Microsoft App-V 4.5
Client in Stand-Alone Mode

June 17, 2009

Tim Mangan

TMurgent Technologies

Abstract

With the release of Microsoft Application Virtualization (App-V) 4.5, enterprises have three major deployment models available to them for virtualized applications:

- Using a Server Infrastructure dedicated to streaming virtual applications
- Using a Server Infrastructure shared to deliver OS Images, Virtualized, and Non Virtualized applications.
- Using the Stand-Alone Client, with or without a backing File Server.

This White Paper focuses on this last option, and introduces usage that we find very useful, even when an enterprise chooses to implement one of first two options. There are two methods for deploying with the stand-alone client addressed in this paper.

The paper first describes the usefulness, and methods for using the stand-alone client as originally documented by Microsoft. This is typically considered for supporting a very small number remote users, typically with poor network connectivity to a main site. We extend this use case with a second use, using this deployment method as part of our recommended Sequencing (virtual application preparation phase) practices.

"When configured without streaming, the stand-alone client is, in our opinion, the best setup for the initial test of a newly sequenced application."

Tim Mangan, *TMurgent*

The paper then describes a potential full-scale deployment using the stand-alone client in conjunction with file streaming. Importantly, used in this manner, virtual applications may be upgraded without loss of user personalization and customizations.

Finally, the paper describes a method we use to customize the MSIs generated by the sequencer.

Stand-Alone Client without Streaming

Microsoft first introduced the stand-alone client mode in a hot-fix rollup release of version 4.2. Along with the ability for the client to operate without a back-end server, Microsoft created a different way to deliver the virtualized application to the client running in this mode. This involved creating an MSI package for the virtual application when sequencing. This MSI, along with the SFT file, would be used by the client to add the virtual application to the App-V client and fully populate the cache. The MSI would be run on the client, which would create the application access shortcuts and file associations on the underlying PC. These shortcuts and associations make the application appear as if installed, but instead instruct the App-V Client to launch the virtualized application instead. The MSI would also fill the App-V

file cache with the complete SFT file. In 4.2 the MSI creation was a separate tool run after sequencing. Microsoft's intent with this was to provide virtual application support to one-off situations where network connectivity to a server infrastructure was assumed to be of too poor quality or nonexistent.

In the 4.5 release, Microsoft integrated the MSI generation directly into the sequencer. When sequencing, this extra MSI file will be generated if the appropriate checkbox is selected when sequencing. At TMurgent, both in our own practice and in the Masters Level App-V classes we teach, we have made it a standard practice to generate the MSI with every sequence we create.

Using App-V in stand-alone mode really highlights what App-V does. Although you "install" an MSI file, what is really being installed into the OS is only the shortcuts to launch the application plus file-type-associations that create alternative ways for the user to launch the application. The application itself is still fully virtualized. The other implementation modes do the same thing; it's just that they don't use an MSI to do it.

In addition to the use case outlined by Microsoft, we incorporate the stand-alone client without streaming into our sequencing practice. We have found it much more productive for the initial, or "smoke test" of a newly virtualized application to be performed on a stand-alone client. The person sequencing only needs a couple of virtual machines, one for sequencing and another for testing. Productivity of the person sequencing increases because no back-end infrastructure is required, and no need to log onto yet another machine to use the management consoles.

When sequencing an application for the first time, it is typical to need to take several passes through this sequence-test phase. We recommend in our classes that even if everything works perfectly in the first pass, that documentation on how the applications was sequenced (sometimes called recipes[1]) be created/updated and the application be sequenced from that documentation. While testing with the stand-alone client does not fully test everything that a full test deployment (using a client with the dedicated server using RTSP/S or with the shared SCCM server) would accomplish, we find that initial testing in this mode to be extremely productive. We follow-up this testing with a more complete test system that more closely mimics the production environment. It is in this full test deployment that we bring in a designated "application expert" who has sufficient knowledge to test and approve the virtual application for production deployment.

[1] I coined the term "Recipe" for this documentation when I was the Vice President of Technology at Softricity when we built the original SoftGrid product. The recipe was a step-by-step procedure that anyone adequately trained in sequencing could reproduce. I created a group at Softricity to publish recipes for applications just before I left the company. Subsequent to that time the company, and now Microsoft, have preferred to use "Prescriptive Guidance" documents which detail issues with virtualizing an application and a work-around for each issue. Such guidance requires significant interpretation and does not lead to reproducible results. Thus, most companies create their own recipes for their internal use and the term has survived.

The beauty of the stand-alone client is that there is no need for any back-end components. While the reality is that you will store the images on a file share somewhere, you are free to distribute applications to users through any method. This includes ANY traditional Electronic Software Distribution (ESD) system (SMS, Altiris, SCCM, etc), putting it on a DVD and mailing it, or allowing access to the file share directly.

There are some limitations to using the client in this way, which you need to be aware. First, you cannot mix and match client modes between applications. If you want to use the client in stand-alone mode, then it will not contact the Application Virtualization Server (RTSP/S). Second, without streaming you do not have the ability to update an application once it is "installed". An update means uninstalling the old and installing the new. User preferences are lost in that process.

Setting up the Stand-Alone Client for use Without Streaming

We find it easiest to set up the stand-alone client using a script. Repackaging of the App-V Client MSI for silent installation is also a possibility, however we prefer to use the setup.exe provided by Microsoft as it ensures that several pre-requisites are installed on the client and adds them if needed. Below is a single line CMD script that we use in some cases. As no fancy scripting tricks are in use, such a script is easily modified to whatever scripting language you prefer.

```
Setup.exe /s  /v"/qn SWICACHESIZE=\"12144\"  SWISKIPDATASETTINGS=\"false\"
SWIGLOBALDATA=\"C:\AppVirt\Global\" SWIUSERDATA=\"^%APPDATA^%\"
SWIFSDRIVE=\"Q:\" REQUIREAUTHORIZATIONIFCACHED=\"0\"
ALLOWINDEPENDENTSTREAMING=\"1\" AUTOLOADONLAUNCH=\"0\"
AUTOLOADONLOGIN=\"0\" "
```

An explanation of this command line, such that you can make modifications as desired, follows.

Item	Meaning
/s	Setup.exe is generated by InstallShield. The /s option makes the setup.exe portion of the install run in silent mode, eliminating dialog prompts.
/v"…"	The /v option makes the setup.exe extract out an imbedded version of the MSI file and run msiexec against this MSI with whatever is inside the following quotation marks as command line arguments.

	Items within these quotation marks sometimes need escaping for setup.exe to properly interpret them. For example, any quotation marks must be preceded by a backslash. Also % characters must be preceded by a ^.
/qn	Instructs the MSI portion of the install to be performed in quiet mode, with no reboot. (A reboot is only needed if a client upgrade is being performed).
SWICACHESIZE	Sets a fixed maximum size for the sft cache in MB
SWSKIPDATASETTINGS	Seemingly required for other SWI to be interpreted
SWIGLOBALDATA	Optional. We use a known location so that testing on WinXP and Vista use the same location in case we need to troubleshoot.
SWIUSERDATA	Optional. In this case we are only showing how to reference an environment variable.
SWIFSDRIVE	Optional. Again this is the default value anyway.
REQUIREAUTHORIZATIONIFCACHED	Turns off client need for app authorization.
ALLOWINDEPENDENTSTREAMING	Enables the client to receive the SFT without a publication server.
AUTOLOADONLAUNCH	Turns off new background streaming.
AUTOLOADONLOGON	Turns off new background streaming.

Using the Stand-Alone Client Without Streaming

Once the App-V client is installed this way, the MSI and SFT that are output by the sequencer must be made available. These can be copied to the client system through any method (email, USB or CD/DVD device, etc) or be made available on a file share or mounted drive. While it is possible send the MSI and reference the SFT remotely, we prefer to keep these two files in the same place.

The MSI is then run. By default this will display a multi-page dialog which the user just clicks through without any chance of customization. The MSI will check that an appropriate version of the App-V client is present and configured, and that this virtual application is not already “installed”. It is necessary to use the control panel feature (called “Add/Remove Programs” or “Programs and Features” depending on the version of the operating system) to “uninstall” the old version if the same package name/GUID is used.

The MSI built by the sequencer includes the ICO and OSD files needed by the client, as well as the _manifest.xml file. It does not include the SFT because the MSI format has a hard limit of 2GB for

embedded files. Because some SFT might exceed this limit, Microsoft decided to always keep the SFT external. (A decision I do not agree with, but then again I don't run the world).

It is important to note that the application publishing information (details about the virtual application shortcuts and file type associations) are duplicated both in the OSDs and in the _manifest.xml. While the dedicated server reads this information from the OSDs, clients in stand-alone mode (or configured for delivery via SCCM) read this information from the _manifest.xml. *This has important implications to manual editing of publication information.* Any manual edits post sequencing should be made to both files (and of course re-opened in the sequencer so that the files internal to the MSI are updated).

The MSI will unpack the embedded files in a new subfolder in the global area, and use the client SFTMIME command to add the package to the client. The client will then cache the OSD and ICO files in the global area. When adding the package using the SFTMIME command, it will use the /GLOBAL option which applies the publishing to any user logged onto the computer. *This has implications to shared computer systems such as a Terminal Server or "hotdesk".* Since only a user with administration privileges can usually install an MSI and the client has no server to provide per-user authorization, using the /GLOBAL option is used to make the virtual application available to all users of this computer.

Finally, the MSI will load the SFT cache with the contents of the SFT. Whether or not the package was sequenced with both feature blocks, the entire SFT will be cached. Note that it is necessary to ensure that the stand-alone client cache be sufficient size. Should the maximum size be reached, old bits will be pushed out of the cache (based on least recent usage), potentially breaking the old application as the client is configured in a way that it cannot locate the original SFT to refill the cache. This is usually not an issue for the sequencer test station, but is a concern for production stand-alone use.

Stand-Alone Client with File Streaming

Poorly documented and understood, is another way of configuring the App-V Client, for stand-alone operation with File Streaming. We believe that this mode has interesting production use possibilities due to the combination of benefits and restrictions it imposes.

*We are currently unaware of **any** customer using this option in production, we believe this to be due to the lack of documentation and understanding (leading to this white paper). We regularly use this in our labs and are happy with the results.*

Use of the Stand-Alone client without streaming has two limitations addressed by use in this mode:

1. The inability to update a virtual application without the user loosing their preferences/customizations.
2. The inability for the client to recover from a too-small sft cache file.

The beauty of the stand-alone client is the absence of any back-end servers. In practice, you still will want to store the original sequenced package somewhere. You still need to have the MSI and SFT available to distribute to stand-alone clients and you want to retain the other files so that you can re-open the package in the sequencer for the future.

By making this folder and contents available as a read-only file share, configuring the stand-alone client to support file streaming, and installing the "virtual applications" on this client differently, we solve the two issues listed above. This enables us to achieve more of the important benefits of the client in RTSP/S or SCCM mode than we could before.

Furthermore, we can add any push style deployment tool, such as SMS or third party tools, to push out the virtual application MSIs to the client with a script. We find many customers that use SMS (or other tool) today and are reluctant to move to SCCM R2 to deliver virtual applications because the change would have such a large impact on their operations.

Setting up the Stand-Alone Client for use With Streaming

The script we use to set up the client in this mode follows:

```
Setup.exe /s  /v"/qn SWICACHESIZE=\"12144\"  SWISKIPDATASETTINGS=\"false\"
SWIGLOBALDATA=\"C:\AppVirt\Global\" SWIUSERDATA=\"^%APPDATA^%\"
SWIFSDRIVE=\"Q:\" REQUIREAUTHORIZATIONIFCACHED=\"0\"
ALLOWINDEPENDENTSTREAMING=\"1\" AUTOLOADONLAUNCH=\"1\"
AUTOLOADONLOGIN=\"0\" "
```

The small change to AutoLoadOnLaunch instructs the client to re-fill anything for this application that was pushed out of the cache when the user launches the application. While technically not needed, we prefer this setup. It is in "installing" the virtual application package that we use different procedures to achieve the benefits of streaming.

Using the Stand-Alone Client With Streaming

When working with a client in this mode, we prefer to keep the MSI and SFT assets stored in a central file share and do not copy them to the client workstation. We also make a copy of a script that will be used to perform the virtual application install, modify it to reference this specific package and place it in the same folder on the server share. Again, this script could be used as part of a push technology (SMS, etc) to automate the install centrally. Below is an example cmd script version.

A better script that does not require editing can be found in the tools section of our website at www.tmurgent.com . This VBscript will take a command line argument of the MSI file on the file share, or will prompt you for that file. The script will parse the files in that folder to determine if there are any Dynamic Suite Composition (DSC) dependencies and warn you if there are any before performing the install.

We can edit the script to replace the reference names for the msi and sft and deposit the copy in the folder with those files to keep this script simple. In practice, a smarter script would be employed that scans a given folder for the msi and sft names; we are showing this script to avoid specific scripting language issues in this paper.

```
msiexec.exe /i
\\Roadhog\Content\DSC\IE\Tim_DSC_IE_BASE_FOR_PLUGINS_iebase.t33\Tim_DSC_IE_BA
SE_FOR_PLUGINS_iebase.t33.msi   MODE=STREAMING
OVERRIDEURL=\\\\Roadhog\\Content\\DSC\\IE\\Tim_DSC_IE_BASE_FOR_PLUGINS_iebase
.t33\\Tim_DSC_IE_BASE_FOR_PLUGINS_iebase.t33.sft   LOAD=TRUE /q
```

When run, this script will run the msi from the file share. As in the case of running it without streaming, the _manifest.xml, OSD, and ICO files are unpacked into a sub-folder of the global area. SFTMIME is run with the global option against the manifest, however due to the MODE=STREAMING option, the client is being instructed to check the source sft each time the application is launched for potential updates. The source will be on our file share (as indicated by the OVERRIDEURL). The LOAD=TRUE option simply tells the MSI to load the application 100% into the cache immediately after adding the application. The use of the /q option makes this install complete silently.

When the application is added in this way, we can now update the package on client systems without loss of personalization/customizations or the need to touch the client system.

UNDERSTANDING OSD SOURCE, ASR, AND OVERRIDEURL

The OSD file contains an HREF reference to the SFT file in the form of PROTOCOL:\\Server:port\Path\Sftfile.sft". In a traditional dedicated server setting this would look something like "RTSP:\\%SFT_SOFTGRIDSERVER%:554\Vendor\Package\Package.sft". When a user launches an application from a shortcut, the OSD is read and this reference is used unless…

If the Windows Registry of the client has a value for ASR, this will override the "PROTOCOL:\\Server:port portion of the OSD. This ASR is a single setting for all applications and all users on the client PC. Unless…

Additionally, if the Windows Registry of the client has a value called OVERRIDEURL for this application (shortcut), this will override the entire HREF of the OSD.

In summary, the client will look for an OVERRIDEURL, and if none then look for an ASR, and if none use the OSD as is.

Updating Applications on Stand-Alone Client with Streaming

When it comes time to update an application deployed in this way, we can use the sequencer to open the package. The process on the sequencer is the same as it would be for any deployment model: Open for Upgrade, run the Sequencing Wizard and update the application, then save.

This produces a new sft file with a new name. By default, the sequencer would add an _2 to the basename of the SFT (or replace an _N with an _N+1, such as _3) each time. When deploying with either of the server models, this file is used with this name, and changes are made on the server to redirect the client to use the new name.

In our case, however, we will need to modify the filename to be the same as the original sft. Although App-V clients open this file in read-only mode, and then only for a short while, we recommend only overwriting the server share copy (after backing it up, of course!) during a time that clients are less likely to be launching the application, such as nights or weekends or designated maintenance windows.

Upon the user requesting to launch the application, the client will check datestamps of the sft and re-stream for updates if needed. Because this is file streaming, the entire new sft will be streamed down[2]. This might make Stand-Alone with Streaming less useful for poorly connected clients but we are still bullish about this mode for both central site and branch deployments.

To avoid the need to rename the updated sft file, we note that the sequencer has an option setting to turn on the version renaming. If the application is only deployed by stand-alone client with streaming, you could turn off this feature.

Limitations

In this section, we address what we see as the limitations of deploying in this mode.

1. We are concerned with overwriting the existing SFT due to potential client issues, but have not seen this to be a problem. Thus we urge caution in deciding when to deploy the updates. We have not tested, but suspect that we could place the _2.sft file on the share and then push out a script to the client to modify the HKLM registry reference to the application "original source" to point to the new name for the next time the user launches. This would be safer in those environments with a good push technology present. Because this involves changing an existing registry value, we do not believe GPOs to be an effective solution for this push.
2. In the case of an application package being updated that affects publishing information, we would recommend saving as a new package, "uninstalling" the old virtual application, and then "installing" the new. Changes to shortcut locations and file type associations, or even entirely new OSD applications entry points are not updatable using the process we outlined in the previous section.
3. Features of the Dedicated Streaming Server are not available. This includes features such as per user authorization, metering, and licensing. But this is no different than the regular stand-alone client or using the client with SCCM R2.
4. Installing the virtual package normally requires admin rights. If you are not pushing out using ESD, consider how users will be able to run an installer. If you are using a file share and pointing users to it, we note that it is possible to achieve some level or authorization for the "install" by

[2] In the 4.5 RC version, Microsoft had an option to produce a DFST file that contained only the changes from the original SFT. This feature was dropped in the RTM release and it is unclear if it will be included in a future release. If it is, it is currently unknown as to how we would be able to direct the client to the DSFT file.

utilizing ACLs and Active Directory group assignments. These ACLS would be placed on the SFTs and MSIs (denying read access to these files)

5. The MSI "installs" using the /GLOBAL flag of sftmime. This means that if this is a shared computer, all uses on the machine will have the application published to them. If an admin performs the install, loading the SFT into the cache, server side ACLs will not help. Instead, the application would need to be sequenced with ACL support enabled and internal package components protected. We do not particularly like this approach as the user would launch the virtual application and then get an application file permission error which would appear differently on different applications.

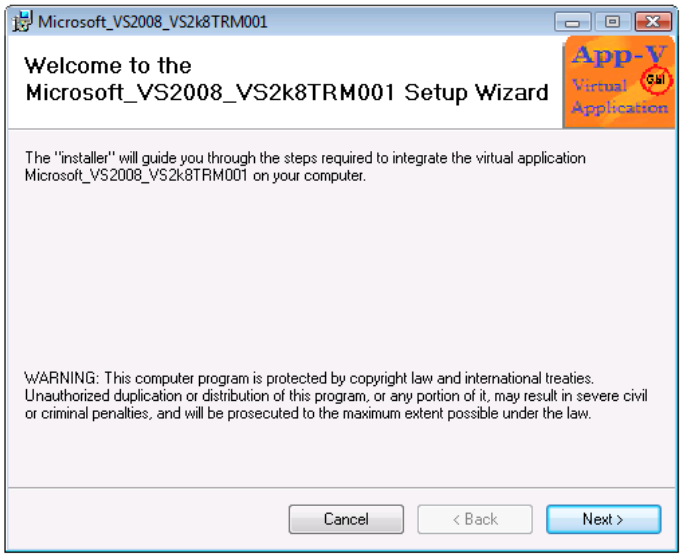
It is possible to alter the behavior of the MSI. The sequencer uses a template MSI to build the final MSI. We have successfully created a modified template and replaced it on our sequencer. In addition to altering the graphics, we changed the wording (to remove the word "Install") and added a new command line option to the resultant MSI that allowed the package to be added without the /GLOBAL flag.

We also note that we look with keen interest at AppLocker, to be released as part of Server 2008 R2 and Windows 7 as a possible future authorization methodology. Unfortunately, these products are not yet released and the implementation is not extended to clients running older operating systems. Third party equivalents do exist today (Tricerat Simplify Lockdown comes to mind) however these products have not been widely deployed to date.

Modifying the Virtual Application Package MSI

The sequencer uses an MSI file as a template for the output MSI produced by the sequencer. We have had success in modifying this template for both cosmetic and practical purposes.

Using an MSI re-packager, (such as Ocra, Camwood, Acresso, or Wise) we can open the template MSI and make modifications. Our template usually modifies the graphics and text for when someone runs the MSI manually. We like to change the wording of things like "install" to make it clear that it is a virtual install and not a regular install.



The other change our template has is that we add a command line option such that the MSI can be installed with a /NOGLOBAL option, in which case the SFTMIME command will be run without the /GLOBAL option. This means that only the user running the install will have the shortcuts published to their desktop account. This might be useful in some shared use situations, however keep in mind that the user still needs install rights.

We believe that it should be possible, with sufficient effort, to create a template that would publish to some kind of pre-determined list of users. We have not done the work for this because ultimately we believe that client changes to check for list changes over time would be necessary. Thus, for now we recommend that customers needing per-user application publishing on multi-user machines stick with the Dedicated Server infrastructure.

We should also note that the MSI is nothing more than a convenient wrapper for a script that runs two sftmime commands. The truly ambitious could roll their own stand-alone deployment system to clients in stand-alone mode without resorting to MSIs.

Conclusions

In addition to the poorly connected user scenario described by Microsoft, we find the stand-alone client useful in two different scenarios.

First, the stand-alone client is the ideal way to quickly test a newly sequenced application before sending it to an “application expert” to verify operation.

Second, when configured with streaming and combined with an existing push technology, the stand-alone client creates a third style of enterprise deployment model for virtual applications. This model consists of the least number of dependent back-end components and provides for flexibility of using any push technology that the enterprise already uses. Furthermore, the benefits of this mode appear to be “on par” with those provided by deploying through SCCM R2. While there may be additional benefits to using SCCM R2 beyond virtual applications, we do not see any if your interest is only in virtual applications.

Finally, nothing can replace the full benefits of using the dedicated server approach. Application metering and per user authentication and licensing are only available through the use of RTSP/S. Customers requiring this support should stay with the dedicated server infrastructure for production use.