

App-V Extensibility Today Before the SDK

White Paper Descriptor

This document presents administrators with the available tools to automate common tasks in App-V 4.5, including sample scenarios and examples using the tools.

Copyright © 2008 MICROSOFT CORPORATION

[Overview 3](#_Toc209414198)

[SFTMIME.EXE 4](#_Toc209414199)

[App-V WMI Provider 7](#_Toc209414200)

[Example Scenarios 10](#_Toc209414201)

[SFTTray.exe 13](#_Toc209414202)

[SFTList.exe 15](#_Toc209414203)

[SFTSEQUENCER.EXE 16](#_Toc209414204)

[Switches 16](#_Toc209414205)

[Examples 18](#_Toc209414206)

[OSD Scripting 19](#_Toc209414207)

[Examples 19](#_Toc209414208)

[Conclusion 20](#_Toc209414209)

# Overview

The new enhancements in App-V 4.5 offer administrators many choices for managing virtual applications. However, in some cases the new deployment and manageability features do not address every infrastructure need. Today, before a Microsoft Application Virtualization 4.5 (App-V) software development kit (SDK) is available, there are many tools included in the App-V 4.5 product. These tools can greatly reduce administrative overhead and provide greater levels of support than traditional GUI-based management. App-V enhancements in manageability include Command Line Interfaces (CLI), a WMI provider, and several other tools. This document will describe the management capabilities available today and provide practical examples to get started.

**Audience**

This document is intended for an App-V infrastructure administrator who wants greater control over the manageability of their App-V environment. It is also intended for Independent Software Vendors (ISV) that want to create utilities to extend on App-V as a platform.

**Topics**

* SFTMIME.EXE
* App-V Windows Management Instrumentation (WMI) Provider
* SFTTRAY.EXE
* SFTLIST.EXE
* SFTSEQUENCER.EXE
* OSD Scripting

After completing this document administrators will have a thorough understanding of the capabilities of App-V’s CLI management tools that are available in the current release, before the development of an SDK. This knowledge will enable administrators and ISVs to begin automating common tasks and more effectively manage an App-V infrastructure. Additional documentation will be referenced when necessary.

## SFTMIME.EXE

SFTMIME.EXE is a familiar tool for users of the previous versions of App-V. This tool provides the most powerful set of features for managing groups of App-V clients. In this version of the App-V client, SFTMIME.EXE has been enhanced with several new switches making it a more comprehensive tool for automating common administrative tasks. This document provides an overview of the tasks that can be accomplished with SFTMIME.EXE and gives some common examples. For a complete SFTMIME.EXE reference please view the **App-V Planning and Deployment Guide** at: <http://go.microsoft.com/fwlink/?LinkId=122063>

Using SFTMIME mimics the tasks that can be accomplished using the App-V Client Management Console, but since it is command line-based, it can be placed in batch files and called through scripts to perform tasks on several machines remotely and automatically. Through the use of a logon script, SFTMIME commands can be used to target specific machines to perform certain administrative tasks. SFTMIME commands are comprised of verbs, objects, parameters, and switches. The following documentation is a list of objects that are available and some common tasks that can be performed using those objects.

SFTMIME.EXE in App-V 4.5 has changed to support globalization. SFTMIME now writes directly to the console instead of stdout. In order to export the output of an SFTMIME command to other commands, you must first write SFTMIME output to a file (use /log parameter) and then use this file for input to the next command.

Example:

sftmime query obj:app /log %temp%\mime.log

findstr “My Favorite App” %temp%\mime.log

**Application (APP)**

The application (APP) parameter enables administrators to manage the applications on App-V client computers. Using the APP parameter, administrators can automate common tasks like adding, deleting, and locking applications.

Scenario: An organization has deployed App-V and is using the benefits of application virtualization. However, there are a few mission-critical applications that the organization wants s to ensure are always in cache. The following command will “lock” the application in cache, so that it can only be removed after unlocking it.

**Sftmime lock app:”Microsoft Office 2007”**

NOTE: Locking an application will lock the entire package and not one individual application.

**PACKAGE**

The Package parameter enables administrators to manage packages on App-V client computers. Packages can consist of one or more applications. Using the PACKAGE parameter, administrators can automate common tasks like adding, deleting, and publishing packages. These tasks will be most commonly used in environments where no Management Server is being used for publishing refresh. One task that can be useful is to publish and unpublish an entire package. This gives administrators control over all the application’s shortcuts and file type associations from one command. This can be implemented in a custom solution where no publishing refresh is being performed on the clients, like MSI-based installation, or Streaming Servers with no Management Server for publishing refresh, such as when integrating App-V with an ESD system.

Scenario: An App-V Lightweight infrastructure was implemented using only Streaming Servers, with no Management Servers. Since Streaming Servers do not perform the publishing refresh operation, the packages will need to be published manually to the App-V clients. An administrator could script the add process of each package and specify the package location using the **OVERRIDEURL** switch to specify the sft file location.

**SFTMIME ADD PACKAGE:"Microsoft\_Visio\_2007\_MNT" /MANIFEST \\\\appvms\\content\\Microsoft\_Visio\_2007\_MNT\\Microsoft\_Visio\_2007\_MNT\_manifest.xml /OVERRIDEURL \\\\appvms\content\Microsoft\_Visio\_2007\_MNT\Microsoft\_Visio\_2007\_MNT.sft /GLOBAL**

NOTE: Users will need appropriate permissions to access the location of the package, have the ADD application permissions, and AllowIndependentFileStreaming will have to be enabled.

**OBJECT**

The OBJECT parameter enables administrators to manage application, file type associations, packages, and publishing servers. This parameter enables administrators to query, delete, and clear specific object types on an App-V client. When used with the query verb, the Object parameter can be used to inventory clients. This information is similar to what can be gathered using the WMI provider.

Scenario: When troubleshooting, administrators want to be able to obtain a list of objects published to the clients through App-V. The following commands will list out objects by their type.

**SFTMIME QUERY OBJ:APP /GLOBAL /LOG:C:\APPS.LOG**

**SFTMIME QUERY OBJ:PACKAGE /GLOBAL /LOG:C:\PACKAGES.LOG**

**SFTMIME QUERY OBJ:TYPE /GLOBAL /LOG:C:\FTAS.LOG**

**SFTMIME QUERY OBJ:SERVER /LOG:C:\PUBSERVERS.LOG**

**SERVER**

The SERVER parameter enables administrators to manage publishing servers on a client. Tasks that can be performed using this parameter would include adding, configuring, and deleting publishing servers on clients. Using this parameter to automate changes to publishing servers can reduce the administrative overhead for this process. One important task that can also be accomplished is forcing a publishing refresh.

Scenario: The original design of the App-V infrastructure used RTSP with non-secure publishing servers. After configuring a PKI infrastructure and enabling RTSPS with secure publishing servers, an administrator wants to automate the process of changing the clients to communicate using RTSPS.

**SFTMIME CONFIGURE SERVER:DISPLAYNAME /PORT 322 /TYPE RTSP /SECURE**

**TYPE**

The TYPE parameter is used to configure file type associations on the client. Using this parameter can resolve file type association problems quickly without involving a publishing refresh or modifying the package. Modifying file type associations should be done with care as it could introduce inconsistency in the client file type association settings that would be more difficult to support.

Scenario: An administrator is testing an application after being given the package from the sequencer. The administrator notices that double-clicking on a file that is normally associated with the application does not properly launch the application. Before providing feedback back to the sequencer or making the change permanently to the package, the administrator first wants to add the file type association on the client computer to ensure that changing the file type associations in the package will correct the problem. The following command will add a file type association for an application.

**sftmime add type:docx /app "Microsoft Word 2007" /description "Word 2007 Document"**

## App-V WMI Provider

App-V includes a client WMI provider to enable querying, or retrieval actions, of common information from virtual packages and applications. The App-V WMI provider includes an application class and a package class. If combined with SCCM this provider allows an administrator to render rich reporting of virtual applications. In cases where SCCM is not used, or not available, the WMI provider can still provide access into the same data in the virtual environments. The current implementation of WMI provider is a read-only “pull” action only and cannot be used to configure App-V clients. Additional information on Windows Management Instrumentation is available at:

<http://msdn.microsoft.com/en-us/library/aa384642.aspx>

Accessing the WMI provider can be accomplished programmatically, via scripts, and Windows PowerShell. This document will focus on Windows PowerShell for examples.

**Accessing WMI from Windows PowerShell**

<http://www.microsoft.com/technet/scriptcenter/topics/msh/mshandwmi.mspx>

**Windows PowerShell Syntax**

<http://go.microsoft.com/fwlink/?LinkId=106434>

**Scripting in WMI (VBScript)**

<http://msdn.microsoft.com/en-us/library/aa393262(VS.85).aspx>

**App-V Package WMI Class**

The Package class allows administrators to gather information about the various packages that have been deployed in the App-V infrastructure. Administrators can use the available properties to inventory machines with specific packages, find out the version of the package that is in cache, and several other properties. This information can be useful in providing quick reports for licensing, check status on application loading, and view the size of the packages in cache. The table below depicts the properties available in the Package class.

**Package Properties**

| Property Name | Data Type | Access Type | Qualifiers | Description |
| --- | --- | --- | --- | --- |
| Name | String | Read-only | None | The user-friendly name of the virtual package. |
| Version | String | Read-only | None | The version of the virtual package. |
| PackageGUID | String | Read-only | Key | The GUID identifier of the package configuration and source files. |
| SftPath | String | Read-only | None | The file path of the SFT file. |
| TotalSize | UInt64 | Read-only | None | The total size of the virtual package in kilobytes. |
| CachedSize | UInt64 | Read-only | None | The total size of the cache for the virtual package in kilobytes. |
| LaunchSize | UInt64 | Read-only | None | The total size of the virtual package primary feature block in kilobytes |
| CachedLaunchSize | UInt64 | Read-only | None | Total size of the virtual package primary feature block that has been cached in kilobytes. |
| InUse | Boolean | Read-only | None | True if any virtual application in the virtual package is running; otherwise false. |
| Locked | Boolean | Read-only | None | True if the virtual package is locked; otherwise false. |
| CachedPercentage | UInt16 | Read-only | None | The percentage of cache files. Based on the following formula: CachedSize / TotalSize x 100. |
| VersionGUID | String | Read-only | None | The GUID identifier of the package version. |

**App-V Application WMI Class**

The application class allows administrators to gather information about the applications that have been deployed in the App-V infrastructure. Administrators can use the available properties to inventory machines with specific applications, evaluate virtual applications currently running, and learn when applications have been launched. This information can be used to provide reports showing when specific applications are being used, find a package associated with an application, and gain status on updated application versions.

App-V has built in reports in a full infrastructure where a Management Server is in place and the clients report usage information. The App-V reports can be used by administrators to find general usage information but don’t provide the customization available with the WMI providers. Also, in a lightweight or stand-alone infrastructure, where the clients don’t report usage information to a data store, the reporting feature would not be available and the WMI provider would be the only way collect information on virtual applications.

For example, with the information provided in the Application WMI Class an administrator could retrieve information on what users have Word for Windows 2003 and how many users have Word for Windows 2007. This information is not only useful for licensing of applications but could be used in gaining knowledge on general usage and learning curves of new versions.

A new feature in App-V 4.5 is the ability to have an alternate OSD path set for the client. Using the application class of the WMI provider, an administrator can discover where the users are pulling their OSD files. This information can be critical in determining proper branch office configurations.

The table below depicts the properties available in the application class:

**Application Properties**

| Property Name | Data Type | Access Type | Qualifiers | Description |
| --- | --- | --- | --- | --- |
| Name | String | Read-only | Key | The display name of the virtual application. |
| Version | String | Read-only | Key | The version of the virtual application. |
| PackageGUID | String | Read-only | None | The GUID of the package that the virtual application is associated with. |
| LastLaunchOnSystem | DateTime | Read-only | None | The last date and time that the virtual application was launched. |
| GlobalRunningCount | UInt32 | Read-only | None | A count of the running instances of the virtual application that were started directly. |
| Loading | Boolean | Read-only | None | True if the virtual application is being started: otherwise false. |
| OriginalOsdPath | String | Read-only | None | The original file path of the OSD file that was registered with the App-V client. |
| CachedOsdPath | String | Read-only | None | The file path of the OSD file if the App-V client has cached the OSD file locally. |

### Example Scenarios

The following examples demonstrate some uses of the WMI provider to help administrators better manage their environment.

Scenario 1:

You have deployed a virtualized application (could be anything from Office to Adobe Reader to a Java plug-in). The vendor releases an important security patch. You need to validate that all instances of this application are updated with the patch.

After deploying the updated package that includes the important security patch, you want to create a report on which clients have received the update.

The App-V client WMI provider permits remote administrative access to query your computers to get this information.

Code in PowerShell:

# fill in the information in {braces}

# need a list of computer names to query

$computerList = @('{computer1}', '{computer2}')

# need package guid that we will be querying for

$pkgGUID = '{PackageGUID}'

# need a package version that we will want to compare.

# VersionGUID may be a better attribute if complex package upgrade scenarios exist.

$desiredVersion = '{PackageVersion#}'

#loop through each computer and check the version for our package

foreach ($remoteComputer in $computerList) {

$pkg = get-wmiobject -namespace root/microsoft/appvirt/client `

-query "select \* from Package where PackageGUID='$($pkgGUID)'" `

-computer $remoteComputer

#pkg should contain one object... Validate GUID and look at the Version and VersionGUID attributes.

if (-not $pkg -or $pkg.PackageGUID -ne $pkgGUID) {

Write-Host "We did not find the desired package on client computer $remoteComputer"

}

if ($desiredVersion -ne $pkg.Version) {

Write-Host "$remoteComputer is not yet up to date"

}

}

Scenario 2a:

You have deployed Microsoft Office or another application suite. There is an application or applications that are part of the package that are confusing users (e.g. VBA projects or Digital Certificates in Microsoft Office). You would like to remove the confusing and possibly unnecessary applications from being published to the users. Before removing the application(s), you need to make sure no-one has run it recently.

Code in PowerShell:

# need a list of computer names to query

$computerList = @('{computer1}', '{computer2}')

# need an application name and version

$appName = '{application name}'

$appVer = '{application version}'

#loop through each computer and check the version for our package

foreach ($remoteComputer in $computerList) {

$app = get-wmiobject -namespace root/microsoft/appvirt/client -query `

"select \* from Application where Name='$($appName)' and Version='$($appVer)'" `

-computer $remoteComputer

#app should contain one object...

Write-Host "Application last launched on $remoteComputer on $($app.LastLaunchOnSystem)"

}

Scenario 2b:

You have deployed a new package that supersedes an old package. It is time to remove the old package; however you need to make sure no-one is still using it. You can query the virtual application LastLaunchOnSystem attribute for all applications in a package to determine how long it has been since it has been used. This will ensure that removing the old package won’t cause any problems for users:

Code in PowerShell:

# need a list of computer names to query

$computerList = @('{computer1}', '{computer2}')

# need package guid that we will be querying applications for

$pkgGUID = '{PackageGUID}'

#loop through each computer

foreach ($remoteComputer in $computerList) {

$app = @(get-wmiobject -namespace root/microsoft/appvirt/client -query `

"select \* from Application where PackageGUID='$($pkgGUID )'" `

-computer $remoteComputer)

#$app will be an array of applications

$app | foreach {Write-Host "Application '$($\_.Name)' last launched on $remoteComputer on $($\_.LastLaunchOnSystem)"}

}

Scenario 3:

You need a list of packages that exist on a client computer and how much of each is in cache.

Code in PowerShell:

# need a list of computer names to query

$computerList = @('{computer1}', '{computer2}')

#loop through each computer and check the version for our package

foreach ($remoteComputer in $computerList) {

$pkg = get-wmiobject -namespace root/microsoft/appvirt/client `

-class Package -computer $remoteComputer

Write-Host " -- $remoteComputer --"

$total = 0

$pkg | foreach {Write-Host "$($\_.PackageGUID) $($\_.CachedSize / 1024) kb in cache"; $total += $\_.CachedSize}

Write-Host "--- $remoteComputer - total of $($total / 1024 / 1024)MB in cache"

}

# SFTTray.exe

The SFTTRAY.EXE application has been available in previous versions of App-V and it was commonly used to automate App-V administrative tasks. SFTTRAY.exe is the actual target application that is called from App-V application shortcuts and appears in the System Tray Notification Area. Although the command does not contain the rich feature set of SFTMIME, it has very well-known switches that complete several common tasks. SFTTRAY can be used in many cases to quickly develop a script for automating tasks; however it doesn’t provide support for many of the new App-V 4.5 features (OVERRIDEURL, Autoload, and others), and can’t be used to query the client for status information. SFTTRAY.EXE can be used for preloading the App-V cache, performing a publishing refresh, and launching applications. A list of the switches is listed in the table below.

**Switches:**

**Note**: All Sfttray.exe command-line options are case sensitive. These options include the application name.

| Switch | Usage | Description |
| --- | --- | --- |
| /launch | /launch "application name" /launch "file\_name.osd" | Specifies an application to be launched. You can specify the application by using a relative path, an absolute path, the path of a network share, or a URL. |
| /load | /load "application name" /load "file\_name.osd" | Specifies an application to be loaded in cache. You can specify the application by using a relative path, an absolute path, a UNC path, or a URL. |
| /loadall | /loadall | Loads all applications completely in cache. |
| /hide | /hide | Hides the App-V client in the notification area. Must be used together with and before the /launch, /load, or /loadall option. |
| /sftfile | /sftfile "file://path/file\_name.sft" | Specifies a .sft file to import directly from the file. Must be used together with the /load option. |
| /help OR /? | /help /? | Displays Help. |
| /exit | /exit | Exits all App-V -enabled applications and removes the icon from the notification area. |

**Examples:**

The following code launches and loads the App-V example application by using the application name. Using the /hide switch the command also does this by hiding the system tray status bar from the user.

sfttray.exe /hide /launch "App-V Example Application"

sfttray.exe /load "App-V Example Application"

The following code starts and loads the App-V example application by using the application name and the version. This information can be acquired in the App-V Management Console in the properties of an application or by using an SFTMIME command.

sfttray.exe /launch "App-V Example Application 7.0"

sfttray.exe /hide /load "App-V Example Application 7.0"

The following code starts and loads the App-V example application by using the UNC path of an OSD file. This type of command is useful to a network administrator who needs to test a script in an alternate OSD file.

sfttray.exe /launch "\\SGVAS01\CONTENT\sft\_ea70.v1\_ea.osd"

sfttray.exe /load "\\SGVAS01\CONTENT\sft\_ea70.v1\_ea.osd"

NOTE: This example would require administrative privileges on the local computer or the “Change OSD” permission on the client.

# SFTList.exe

SFTlist.exe is a command line utility that can be used to change the log level of the client or to reset the client log file. This tool is the only method for managing the log from command prompt or script.

| Switch | Description |
| --- | --- |
| /resetlog | Moves the existing log file to a backup location and renames it to sftlog0001.txt and incrementally from there. |
| /verboselog | Change the log level to Verbose. |
| /linfolog | Change the log level to Information. |
| /warninglog | Change the log level to Warning (the default on install). |

# SFTSEQUENCER.EXE

The App-V Sequencer sequences applications into packages so that they can be run in a virtual environment. When an application is sequenced, it can either be streamed from the App-V Streaming Servers to computers running the App-V Client or delivered as an .MSI file. Traditionally, sequence engineers have used the GUI version of the Sequencer to individually go through the Sequencing Wizard to package their applications. SFTSEQUENCER.EXE is a command line version of the App-V Sequencer. This allows a sequence engineer to fully package applications for a virtual environment without using the GUI wizard.

This tool could be used to take preexisting, automated installations and quickly perform the sequencing process on them. One of the primary use case scenarios is when a company has already developed an automated process to physically install their applications. This could be for use with an electronic software delivery (ESD) solution such as SCCM. In this scenario the company doesn’t want to go through the traditional sequencing process as they have already invested time in developing an installation for each application with the appropriate settings. SFTSEQUENCER.EXE would allow them to use the automated setup and put that existing installer and settings file through the sequencer without having to open and use the GUI.

Although the Command Line Sequencer is a convenient way to automate the packaging of an organization’s existing application into a virtual environment, the sequence engineer’s expectations need to be properly set. It is possible that in an organizations application library, some of these packages may need fine tuning for the virtual environment. However, many of the applications will work and the organization can more quickly sequence their applications for virtualization.

Even in the event that an application does not sequence 100% correctly by using the Command Line Sequencer, the sequence engineer is still saving valuable time by first using the CLI. Afterwards the package could be opened for upgrade in the GUI Sequencer and tuned to run properly.

NOTE: The use of the SFTSEQUENCER.EXE does not allow the sequencing engineer to configure Primary and Secondary Feature Blocks for the package.

### Switches

Use the following Application Virtualization Sequencer parameters to sequence an application and to upgrade a sequenced application package at the command prompt. In the Microsoft Application Virtualization Sequencer directory, you would enter **SFTSequencer.exe**, followed by the appropriate parameter.

| Switch | Description |
| --- | --- |
| **/HELP or /?** | Use to display the list of parameters available for command-line sequencing. |
| **/INSTALLPACKAGE or /I** | Use to specify the installer or a batch file for the application to be sequenced. |
| **/INSTALLPATH or /P** | Use to specify the package root directory. |
| **/OUTPUTFILE or /O** | Use to specify the path and file name of the SPRJ file that will be generated.  **Note:** **The /OUTPUTFILE** parameter is not available when opening a package that you do not intend to upgrade. |
| **/FULLLOAD or /F** | Use to specify whether to put everything in the primary feature block. |
| **/PACKAGENAME or /K** | Use to specify the package name of the sequenced application. |
| **/BLOCKSIZE** | Use to specify the SFT file block size as it is streamed to the client. |
| **/COMPRESSION** | Use to specify the method for compressing the SFT file as it is streamed to the client. |
| **/MSI or /M** | Use to specify generating a Microsoft Windows Installer (MSI) for the sequenced application. |
| **/DEFAULT** | Use to specify the default SPRJ file to use when launching a sequenced application package. |
| **/UPGRADE** | Use to specify the directory in which the sequenced application package about to be upgraded is located. |
| **/DECODEPATH** | Use to specify the directory to unpack the SFT file. |

### Examples

**How to Sequence a New Application Package Using the Command Line**

This example will create a package and load all of the blocks of data into the Primary Feature Block.

SFTSEQUENCER /INSTALLPACKAGE:"Path to installation program" /INSTALLPATH:"the package root directory " /OUTPUTFILE:"The name and path of output project file" /FULLLOAD

**Note:** If the installer or Windows Installer package has a graphical user interface, it will be displayed after you specify the command-line parameters.

**How to Open a Sequenced Application Using the Command Line**

This example will open an existing package and create an MSI install. This could be useful in automating the process of creating MSI-based installations for stand-alone clients from pre-App-V 4.5 packages.

SFTSEQUENCER /OPEN:"the sequenced package project file (SPRJ)" /MSI

**Note:** If the installer or Windows Installer package has a graphical user interface, it will be displayed after you specify the command-line parameters.

**How to Upgrade a Sequenced Application Package Using the Command Line**

This example will upgrade a package with an update or additional application in the suite, save it as a new package version, and compress the new package file.

SFTSEQUENCER /UPGRADE:"The path to the project file (SPRJ)" /INSTALLPACKAGE:“Location of application (upgrade) to be applied to the package” /DECODEPATH:“the directory in which to unpack the SFT file” /OUTPUTFILE:“Location and name of package to be created (SPRJ)” /COMPRESSION

**Note**: If the installer or Windows Installer package has a graphical user interface, it will be displayed after you specify the command-line parameters.

# OSD Scripting

Modifying the OSD file of an application can be a very powerful tool for an experienced sequence engineer. By using the scripting capability within an OSD file, any number of commands or actions can be executed before, during, or after an virtual application has been launched. These scripts can be used to set up or alter the environment before an application executes or can even be used to clean up the environment after the application terminates. Additional detail and examples are available in the App-V Sequencing Guide at: <http://go.microsoft.com/fwlink/?LinkId=127120>

### Examples

When troubleshooting virtual applications, it is often important to gain access to the virtual environment the application is running in. The following example launches a command prompt before launching the virtual application. This allows troubleshooting tools, like process monitor to be run to investigate any problems.

<DEPENDENCY>

<SCRIPT TIMING="PRE" EVENT="LAUNCH" WAIT="TRUE" PROTECT="TRUE">

<SCRIPTBODY>cmd.exe</SCRIPTBODY>

</SCRIPT>

</DEPENDENCY>

Another use of OSD scripting is to add registry values to the virtual environment. The example below adds registry values into the virtual environment.

<IMPLEMENTATION>

<CODEBASE … />

<VIRTUALENV>

<REGISTRY>

<REGKEY HIVE="HKLM" KEY="Software\Test">

<REGVALUE REGTYPE="REG\_SZ" NAME="string">Holas</REGVALUE>

<REGVALUE REGTYPE="REG\_DWORD" NAME="dword">5051</REGVALUE>

<REGVALUE REGTYPE="REG\_BINARY" NAME="hex">50,51,52</REGVALUE>

<REGVALUE REGTYPE="REG\_SZ">Value of Default Key</REGVALUE>

</REGKEY>

</REGISTRY>

</VIRTUALENV>

</IMPLENENTATION>

# Conclusion

With the set of tools listed in this document, administrators can automate many of the tasks associated with an App-V infrastructure.