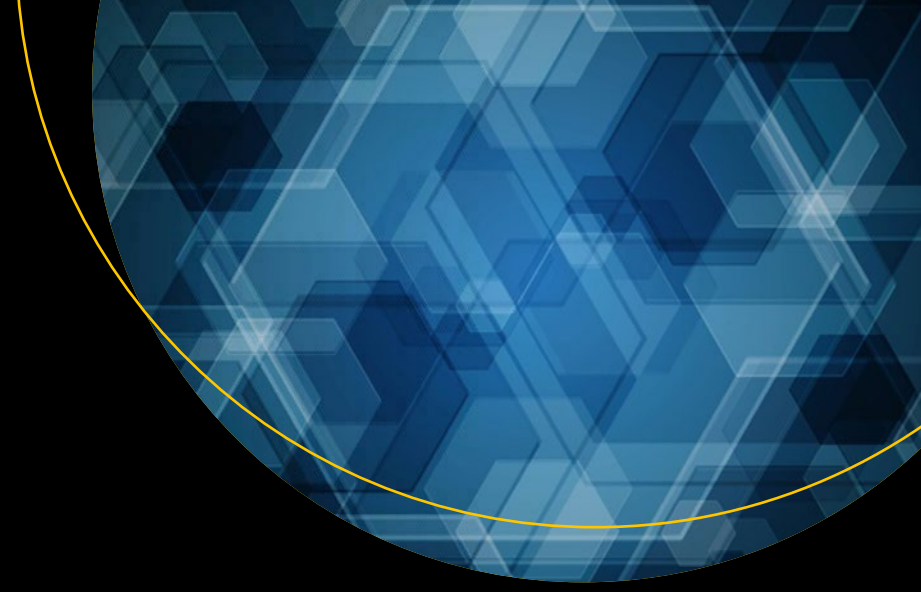


Delivering Applications with VMware App Volumes 4

Delivering Application Layers to Virtual Desktops Using VMware

Peter von Oven

Apress®

Delivering Applications with VMware App Volumes 4

Delivering Application Layers to Virtual Desktops Using VMware

Peter von Oven

Apress®

Delivering Applications with VMware App Volumes 4

Peter von Oven
Wiltshire, UK

ISBN-13 (pbk): 978-1-4842-6688-5 ISBN-13 (electronic): 978-1-4842-6689-2
https://doi.org/10.1007/978-1-4842-6689-2

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Matthew Moodie
Coordinating Editor: Aditee Mirashi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

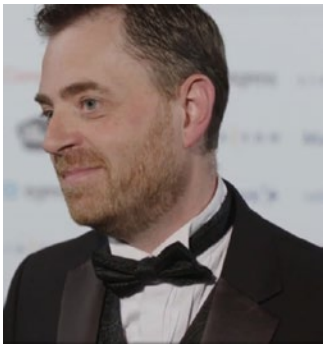
Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6688-5. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my family, for their continued support and for believing in me.

Table of Contents

About the Author



Peter von Oven is an experienced technical consultant and has spent the past 25 years of his IT career working with customers and partners in designing technology solutions aimed at delivering true business value as well as authoring 14 books on VMware EUC technology solutions. During his career, Peter has been involved in numerous large-scale enterprise projects and deployments and has presented at key IT events, such as VMworld, IP EXPO, and various VMUG and CCUG events across the United Kingdom.

He has also worked in senior presales roles and presales management roles for some of the giants of IT, such as Fujitsu, HP, Citrix, and VMware, and has been awarded VMware vExpert for the last 6 years. More recently, he has become part of the VMware EUC vExpert program. Outside of work, Peter volunteers his spare time as a STEM Ambassador, helping, coaching, and mentoring young people in taking up careers in technology.

About the Technical Reviewer

Darren Hirons is a Senior End User Computing Solutions Engineer with VMware, specializing in Horizon and Workspace ONE technologies.

Prior to joining VMware, Darren has held a variety of IT roles in the public and financial sectors and has 25 years of operational experience.

Acknowledgments

This is, amazingly, the fourteenth book that I have written, and I would like to say thank you to Apress for giving me the opportunity to write with them. Even after having written several books, the whole process of planning and writing a book still presents a huge challenge. Given the current situation we all find ourselves in at this time of writing, authoring this book has come as a welcome distraction.

I would also like to acknowledge and thank the people who have helped with preparing this book. Thanks to Aditee at Apress for making sure I kept on track and answering questions about the process, and a really big thank you to Darren Hirons at VMware for not only providing his technical editing skills but for also helping with some of the questions and support issues I had along the way.

Finally, I would like to acknowledge you, the reader. Without you, this whole project would not be worthwhile, so again, thank you. I would also love to hear from you any suggestions or questions regarding this book or any other end user computing solutions book for that matter. You can find me on Twitter at @pvo71.

Introduction

"Deliver apps to virtual desktop environments in seconds, and at scale with the click of a button."

App Volumes delivers applications in real time to virtual desktop machines enabling VDI deployments to return even greater flexibility, agility, and cost reduction. Enterprises can now fully utilize the nonpersistent, floating desktop model in all VDI use cases, and before, where users such as developers required a persistent virtual desktop of their own, they can now take advantage of the lower-cost nonpersistent delivery model.

This book will focus on how to get started with VMware App Volumes and how to deliver applications to virtual desktop machines and RDSH-based application publishing solutions. In this book, we will not only look at VMware's own virtual desktop and app delivery solutions with Horizon View and Horizon Apps but also Microsoft RemoteApp and RDSH.

Throughout this book, we will work through the solution to enable you to design, install, configure, and manage an App Volumes deployment, using step-by-step instructions with real-life screenshots as you follow the test lab that is used throughout the book to demonstrate each key feature.

Starting with an in-depth overview of where the solution fits within the market, and its key features, we will then move on to explaining the architecture and components and then look at how to design an optimized solution. The next phase of the title is to start installing and configuring App Volumes for the different use cases such as VMware Horizon View, Horizon Apps, VMware ThinApp, and Microsoft RDSH.

Throughout the chapters, you will be given hints and tips, along with best practice, all seen from the eyes of somebody who works with this technology day in, day out, and in many different types of environments and scenarios.

By the end of this book, you will have acquired the skills to build an App Volumes environment for a proof of concept, a pilot, or in a live production environment.

What we will cover in this book:

- Learn how the VMware App Volumes solution can enhance the management and delivery of applications in your desktop environment
- Design a real-life App Volumes solution, using best practice and following the recommended sizing guides
- Install, configure, and deploy App Volumes ready to start delivering applications
- Create and prepare applications ready for delivering to end users
- Learn how App Volumes can enhance other desktop solutions by looking at how it integrates with VMware Horizon View, VMware ThinApp, and RDSH
- Learn how to configure the advanced options within App Volumes

Finally, I would like to say thank you for picking up this book. I hope that you enjoy reading the chapters and that they help you learn all about VMware App Volumes and how to deploy and manage the solution.

CHAPTER 1

Introduction to App Layering and VMware App Volumes

Application layering is a software-based technology solution that delivers applications to endpoint devices. It does this by capturing or containerizing just the application, including the application files and settings, and storing that on a virtual hard disk – hence the use of the word container. The virtual hard disk is often referred to as the container or in some cases the application package or application layer. This means you have abstracted the applications from the underlying operating system, and they are in a format whereby you can deliver them to your endpoints.

To deliver an application using application layering, the virtual hard disk containing all the application runtimes, settings, and files gets attached, over the network, to the endpoint device. Once attached, the application layering software will have an agent running on that endpoint that makes the application appear as if it were natively installed. Basically, this means that it will insert or layer, hence the name app layering, any settings such as registry settings and other pointers into the OS on the endpoint device that ensure any calls made to the application get redirected to the virtual hard disk containing the application. The agent also has the job of ensuring the correct files are called, especially where DLL files are concerned.

The virtual hard disks containing the application layers are stored on a file server or some form of network share and then attached to endpoint devices when required, and on demand, as the end user logs in or the endpoint machine boots.

P. von Oven, *Delivering Applications with VMware App Volumes 4*,
https://doi.org/10.1007/978-1-4842-6689-2_1

Application layering use cases

Having started the chapter with a high-level overview of what application layering is, let us now move our attention to why and where it would be used.

Helping desktop administrators

Application layering came about to try and solve the issue of delivering applications to fully stateless or nonpersistent virtual desktops. Before the advent of application layering solutions, desktop administrators had to manage multiple virtual desktop images to accommodate all the different applications required by their end users. One of the reasons they moved to a virtual desktop environment in the first place was to reduce management overheads and cut down on the number of images that they had to manage. But here we are back to managing multiple images all over again, albeit virtual images this time.

Multiple images also typically meant that a stateful or persistent virtual desktop model would be adopted, meaning every user had their own virtual desktop. Again, not very effective, and this certainly will not deliver the full benefits that virtual desktop infrastructures offer.

But the question is why have multiple images and persistent virtual desktops? The answer is simple: so that the apps can be delivered to the end user. In a stateless virtual desktop model, when the end users log out, either the virtual desktop gets deleted completely, or it gets reset back to a base level. Either way, the apps could or will be lost, and also the end users' personal documents and data. You could of course just install every app into the image, but that means you will have one big old image to manage, and also you will need to license every app within the image regardless of whether the end user uses it or not. That is not very cost-effective.

With application layering, applications are abstracted away from the underlying OS of the machine and can therefore be managed independently. That immediately takes away the need to have multiple OS images for different applications, as they can now be delivered independently of the OS image. So straightaway, savings can be achieved by deploying the more cost-effective stateless desktop model and only needing to have virtual desktops available when required – not just cost savings but management time in building and maintaining multiple images, as well as lower infrastructure costs as you do not need to deploy as many virtual desktop machines up front.

About management time and overheads, where else can savings be made? The first one is when updating applications. No longer is there the need to build a whole new OS image just because an app has gone from v1.2 to v1.3. Instead, the app layer can be created away from the OS, tested, and then easily delivered out. If by any chance something goes wrong, as long as the old virtual hard disk with the previous version of the app layer has been kept, then it's a simple case of detaching the nonworking app and going back to the known working one and allowing the end user to carry on while the problem app is remediated.

Enhanced end user experience

For an end user, they will have access to their applications much quicker than they would before, especially if it was an application they did not already have. No longer would they need to wait for the desktop admin team to come and install the app or give them a new virtual desktop OS image.

Instead, the desktop admins would simply click a mouse and allow that user to have access to the apps they need. Or if the end user changes role, the desktop administrator can simply change their AD group membership, and the user will automatically inherit the apps assigned to that AD group. The apps will simply appear on their desktop.

In this chapter

In the introduction to this chapter, I have given you an overview of what application layering is all about. A high-level overview of what it is, and then the use cases for an end user and a desktop administrator.

In the next sections, we are going to start diving deeper into the solution, and how it works, first from a general app layering perspective and then at a VMware App Volumes level.

This book will cover the theory of App Volumes, its components, how it works, and where you should use it, but then also covers the more practical elements and how to deploy and manage the solution as well as how to deliver the end user experience. Following these practical examples will allow you to build, configure, and deploy your own App Volumes environment.

Let us start by delving a bit deeper into how application layering works.

How does application layering work?

As we have already discussed in the introduction to this first chapter, application layering provides a solution that enables desktop administrator to abstract applications from the underlying operating system, so that those applications can be managed and delivered independently. But that was at a high level, so now we are going to dig deeper and see how it does it.

The concept of application layering is not too dissimilar to what we already do with other virtualization solutions, whether that's desktop or server based. The keyword here is abstraction.

With any virtualization solution, you are abstracting different components. In traditional virtualization technology, you are abstracting the server OS or the desktop OS in the case of VDI away from the physical hardware. Application layering performs its level of abstraction at the next level up, between the OS and the applications. In fact, it goes a bit deeper than just the applications as by deploying fully stateless virtual desktops, you also need to consider any user-authored data and other user-specific settings or end user personalization – the bits that make it personal to that particular end user. Basically, all the things that come together to build a full desktop experience for the end user.

I refer to this as the composite desktop model. The end user desktop experience comprises all those elements that come together to form a complete end user desktop experience and are abstracted, managed, and delivered independently. Elements such as the operating system, applications, user profiles, and user-authored data are all extracted from the operating system, centrally managed, and then delivered back independently in order to "reassemble" the complete desktop environment on demand. In this instance, with application layering, we are talking about the application element of this process.

Prior to this way of building the desktop experience on demand, all the desktop components would have been tightly integrated into the device operating system and therefore managed as a single entity.

Now for the technical bit, starting with how to create an application layer.

Creating application layers

Application layering can be described as a two-step process:

- **Step 1**: Creating and capturing the application layer
- **Step 2**: Delivering the application layer to the end users

In this section, we are going to look at the starting point for application layering, the create and capture process. This can be summarized using the diagram in Figure 1-1.

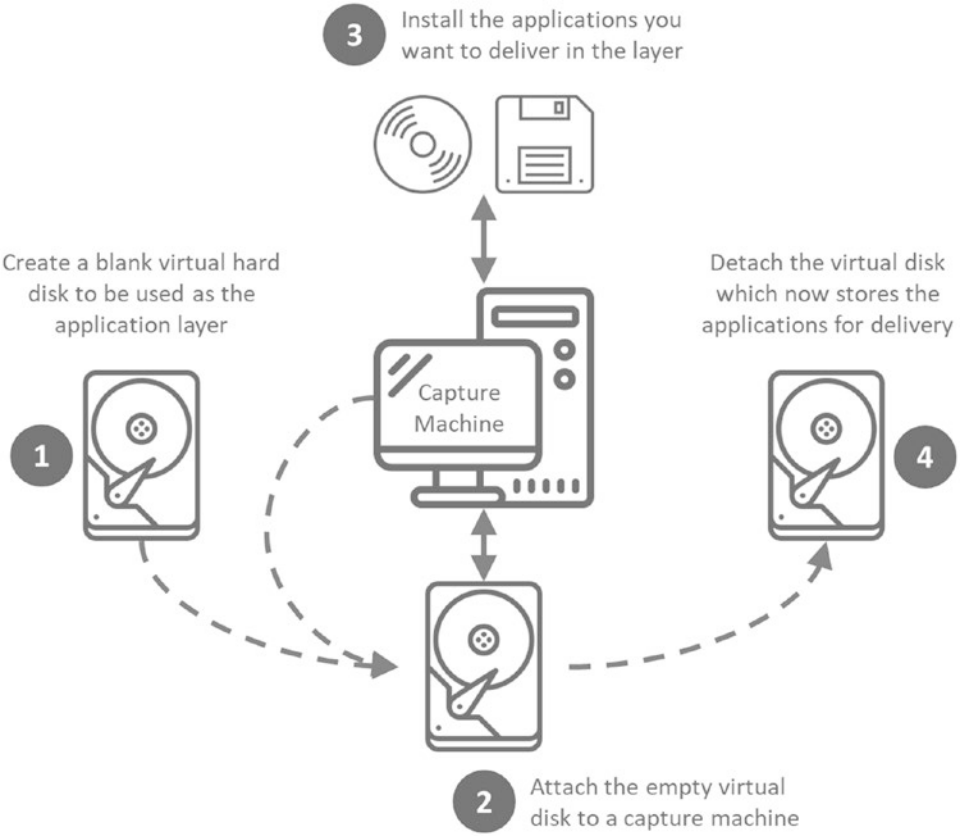


Figure 1-1. *Creating and capturing application layers*

To look at the steps in more detail:

- **Step 1**: The first part of the process is to create the virtual hard disk that will be used as the application layer. This virtual hard disk is where the applications will be installed to create the layer.
- **Step 2**: The next step is to attach the virtual hard disk to a vanilla virtual desktop machine. By vanilla I mean a brand-new virtual desktop with a patched and up-to-date OS and no other apps or software installed other than the app layering software that is required.

- **Step 3**: You can now install your apps by switching your app layering software into "record" mode and then running the application installer. As the installation files are installed, they are redirected to the virtual hard disk.
- **Step 4**: With the application now installed, you can switch off record mode and detach the virtual hard disk, which now contains your application ready to be delivered to end users.

Delivering application layers

Once an application layer has been created, as described in the previous section, the resulting applications can now be delivered dynamically and on demand, based on the end user entitlement.

The process is almost like the capture process in reverse as the virtual hard disk is again attached to the end user's virtual desktop machine, and the application files appear as if they are natively installed on the OS of the virtual desktop machine.

Under the hood, however, the application layering software is managing the access to these files, especially should a file conflict occur.

Figure 1-2 shows what happens when an application layer is attached to an end user's virtual desktop machine.

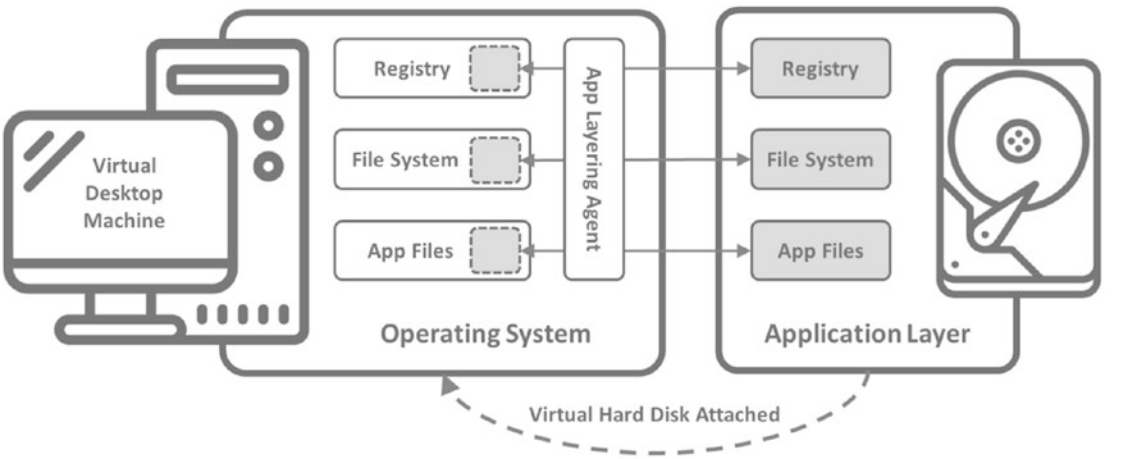


Figure 1-2. *Architecture of delivering an application layer*

When the end user logs in to their virtual desktop machine, the virtual hard disk or app layer that they have been entitled to is mounted to the operating system of the virtual desktop machine.

The application layering software on that virtual desktop machine, typically known as a filter driver or agent, temporarily inserts or layers all the files and settings that the application needs into the operating system of the virtual desktop machine. As far as the operating system is concerned, these application files and settings are installed and available locally. This is also true for the end user experience, with the icon for the layered application appearing on the desktop of the virtual desktop machine. They simply click it to launch it in the same way as they would for any other applications. As far as the end user and the virtual desktop operating system are concerned, the app is actually installed locally in the virtual desktop machine.

You can see this by opening and editing the registry of the virtual desktop machine. If you did this, then you would see the layered application registry settings present in the registry, even though in reality it has not been installed on that virtual desktop machine. The same is true for the application files. If you open Windows Explorer, for example, and navigate to something like C:\Program Files, there you would find all the application files, its executables, its DLL files, and so on. You would see everything that references the application, and to make it run, however, these files and settings reside on the virtual hard disk or layer that is mounted on the virtual desktop machine and not on the virtual desktop machine itself.

So, the end user is now running their layered applications. What happens when they log out of their virtual desktop machine?

When the end user logs out or shuts down their virtual desktop machine, the currently mounted layer, or virtual hard disk, is unmounted which means that the settings and the application files for the layered app no longer appear within the operating system of the virtual desktop machine. Essentially, it is as if the application has been uninstalled. The registry will have no mention of the application and neither will you find any other reference to the application, its files, or its settings. It is like it was never even there!

If the end user had shut down the machine rather than logging out, and that virtual desktop machine was a nonpersistent or stateless virtual desktop machine, then it would revert back to a clean state in preparation for the next user to log in.

When that next end user logs in, they may have a completely different set of applications that get layered into the operating system.

Having now covered the essentials to what application layering is, why you would need it, and how it works, in the next section, we are going to focus on the VMware implementation of application layering and look at VMware App Volumes.

VMware App Volumes

App Volumes came about when VMware acquired CloudVolumes in August 2014. CloudVolumes enabled the real-time delivery of applications to virtual and physical desktops. Then, in December 2014, after the acquisition, VMware rebranded CloudVolumes and changed the name to App Volumes and in doing so added an application layering solution to the VMware Horizon solution stack.

As an application layering solution, App Volumes works in the same way in which we have described application layering in the previous section. It is designed to enable the real-time delivery of applications within a virtual desktop environment.

Application layers are captured and stored on virtual hard disks within your virtual infrastructure environment. Then, when an end user logs in to their virtual desktop machine and is entitled to use a particular application, then the virtual hard disk containing that application is mounted, and the applications layered into the operating system ready to be launched. The end user will see just the application icon.

When the end user logs out, the layer is removed from the OS of the virtual desktop machine, and the virtual hard disk is unmounted.

How does App Volumes work?

We have already discussed the general mechanics around how application layering works, so now we are going to take those principles and look specifically at the VMware implementation of those with App Volumes.

As with any other application layering solution, App Volumes sets out to address the issue of how to deliver tightly integrated applications that are essentially part of the operating system image and take away that operating system dependency so that the applications can be delivered on demand and allow VMware customers to take advantage of the full Horizon virtual desktop solution and deploy a fully stateless virtual desktop.

App Volumes provides the layer of abstraction between the operating system and the applications, allowing the applications to be delivered to the end user's virtual desktop machine on demand and based on policy. The applications are effectively containerized within a virtual hard disk file (either VMDK based for VMware environments or VHD based for other virtual infrastructures). These virtual hard disk containers or layers were called AppStacks in previous versions of App Volumes, but this has now been replaced with a more application lifecycle management–based approach referred to as simplified application management (SAM).

SAM comprises the application, package, and program that go to make up the application layer as shown in Figure 1-3.

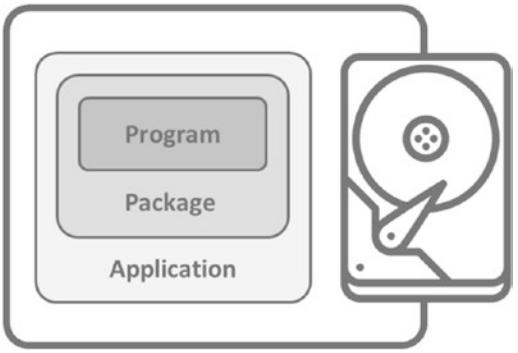


Figure 1-3. *Simplified application management (SAM)*

So that is at a high level. How does SAM look under the covers?

Application

The first part of creating a layer is to create an Application. In App Volumes 4, an application allows you to create a logical construct for an individual application or a group of applications. It is the application that is assigned to your end users.

As part of the application lifecycle management process, an application could be made up of multiple different packages and programs, so one thing to bear in mind is not to give the application a version number as it could well be made up from multiple different versions.

The next part to look at is the Package.

Package

The Package stage is more like the AppStack creation process of the previous version. Using a template virtual hard disk file from which to create the layer, a packaging virtual machine is used to capture the installation process of the application or applications you want to include in the Application construct, with the files and settings captured during install, redirected to a virtual hard disk that is temporarily attached to the package machine during app installation.

When the capture process starts, a copy of the template virtual hard disk file is mounted on the packaging machine. Once mounted, you can install the individual applications as you would do on any other machine with the application files and settings being redirected to the virtual hard disk. It's like recording the app installation process.

Once the installation of the applications is complete, you switch out of that record mode, and the virtual hard disk is unmounted. It is then ready to be assigned to end users based on AD group policy.

A new feature introduced with App Volumes 4 is the ability to choose the package stage. This is the lifecycle of the Package, and you can select either New, Tested, Published, or Retired. It is also worth noting that only one Package inside an Application can be current at any one time.

Finally, there is Program.

Program

The Program elements are automatically generated based on the applications you install during the Package stage. It is made up of the actual application executables and files that were captured during the Package phase.

So now that we have covered the three stages of creating an application layer, how does that translate when creating actual layers? First, let us look at the theory of how to manage an individual application.

Managing a single application

Taking the SAM approach with Application, Package, and Program, what would that look like in a real-life application lifecycle scenario?

So first you create the Application itself, so in this example, maybe that is for a PDF reader. This is the element that gets assigned to the end users. Once the application has been created, the next step is to capture the PDF reader application by installing it on the packaging machine. The files and settings for the PDF reader are then automatically created in Programs. Our PDF reader example would look something like Figure 1-4.

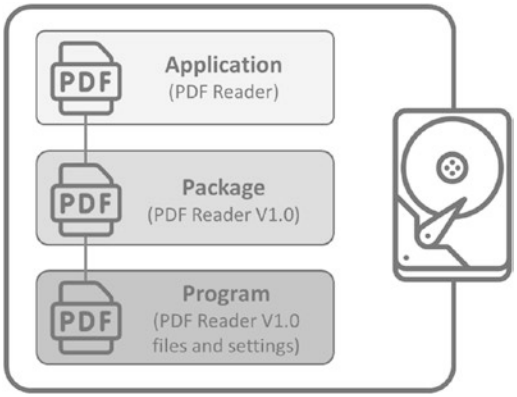


Figure 1-4. *Creating a single application using the SAM methodology*

The preceding example shows the process for a single application and a single version, but as we all know, applications do not stay on the same version for very long. So, what happens when the PDF reader updates to a new version?

Unlike when you create the application as we described previously, you do not need to create a new application. We already have an application for the PDF reader in place. Instead, you go straight to the Package process and capture the new version of the PDF reader using the packaging machine, which in turn creates the Program elements automatically. Once completed, you will have updated the Application with the new version of the PDF reader, which will look something like Figure 1-5.

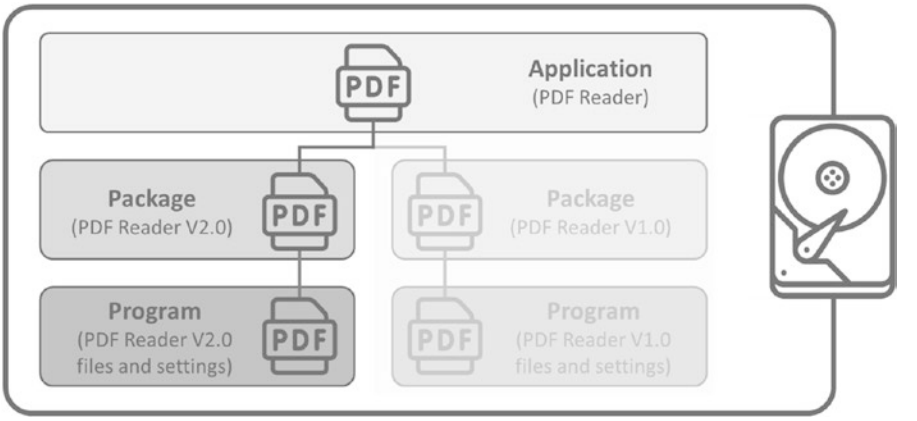


Figure 1-5. *Updating a single application using the SAM methodology*

As part of the simplified application management methodology, you could now mark the old version as Retired and the updated version as New or Published.

So that is how to manage an individual application. How do you manage a group of applications?

Managing a group of applications

Groups of applications are typically used to deploy multiple applications based on departmental use. So, for example, the finance department will have a set of specific finance-based apps, and the sales department will have a set of sales-based apps, and so on. In this example, we are going to create an Application for the finance department which contains a PDF reader and a spreadsheet application as shown in Figure 1-6.

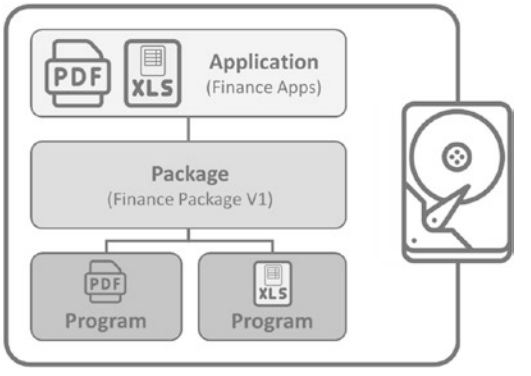


Figure 1-6. *Managing a group of applications using the SAM methodology*

So, let's run through a similar scenario as we did for single applications. First, you create the Application itself, so in this example, we are going to call it something department related rather than app specific, as this Application is going to contain more than one individual application. We have chosen the finance department for this example.

Again, the Application is the element that gets assigned to the end users, but in this case, it means more than one application. It is a set of applications. Once the Application has been created, the next step is to capture the applications by installing them on the packaging machine. The files and settings for each individual application are then automatically created in Programs. In this example, we have created a Package that contains Programs for the PDF reader and the spreadsheet app.

The next question is how do you update the Application should there be an update to one or more of the Programs? It is the same story as we covered with updating a single app.

You do not need to create a new application as that is already in place. Instead, you go straight to the Package process and capture the new version of the application you want to update, using the packaging machine, which in turn will update the Program elements automatically. This means you could update each application independently. For example, if the PDF reader updated and the spreadsheet app did not, then you simply just update the PDF reader, capturing that update using the packaging machine.

The new package will contain the updated app, as will the Programs. Once completed, you will have updated the Application with the new version of the PDF reader and spreadsheet app, which will look something like Figure 1-7.

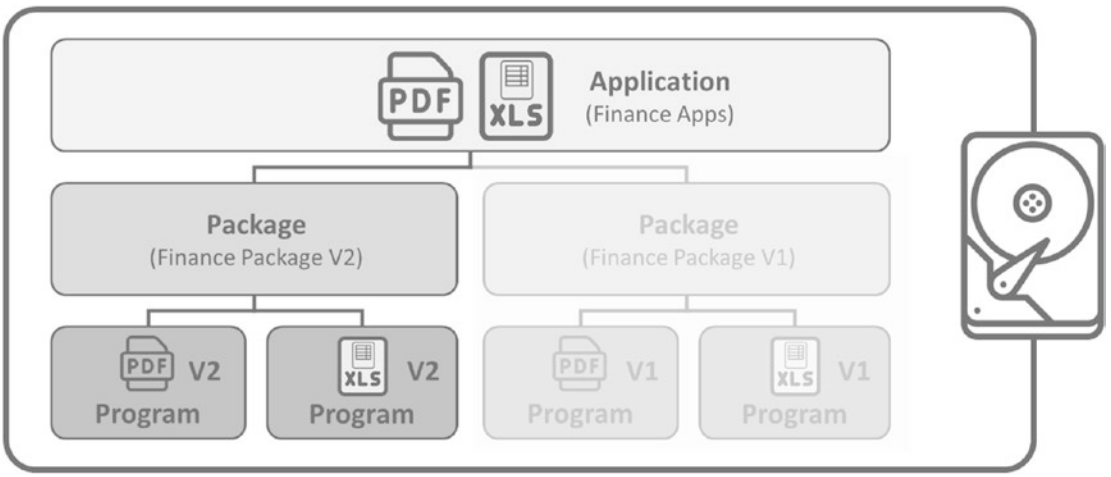


Figure 1-7. *Updating a group of applications using the SAM methodology*