

---

## A Technical Overview of Citrix Application Layering

---

This document provides an overview of Citrix App layering technology, including how Citrix App layering works and how it fits into XenApp and XenDesktop environments as well as other virtualization platforms and the cloud. It will also cover a high level breakdown of how the underlying code is organized to deliver these features.

Citrix App Layering uses a single virtual appliance to manage the layers and then hands them off to other platforms for image and application distribution. This will be covered in detail in this document with the fundamental understanding that Citrix App Layering can integrate with major hypervisor and cloud platforms.

## Understanding Citrix App Layering

Citrix App Layering is a Windows Operating System and application management solution designed for on premise private clouds and public clouds. Citrix App Layering's underlying technology, called layering, enables all components of a virtual machine to be independently assigned, patched, and updated. This includes the Windows OS, applications, and user's settings and data. Built around this core innovation is a management system that encompasses application conflict resolution, image creation, application assignment, and integration technologies that can be used for any virtualization platform.

The primary goal of Citrix App Layering is to create a simple, easy-to-manage, application environment that enables anyone in IT to manage Windows and Windows apps using one interface and one seamlessly integrated technology, regardless of the underlying hypervisor or cloud infrastructure.

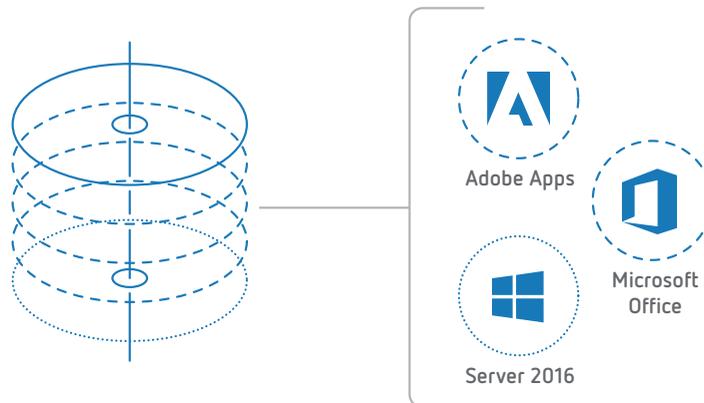
Citrix App Layering makes IT more efficient by reducing packaging and delivery times for all applications to minutes; reducing the many Windows gold images that need to be

patched in most environments to a single OS Layer that only needs to be patched once; and enabling helpdesk staff and junior IT administrators to take on everything from application packaging and updating to desktop provisioning and repair, freeing up senior IT staff for more strategic projects.

### How Citrix App Layering Works

Citrix App Layering enables IT to deliver applications that look, act and feel as if they are installed locally in the VM/Gold Image, but these applications are actually stored as separate manageable objects in their own virtual disks. With Citrix App Layering, any application can be separated from the Windows OS. As a result, IT will only have a single OS Layer to manage regardless of the number of machine configurations (pools, silos, delivery groups). This simplifies the environment while reducing management time/complexity and the costs associated with OS and app management.

The Windows OS and applications (or groups of applications) are stored as their own virtual disks (VHDs or VMDKs) and contain only the files, system objects, and registry entries for that specific layer.



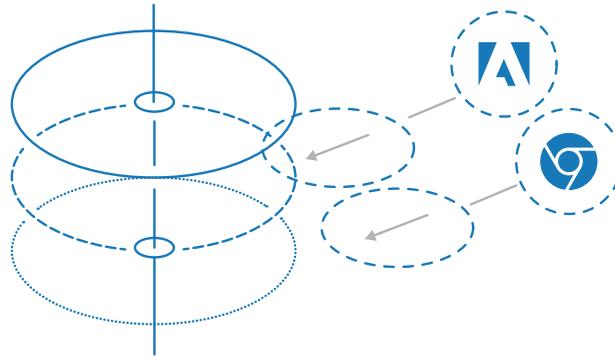
Creation of a Citrix App Layering Layered Image

By separating the applications from the operating system, and separating the personalization (unique) changes for the machine from the operating system, you create a model where IT has one copy of any given OS or application, regardless of the number of desktop or session host configurations.

Application layers can be "attached" to the virtual machine in one of two ways: 1- App Layers can be combined with an OS Layer, in a process called image publishing, and pushed to existing provisioning systems such as Citrix Provisioning Services, Citrix Machine Creation Services, or VMware View Composer, or 2- App Layers can be attached

to a VM at user login based on user AD group membership and app assignments. Each user can also receive a unique "Personalization Layer." This Personalization Layer will contain

unique information for that user that will include things like local Windows profiles, application settings, files and folders created by the user and even user-installed applications.



Adding Elastic Layers to a VM running a Layered Image

## Understanding Citrix App Layering

A Layer is simply a container for the file system objects and registry entries unique to that layer. As an example an "Application Layer" is just the files and registry entries that have been added, changed or even removed during the application installation onto an operating system.

base Layered Image and a series of virtual disks (one per Layer) mounted to the VM.

In addition to App Layers, Citrix App Layering has three other layer types: OS Layers Platform Layers, and User Layers. IT will create and manage three of the four types of layers. The OS Layer (often combined with some App Layers) are used to create a "Layered Image" that can be distributed by any image

Name	Date modified	Type
P786436.R1	2/14/2017 11:16 AM	VHD File
P786438.R2	1/31/2017 2:54 PM	VHD File
P786443.R1	2/23/2017 1:44 PM	VHD File
P786454.R1	6/29/2017 3:57 PM	VHD File
P786468.R1	2/14/2017 10:25 AM	VHD File
P786450.R1	5/9/2017 1:52 PM	VHD File
P786439.R1	5/24/2017 5:36 PM	VHD File

Individual Application Layers stored in the layer repository

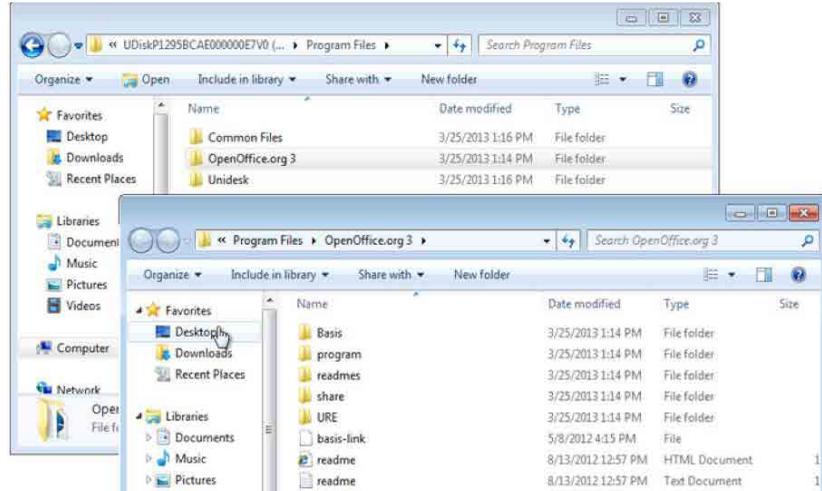
Citrix App Layering leverages a Microsoft Minifilter driver to intercept I/O requests from Windows and respond to the request from the appropriate virtual disk. Within Windows, the applications just "see" a holistic C:\ drive that looks like the apps are installed. But because of the Composite File System, the C:\ drive is in reality a logical drive that is made up of the

management system. The App Layers can also be mounted Read-only to the desktop at login based on app assignments. Finally the third layer type is the User Personalization layer. This layer is created when the user first logs in and is a unique virtual disk for each user. Each of these layer types will be covered in detail below.

## The Contents of a Layer

In the image below we have mounted the virtual disk for the OpenOffice layer to another machine and are browsing it within Windows. We can see the directory structure for Pro-

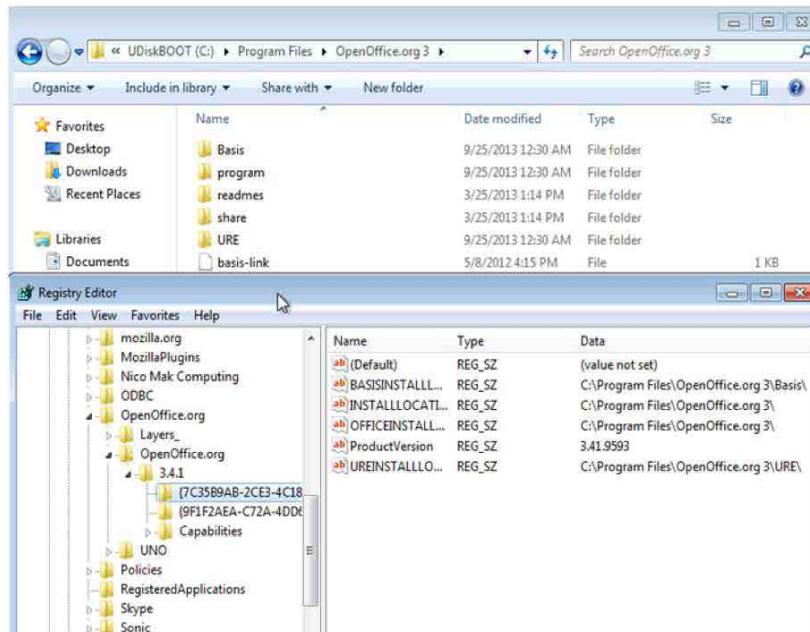
gram Files\Common Files, OpenOffice.org 3 and the Unidesk directory. If you browse into any of these directories you can see other files/directories that were modified during this application's installation.



Contents of the OpenOffice Application Layer

The “trick” with App Layering though is that when this virtual disk is mounted in the machine, the file system is blended with the other layers and presented to Windows as if

the application was natively installed. In the image below we show the contents of the C:\Program Files\OpenOffice.org layer in a running virtual machine.



Application of the OpenOffice layer to a Citrix App Layer virtual machine (registry & file system merge)

---

As you can see, not only is the file system merged, but when you open the registry you can see the registry entries for the application are also merged with the OS Layer's registry.

### Types of Layers

As mentioned above there are four types of layers with App Layering. While operating systems, application, and Platform Layers are fairly straight forward, the User Layers are often misunderstood. Layer types in this context are OS layer, Platform Layer, App layer, and User Layer, but we'll spend a bit of time on the User Layers, as they are sometimes confusing to those new to Citrix App Layering.

### OS Layers

OS Layers are the first layers created within Citrix App Layering. App Layering requires only two items in the OS image“:

- App Layering Drivers
- The host server virtual machine tools (XenTools for XenServer and VMtools for VMware and Host Integration Services for Hyper-V)

In environments where the native provisioning mechanism does not manage domain Join, machine naming, and the like, you can use an unattend.xml for mini-setup to set up domain join, type of licensing, etc..

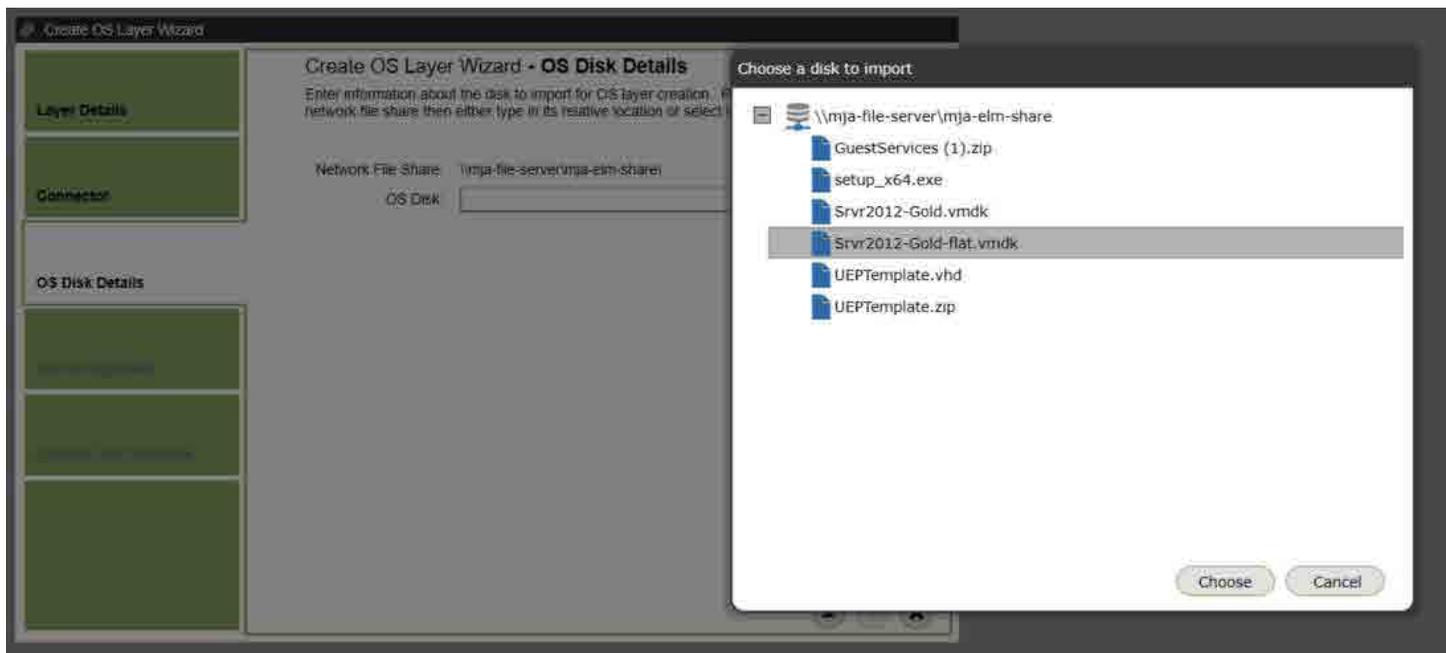
Because App Layering can layer any application, there is no need for applications to be installed on the OS image itself. The OS Layer you are creating may act as a base for numerous pools/delivery groups within the environment, and separation of apps from the OS is key to limiting the number of OS copies you have to manage. Even applications with drivers, services, and kernel devices are supported as App Layers and (with very few exceptions) should not need to be put on the OS image. The key is that the OS Layer is just a virtual disk that contains Windows, the basic VM tools/drivers and the App Layering drivers. This allows maximum flexibility in management of the environment regardless of complexity, number of desktops, or number of desktop configurations.

### Creation of an OS Layer

Creating an OS Layer is simple. Here are the basic steps:

- Build a machine on XenServer or your preferred hypervisor, with your preferred Operating System. Do not join it to a domain.
- Run Windows Update to patch the machine so that it is current.
- Install any VM tools, such as XenTools for XenServer
- For provisioning systems that require it, create an unattend.XML, which will be read at machine creation to join the domain, assign licensing, etc..
- Optionally, fine tune or optimize the OS for the virtual environment.
- Install the App Layering OS Machine Tools.
- Download the Virtual Disk used on the OS Image to the network share configured on the App Layering Enterprise Layer Manager.
- Import the OS image onto a new App Layering OS Layer.

All of the steps listed above, except for the final step, are done within the management framework for your hypervisor. Once you have installed the App Layering drivers, the final step takes you into our console to “Create an OS Layer”



Conceptual look at Layer Creation/Packaging Machine

In the above screen capture, the VMDK is the disk from the VM as it shows up in the file share. During this process, App Layering converts the original OS Image into a new virtual disk layer created for this OS. Once converted, the original VM used to create the OS image is no longer required.

The contents of the file system and registry from that original VM are copied to a new virtual disk. This new virtual disk is the actual OS Layer. As with all layers, it is a thin-provisioned disk and once the import is completed, it will be marked as read-only and can be used to create new App Layers, or to build Layered Images for integration with other provisioning systems.

### Versioning (updating) an OS Layer

Updating OS Layers is a simple operation with built-in version control. When you create a new version of the OS Layer, the latest version of the layer is copied, and this copy is marked as read-write. A special virtual machine called a "Packaging Machine" is created on the infrastructure and the copy of the OS is attached. The machine is then booted with this new writable version of the OS, and the admin can update the OS Layer as needed.

Once all of the changes are complete and any required reboots are finished, the OS version can be assigned to Image templates and updated images can be published to your image provisioning system. App Layering will create updated images with the new OS versions which can then be published to your defined targets (such as the Citrix PVS Image Store directories).

### Application Layers

Application layers are simply layers that contain the file system and registry objects for that application (or group of applications). The layers are created by the App Layering ELM creating a "packaging disk" This is a single virtual disk that contains two volumes. The first volume (also the boot volume for the VM) contains the target Operating System layer and any pre-requisite / dependent layers. This disk also contains a second writable volume where the new application will be installed.

Once the machine boots, changes made within the VM (like the installation of the application) are directed to the writable volume by the App Layering file system filter. This is not a pre-scan/post-scan model, nor is this

a block-based delta like a delta disk. These changes are captured at the File System and Registry level within Windows and redirected to this disk. One exception to this is that boot level files or registry entries are allowed to pass through to the "Read only" volume. These changes will still be saved in the layer, but allow for the machine to reboot and function as it should with applications that require a machine reboot during installation.

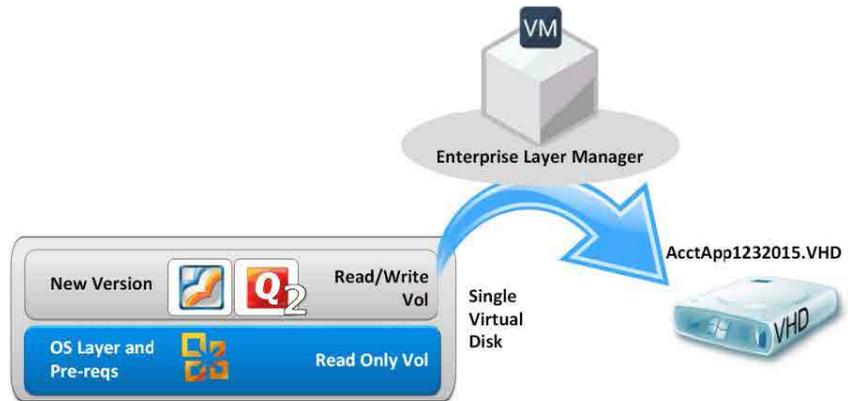


### Conceptual look at Layer Creation/Packaging Machine

The graphic above shows the conceptual components of a Packaging Machine when creating a new App Layer. The new App Layer is logically “above” the read-only OS Layer. Changes being made during the application installation are written to the new layer, while all other layers are read-only.

Packaging Machines also allow you to attach what are called Prerequisite Layers. Prerequisite Layers can be used when a new App Layer is dependent on other applications to install or function properly. A good example of this is MS Office Plug-ins. In the image

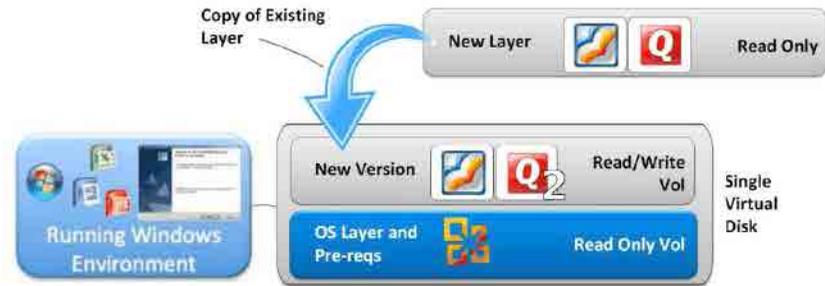
above, you can see the Windows OS Layer and a Prerequisite Layer with MS Office in the boot volume. When the machine is booted, the Windows environment will show MS Office installed on the VM. When the admin installs a new application, the changes will be written to the new App Layer, but the installation process will be able to “see” that Office is installed. This negates the need to package together applications that are dependent on one another. This model also simplifies application management by keeping organizations from including dependent applications in more than one package.



Whether versioning or creating a new layer, once packaging is complete, the layer is finalized, the machine is powered down, and the new App Layer is copied from the packaging disk into its own virtual disk in the primary Layer Repository.

### Versioning (updating) an App Layer

Version control and the ability to update App and OS Layers is a key Citrix App Layering feature. Not only does this allow admins to simplify the deployments of app and OS updates, it also allows for IT staffs to “roll-back” to a previous version if there is a problem with an update.



Logical example of layer versioning

When updating an App Layer, a copy of the existing layer is made. The virtual disk of the most current version of the layer is copied and attached to a App Layering Packaging Machine. The Admin would then update or patch the layer as needed. Once the update is complete the layer can be pushed out to users as Elastic Layers, or assigned to existing Layered Images. When applications are versioned in this way it also ensures that two different versions of the same application will not be assigned to a virtual machine simultaneously.

A note on versioning layers: A new layer version can be created for a layer when IT needs to modify the existing app installation, or the application needs to be upgraded. You can create a new App Layer for a major application version (such as moving from Office 2013 to Office 2016) but in most instances App Layers are simply versioned during upgrades.

### User Layer

The User Layer is a special layer attached to a desktop OS Virtual Machine during the logon process. The User Layer allows a user to save files, install applications, and customize a virtual machine the same way they would any other dedicated machine.



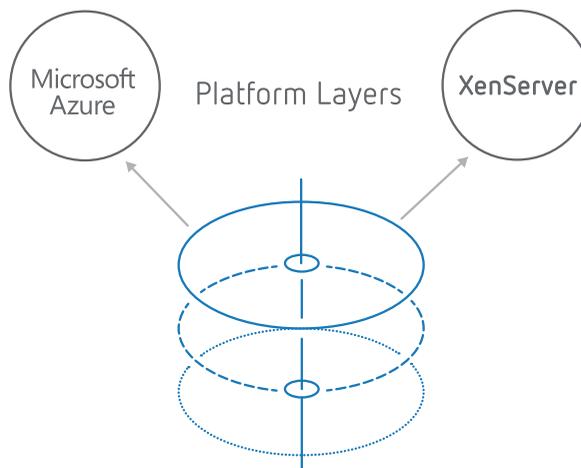
The User Layer is dedicated to an individual rather than a VM, and is the only read/write layer. When the user logs into a Virtual Machine, the User Layer is added to the system. Any changes that are made by the user that would modify the registry or file system of the machine are then redirected by our filter driver to the writable volume. When a user logs off of a machine the User Layer is removed from that machine so that it can follow a user to the next machine.

### Interactions with other layers

Since the User Layer is read/write, users cannot only install standalone apps, but they can also modify applications that have been provided by a layer. A common example is when a user modifies the dictionary for Microsoft Word that has been provided by the Office Layer. A copy of the dictionary is made and placed in the user's read/write Personalization Layer. The user can then modify that dictionary as they see fit. The dictionary in the User Layer will have a higher priority than the one in the Office layer. Layer priority will be covered more in-depth in a later section.

### Platform Layers

A Platform Layer is a special layer that contains all of the information about a particular hypervisor, the provisioning service to be used, and the connection broker. Since App, OS, and User Layers contain OS-specific information, it is very easy to use the concept of a Platform Layer to move other layers between different hypervisors.



Logical example of layer versioning

Platform Layers enable an administrator to update applications and operating systems one time, and distribute them to multiple sites. It doesn't matter if the sites are internal XenServer based environments, or if one is an on premise XenServer and the other an Azure cloud DR deployment, all deployments will use the same base layers in Citrix App Layering.

## Understanding File System Layering

Start with the knowledge that the C:\ drive is just a virtual concept. Whether you are running a physical laptop, a server, or a virtual machine, C:\ is just a logical assignment to whatever physical or virtual "drive" Windows can see at the hardware level. The logical drive contains a hierarchy of folders and files to enable Windows to boot, services and applications to run, and users to interact with them.



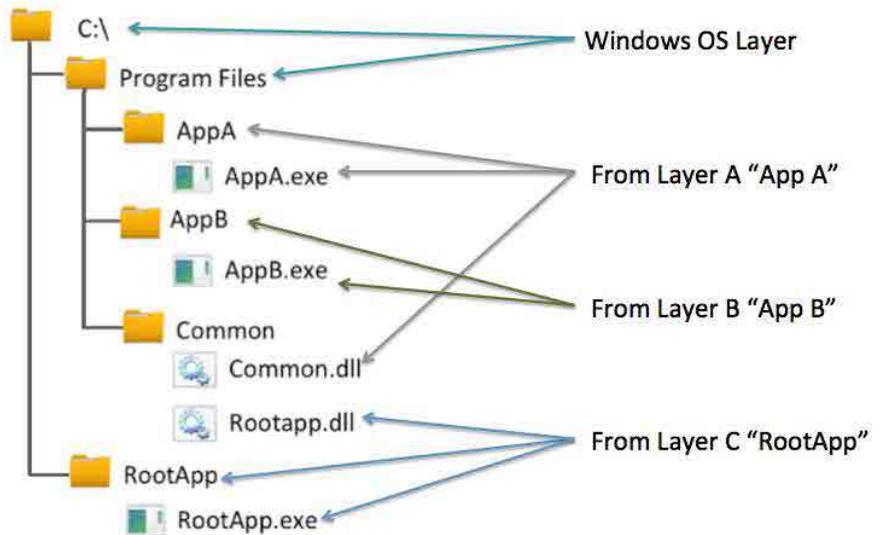
Remember that to layer a C:\ drive is really to create a logical C: drive from a series of Layers, either in a single virtual disk (as in the case of Layered Images), or series of individual virtual disks that are added to the session

at login as additional logical volumes merged into C:\ (Elastic Layers).

Understanding that App Layering layers are stored in separate virtual disks, you can then begin to understand that a Virtual Machine may have several virtual disks attached to it or may run from a single virtual disk that contains the contents of many layers. In any case, the C:\ drive that the machine sees is really made up of one merged view of all these layers. We will get to how this is accomplished shortly, but let's understand the results of this merging first.

Starting with the image above, let's assume that these three layers make up a desktop. Each layer contains one application that has only two files: an EXE and a DLL.

When these layers are assigned to a VM, they are logically merged to make the applications look like they have just been installed into a "regular" C:\ drive, but underneath they actually reside in separate VHD files.



Within Windows, if you were to browse the C: drive in explorer, you would see a structure like the image above. The files and directories that come from different layers are merged together. Program Files is a great example of this layer merging. You can see AppA, AppB and the Common folders from numerous layers. You may also notice that in the Common directory there are DLLs from different layers. This is not simple Windows mount points in a directory. This is a function of file system level virtualization, and the blending of different file systems.

The file system virtualization is based around App Layering conflict resolution logic and a Microsoft file system mini-filter driver. Logically, all of the "magic" lives at the file system. The layers themselves are NTFS within their virtual disks. App Layering is virtualizing the name space for the files system to intercept I/O requests for files and direct them to the proper, layered file system.

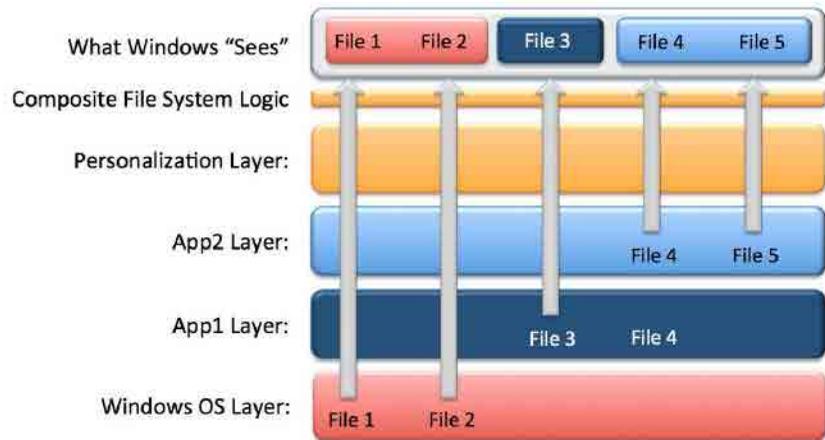
As an example, if you double-click a shortcut for AppA.exe (above), the call to the file system actually goes to our mini-filter, which passes the call to the proper virtual disk. If that app then calls AppB, the same process occurs again for the new files being sought, in this case AppB.exe and possibly its DLL(s).

### Understanding Layer Priority

In the example used in the previous section you may have noted a conflict between AppA and AppB. Both of these layers had a Common.DLL that was placed in the C:\Program Files\Common directory. The conflict was resolved by the system in such a way that AppA layer's DLL is taking priority over AppB and being presented to Windows.

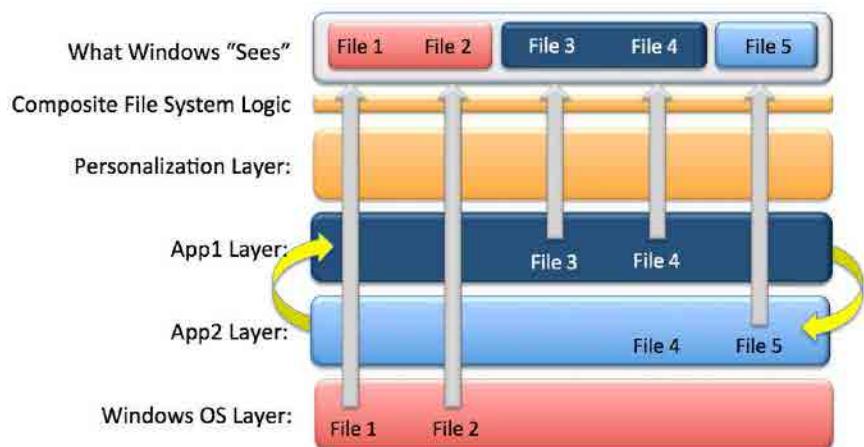
This priority mechanism begins at layer creation and is based on the order in which the layers are created. When Windows views these layers, it is from a top-down model where the highest priority wins. So if a file (or registry entry) exists in two layers, but only one can be presented to an executing Windows environment, the layer with highest priority "wins".

Before you dive into priority it is important to note that the Personalization is always "on top," or the highest priority, and the OS Layer or Layered Image is always "on the bottom" or the lowest priority. App Layers are what receive specific priorities relative to each other and not to the OS or User Layer.



In the image above we can see a total of four Layers: the User Layer, OS Layer, and two App Layers. The conflict here is between App1 and App2 with regards to "File 4". In this default priority, "File 4" from App2 "wins" and is pre-

sent to Windows. But, let's assume there is a problem and we need to expose "File 4" from App1 to the Windows environment.



This is where layer priority overrides come into play. The IT admin can adjust the priorities so App1 is a higher priority than App2. Thus "File 4" from App1 is presented to Windows.

### Object Delete Tokens

Understanding how items are "deleted" in a layered world is not as simple as it seems. In order to fully understand it we have to create a situation where App Layering delete tokens

can be understood. Let's use a corrupt or missing file from a user's desktop:

Let's assume a user corrupts or overwrites a file on their desktop. The file in question was from an App Layer. At this point, can't you just delete the offending file from the User Layer since, because of layering, if the file doesn't exist in the User Layer, the lower level layer's file should "show through" to Windows?

The answer is NO. Just deleting the file from within Windows in the desktop or session does not allow the file in the lower level layer to just pass through. And this is because of something called a Delete Token.

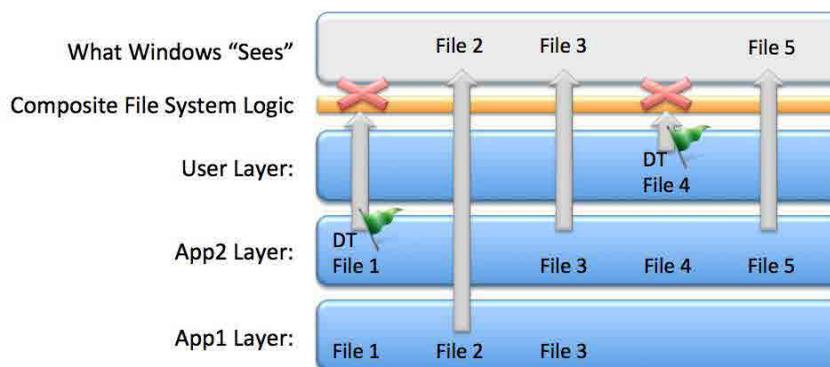
In the situation described above, we are articulating this as a “problem” to be solved. We need to delete the file from the User Layer so the lower level layer shows through. And while this is possible in our interface, it is actually done by the App Layering system and not by deleting a file from the User Layer. Here is how this works.

Whether the file is just a flag that is deleted when the file is launched by the app, or it is a script that runs the first time a machine is booted and then is deleted, or it is a function within an app that you configure by removing the files or registry entries, or it is a shortcut that is in the All Users\Desktop folder that a user deletes and doesn't want to come back, the layering system understands deleted objects vs objects that don't exist in a layer. App Layering does that by using something called a Delete Token. Here's how Delete Tokens work. Say you have two App Layers and a User Layer, all with delete tokens in them.



In this scenario, the App2 Layer deleted “File 1” when it was created as a new App Layer. Then, somewhere in the process of using the application or desktop, “File 4” was deleted in the User Layer. While the file is “deleted”

and gone from view in both cases, what really happens is that a Delete Token is left in its place. This tells the Composite File System that the file is not there, and the running Windows environment “sees” this:



Here the Delete Token from the App2 Layer has “deleted” File 1 and the Delete Token from the User's User Layer has “deleted” File 4. In the event that App 1 was given an increased

priority (stacked higher than App2), File 1 would then be presented to the Windows environment, as the Delete Token in App2 would be lower in the layer conflict resolution logic.

---

## Understanding Registry Layering

When you interact with the registry, the hives you see, such as HKLM\SYSTEM or HKLM\SOFTWARE, are stored in files on the C: drive. There are also some registry hives that are volatile and are created purely in memory, but the major ones you see, and we interact with, all have files associated

Files associated with registry hives:

```
HKEY_LOCAL_MACHINE \SYSTEM
\system32\config\system
HKEY_LOCAL_MACHINE \SAM
\system32\config\sam
HKEY_LOCAL_MACHINE \SECURITY
\system32\config\security
HKEY_LOCAL_MACHINE \SOFTWARE
\system32\config\software
HKEY_USERS \UserProfile
\winnt\profiles\username
HKEY_USERS.DEFAULT
\system32\config\default
```

Knowing how App Layering merges the file system, you quickly realize that the registry files can't simply be replaced and presented to the C: drive like other files. If that were to occur, a file from one layer would "win" and you would only have registry entries from a single layer, when you actually need them from the OS and all of the App Layers.

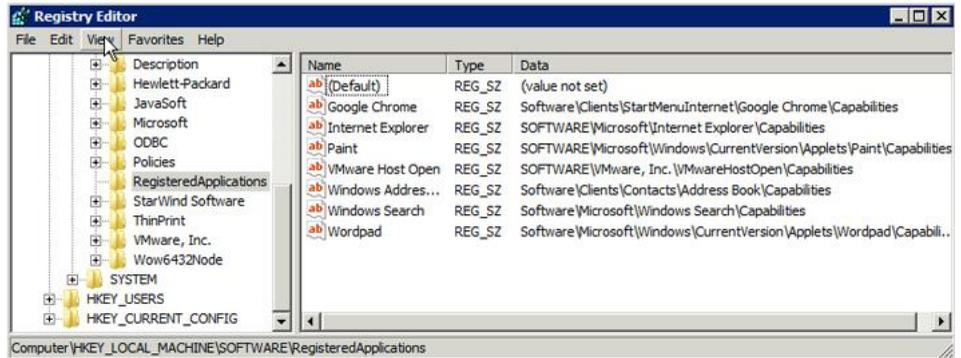
To solve this problem, App Layering uses what is called a Composite Registry. Registry merges are handled based on how the layer is being deployed. If the layers are being deployed into a Layered Image, say for deployment by a provisioning system like Citrix PVS, App Layering will actually build these files with the contents of the registry changes from each layer merged into the proper registry files. The result is that the registry in the image is a complete registry merged from registry components of each assigned layer. A benefit of creating the registry this way is that the files are there pre-boot, meaning services and drivers all work as expected, regardless of which layer these hard-to-virtualize items are located in.

### How elastic layering loads a registry

When an elastic layer is added to a virtual machine, the registry hive is loaded in from the layer on the layer repository. The hive is then mounted in memory. Once the hive has been mounted, the operating system can access the registry data as if the application had actually been installed on that virtual machine.

### How the registry merging is like and not like the file system merge

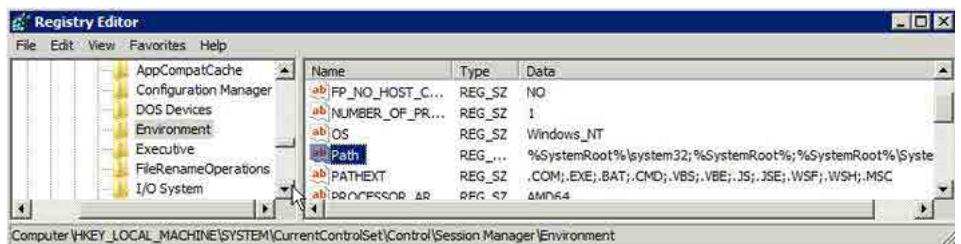
The registry merge process is not unlike the App Layering composite file system. Objects from different layers have to reside in the registry in the proper location. Below is an example of how layers show up in HKLM:



Here in HKLM\SOFTWARE\RegisteredApplications you can see registry entries from both Google Chrome and Internet Explorer. In this example the registry entries are contained in the file “\system32\config\software,” but because App Layering merges inside the registry and not just the registry files, you can see results here loaded into Windows. Internet Explorer entries are from the OS Layer, and Chrome is its own App Layer.

### Registry merge that is not simple override

The App Layering process for building the Composite Registry is actually much more complex than the example above conveys, and it is more complex than the file system merging process. This is because the registry uses things like Multi-String (REG\_MULTI\_SZ) Values, where registry virtualization cannot be a simple replacement based on priority. Let’s use the example of the entry for System Path variable shown below:



If multiple App Layers have modified the System Path by appending extra directories to it, rather than a simple overwrite of the entire Path value, intelligent registry composition is required to append the changes to a Multi-string registry entry. This registry-aware intelligence is critical throughout the Windows world. Some of the key intelligence in App Layering is proprietary, but we can share a few examples of how our composition model impacts your desktops.

- In .NET applications, Microsoft relies heavily on Fusion keys, and their index values, to execute the applications. If you layer .NET applications, App Layering will intelligently merge their information, which allows apps like Microsoft Office, Office Plug-ins, Visual Studio, Visio and Project to be layered separately, yet still function on a single desktop.
- App Layering allows drivers to be installed in different layers by merging the Windows Driver store and placing the references to the drivers in the registry. This allows you to layer as many apps as you want with drivers and not have to place them in the OS image or a specific “drivers layer”.

## Understanding How App Layering Layers Are Deployed

The contents of a virtual machine using App Layering Layers is a composite of layers that provide the operating system, applications, and personalization data. App Layering can create Layered Images that can be deployed via existing image management tools, such as Citrix PVS and View Composer, and also supports more App Layers, attached at login, based on user identity. In this section we will review the unique layer deployment characteristics and processes.

App Layering supports two models for layer deployment in Windows-based virtual machines. The first is a Layered Image, where a virtual disk image is created from a combination of OS and App Layers. This image is then used by an image provisioning system, such as Citrix Provisioning Server or VMware View Composer, as the base image for a pool or catalog of VMs. The second supported model is the dynamic assignment of an App Layer at login. This is accomplished by Layering software that presents the application to the users' desktop or session, based on AD credentials and group membership.

## Supported Operating Systems

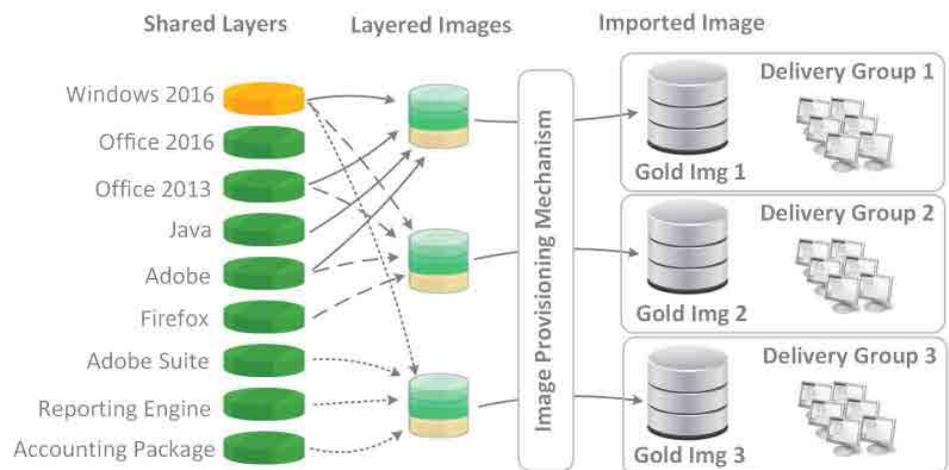
Citrix App Layering is a Windows-only platform. While Linux is used for the virtual appliances, the OSes supported are focused on Windows applications, and therefore VDI and XenApp/RDSH environments.

Currently supported managed operating systems in App Layering 4.x include:

- Windows Server 2012 R2 64-bit Datacenter – As a Single User Desktop or Session Host/XenApp Server
- Windows Server 2008 R2, 64-bit Datacenter – As a Single User Desktop or Session Host/XenApp Server
- Windows 7 64-bit & 32bit
- Windows 10 64bit

## Layered Images

A Layered Image is a collection of shared layers composited into a single virtual disk. This concept allows the IT admin to manage a single copy of any App or OS Layer while still providing numerous image configurations to their users.



Using the example above, we have 3 different Citrix Delivery Groups. All three delivery groups are using Windows Server 2016, but have differing application requirements. In a Layered Image model, only a single copy of the OS is maintained and patched. This OS Layer is then combined with the required App Layers to create a Layered Image. The admin

would select the OS and Apps, then select the provisioning mechanism (in our example Citrix PVS). The image is output from App Layering as a Single VHD file that contains all selected layers. That VHD is automatically copied to the PVS Store where it can then be applied to individual devices or collections. In our example, this reduces the Citrix admin's workload

by at least two-thirds, as his OS and applications only have to be updated once across the environment, versus once for every image and configuration they are supporting.

### Layered Image Creation

When a Layered Image is published to a provisioning system, App Layering will create

a single virtual disk containing all layers assigned to that image. This is done by simple calls to the hypervisor to create a VM and a virtual disk to which layer contents can be copied. Once the VM and target virtual disk are created, the Layer Manager will create the Bootable Image. It does this through the following multi-step process.



During this process, the Layer Manager virtual appliance attaches to the target virtual disk, formats it, and then copies the required files from the Windows and App Layers to the disk. It also copies the base registry from the Windows OS Layer and reads the changes to the registry from each App Layer into the composite registry on the Layered Image.

The iteration through the layers for both files and registry entries is done in layer priority order, from the lowest priority to the highest. Once the Layer Manger has copied the files and registry settings needed, the virtual disk is detached from the appliance and copied to the target location, based on the settings the administrator has configured in the platform connector.

### Elastic Layers

The Elastic Layer architecture has been designed for enterprise environments that require maximum flexibility, scalability and recoverability. To that end, App Layers that

are published in this model require no infrastructure between the user's session (VM) and the layer being delivered. In most layering solutions, the client software installed in the OS will connect to some sort of management plane or server, that will then make the appropriate calls to a database, the hypervisor, or other management tools, to mount the assigned layers for the user. This architecture can make things like replication for DR or scaling and even use in hybrid environments (on-prem and in the cloud) extremely difficult, as all this infrastructure must be made redundant and replicated, not to mention, it often requires that you run identical hypervisors on all sites and bars the use of the layers in cloud or hybrid environments.

In an App Layering environment, the Layering Services that run in the target VM are configured with a simple UNC path to what is called the Layer Repository. This repository contains the layers and their self-describing information required for use. The Layering Services will simply read a pair of control files that are located in the UNC path, and then mount the appropriate layers assigned to the user. This is designed to work within the guest operating system using native Windows mounting and file share technologies. And it creates an environment where IT Architects can replicate the share in a single location for maximum scalability, replicate the share to a secondary site for active use or DR purposes, or even replicate or move this share to a cloud environment. Leveraging software in the native Windows OS eliminates dependencies on the hypervisor and removes all needs to replicate

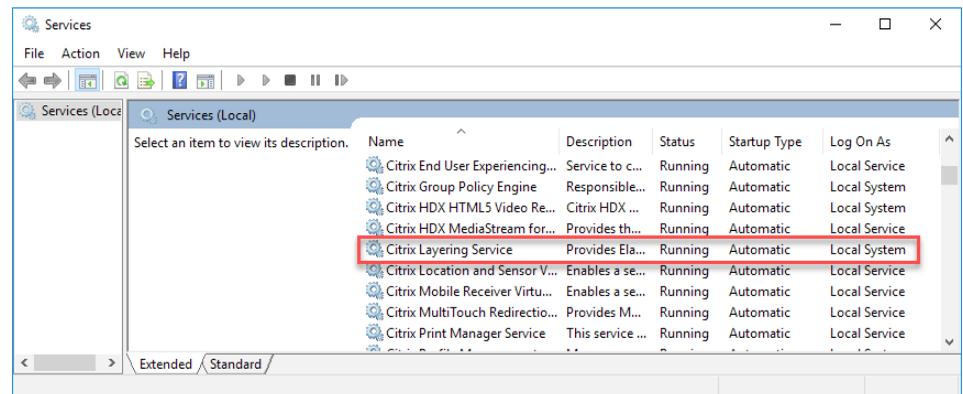
databases and management servers in multiple environments.

## Components

To understand this better, we can look at the actual layer repository and how the Guest Layering Services interact with its components.

## Guest Layering Services

The Layering Services you see below is a simple application that can be placed in a Layered Image automatically, or deployed via a separate process. It is this service that actually deals with enumerating authorized App Layers and initiating their addition to the desktop or session.



The Layering Services will read configuration information out of the registry (under HKEY\_LOCAL\_MACHINE\SOFTWARE\Unidesk\ ) to find the location of the SMB share that holds the elastic layers and elastic layer policy database. Once the Layering Services locates the share, it will then read two files in the share that allow it to mount and present the layers into the session, assuming the user has read access to the share.

The ElasticLayerAssignments.json contains the information about user and group mappings to individual App Layers. Each mapping is essentially a single json object that links an AD object (user or group,) to a specific Unidesk application revision. At login the user's AD user sid and all group memberships are used to pull out any assignments from this file.

## Control Files in the Layer Repository

Once the Layering Service has located the layer repository, it then reads the following files, in order:

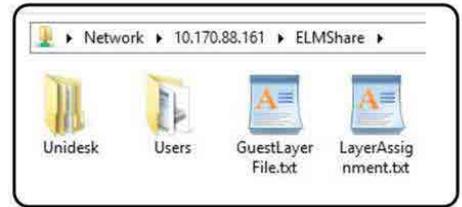
- ElasticLayerAssignments.json
- Layers.json

Whenever there is a conflict (say a user is in the marketing group, which is assigned revision 3 of the Office layer and also a member of the Sales group, which is assignment version 2 of the Office layer, the highest revision wins.

The Layers.json file is a simple mapping between Layers and UNC path to those layers.

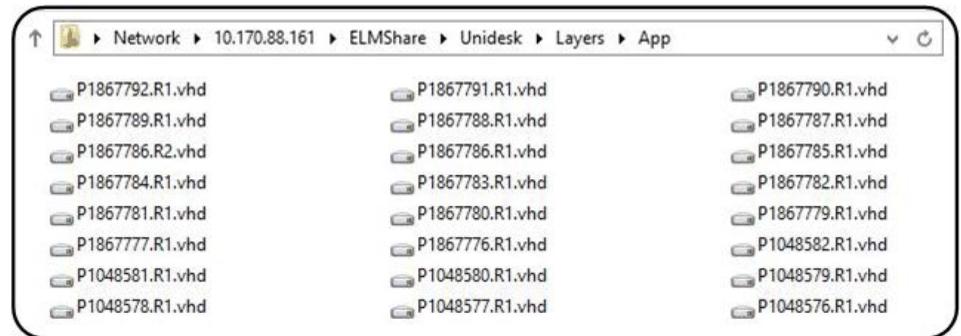
### Layers

The layers themselves are simple VHD files located in a subdirectory in the Layer Repository.



From the root of the Layer Repository you can see the following:

- Unidesk directory – contains the shared layers
- Users directory – contains user personalization disks
- The control files of GuestLayerFile.txt and LayerAssignment.txt described early in this section

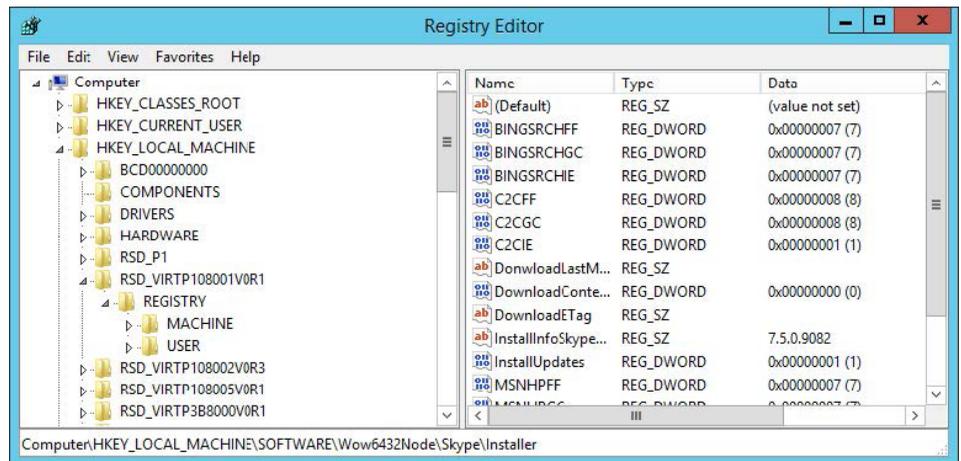


The App Layers are located in the Unidesk\Layers\App directory. Each VHD maps to a specific App Layer and version of the App Layer. It is this directory and the contents of the root of the share that can be replicated for large scale and DR environments.

Once a layer has been mounted in the VM the entire registry hive is loaded into the registry of the VM under HKEY\_LOCAL\_MACHINE. For each application that is present there will be an entry that starts with "RSD\_VIRTP".

### Elastic Layer at-login process

The entire Elastic Layering process takes place over the course of seconds when a user logs into a desktop. As the user logs in to the VM, the LayerAssignment file is read, and the layer assignments are located for that user. Once the layers have been located, they are mounted inside the guest.



Inside each of these keys is where you will find the regular registry data that gets used for applications. When a registry value is accessed, it is redirected to the appropriate RSD\_VIRTP key instead of to the location where that application would normally live in the registry.

The virtual disk is then made available as part of the App Layering Clustered Filesystem. An end user can click on an executable and launch the program.

### Session Based Elastic Layers

Session based elastic layers work just like normal elastic layers, with one exception. They are restricted to a particular user. No other user logged into that session host can see the application unless they have been authorized for it by the App Layering Administrator.



Multiple Users accessing different session based layers.

Since multiple users can be logged into a session host, the first thing that layering services will do is check to see if the requested layers are already present on the VM. If the layer is found, that user is simply “authorized” to see the registry and file system data. Once the user is logged in, they will see that application, just as other authorized users see it. When a layer is not already available on a session host, it is added during the logon process the same way it would be during a desktop logon. The only exception to that rule is that only the logged on user will be able to see that layer. No one else will be able to.

When a user logs off from a session host, the applications associated with them are left on that host. The assumption is that there could be other logged on users who are accessing that data. If for some reason a layer must be removed from a VM, the administrator will have to wait until all users are logged off and the session host will have to be rebooted.

## Elastic Layers

The App Layering management components for OS and Applications are deployed as virtual appliances. No external servers or physical hardware is required. This section describes the App Layering appliance and functions, and how App Layering communicates with the environment into which it is deployed.

### Enterprise Layer Manager (ELM)

The App Layering virtual appliance that houses these management components is called the Enterprise Layer Manager, or ELM. The Layer Manager hosts the web management interface, a small configuration database, and the layer logic engine. The ELM is also responsible for storing layers on the layer work disk and copying Layered Images via the Platform Connector to targeted provisioning systems. Depending on the broker

and third party provisioning system used, a single Layer Manager can support tens of thousands of virtual machines and thousands of App Layers.

The administrator interacts with the Layer Manager, which directs activity in an App Layering environment. The Layer Manager also maintains a master copy of all IT-created layers. These layers are then used to create Layered Images or used as elastic layers and are hot added into running desktop sessions. These layers are stored on a 200 GB work disk attached to the ELM.

This model allows App Layering technology to scale as large as the supporting infrastructure and provisioning systems are designed to. App Layering does not provision or update machines from this appliance, so it is not a single point of failure nor a limit to scalability.

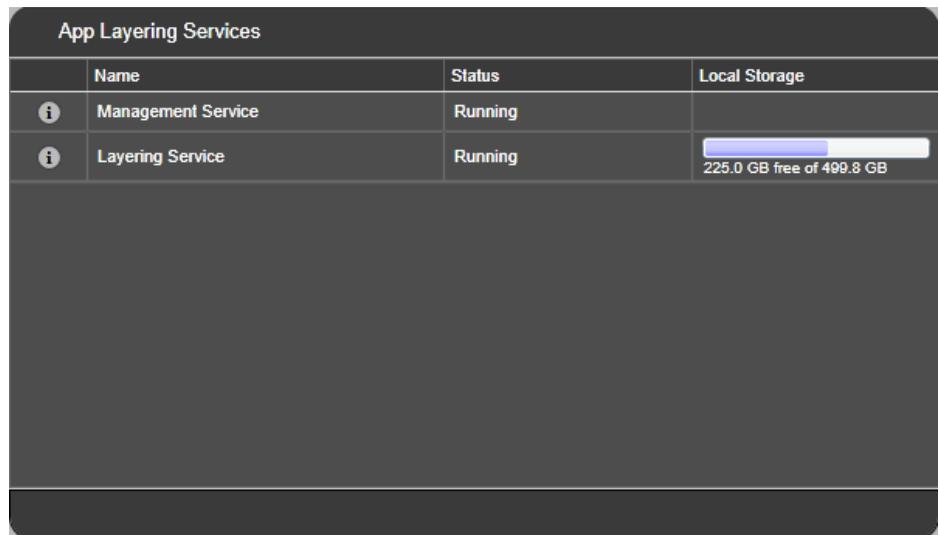
Component	Description
Enterprise Layer Manger	The primary App Layering virtual appliance used to manage the following components: <ul style="list-style-type: none"> <li>• Operating System Layers</li> <li>• Application Layers</li> <li>• App Layering Platform Connector Engine and individual Platform Connectors</li> <li>• Layer Work Disk</li> </ul>
Platform Connector	The software that controls the connection and configuration Unidesk will use to build layers and export Layered Images.
Layer Repository	The ELM is configured with a UNC path to a share that will act as a Layer repository. This is where elastic layers are stored. Layered Images can also be deployed here.

## The Layer Work Disk and the Remote Layer Repository

### Layer Work Disk

All the Layers created on the ELM are stored on the layer work disk. This disk is attached to the ELM and is part of its local Linux-based file system.

The Layer work disk will also be used as a “scratch disk” when the Layered Images are put together. After they are assembled, they’ll be placed on the provisioning platform of your choosing. The ELM will keep track of the space used, so that additional storage can be added to the system, if needed.

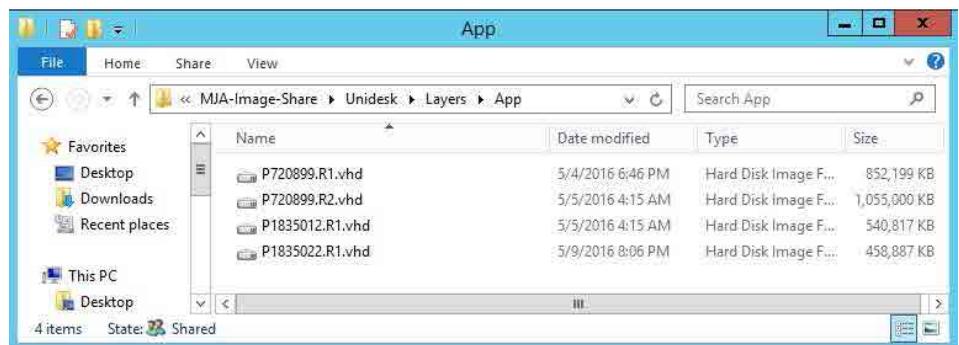


App Layering Services			
	Name	Status	Local Storage
	Management Service	Running	
	Layering Service	Running	 225.0 GB free of 499.8 GB

### Layer Repository

The Layer repository is an external network share where Elastic layers are stored for distribution. When a layer is elastically assigned to an AD user, a task kicks off to sync the layer work disk and the layer repository. During this operation the layer is copied to the repository and the LayerAssignment.txt file is updated.

Regardless of what platform the layers will be delivered to, they will always be stored in the .vhd format. These vhd files are mounted internally on the guest VM. The Layer Repository can also be used as an export target for Layered Images, if no other connector is configured.



---

## App Layering Integration with the Provisioning Tools and Hypervisors (Platform Connectors)

To distribute our Layered Images using various provisioning tools, App Layering communicates with the underlying hypervisor to create layers and or deploy images. Each hypervisor uses different APIs for these tasks, in addition to the various available APIs for different provisioning tools (such as Citrix PVS, or VMWare View Composer). This means that each connector for publishing images is specifically is written to support both a hypervisor AND a provisioning system. As an example you may see a single system with two connectors: one for Citrix MCS on vSphere and View Composer on vSphere. While the code is similar for hypervisor operations, each provisioning system has different requirements for the underlying disk, the OS preparation state, and the configuration of the VM itself (if used).

In this model there are two distinct types of platform connectors:

### Platform Connectors for Layer Creation

A platform connector used in layer creation allows for Unidesk to communicate with the underlying hypervisor. During this process the connector will create a virtual machine on the target hypervisor or cloud service, allowing you to install a particular application. For some hypervisors, App Layering doesn't yet include connectors, but the network share configured during initial setup will always be available to export a disk.

### Platform Connectors for Image Publishing

A platform connector used in image publishing is used slightly differently than the one used in layer creation. App Layering does not include methods for provisioning virtual machines but instead integrates other technologies, such as Citrix PVS, Citrix MCS and VMWare Composer. The platform connector used in image publishing is used to communicate with those services. When publishing an image you will select the platform connector you are going to use, and App Layering will composite an image and send it directly to that service, with no other action required. As with the layering connector, the imaging

connector always has the network share as an option so that an image VHD can be moved to a platform for which there is no automated connector yet.

### Supported Platform Connectors

Currently automated Platform Connectors are available for the following platforms and deployment systems:

#### Hypervisor

- Azure Resource Manager
- Citrix XenServer 6.5, 7.0, 7.1, 7.2
- Microsoft Hyper-V, Windows Server 2008 R2, Windows Server 2012 R2, Windows Server 2016
- Nutanix Acropolis
- vSphere vCenter 5.5.x, 6.0.x, 6.5.x

#### Image Publishing

- Citrix MCS for Nutanix AHV
- Citrix MCS for vSphere
- Citrix MCS for XenServer
- Citrix PVS 7.1, 7.6 - 7.9, 7.11 - 7.14 with recommended network speeds to the PVS Store of 10 GB.
- Citrix XenApp and XenDesktop 6.5, 7.0 - 7.14
- Microsoft Azure, with recommended network speeds to the Azure publishing location of 10 GB.
- VMware Horizon View 6.x, 7.0.x, 7.1.x

#### Operating System for Layered Images

- Windows Server 2016, 64-bit (Standard and Datacenter Editions)
- Windows Server 2012 R2, 64-bit (Standard and Datacenter Editions)
- Windows Server 2008 R2, 64-bit (Standard and Datacenter Editions)
- Windows 10, 64-bit (Education, Enterprise, and Professional Editions)
  - Supported versions: 1511, 1607
  - Unsupported version: 1703
- Windows 7, 64-bit (Enterprise and Professional Editions)
- Windows 7, 32-bit (Enterprise and Professional Editions)

#### Directory Service

- Microsoft Active Directory

#### Internet Browser

- Internet Explorer v11
- Firefox v45 and later versions that support MS Silverlight 4.0.

---

## Glossary of App Layering terms

The following table provides definitions of terms that are specific to the App Layering product.

Unidesk Term	Definition
App Layer	A virtual disk layer containing one or more applications that you can use in any number of Layered Images. When publishing a Layered Image, you can combine an App Layer with the OS Layer used to create it, other App Layers, and a Platform Layer.
Enterprise Layer Manager (ELM)	A virtual appliance that coordinates communication in the Unidesk environment, and hosts the Unidesk Management Console (UMC), the administrator interface for the Unidesk environment. The ELM also manages copies of all Layers.
Elastic Layer	An App Layer that is attached at logon to a XenApp server session or VDI machine based on user criteria. This allows simultaneous use of different applications from the same XenApp server as well as VDI flexibility that ultimately leads to resource consolidation and improved efficiency.
Layered Image	A combination of an OS layer plus one or more platform and/or one or more app layers that are merged together to form a standard bootable virtual disk that can be used with Citrix Machine Creation Services (MCS), Citrix Provisioning Services (PVS), or another mechanism from a third party.
Platform Layer	A virtual disk layer that includes configuration settings, tools, and other software required for Images to run on a particular platform. For example, a platform layer for XenServer would include the XenTools software.
Platform Connector	A stored set of values for connecting to a specific environment. A configuration typically includes credentials for authentication, a storage location, and any other information required to interface with the environment where you will be creating layers or publishing images.
Packaging Machine	A virtual machine that acts as a staging area for the creation of App Layers, App Layer Versions, and OS Layer Versions. The Packaging Machine is booted from a Packaging Disk using the credentials and location specified in the selected Connector Configuration.
Management appliance	See Enterprise Layer Manager.
Layer Repository	A Layer storage where the ELM creates, composites, and stores Layers and Layered Images. Local storage is used for temporary files during the creation of Layers and Layered Images, and for persistent files, for example, Layers and Image Templates. Administrators can define the Network File Share location that will be used for Elastic Layers in the UMC's System and Settings.
OS Layer	A virtual disk layer containing the operating system. You can use an OS Layer with any compatible Platform and App Layers in any number of resulting Layered Images. You can create a new version of the OS Layer for every patch you need to roll out, and continue deploying all versions of the layer as you add patches.
User Layer	A writeable layer for use with Windows 10 or Windows 7 VDI instances that allows users to install applications in non-persistent environments. This provides user flexibility with administrative control in situations where dedicated VDI would traditionally be required.



**Enterprise Sales**

North America | 800-424-8749

Worldwide | +1 408-790-8000

**Locations**

**Corporate Headquarters** | 851 Cypress Creek Road Fort Lauderdale, FL 33309, United States

**Silicon Valley** | 4988 Great America Parkway Santa Clara, CA 95054, United States

© 2017 Citrix Systems, Inc. All rights reserved. Citrix, the Citrix logo, and other marks appearing herein are property of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered with the U.S. Patent and Trademark Office and in other countries. All other marks are the property of their respective owner(s).