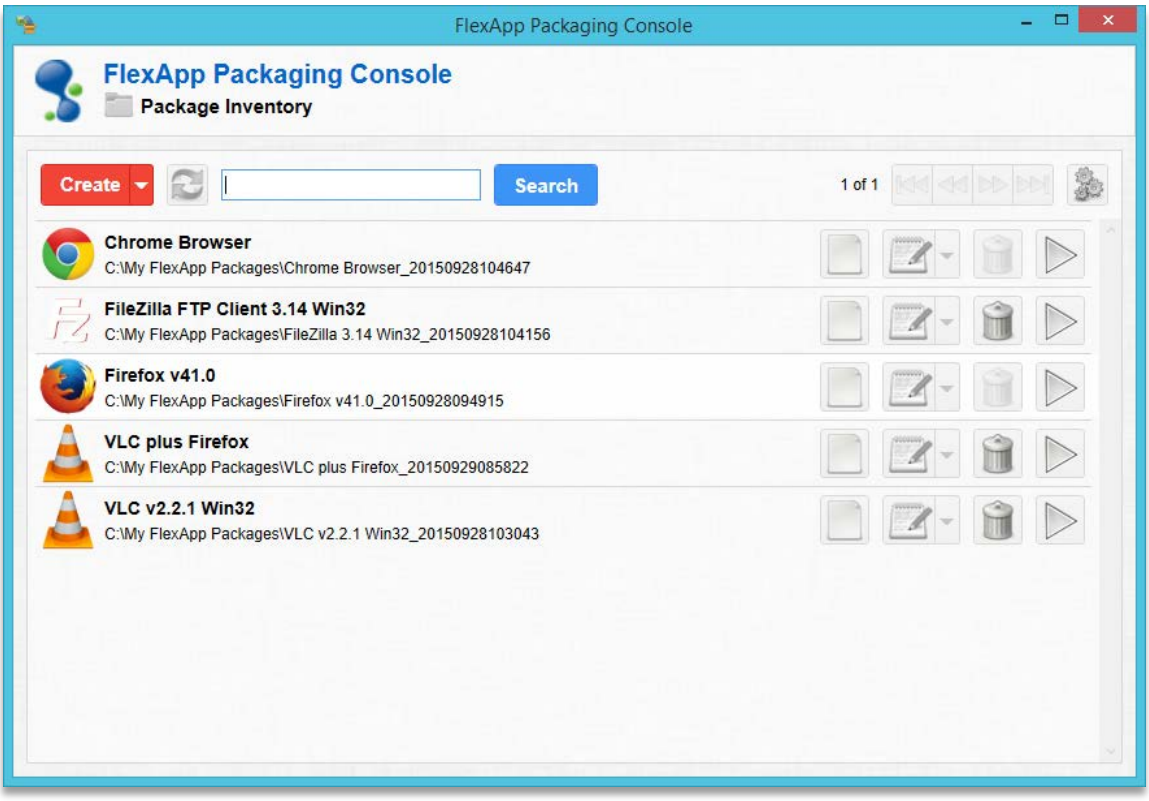
There are many application testing approaches leveraged by enterprises today. Here are a couple of common FlexApp Testing techniques for your reference:

- Primary Launch Test
    - Perform a simple test launch of the completed FlexApp Layer prior to reverting the FlexApp Packaging Console VM.
    - This will quickly identify the status of the target FlexApp Layer.
- Secondary Launch Test
    - Leverage the same FlexApp Packaging Console VM.
    - Revert to a prior snapshot essentially rolling back to a clean baseline.
    - This is sometimes referred to a “Clean” test considering the corresponding application within the FlexApp Layer has never existed within the VM before.
    - So this test provides important data with respect to real time deployment success capabilities of the FlexApp Layer.
- Pre-Production Test
    - Leverage a different FlexApp Packaging Console VM with a different windows OS version
    - This test provides critical data with respect to FlexApp Layer deployment capabilities to different OS versions.
- Basic Application Launch Testing
    - When the corresponding FlexApp Layer launches, perform any simple usage tests to gauge application functionality:
        - Click on drop down menus.
        - Attempt to change application settings.

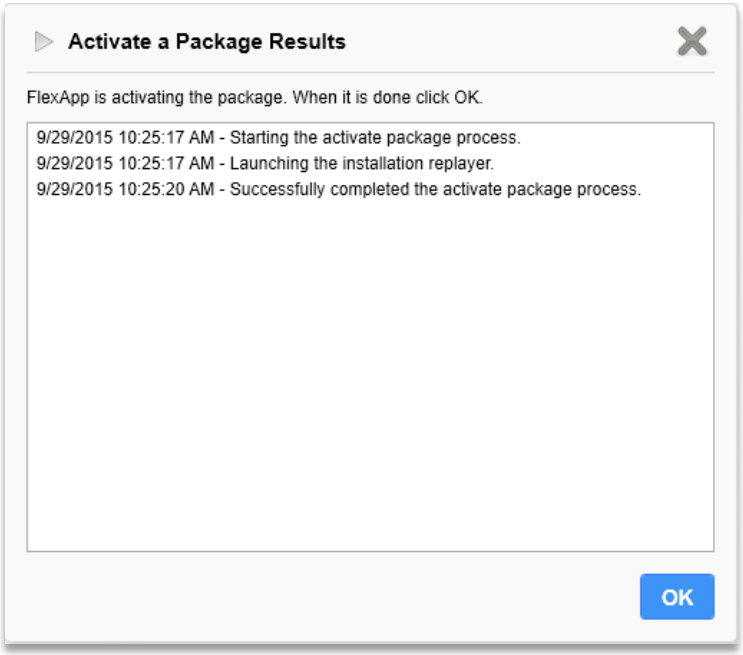
1. From the FlexApp Packaging Console, open a web browser and connect to the corresponding address of your ProfileUnity Management Console, typically HTTPS://servername:8000. Login to the ProfileUnity Management Console with your admin credentials.

2. Initial test
    a. The final step of the FlexApp Layer packaging process basically uninstalls the application from the FlexApp Packaging Console.
    b. Simply activate the new FlexApp Layer from the FlexApp Packaging Console and take note of the results.
    c. If the FlexApp Layer deployed successfully and PASSES initial launch testing, then you can move onto the CLEAN test phase.
    d. If the FlexApp Layer fails, then additional troubleshooting steps should be taken.
    e. See the **FlexApp Layer Troubleshooting** section for additional details.

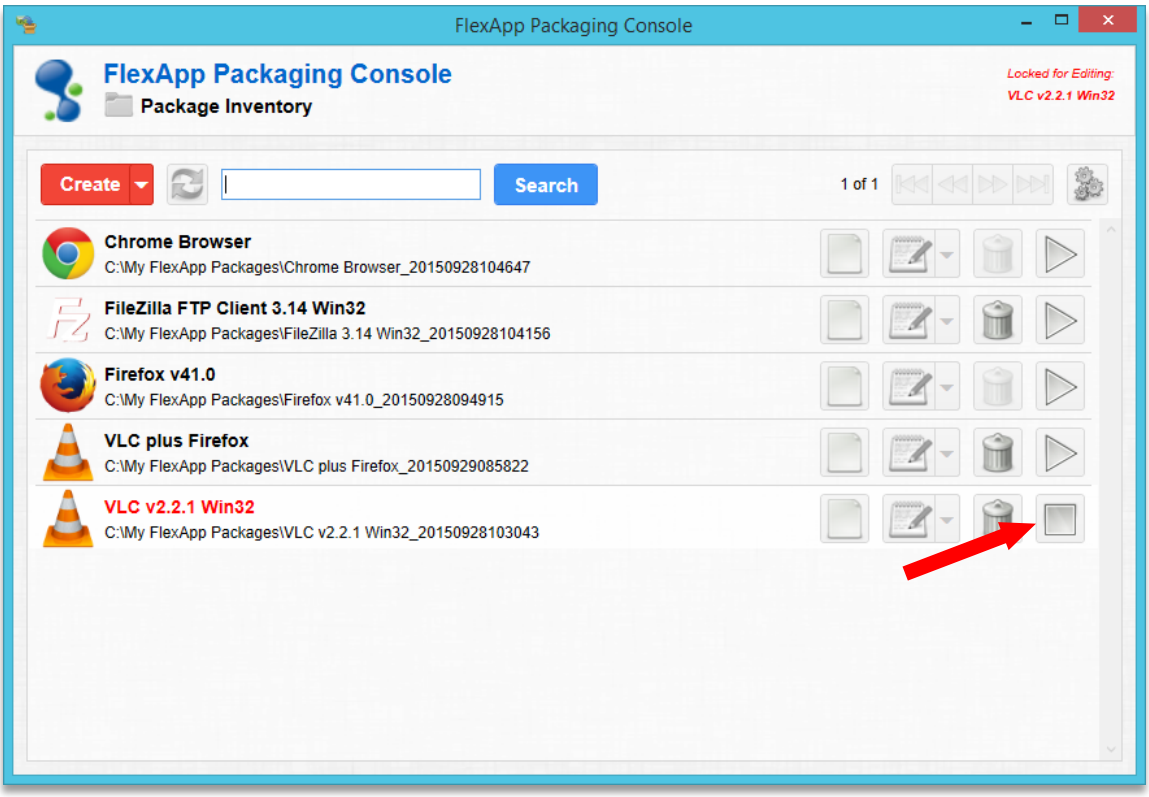
3. Click on the **Activate the package** button.



4. The FlexApp Packaging Console will show its progress on activation. Click **OK** when it is finished.



The **Activate** triangle icon will be replaced by the **Deactivate** square icon.



5. To deactivate an application, click on **Deactivate the Package**. Again, the FlexApp Packaging Console will show its progress towards deactivation. Click **OK** when it is finished.
6. Secondary Launch test
    a. Revert the FlexApp Packaging Console VM to a prior snapshot.
    b. This eliminates any trace of the corresponding application.
    c. Log into the FlexApp Packaging Console.
    d. Select the corresponding FlexApp Layer.
    e. Click the Activate button.
    f. This deploys/attaches the corresponding FlexApp Layer to the FlexApp Packaging Console.
        i. Passing the CLEAN test also means that the FlexApp Layer has proven functionality without any existence of the original install elements during the FlexApp Packaging process.
    g. If the FlexApp Layer PASSES the CLEAN launch test, then it is safe to proceed to UAT or production ready testing.
    h. If the FlexApp Layer FAILS the CLEAN launch test, then additional troubleshooting steps will be necessary.
    i. See the **FlexApp Layer Troubleshooting** section for additional details.
7. Production ready launch test
    a. Launch the FlexApp Packaging Console from different VM with a different OS version.
    b. Log into the FlexApp Packaging Console.
    c. Select the corresponding FlexApp Layer.
    d. Click the **Activate** button.

e. This deploys/attaches the corresponding FlexApp Layer to the FlexApp Packaging Console.
    i. Passing the CLEAN test also means that the FlexApp Layer has proven functionality without any existence of the original install elements during the FlexApp Packaging process.
f. If the FlexApp Layer PASSES the CLEAN launch test, then it is safe to proceed to UAT or production ready testing.
g. If the FlexApp Layer FAILS the CLEAN launch test, then additional troubleshooting steps will be necessary.

8. See the **FlexApp Layer Troubleshooting** section for additional details.

Additional Testing Steps

- App Owner Testing on FlexApp Packaging Console
    - Open all shortcuts right after packaging, use the app like normal, and license the application like you would normally.
    - Roll back the FlexApp Packaging Console snapshot.
    - Activate the application, open all shortcuts and check licensing status.
- App Owner Testing simulated production pool
    - Activate the application, open all shortcuts and check licensing status.
- User Acceptance testing (UAT)
    - Assign the application, use it like normal and check licensing status.

FlexApp Layer Troubleshooting

- Does the app work right after capture? Attempt to launch the target application after install but before clicking finish and closing the layer.
    - If no, stop here.
        - Check that the app works natively to be sure you don't have a corrupt installer or compatibility issue.
- Does app work and does licensing check out after rolling back the FlexApp Packaging Console?
    - If no, stop here.
        - Document error.
        - Check event viewer for errors or warnings that could apply to the application.
        - Some error messages have reference to:
            - If reg key
                - Try and export reg key from native install and import on the FlexApp Packaging Console as a test?
                - Is the reg key in the cap file but not being put down?
                - Is the reg key missing from the cap file?
                - Is the reg key being excluded by the virt config?
            - If file
                - Is the file in the virtual disk?
                - If file is missing from virtual disk, can you copy it in from native install and have the app work?
            - Service
                - Virt log should show installing services, failure?
            - Printer
                - Virt log will show installing printer, failure?
        - Compare application behavior to native install.
        - Does the application have drivers?
- Does the app work on the endpoint or desktop pool outside of the FlexApp Packaging Console?
    - If there is a licensing error, then it's possible the corresponding licensing can't be moved from machine to machine.

How to Handle Dependent Applications

Sometimes target applications may need prerequisite software installed first. The FlexApp Packaging Console allows the packaging of applications separately or the packaging of applications into one FlexApp Layer. To package applications into one layer, the admin would package the dependent application first and then "Extend" the FlexApp Layer to include another application, finally ending with the target application.

Dependent Apps Excluded from FlexApp Layer

1. Install the dependent application on the FlexApp Packaging Console OS before starting the packaging process. (For example .NET)
2. Start the FlexApp Packaging Console and capture the target application. (For example Paint.NET)
3. Perform test launches of the application.
4. Save and finalize the target FlexApp Layer.
5. Revert to a clean snapshot on the FlexApp Packaging Console VM.
6. Start the FlexApp Packaging Console and capture the dependent application.
7. Save and finalize the target dependent FlexApp Layer.
8. Revert to a clean snapshot on the FlexApp FlexApp Packaging Console VM.
9. Deploy both the target application and the dependent application to the user's desktop.
10. Test functionality.

The Paint.NET FlexApp Layer will attempt to find the .NET dependency on the host OS during the launch process.

Dependent Apps Included within FlexApp Layer

1. Start the FlexApp Packaging Console.
2. Start the packaging process.
3. First, package the dependent application on the FlexApp Packaging Console. (For example .NET)
4. Next, extend the same package to include the target application. (For example Paint.NET)
5. Perform test launches of the application.
6. Save and finalize the FlexApp Layer.
7. Revert to a clean snapshot on the FlexApp Packaging Console VM.
8. Deploy the one FlexApp Layer that contains both the target application and its dependent application to the user's desktop.
9. Test functionality.

The Paint.NET FlexApp Layer will leverage the .NET dependency inside the same FlexApp Layer during the launch process.

More on extending, cloning, and patching FlexApp Layers can be found in the ***FlexApp Packaging Console Manual***.

Updating or Patching Application Layers

Just like any application packaging process, FlexApp Layers can be updated on an as needed basis. The following guidance represents a basic quick start guide for updating FlexApp Layers. Most enterprises will simply integrate FlexApp Layering techniques into their existing workflows.

One common best practice technique is to make a "Clone" of the corresponding FlexApp Layer before updating it. Essentially this is similar to taking a snapshot of a VM. Making a clone of the corresponding FlexApp Layer first, provides an additional "Roll Back" capability within the update process.

Key questions:

- Is it easier to simply create a new FlexApp layer with the corresponding application or update?
- Is it easier to clone a FlexApp Layer and perform the update within this cloned Layer?

There are no wrong answers. Every application will present different requirements or best practices with respect to update strategy. Enterprise application packaging teams will often have best practices for both application packaging and updating that FlexApp Layering can simply integrate with.

More on extending, cloning, and patching FlexApp Layers can be found in the ***FlexApp Packaging Console Manual***.

Getting Help with ProfileUnity

If you have questions or run into issues while using ProfileUnity with FlexApp, Liquidware Labs is here to help. Our goal is to provide you with the knowledge, tools, and support you need to be productive.

Using Online Resources

Liquidware Labs 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 http://www.liquidwarelabs.com.

Troubleshooting with the Software

ProfileUnity with FlexApp provides full logging capabilities to track activities. Once you have tried to duplicate the issue with full logging turned on, the log file details can be used to help pinpoint the source of the problem you are experiencing. To turn logging on, go to your login user ID at the top right of the Management Console interface and select **Administration**. At the top of the Administration area, select **Settings**. As you scroll through the list you will see the Logging category. Set the **Level of Logging** to **Debug** to provide the most information. To view the log file, click on **View** under **Latest Log**. If you are still experiencing issues and need to contact technical support for additional help, the log file can be sent to support for further evaluation.

Not sure about your configuration settings? Don't forget that ProfileUnity offers summary reports for all your configurations, filters, and portability settings. Simply click on **Report** next to the name of the configuration, filter, or portability setting for which you need a report. You can download and save the report in either a Portable Document Format (PDF) or text format. These summary reports can also be very helpful in troubleshooting issues.

Contacting Support

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

- Check the online help included with your Liquidware Labs Product.
- Check the Product Documentation included with your Liquidware Labs 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 Labs product and environment details such as operating system, virtualization platform version, etc.

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

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

International: + 1-678-397-0460