At a glance:
- App-V architecture
- Managing virtual apps
- Using the App-V sequencer
- Integrating App-V with Configuration Manager

Getting started with Microsoft Application Virtualization

Anthony Kinney

Microsoft Application Virtualization (or App-V) is near and dear to my heart. App-V was formerly known as SoftGrid, and I came to Microsoft with the acquisition of the company that created SoftGrid, called Softricity. I am very excited to have the opportunity of writing this article for *TechNet Magazine* now, as a lot has changed since the acquisition.

The best way to approach App-V is first to talk about the challenges IT professionals face in terms of enterprise management. Today's business desktop is awash in applications. Before an application is installed, it must go through lengthy regression testing to ensure that it can coexist with the other applications installed on the system without impacting their ability to run properly. The application must then go through a series of deployment processes before it reaches production. And because an application is essentially only available where it is installed, your users are tied to specific computers. This further complicates complex yet critical projects, such as OS and application migrations, security refreshes and disaster recovery planning.

App-V changes all that. Rather than being a complex series of time-consuming steps that take up resources, desktop administration becomes a simpler, more automated process with App-V. You can more easily deploy, patch, update and terminate applications with better results.

With App-V, a user can sit down at any desktop and access his applications. The applications are delivered on demand but run as if they were actually installed locally. Thus, there is no need to install the application components or alter the host device.

This use of virtualisation could drastically change how IT professionals manage desktops. Not altering the host device and running virtualised applications instead introduces numerous advantages, including:

- Fewer application conflicts
- Faster and easier application updates
- The ability to run multiple versions of the same app side by side
- Flexible applications that follow users online and offline
- Reduced application-to-application regression testing

This article is based on a prerelease version of App-V. All information herein is subject to change.

The App-V architecture

So now let's look at what is really happening behind the scenes of the App-V platform. The platform consists of a few main components: a sequencer, a database, clients, a management server, a streaming server and a management console (see **Figure 1**).

At the core of the App-V system is the App-V client. There are two types of clients that can be used – the Terminal Services Client and the Desktop Client. In either case, the client must be installed on every desktop and terminal server on which you plan to deploy virtual applications. The client takes up relatively little disk space. It installs a driver and has a visible user runtime component that shows up as a tray indicator.

The client gathers a list of virtual applications from the App-V Management Server and displays the available virtual apps. It handles launching those applications (when initiated by the user) and managing the client-side cache. The client is also responsible for managing creation of the virtual runtime environment and ensuring that each environment runs in its own virtual bubble. This virtual environment includes several compo-

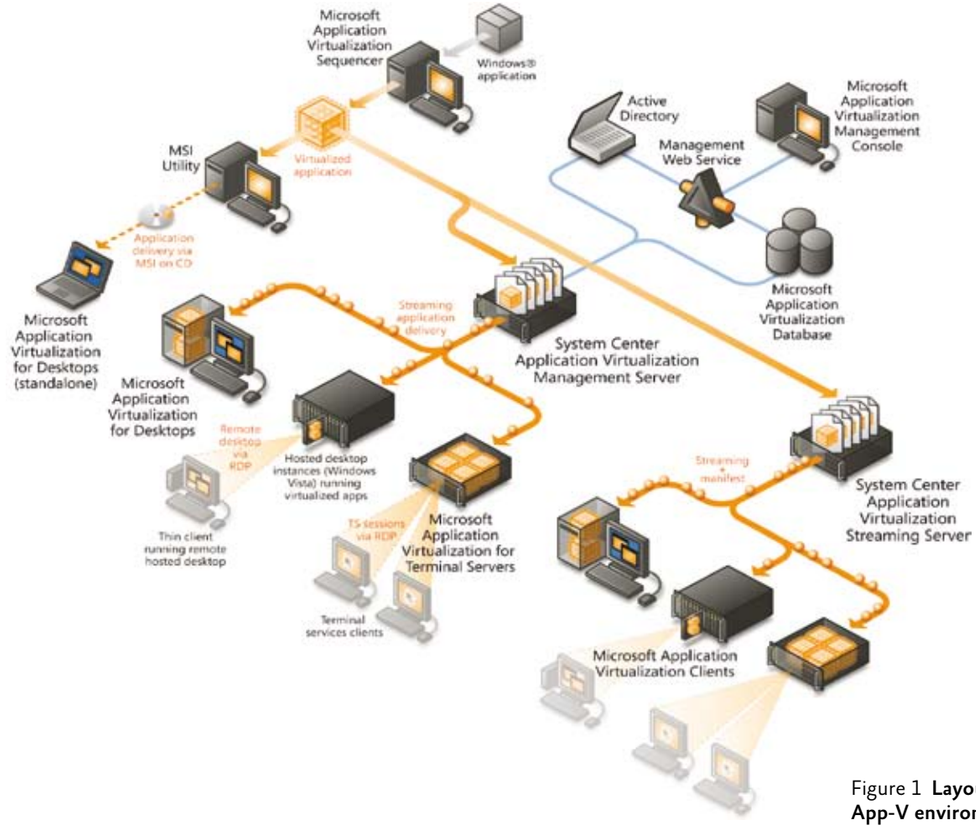

Figure 1 **Layout of an App-V environment**

nents, including a Virtual Registry, Virtual File System and Virtual Services Manager.

There are three infrastructure deployment options available in App-V 4.5: full infrastructure, lightweight infrastructure and standalone mode. When you deploy a full infrastructure, the back end includes the App-V Management Server and the App-V Streaming Server (this is a new component I will discuss in a moment). The App-V Management Server hosts and delivers the centralised virtual applications, as well as updating the virtual applications when patches or updates are applied.

> The App-V Management Server hosts and delivers the centralised virtual apps, as well as their updates

This management server relies on SQL Server to host the App-V database, which contains configuration and settings for virtual applications. You should use Active Directory groups as the central management tool for provisioning and controlling permissions to virtual applications.

To manage the settings and configuration, the App-V platform provides a Microsoft .NET Framework web service that can be loaded on the same server as long as IIS is installed. This web service acts as a liaison between the App-V Management Console – a Microsoft Management Console (MMC) Snap-in – and the App-V database. Administrators can use the console to publish and manage virtual applications, assign Active Directory groups and control server settings, and run reports on usage of virtualised applications (see **Figure 2**).

The lightweight infrastructure includes the App-V streaming server, which enables streaming capabilities such as active/package upgrade. This option does not require Active Directory or SQL Server, it has no desktop configuration service, and it lacks licensing and metering capabilities. The lightweight infrastructure does, however, allow streaming capabilities to be added to System Center Configuration Manager (SCCM) and other third-party enterprise software deployment (ESD) solutions.

In standalone mode, the App-V sequencer can create an MSI file that automates the addition of the virtual application (see **Figure 3**). The MSI file contains metadata that allows any ESD system to recognize it and control the virtualised application. This mode requires the client to go into standalone mode, which only allows MSI-based updates of the virtual applications, and streaming is not allowed while in standalone mode. This mode gives organisations the ability to make use of the App-V isolation capabilities.

MSI files are very flexible, they can run completely standalone with just an App-V client, and they don't require any server components. This means they can be deployed manually, with a disk, or by any traditional deployment tools.

In App-V 4.5, HTTP and HTTPS are now supported protocols for streaming. This enables better performance of streaming along with a more widely adopted protocol, especially for streaming across secure wide area network (WAN) environments and across the Internet.

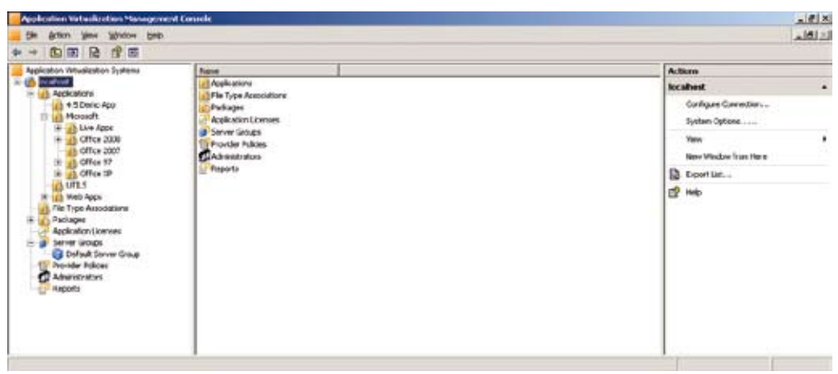

Figure 2 **Management Console**

How App-V full infrastructure works

A user logs on to a device that has one of the clients (either App-V Terminal Services or Desktop Client) installed, and the client sends a request to the server for a list of applications assigned to the current user. The server communicates with Active Directory to determine which groups the user is a member of and then returns the list of applications back to the client. The client begins building advertisements for the virtual applications that have been assigned to that particular user.

In this publishing process, several actions are performed:

- Configuration files are copied
- Desktop icons are created
- Send To links are created
- Start menu folders are created
- File types are configured

This process is very fast and, most importantly, ensures that the environment will look exactly as the user expects with no visual changes. The virtual applications act as if they are installed locally, but, of course, they don't alter the host machine. The icons, instead of pointing to executables that reside in the program files directory, point to the App-V client, which relies on a launcher file (an OSD file) for its configuration.

It's important to note that this process has very little impact on the network because, unlike traditional software deployments, nothing is being installed. This has tremendous benefits, especially in roaming user environments, since the applications are available to the user but nothing is actually delivered until an application is launched. This advertisement method is also what provides the on-demand and roaming application features of App-V.

When the user launches a virtual application, the client reads an OSD configuration file, which has been stored on the local machine. This tells the client which protocol to use when communicating with the App-V management server and on which server the application resides.

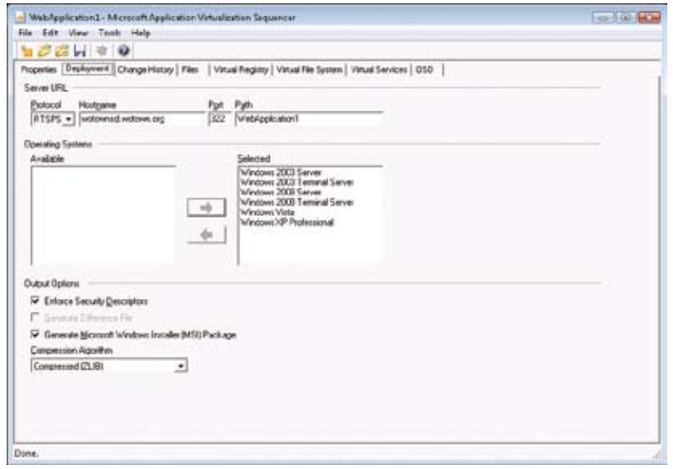

Figure 3 **App-V sequencer**

The appropriate server responds to the client by streaming down the initial launch threshold, which is typically 20–40 percent of the full application. Once the entire launch threshold has been streamed (again, just 20–40 percent of the application), the virtual application is ready to run.

The streaming really is one of the key elements of the paradigm shift being introduced with App-V. It can send just enough of an application for it to run without wasting valuable network bandwidth. All the data being delivered to the client resides in a local cache file on the device and any subsequent launches of the application are launched from the local cache, eliminating additional network traffic.

Once the virtual application is finished streaming, the client builds an isolated environment that prevents the application from altering the local machine (in other words, the application has no client footprint). The client does, however, allow the virtual application access to the local file system when saving and editing files, and it also allows the application to interact with local services (such as printing) as long as the user has the appropriate privileges on the local system. But any changes made by a virtual application to the local system's files and registry are

redirected to the virtualised environment so the host device will remain unaltered.

When the application is being run, any features that haven't been used previously are delivered as needed and cached for subsequent use. The upside to this is that only components needed by the user are loaded on initial launch and features that aren't needed don't consume network resources. (The new version provides some client-side caching enhancements that allow for smarter cache usage and background streaming.)

Consider Microsoft Office Word, for example. Almost all users use spell checker (heck, I couldn't write this article without it!); therefore, it would be part of the initial launch. But what about the Help feature in Word? Not as many users use this feature,

Only components needed by the user are loaded on the initial launch; thus unneeded features don't consume network resources

and, thus, it wouldn't need to be delivered in the initial launch. Instead, it would be sent to a user the first time he accesses it.

When the user is finished and closes the application, the client tears down the virtual environment and stores all the user settings in a user-specific location so the environment can be retained and rebuilt upon next launch. Whatever percentage of the virtual application had been streamed remains in the local cache and is available for the next launch. And if another user logs onto the same host system and launches the same virtual application, the new user benefits from the application already stored in the cache.

To remove the virtual application advertisements, simply remove the user from the appropriate Active Directory group. And to uninstall the virtual application from a desktop completely, you can simply clear the cache. Since the application was never really installed locally, there are no annoying prompts asking, "Do you want to remove this shared component?"

Note that even if a virtual application is stored in a cache, that doesn't mean all users have access to it. Unlike locally installed applications where users can simply search or browse for executables that they don't have rights to, there are no visual or physical representations that the virtual application exists unless the user has been given explicit rights to it through Active Directory groups.

Updating virtual applications

Updating is done using the sequencer. Once an application has been revised to include an update, it is placed on the App-V management server right next to the previous version. The server then notifies the client upon next launch that a change has been made. If the previous version is still being used, the user continues to have access to that version until the virtual application is closed. Upon the next launch, the deltas that make up the update are streamed down to the client and loaded into cache, resulting in an updated version of the application.

Say you have 1,000 users running Word 2000. An administrator needs to update Word 2000 (word2K.sft) to Word 2000 SP3, so she copies the word2K.sft file over to the sequencing station and selects Open for Package Upgrade in the sequencer. By selecting Open for Package Upgrade, the admin begins working from the last package state. She can then copy DLLs, run updates or execute patches within the virtual application to update it to Word 2000 SP3. The admin then saves this updated package.

The sequencer automatically assigns a new file name, word2K_2.sft, to prevent duplicate file names and to indicate the version of sequencing. This new package is placed in the same directory as the old package on the App-V management server so that Word 2000 (word2K.sft) and Word 2000 SP3 (word2K_2.sft) end up residing in the same directory. The admin then uses the App-V management console to link these two SFT files together.

On the client side, users who have an active session of Word 2000 without SP3 open continue to function normally. Users who launch a new session of the application after the admin has done this linking will receive a message that a change has been detected.

Figure 4 **The sequencing wizard**

The client then begins streaming down only the delta changes between word2K.sft and word2K_2.sft, automatically updating the application to Word 2000 SP3.

Because of the dynamic nature of virtual applications, rolling back is also quite easy. Simply go back to the App-V management console and remove the newly added version. This causes the client to roll back to the previous version upon relaunch. To ensure no crossover of package data, the client automatically purges the cache and restreams the proper SFT file. This is a fair trade-off when you consider what must be done to roll back an application update that has been installed physically using more traditional software deployment tools.

Sequencing

To realise the benefits of App-V, you must create virtual application packages. This is where the App-V sequencer comes into play. Any knowledge and experience you have in scripting and creating packages for traditional software deployment tools will ease your transition into sequencing.

Most software deployment solutions rely on scripts that capture the way an application installs itself and then duplicate the process on other machines, eliminating the need to visit every machine to install or update applications. Once the application has been installed, common software deployment tools wipe their hands clean of the package. You must then install any dependencies on which the application may rely, run other scripts, or perform manual steps to configure the application for your needs.

The fundamental change in App-V is that the sequencing process generates an image of an already installed app, complete with its dependencies and configurations. This can be "played" by the App-V client without altering the device on which it's being played.

The sequencer generates a variety of files, the most important of which is the SFT file, which contains all of the application assets, dependencies and configuration information. In some cases it may also contain multiple applications. Not surprisingly, this file can be quite large. There are some compression options, but a sound knowledge of your network and device performance is essential. The Icon file (.ico) that the sequencer creates is used to advertise the virtual application so that it acts as if it is locally installed.

The OSD file is also very important, and its options are endless. By default, this is an XML-based file that is used to tell the App-V client how to launch the virtual application. The OSD file can also be modified to configure and control how the virtual application launches and runs. I strongly suggest you read the Sequencing Admin Guide and the Sequencing Best Practices document to familiarise yourself with the properties and values available through the OSD file.

Lastly, the new manifest.xml file contains package-based configuration information and can be used for integration with third-party ESD solutions and MSI deployments. The sequencer can also generate an MSI file for the virtual application package. This can be used to load the virtual applications onto standalone (serverless) clients and through an ESD system.

The sequencer itself is a wizard-based tool that walks you, the packager, through the process of installing an application and transforming it into a virtual application (see **Figure 4**). The first step allows you to configure default properties for the package. These properties, which are stored in the OSD file, include package name and comments. Some of the advanced settings let you specify the server from which to stream, the content directory, and what operating systems the package should support.

The second step is to install, configure and

test the application. During installation, the sequencer captures all the changes being made to the local system, including file system, registry and system. There are also a few utilities in this wizard that enable, for example, integration with Windows Update.

The next step is to configure file type associations and specify where the shortcuts should be placed. Standard placements include the Start menu, the desktop and the quick launch bar, but you can also create customised locations.

Then you need to launch the application and configure the initial launch threshold. This is the step where App-V determines the initial portion of the application that needs to be delivered to the client to allow the application to begin.

To configure this initial code (typically referred to as Feature Block 1, or FB1), simply launch the application and use the most common features needed by users. For example, launch Word and then initiate the spell checker. Any DLLs, files or registry keys called by the application during this phase are automatically designated as part of FB1. Any files, settings or components not used at this point are added to FB2. When the application is then used, the client will receive a map of the SFT file that indicates where FB1 starts and stops and where other files exist in FB2 so the client can retrieve those files when needed by the application.

The final step in the sequencing process is to ensure everything is configured properly. The sequencer displays the dialog shown in **Figure 5**, which represents the SFT and allows you to make any final additions or changes to the package.

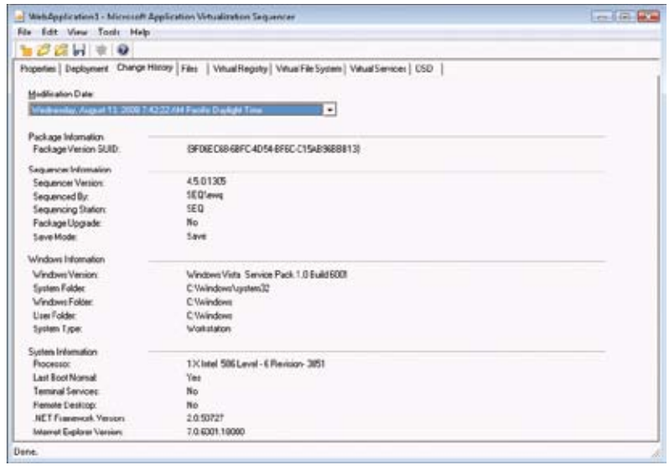

Figure 5 **Confirming and tweaking a final package**

Version 4.5

After two years in the making, App-V 4.5 is due to be released later this year. This is the first Microsoft version of the product to be released, and it promises to elevate application virtualisation by introducing several key enhancements, such as Dynamic Virtual Application Interaction, extended scalability and improved alignment with Microsoft internalization and security requirements.

Dynamic Virtual Application Interaction allows virtualised applications to interact with one another. This interaction is referred to as Dynamic Suite Composition (DSC). DSC doesn't replace the ability to add multiple applications to the same package. Instead, it offers a new way to integrate dependencies, middleware and plug-ins shared among virtual applications.

Administrators can specify which virtualised applications can interact with each other. For example, suppose you have five web applications that require the same Java version. In App-V 4.1, you would have to add that same Java version to each of the five separate packages. And suppose that Java version needed a patch. The administrator would then have to patch the five different packages. Using DSC, Java can be packaged once and then configured as a package to be used by all five web applications. As a result, patching Java would only require the administrator to patch the Java package once.

The same scenario applies to middleware and plug-ins. I plan to blog about other use case scenarios as Microsoft gets closer to the release and finalises any late additions.

Enhancements to scalability involve both streaming and back-end infrastructure. Back-end components have been modified to better support clustering and failover scenarios and streaming is more WAN and LAN friendly. The enhancements come from several key additions.

First is the new Streaming Server Component, which allows for streaming without the need for a back-end infrastructure of Active Directory and SQL Server. You still get all the great benefits of on-demand delivery and centralised updating of packages, but without the heavy back-end requirements. This will be widely used in branch office scenarios and for integration with third-party ESD solutions.

The App-V client has received a few enhancements as well. For instance, the client now stores all usage information locally so this information can be tracked whether the client system is on or off the network. The client cache has also been expanded and improved for better performance in scenarios with limited disk space. There is also now support for sequencing non-English applications, running App-V on non-English operating systems, and localisation into several other languages. ■

Integrating App-V with Configuration Manager

A number of enhancements and new features in App-V 4.5 have been designed for integrating App-V into SCCM 2007 R2. Virtualization and streaming offer ways of delivering applications that traditional software deployment tools don't provide. I'm not implying that App-V will replace these tools, rather that App-V can complement and extend them.

With this integration, you get all the scalability, reporting, device recognition and WAN features of SCCM with all the streaming and isolation features of App-V. Here are some of the areas that benefit from integrating these two technologies:

Application Delivery The SCCM R2 integration supports all the capabilities of on-demand delivery, roaming applications, initial launch thresholds and deployment of applications without altering client PCs.

Updates SCCM distribution points (DPs) can deploy just delta changes of virtual applications when packages are updated. This introduces the centralised ability to revert virtualised applications to previous versions with a single click.

Management Version R2 introduced a new virtual application advertisement wizard that allows administrators to deploy virtualised applications as well as traditional software packages and advertisements from a single console.

Packaging There is no need to repackage applications when integrating App-V with SCCM. The initial sequencing of an application needs to be done using the App-V sequence outside of SCCM, but administrators do have the ability to update existing packages using SCCM.

Licensing Virtual applications can be tracked for licensing and metering using the existing tools in SCCM.

BITS SCCM offers a new method for deploying virtualised applications to App-V using the industry standard BITS protocol. Though SCCM DPs can stream, there are cases when streaming isn't the ideal method for deploying virtualised applications. When deploying via SCCM, you have two options. You can use standard streaming or you can use the Quality of Service (QoS) features of BITS for a more controlled deployment. This is also useful in scenarios where you'd like to pre-load the cache prior to users launching the virtual application.

Machine Deployments SCCM provides the ability to deploy virtualised applications to specific machines as well as continue to support the user-based targeting approach of the App-V platform. This can be beneficial when you deploy virtual applications to laptops, kiosks and lab machines. This is also handy for assisting with licence control when your software may be licensed by device, not by named users.

Scalability The need to deploy two separate tools that have a great deal of overlap is a common concern. By integrating the scalability and WAN benefits of SCCM with the isolation and streaming benefits of APP-V, you can use your existing SCCM, which provides a single tool to handle both management and deployment without having to add more complexity.

For more information on Microsoft Application Virtualisation, visit:
http://technet.microsoft.com/en-gb/appvirtualization
or visit the blog at:
http://blogs.technet.com/softgrid/archive/2008/09/03/microsoft-application-virtualization-4-5-rtms.aspx

Anthony Kinney *is a technical sales professional responsible for the Microsoft Desktop Optimization Pack. Anthony came to Microsoft in 2006 with the acquisition of Softricity. While at Softricity, Anthony wrote and designed the first training program for SoftGrid (now App-V). He can be reached at Anthony.kinney@microsoft.com.*