The Software Developers' Guide to

Making Your Program Work With

Microsoft App-V

Tim Mangan

TMurgent Technologies, LLP

January, 2016

Introduction

When you sell your software into a company, especially the larger companies we refer to as Enterprises, you actually have two customers. The one you primarily think about in design is the end-user that makes use of the software to do amazing things. The end-user loves your software.

The other customer is the internal IT department. They are much more likely to have a dimmer view of your software as they deal with all of the problems it creates as they try to install and distribute the software along with all of the other software that they purchase. And the installer that you provide is probably the worst part of your software.

In the last fifteen years, I have seen software vendors make great strides with their software designs. Moving from unmanaged code that crashed way too often to modern programing languages that automatically takes care of memory management to prevent the most common causes of those problems.

Software vendors have also (mostly) built in support for user profiles, allowing a user to retain their application personalization settings and data as they roam from OS to OS. Typically, they also support use by multiple users on a server with Remote Desktop Services (RDS) enabled (you might know this situation under the name Citrix XenApp as the Enterprise will probably add that to RDS). Additionally, some vendors also support the multi-tenancy needs of deployments in a Cloud Services environment (what we used to call an Application Service Provider, or ASP).

But surprisingly, we have rarely (until recently) seen software vendors support the primary "go to" technology used by the enterprise to manage and distribute their application within the company, Microsoft Application Virtualization (App-V).

Why Does the Enterprise Use App-V?

1. They can't use your installer.

   To begin with, the enterprise does not use the software they way that you designed it. The end-user does not have permissions to install the software and make the choices provided during installation. This is done for them by the IT department. They take your installer, remove the dependencies (such as VC++ Runtimes) and handle them separately. IT performs the installation and chooses the install options and features. If the app is going to be used by, let's say call center operators, they remove every possible feature not absolutely required by the operator. They pre-configure the app for the back-end components that the app will need to use. They then configure the remaining options for how the internal business unit has indicated they want it configured.

   In the last 15 years, installer technology has not really advanced much. The enterprise prefers that you give them an MSI over an exe based installer, because the tooling they use to customize and prepare your

The installation and deployment experience as perceived by the enterprise IT department, needs to be as delightful as all of those user interface dialogs that you spend so much time designing. And it isn't.

application is easier with MSI. Plus they have much more visibility on what your installer is doing, with the exception of those pesky MSI "Custom Actions" that you probably include (simply because MSI hasn't kept up with your needs). So the enterprise often repackages your installer (using tools like Flexera Admin Studio) and uses their own. This process is expensive, as getting and retaining people who are good at it is really tough.

App-V provides a way to accomplish this for most apps at a lower cost.

2. Application Conflict

   When software is installed onto an operating system, the bulk of the software goes into unique locations. Files go into a new folder under "Program Files", typically using the program or vendor name. Most of the settings go into a new keys in the registry under the "HKLM\Software\" and "HKCU\Software\" keys, typically using the program or vendor name also.

   But your installer also adds dependencies and writes to locations shared by other applications. And sometimes these changes interfere with other software the end-user needs.

   The use of managed code has largely removed the biggest issue caused by installers writing to shared locations faced by the enterprise in the past, "Dll Hell". This term refers to an installer that updates a shared system dll to the version they need. Meanwhile another app needs a different version of the dll, and the dll is not 100% backwards compatible. This situation is a nightmare for the IT department, because a end-users system will be fine one day, and then suddenly an app that was working great starts crashing. And it might be either of the conflicting apps, depending on which was installed first on that system. While the use of .Net programming languages can eliminate DLL hell, the same kind of issues remain in other shared location changes. We use terms such as "GAC Purgatory", "COM COMfusion", "Environment Variable Shenanigans", and "Registry Hijacking" to identify the primary ones, but underneath it is the same kind of issue. You can't know what other apps depend upon. And even if you did know today, every Enterprise is running some apps that are at least 10 years old so you would have to consider the potential needs of apps built in the next 10 years too.

   App-V runs your installer and application in a virtualized environment, preventing these conflicts. It is sort of like running an OS is a virtual machine, but instead of virtualizing the entire OS, it just virtualizes the app.

3. Making Software Availability Dynamic

   The era of the "Personal Computer" has evolved in business. The device the end-user touches is not necessarily the one running the apps they use. RDS/XenApp servers are a much more efficient way for IT to make applications available. Virtual Desktop Infrastructure (VDI) is another very similar situation. In both cases, the software application is run on a machine (or virtual machine) inside the data center. Not only is this easier for IT to maintain, it is often much more secure (as data doesn't roam around on people's laptops). Plus, by remoting the user interface to where the user is, the user is not tied to their desk. They can work from a conference room, or home. In medical offices, the doctor or nurse can walk up to the device in

the room they are in and look up the patient information by logging back into their session from that device.

App-V enables the IT department to package up your app one time, and then deploy it to any user on any operating system version on demand. They aren't interested in violating your license agreements – they are quite willing to pay those licenses – they just want to instantly get the app into the hands of whatever user needs it on whatever device they log into. App-V also tracks the application use, allowing the IT department to correctly determine actual usage to ensure license compliance.

What Not Supporting App-V Costs Your Company

Today, the enterprise doesn't choose one package over another simply because it supports App-V. This might change in the future as more vendors support App-V, but we aren't there yet.

However, Application Virtualization doesn't work with all applications. While most applications are easy to package up and deploy using App-V, there are a few that are not. I only hear about the ones from App-V customers that don't work, but the list isn't big. The inability for the customer IT department to get the app packaged successfully in App-V can, and has cost, software vendors revenue from renewals. If your software does not support App-V, the IT department will look for alternatives down the road and push back on the business unit requesting the software. I have seen a vendor loose a multi-million dollar renewal order over this.

But on a regular basis, App-V is costing your company money in the form of increased Customer Support calls. And you probably don't know it.

When the customer is using App-V and has a problem, they will call your support line, but if at all possible they will not mention App-V. They know that if they do, the following happens:

- The person at the end of the call has never heard of App-V.
- That person will check a list of supported platforms, not find App-V on the list, and (politely) tell the customer to go pound sand.

It is just like how customers handled your apps on Citrix servers years ago when your customer support department didn't have Citrix on the list (most of you do provide support on Citrix these days).

So the customer gets creative. If they can, they will reproduce the issue without App-V, but when they can't they just omit App-V and try to get your support to figure out what is going on without all the facts. This is expensive, and frustrating for both you and the customer. And it happens every week, all year long.

Wouldn't it be easier to embrace and understand you your software works with App-V once, and then enable your Customer Support personnel and Customers delighted with the support?

The Bottom Line

The installation and deployment experience as perceived by the enterprise IT department, needs to be as delightful as all of those user interface dialogs that you spend so much time designing. And it isn't.

An ISV Project to Support App-V

So what does this take? In the remainder of the document I will outline the typical project.

Phase I – Try it

Before spending time analyzing, have someone spend a week to try it. This could be someone in the development organization or in your own IT department. There is a very good chance that they will have success with it out of the box, and this will help to set the scale of the project.

If it does not work right away, you might need to include some training or outside consulting by third parties that know App-V. While it is extremely unlikely that there will be a need for changes in your software to resolve issues[1], engaging with an outside expert that understands enterprise IT needs may provide advice on some changes that make your software more deployable by the enterprise with or without App-V involved.

Phase II - Determine Scope of Support

Put some goals in place for what you want to accomplish. Maybe you just want to see it work, explain it to your customer support personnel, and add it to the supported list.

You might consider a small white paper for your customers explaining the support.

You might include an "App-V Recipe" for the customers. This is typically a document with screenshots walking them through the App-V packaging, which is similar to what you might already provide for manual installation.

You might create a package and release it in the form of an App-V Accelerator. An Accelerator is a version of your app per-packaged in App-V by you. The accelerator file has all of the executable bits removed, but with placeholders left in place. This file is then made publicly available on the Microsoft TechNet Gallery, or on your own website. The customer would use this file, along with the App-V Sequencer and your full installer to re-hydrate the package into a usable form. While this is an option, I will caution you that our experience is that the enterprise IT would prefer the recipe. Thus I don't recommend you creating the Accelerator.

You could consider releasing in both MSI and App-V format as part of your release process. This requires more than just creating the package. To be successful, you will need to consider how the enterprise can take this package and customize it. While I would love to see every ISV provide this support to their customers, clearly the biggest bang for the buck for most will be to publish the recipe and get your customer support engaged.

Finally, I will mention Microsoft Project Centennial. Project C is (currently) a work in progress, but Microsoft sees it as a future bridge to modernize your app. Project C will basically use the App-V technology "under the covers", but it allows your application to get into the Windows Store, deployed and licensed just like any other Universal Windows App. When Project C becomes available as a

---

[1] The primary issues I find are device drivers without a separate installer, licensing, and file path lengths as App-V will add 100 – 120 characters to some file paths. We also find some out-of-process COM issues due to security context issues, but these are more likely resolved in packaging. Given enough time to debug, we can package over 95 percent of software applications without changes to the software.

product/feature, you will have to make software changes to work with Project C. But working with App-V now will be a great start toward potential future work with Project C.

Phase III – Implement the Project

Put the team together and make it happen. Depending on what you found when you tried it, this might consists of a one week project to document and distribute the success that you have already had.

More likely, you'll need a couple more weeks to learn and try things to resolve issues.

And in a few isolated cases, it might require more significant investment that may take 2 to 3 months.

About TMurgent Technologies

We are a small independent Microsoft Partner that supports both enterprises and software vendors with Microsoft App-V. The company is run by Tim Mangan, who created the original version of App-V (then known as SoftGrid) as VP of Software Development at Softricity. TMurgent is best known for their training programs on App-V, but also offers consulting services. We can help software vendors with a project to support App-V, if you need it. Find us at www.tmurgent.com or reach out to Mary Jane at murjur@tmurgent.onmicrosoft.com