

**Microsoft Application Virtualization 4.6**

**Sequencing Guide**

White Paper Descriptor

This whitepaper is designed to provide administrators with guidance for sequencing applications to create virtual packages that can be delivered to the end user. This document discusses setting up the Sequencer, sequencing best practices, an example of sequencing, important information related to updating packages, and finally, examples of advanced OSD scripting.

Copyright © 2008 MICROSOFT CORPORATION

Contents

[1 Introduction to Application Sequencing 5](#_Toc253150976)

[2 Sequencer Workstation Configuration 6](#_Toc253150977)

[3 New features of App-V 4.6 Sequencer 8](#_Toc253150978)

[3.1 Changes to the Sequencer in 4.6 8](#_Toc253150979)

[3.1.1 Improved UI 8](#_Toc253150980)

[3.1.2 Support for 64-bit Applications and 64-bit Operating Systems 8](#_Toc253150981)

[3.1.3 New and Existing App-V Sequenced Packages 9](#_Toc253150982)

[3.1.4 New OS Value Tags to Support 64-bit Operating Systems 10](#_Toc253150983)

[3.1.5 Removal of Block Size Selection 10](#_Toc253150984)

[3.2 App-V 64-bit Sequencing and SxS Conflict Detection 11](#_Toc253150985)

[4 Recommended Best Practices for Sequencing 12](#_Toc253150986)

[5 Classifying Applications for Sequencing 14](#_Toc253150987)

[6 Sequencing Limitations 15](#_Toc253150988)

[7 Sequencing Phases 16](#_Toc253150989)

[7.1 Installation Phase 16](#_Toc253150990)

[7.2 Launch Phase 17](#_Toc253150991)

[7.3 Customization Phase 18](#_Toc253150992)

[7.4 Save Phase 19](#_Toc253150993)

[8 Sample Sequencing 20](#_Toc253150994)

[8.1 Welcome Screen 20](#_Toc253150995)

[8.2 Package Information 21](#_Toc253150996)

[8.3 Advanced Options 22](#_Toc253150997)

[8.4 Monitor Installation 23](#_Toc253150998)

[8.4.1 Applications that Require a Reboot 23](#_Toc253150999)

[8.5 Configure Applications 25](#_Toc253151000)

[8.6 Launch Applications 26](#_Toc253151001)

[8.7 Package Deployment 27](#_Toc253151002)

[8.8 Saving the Package 28](#_Toc253151003)

[9 Application Package Upgrade 29](#_Toc253151004)

[9.1 Updating a Package that will Replace the Existing Package 29](#_Toc253151005)

[9.2 Updating a Package that will Run at the Same Time as the Existing Package 30](#_Toc253151006)

[10 Dynamic Suite Composition 31](#_Toc253151007)

[10.1 Advantages 31](#_Toc253151008)

[10.2 Create a Secondary Package for Shared Base Application using Dynamic Suite Composition 32](#_Toc253151009)

[10.3 Create a Secondary Package for a Shared Middleware or Utility using Dynamic Suite Composition 32](#_Toc253151010)

[10.4 Defining the Dependency in the Primary Package 33](#_Toc253151011)

[10.5 Dynamic Suite Composition Tool 35](#_Toc253151012)

[11 Compression of the SFT 37](#_Toc253151013)

[12 Advanced OSD Scripting 38](#_Toc253151014)

[12.1 OSD Scripts 39](#_Toc253151015)

[12.2 OSD: SCRIPTBODY vs HREF 41](#_Toc253151016)

[12.3 <SCRIPTBODY> Examples 42](#_Toc253151017)

[12.4 <HREF> Examples 43](#_Toc253151018)

[12.5 <PARAMETERS> Examples 44](#_Toc253151019)

[12.6 <WORKINGDIR> Examples – Specifying a Working Directory 44](#_Toc253151020)

[12.7 <REGISTRY> Examples – Registry Customizations 45](#_Toc253151021)

[12.8 Launching Local Applications 45](#_Toc253151022)

[12.9 Setting Environment Variables 46](#_Toc253151023)

[12.10 Launching Application via Non-Default Provider Policy 46](#_Toc253151024)

[12.11 Running an Application as a CONSOLE application 47](#_Toc253151025)

[12.12 OSD XML Syntax Check 48](#_Toc253151026)

[13 Advanced Sequencing Techniques 49](#_Toc253151027)

[13.1 Sequencing Applications That Cannot Install to Q:\ 49](#_Toc253151028)

[13.2 Sequencing Web-Based Applications 50](#_Toc253151029)

[13.3 Sequencing Applications that Require Access to Local Devices and/or Resources 50](#_Toc253151030)

[13.4 Sequencer Log Files 51](#_Toc253151031)

[14 Finding Additional Information 52](#_Toc253151032)

[14.1 Error Codes and the Microsoft Knowledge Base 52](#_Toc253151033)

[14.2 App-V Team Blog 53](#_Toc253151034)

[14.3 App-V Related Technical Discussion Forums and Web Sites 53](#_Toc253151035)

[15 Appendix A – Creating a Dummy ODBC 54](#_Toc253151036)

[16 Appendix B – Creating a Dummy Printer 62](#_Toc253151037)

# Introduction to Application Sequencing

The Microsoft Application Virtualization (App-V) Sequencer is a component of the App-V suite used to package applications to be deployed to systems using the App-V Client. Properly sequencing applications is the key to a successful App-V implementation. As such, it’s important to follow Microsoft’s recommended practices and be aware of the different options when sequencing. This document covers setting up the Sequencer, sequencing best practices, an example of sequencing, important information related to updating packages, and finally, examples of advanced OSD scripting.

# Sequencer Workstation Configuration

**Proper configuration of the sequencing station is imperative to ensure that applications will function properly when launched on a client. Microsoft recommends the following configuration when sequencing:**

* **Sequence on a machine that matches the operating system (OS) and configuration of the target clients**. It is often possible to sequence on one OS and run the virtualized application on a different OS; however this scenario is both application- and OS-dependent and is not guaranteed to work for all application/OS combinations since App-V is not a general-purpose OS compatibility solution. If problems are encountered, the application may be required to be sequenced on the same OS environment that the App-V Client is running on order to resolve those problems.
* **Include Microsoft Office on the base sequencer image if Microsoft Office will be installed traditionally on all desktops in the organization.** Many applications will install differently if they recognize that Microsoft Office is already installed on the machine. Thus, if an application is expected to integrate with Microsoft Office, it’s best to attempt sequencing on a machine with Office already installed and activated. This assumes that a Microsoft Office suite will be installed locally on all client PCs. In addition, you may want to install any other programs that could be used by the application you are sequencing if they are not going to be a part of the sequence.
* [**Create an ODBC DSN setting as part of the Sequencer base image.**](#_Appendix_A_–)  If no ODBC DSN setting exists on the base Sequencer image and the application being packaged creates one, the entire registry key associated with ODBC settings will become virtualized. This will prohibit the packaged application from seeing any ODBC DSN settings that exist on the base client machine. If an ODBC entry already exists on the Sequencer machine, only the ODBC settings will become virtualized, and the ODBC settings on the client will be merged with the ODBC settings in the package.
  + The following locations can be checked to determine ODBC information was captured:
    - Search for odbc.ini: It will be located in the VFS\%CSIDL\_WINDOWS% folder
    - HKLM\Software\ODBC\ODBC.INI\ODBC Data Sources
    - HKCU\%SFT\_SID%\Software\ODBC\ODBC.INI

For step-by-step instructions on how to setup a dummy ODBC entry, see [Appendix A](#_Appendix_A_–).

* [**Add a dummy printer device as part of the Sequencer base image**](#_Appendix_B_–). Printers act in the same manner as ODBC settings. It is necessary to include a dummy printer device in the Sequencer PC image. For step-by-step instructions on how to create a dummy printer device, refer to [Appendix B](#_Appendix_B_–).
* **Setup your sequencer machine with multiple partitions.** It is recommended that the Sequencer machine be configured with at least two primary partitions. The first partition C:, should have the operating system installed and should be formatted as NTFS. The second partition should be assigned the drive letter that will match the virtual drive letter assigned on the client. The default value is Q: This assignment is used as the destination path for the application installation and should also be formatted as NTFS.
* **Configure the temp directories to ensure successful sequencing and improve performance.** The Sequencer uses the %TMP%, %TEMP%, and its own Scratch Directory for temporary files. These locations should contain free disk space equivalent to the estimated installation size. The Scratch Directory is where the Sequencer will temporarily store files generated during the sequencing process. You can check the location of the Scratch Directory by launching the Sequencer, clicking Options from the Tools menu, clicking the Paths tab, and then noting the Scratch Directory box. Placing the temp directories and the Scratch Directory on different hard drive spindles can improve performance during sequencing.
* **Sequence using Virtual PC.** Most applications will be sequenced more than once. This may be due to additional configuration changes or simply starting over to correct a mistake. The point is that you will be going back to your original configuration on the PC several times. To help facilitate this you may want to use a virtual machine. This will let you sequence an application and, with a simple click of a button, revert back to a clean state so you can continue sequencing with no down time. Additionally, whenever you start a new sequence, start with a clean sequencing system.
* **Shut down Other Programs.** Processes and scheduled tasks that normally run on your computer can slow down the sequencing process and cause irrelevant data to be gathered during sequencing. These programs should be shut down before you begin sequencing. Some of these programs include:
  + - Windows Defender
    - Antivirus software
    - Disk defragmentation software
    - Windows Search
    - Microsoft Update
    - Any open Windows Explorer session

**Note**: The Sequencer workstation should be fully scanned for viruses and malware and then the antivirus and antimalware software should be disabled before creating a snapshot image of the Sequencer workstation.

# New features of App-V 4.6 Sequencer

Application Virtualization 4.6 adds support for 64-bit operating systems and 64-bit applications. Configuration and understanding of the changes for App-V 4.6 as they pertain to the Sequencer are important to successfully deploying App-V 4.6 and supporting virtual applications. The following section describes specific details about the 4.6 Sequencer.

## Changes to the Sequencer in 4.6

The following list describes the changes to the App-V 4.6 Sequencer that will be detailed in the section below.

* Improved UI to simplify sequencing process
* Support for 64-bit applications and 64-bit operating systems
* New OS value tags in the OSD to accommodate 64-bit operating systems
* Removed block size selection

### Improved UI

The App-V 4.6 Sequencer interface was changed to improve the sequencing experience. [Section 8](#_Sample_Sequencing) in this document provides a detailed walkthrough of the sequencing interface. More information about the changes to Sequencer interface is available at: <http://blogs.technet.com/softgrid/archive/2009/08/25/sequencer-ui-changes-in-app-v-4-6.aspx>.

### Support for 64-bit Applications and 64-bit Operating Systems

Sequencing 64-bit applications requires sequencing on a 64-bit operating system. The resulting package can only be deployed to a 64-bit operating system with the 64-bit App-V 4.6 Client. All other recommendations for sequencing, including sequencing on the same OS as the target OS, still apply.

### New and Existing App-V Sequenced Packages

The goal of App-V 4.6 is to ensure that the transition will have minimal impact on an organization. Packages that have been sequenced with previous versions of the App-V Sequencer can be deployed to App-V 4.6 Clients, with only changes to the OSD files to include the new **OS Value** tags that support 64-bit operating systems (which are described next in this section). The following is a support matrix of coexistence with pre-existing and new App-V sequenced packages:

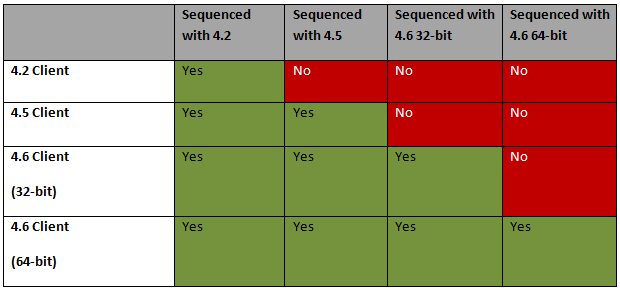


Table 1: Sequencing Support Matrix

**Note: The column “Sequenced with 4.2” also includes App-V 4.1 SP1 sequences.**

**Note: It is recommended to only sequence 32-bit applications on 32-bit sequencers, as 32-bit applications sequenced on a 64-bit sequencer are only supported on 64-bit clients.**

In summary, sequenced packages created with previous versions of the App-V Sequencer 4.1 SP1 and later should work with the App-V 4.6 Client. Some applications that have been sequenced in previous versions of operating systems may need to be sequenced again if deploying to newer operating systems.

### New OS Value Tags to Support 64-bit Operating Systems

The following table shows the differences between **OS Value** tags from previous versions of App-V to App-V 4.6.

|  |  |  |
| --- | --- | --- |
| OS Value | App-V 4.6 | App-V 4.5 |
| WinXP | Windows XP Professional 32-bit | Windows XP Professional |
| WinXP64 | Windows XP Professional 64-bit | n/a |
| Win2003TS | Windows 2003 Terminal Server 32-bit | Windows 2003 Terminal Server |
| Win2003TS64 | Windows 2003 Terminal Server 64-bit | n/a |
| WinVista | Windows Vista 32-bit | Windows Vista |
| WinVista64 | Windows Vista 64-bit | n/a |
| Win2008TS | Windows 2008 Terminal Server 32-bit | Windows 2008 Terminal Server |
| Win2008TS64 | Windows 2008 Terminal Server 64-bit | n/a |
| Win2008R2TS64 | Windows 2008 R2 Terminal Server 64-bit | n/a |
| Win7 | Windows 7 32-bit | App-V CU1 |
| Win764 | Windows 7 64-bit | n/a |

Table 2: OS Value tags

### Removal of Block Size Selection

App-V 4.6 removes the selection of block size when sequencing. All App-V 4.6 packages are sequenced with 64KB block size for optimal network performance.

## App-V 64-bit Sequencing and SxS Conflict Detection

Sequencing with a 64-bt Sequencer is the same process as 32-bit sequencing. However, in some cases the Sequencer will detect a conflict when performing privatization of Side-by-Side (SxS) assemblies during normal sequencing. Typically, the Sequencer creates private side-by-side assemblies for all public assemblies that are used by the package. However, it is not possible to create 32-bit and 64-bit versions of private assemblies in the same directory. When the Sequencer detects that a directory would contain both the 32- and 64-bit privatized versions of the same public assembly, the Sequencer will report a SxS privatization conflict and display the dialogue below.

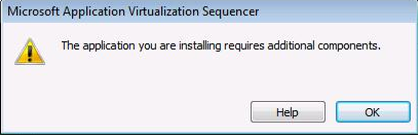


Figure 1: Side-by-Side Conflict Detection Message

When this occurs, it is recommended that you perform the following steps:

1. Stop sequencing and revert the sequencing workstation back to the base configuration.
2. Natively install either the 32-bit or 64-bit conflicting assembly on the sequencing station as a public SxS DLL (64-bit assembly is recommended for the native installation).
3. Sequence the application that caused the conflict detection.
4. Natively install the public assembly from step 2 on the each client before deploying the package.

**NOTE: If performing command-line sequencing, the message will appear in the log file.**

This process ensures that only one version of the private assembly gets captured in the package. Visual Studio 2008 is an example of an application that would display this behavior if installing on a 64-bit operating system. In order to complete the tasks above, the specific assembly for 64-bit or 32-bit that will be installed natively on the Sequencer and client computers will need to be located. For Visual Studio 2008, the 64-bit assembly can be found at: <http://www.microsoft.com/downloads/details.aspx?familyid=BD2A6171-E2D6-4230-B809-9A8D7548C1B6&displaylang=en>.

# Recommended Best Practices for Sequencing

This section covers Microsoft’s recommended best practices for sequencing applications.

It is recommended that you familiarize yourself with the installation and execution of the application prior to sequencing. Be sure to read all installation instructions associated with the application. It is also recommended that you learn how the application runs and the components of the application that the user will need.

To improve the process of sequencing an application, one should document step-by-step the installation and post-configuration procedures for the application. Step-by-step documentation will ensure that no unnecessary troubleshooting occurs during the sequencing process since no important steps will be skipped. Always document the sequencing process step-by-step creating a “recipe” using a standardized template. Documenting the sequencing process step-by-step will allow you to hand the recipe to someone else in your organization and have them recreate the same package.

Items to document in a recipe include:

* What application components are needed and will be required to complete the installation of the application?
* What updates, such as adding new files to the package, need to be performed in the Sequencer after the installation?
* What post-installation configuration steps need to take place in the Sequencer?
* How do users commonly use this application immediately after its launch?
* Does this application do something that App-V currently does not support? If so, check the Microsoft Knowledge Base to see if a workaround is available.

Recommended settings, configurations, and process include:

* When sequencing on Windows Vista or Windows 7, ensure sure you have User Account Control (UAC) enabled on the sequencing machine if the client machine you are deploying the application to will have UAC enabled as well.
* Use the Comments field in the Sequencer (Abstract Tag) to note any details about the package you may want to include. This will allow you to revisit the Sequence later and have a record of this information.
* The installation drive on your Sequencer should match the virtual drive on the client. In some cases this may not be possible (if, say, packaging for multiple organizations), and you may be required to go back and edit the virtual registry or some individual files to point to the correct locations.
* Sequence to a unique, 8.3 directory name. This applies to both the Asset and Installation Directories. (‘Q:\MYAPP’ is correct, ‘Q:\My Application’ is incorrect. Q:\MYAPP.001 is also correct.)
* Sequence to a folder in the root of the drive, not to a subdirectory. (‘Q:\MYAPP’ is correct; ‘Q:\’ is incorrect; ‘Q:\Temp\_Junk\MYFOLD’ is incorrect). If the suite has multiple parts, install each application in a subdirectory of the Asset Directory. For example, if a package contains a Line of Business Application along with the Oracle Client, use Q:\AppSuite as the Asset Directory; sequence the LOB application to Q:\AppSuite\LOB; and sequence the Oracle client to Q:\AppSuite\OracleClient.
* Always use globally unique paths and package names across the set of application sequencings. Do not install, for example, multiple Microsoft Office sequencings to the same Asset Directory name. Use a standardized naming scheme for the Asset Directory that can be incremented for new revisions, for example Q:\OFFXP.v1 or Q:\OFFXP.001. Failure to make these unique can cause conflicts with your applications.
* Configure and test the application in the Installation Phase. Completing the installation of an application often requires performing several manual steps that are not part of the application installation process. These steps can involve configuring a connection to a ‘back-end’ database, copying updated files, etc. Do this configuration in the Installation Phase and run the application to make sure it works.
* Execute the application, multiple times if necessary, until the program is in a static state in the Installation Phase. For example, run the application multiple times to get past all registration and dialog box requests. Some applications perform different tasks on first launch, second launch, and sometimes subsequent launches. The multiple launches will make sure that any post-installation tasks that are required by the application can be completed (e.g. accepting a license agreement or setting file type associations).
* Use the Application Wizard to launch each executable in a suite of applications. This will ensure that each application will have the required initial launch data on the App-V Client.
* Disable “Install on First Use.” Some applications have the option to “Install on First Use” for certain components. It is required that none of the components are sequenced with this option. It is necessary to choose either “Run from My Computer” (install this component) or “Not Available” (do not install this component). For application components that will not be used by any of the targeted users ,it is recommended that the components not be installed.
* Disable “Auto Update” features. Some applications have the ability to check a web site or a server for the latest application updates. This feature should be turned off, as version control should be performed via sequencing new versions.
* Operations made during the Launch Phase will be included in the primary feature block. As a general rule, when building the primary feature block make sure you execute the application’s most common operations (e.g., in Microsoft Word, open a document and misspell a word to include spell check function) so that they are included in the initial streaming of the application and you have an accurate primary feature block. If this is not done, users will see delays as they start to use the application and will regard it as being slow if many of the features they use are not in primary feature block. Additionally, if you are in an environment where bandwidth is limited, then you want to have an accurate primary feature block so that users are not constantly making calls to the server to download additional files in cache.
* There are several online articles relating to sequencing best practices :
  + Microsoft Support: <http://support.microsoft.com/kb/932137>
  + App-V Team Blog: [http://blogs.technet.com/App-V/archive/2007/07/12/sequencing-best-practices.aspx](http://blogs.technet.com/softgrid/archive/2007/07/12/sequencing-best-practices.aspx)

# Classifying Applications for Sequencing

All applications are different and, therefore, no application will require the same amount of time to sequence. However, sequencing estimates can be put into three categories based on the complexity of the application, size (both size on disk and number of files), and finally, reliance on resources outside of the virtual application. Another item to take special note of before sequencing begins: **nothing will slow down the sequencing process more than not having access to someone who inherently understands the full functionality of the application.**

This section includes approximate sequencing times. However, these are very general in nature. Every application will be different. The times are presented only to assist in *estimating* the time required for a project.

|  |  |  |
| --- | --- | --- |
| Application Type | Description | Time Scale |
| Simple | These applications are normally small. An example of an application in this category would be WinZip or Adobe Reader. These applications are very straightforward and normally small in size (usually under 100MB). Very little, if any, modifications are needed to run these applications. | **Typical sequencing time: less than 1 hour** |
| Moderate | This is probably the most common application type. These applications might require some changes while sequencing to function correctly. In some cases, they may require no changes but have a larger install that takes more time. In rare occurrences you will encounter both. Changes you might encounter in these packages would be on the order of making changes to the registry, altering the .osd file to launch with additional parameters and scripts, or there may be additional applications needed to install together as a suite so they can function together.. | **Typical sequencing time: 1-4 hours** |
| Complex | These are large applications or applications that take four or more hours to install, significant amounts of customization to function in the virtual environment, or both. Packages like these will normally be 3-4 GB in size and may require compression to get the package under the 4GB App-V limit. Other hurdles you may encounter are the application’s reliance on files being in a specific place and functions hard-coded to that install. These applications may require you to manually edit batch and command files to point to resources in the virtual environment. If this is the case, it is highly recommended utilizing a program that can scan multiple files and make several changes at once. You also may be required to install a device driver separately since drivers cannot be virtualized. Applications of this complexity can be sequenced, however it is imperative that, before you begin, all the pieces must be in place. All knowledgeable resources should be engaged and available, sequencing hardware should be better than average, and finally, sequencing applications such as these should be done by an experienced sequencer who has used the App-V product. | **Typical sequencing time: 4-8 hours, but could be longer depending on the size and number of files** |

Table 3: Classifications of Applications

# Sequencing Limitations

Sometimes there are applications that cannot or should not be sequenced. Also there are certain limitations with App-V. Here is a short list of application functions and limitations of the software.

|  |  |
| --- | --- |
| Limitation | Description |
| Sequenced package over 4GB (SFT has 4GB limit) | * By utilizing compression within the Sequencer it’s possible to take applications that are larger than 4GB and get them below the limit. * However, after compression the application must be smaller than 4GB. * If the application is too large then the sequencer will not save the application. It will, however, attempt to compress the file. |
| Applications that start services at boot time | * App-V requires a logged in user to initiate the launch of an application. |
| Applications that require device drivers | * App-V cannot virtualize drivers. It is possible to bypass this issue and install the driver locally on the target computer. * Some user-mode device drivers can be virtualized. |
| Applications that are required by several applications for information or access | * For example, a program that will initiate a command and launch another program. Normally you would include both programs in the same suite, however if this application launches or initiates commands in several applications it may not be feasible for you to include all of the applications in the same suite. * This is especially true if one of the reasons you are deploying App-V is to avoid application conflicts. Always remember that the virtual “bubble” can see the OS and what’s installed on it but the OS cannot see the “bubble” and interact with it. On the same note, remember that one “bubble” cannot see another. |
| Applications that are a part of the OS | * Such as Internet Explorer |
| Applications that use COM+ | * Because COM+ is dynamic and happens at runtime, there’s no way for the Sequencer to capture this information. |
| COM DLL surrogate virtualization | * For example, DLLs that run in Dllhost.exe |

Table 4: Sequencing Limitations

# Sequencing Phases

There are four phases to sequencing a package: installation, launch, customization, and save. This section provides a high-level overview of each phase.

## Installation Phase

During the installation phase, the Sequencer will be used to configure the package information, monitor the installation of the application(s), and finally, process the installation information. After completing the installation phase, the Sequencer will move to the launch phase. The figure below describes the operations that occur in the installation phase:

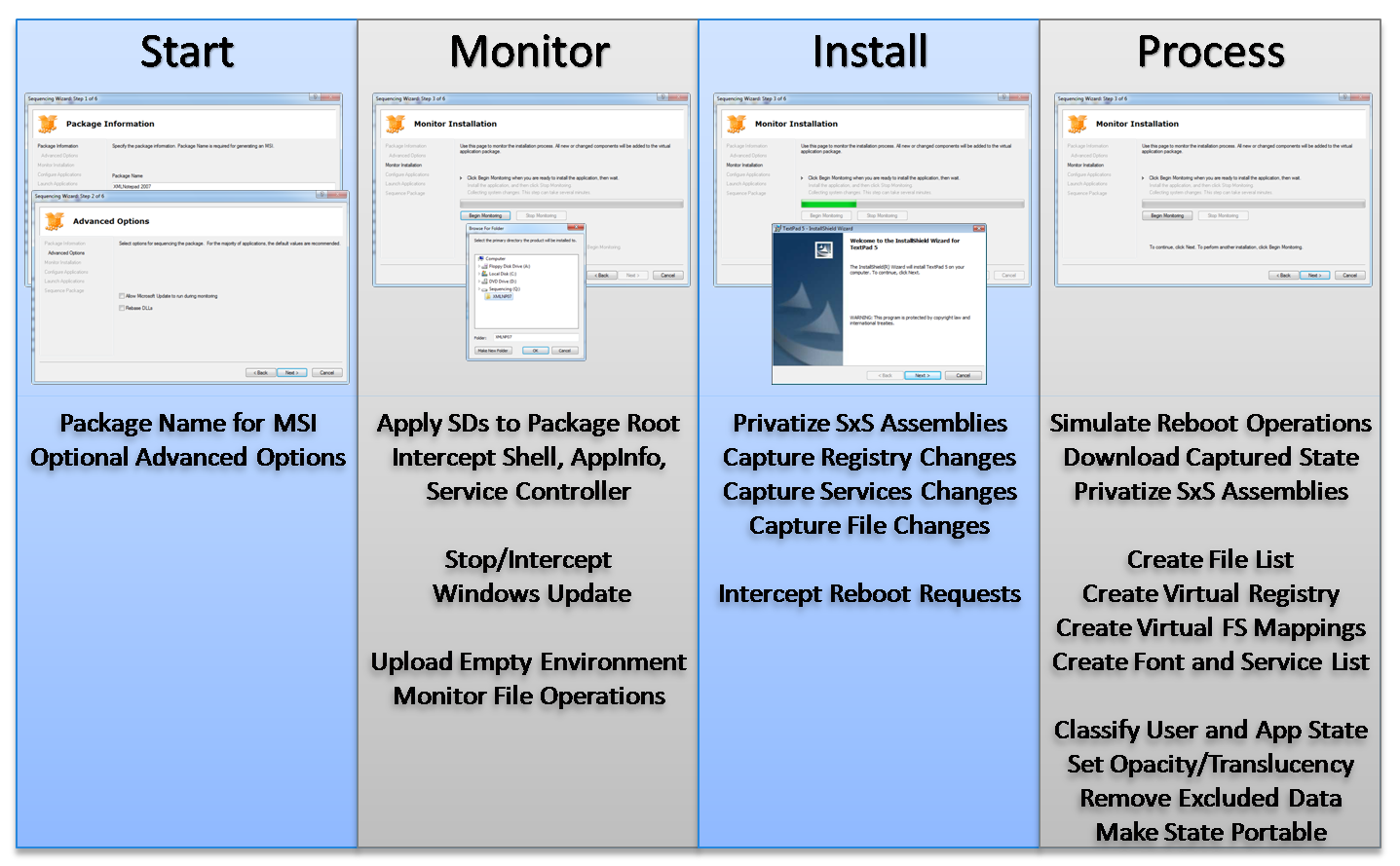


Figure 2: A walkthrough and brief description of the different steps of the installation phase

## Launch Phase

The next phase, or launch phase, is used to complete the following tasks: configure applications, launch selected applications to set proper primary feature block, process the package, and create the package.

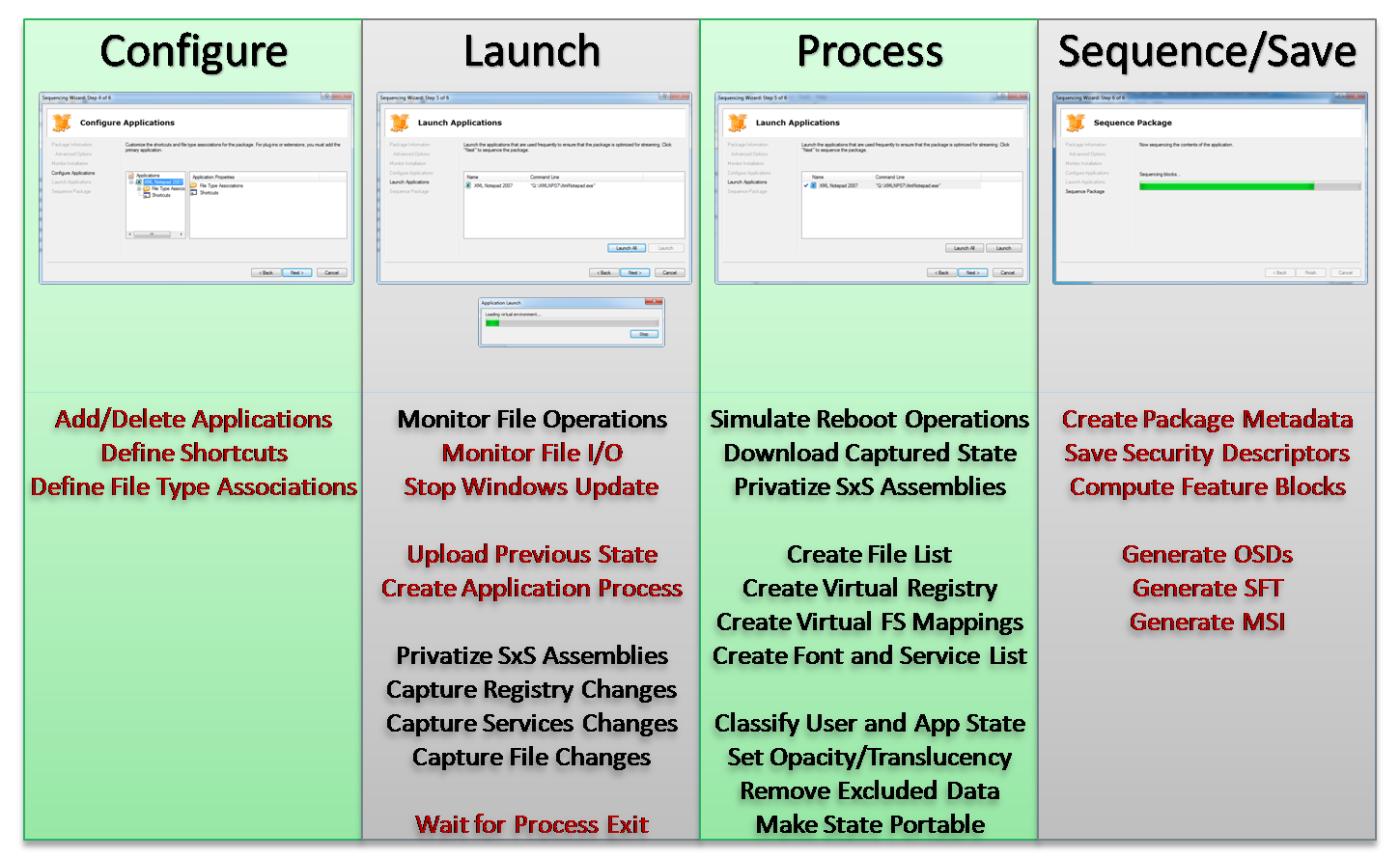


Figure 3: A walkthrough and brief description of the different steps of the launch phase

## Customization Phase

During this phase you can make customizations to your package in the OSD files. The following is a brief example of what is in the OSD file. A more detailed description can be found in the “Advanced OSD Scripting” section later in this document.

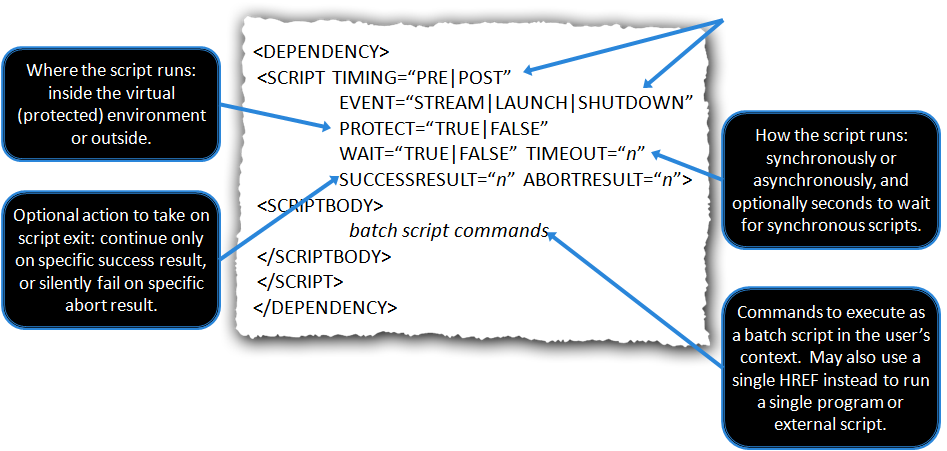


Figure 4: A description of OSD elements

## Save Phase

When you save your package you will have several files. The following is a list of those files and descriptions of each of them.

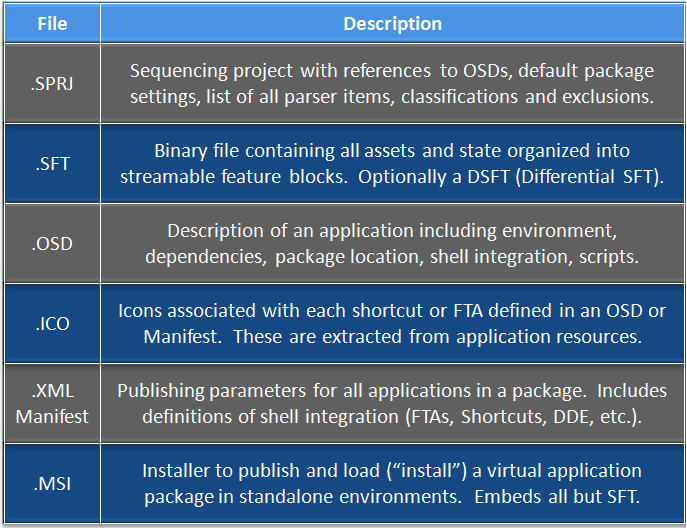


Table 5: Sequencing Output Files

# Sample Sequencing

This section describes some of the key points to remember during sequencing. To begin open the Application Virtualization Sequencer and select **Create a Package** from the **Welcome Screen**.

## Welcome Screen

The Welcome Screen is new in App-V 4.6 and provides quick access to the common tasks of creating, editing, and upgrading a package.

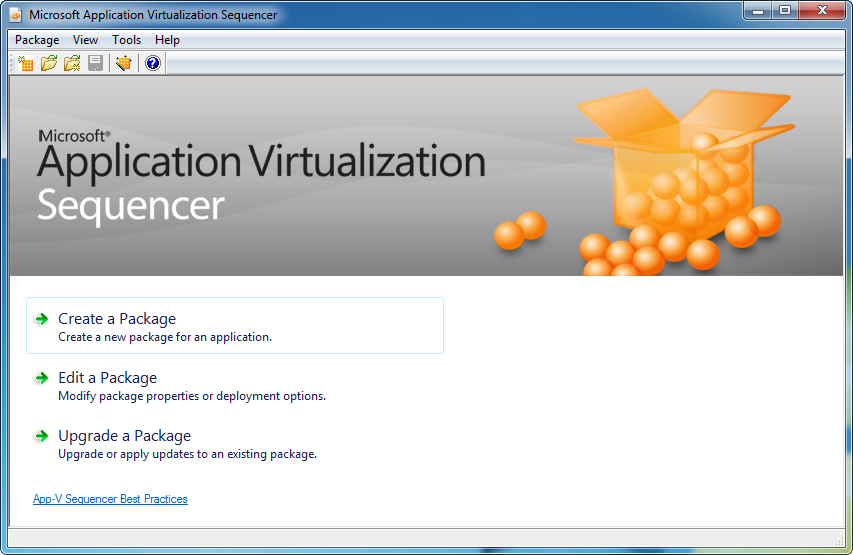


Figure 5: Welcome Screen

|  |  |
| --- | --- |
| Welcome Screen Component | Description |
| Create a Package | The Package Name must be unique. Microsoft uses the following naming convention: <Appname>\_<AppVendor>\_<Version>\_<MNT | VFS>, where MNT indicates the application was installed to Q: and VFS indicates the application was installed to C: |
| Edit a Package | Indicates the person who performed the sequence, the date the sequence was created and the version of the Sequencer. This information is recorded in the ABSTRACT element of the OSD file, and can be used to understand the pedigree of the sequence. |
| Upgrade a Package | Checking this box will display the Advanced Options dialog seen in 8.2 below. |

Table 6: Welcome Screen Component Description

## Package Information

Creating a package starts with providing information about the package. The picture and table below describe the different options that are available.

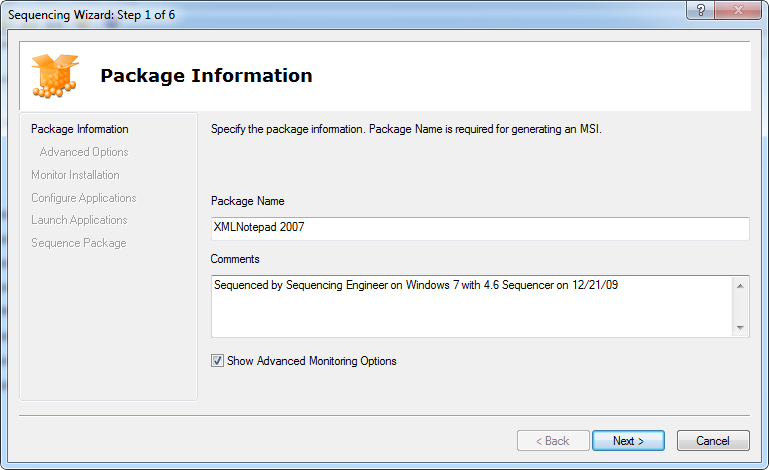


Figure 6: Package Information

|  |  |
| --- | --- |
| Package Information Component | Description |
| Package Name | The Package Name must be unique. Microsoft uses the following naming convention: <Appname>\_<AppVendor>\_<Version>\_<MNT | VFS>, where MNT indicates the application was installed to Q: and VFS indicates the application was installed to C: |
| Comments | Indicates the person who performed the sequence, the date the sequence was created and the version of the Sequencer. This information is recorded in the ABSTRACT element of the OSD file, and can be used to understand the pedigree of the sequence. |
| Advanced Monitoring  Checkbox | Checking this box will display the Advanced Options dialog seen in 8.3 below. |

Table 7: Package Information Component Description

## Advanced Options

When advanced options are selected in the package information screen, the Sequencer will be presented with options for Microsoft Update and Rebasing DLLs. The table below describes these selections in detail.

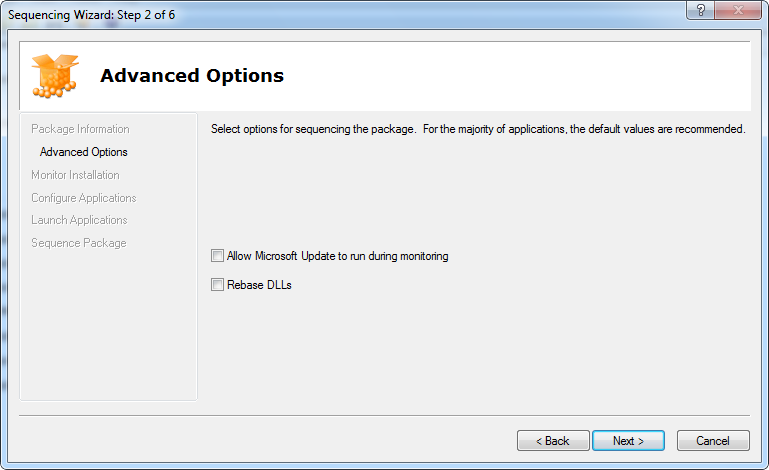


Figure 7: Advanced Options

|  |  |
| --- | --- |
| Advanced Options Component | Description |
| Allow Microsoft  Update | Allows Microsoft Update to run as part of the sequencing process. Typically this is used to allow Microsoft Update to update Microsoft Office during the package update process. However, it is important to remember that the Sequencer cannot capture OS components. |
| Rebase DLL’s | Results in DLLs being remapped to a contiguous address space in RAM to increase efficiency. This option is primarily used for applications being deployed to a Terminal Server environment. Some applications may not support this feature and, therefore, rebasing is disabled by default. |

Table 8: Advanced Options Component Descriptions

## Monitor Installation

After the package information has been collected, the next step is to monitor the installation of the application. Selecting Begin Monitoring will prompt for an installation directory. Next, the virtual environment will be built and the Sequencer will start monitoring specific locations and processes for activity. In App-V 4.6, one change is that the Monitor Installation page will not minimize after choosing to Begin Monitoring. After selecting Begin Monitoring, most operations will be recorded as part of the package, unless a specific exception has been configured on the Sequencer.

### Applications that Require a Reboot

Some applications will require a reboot of the operating system after installation to accommodate files that are in use during installation that need to be replaced. The Sequencer is aware of reboot tasks and the operating system will not need to be rebooted. While sequencing an application that requires a reboot, perform the following tasks:

1. When prompted, select to reboot. The Sequencer will capture this request and the machine will not restart.
2. Next select **Stop Monitoring** from the Sequencer.
3. Then select **Begin Monitoring** again. All of the reboot tasks will be processed once the Sequencer starts monitoring again.
4. Proceed with additional installations or click **Next** if no additional installations are required for the package.

The process of App-V sequencing provides an additional benefit over traditional software deployment and installation. Applications that require a reboot to complete the installation will no longer require the reboot when delivered to the client machines. This is because the Sequencer will monitor the operations that need to be completed during reboot and process them prior to saving the package and prior to delivery to client computers.

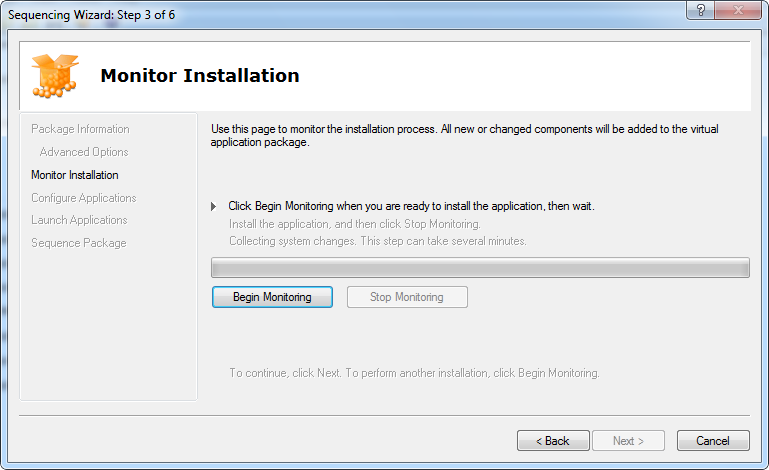


Figure 8: Monitor Installation

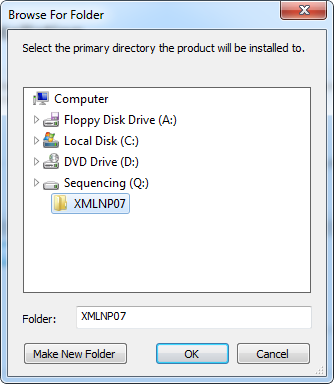


Figure 9: Installation Folder Selection

|  |  |
| --- | --- |
| Monitor Installation Component | Description |
| Begin Monitoring | Click the Begin Monitoring button then install your application. When done, Click the Stop Monitoring Button |
| Create installation Folder | After you click “Begin Monitoring” you will be prompted to create the installation folder. In version 4.2 this was done when you clicked “Stop Monitoring.” |

Table 9:Monitor Installation Component Selection

## Configure Applications

Next, the Sequencer will evaluate all of the applications identified during monitoring. Applications may have to be added or removed from this list, based on what is wanted in the final package. Remember, any application in this list will have an associated OSD file created for launching. Also, with Web-based applications it is often required to add Internet Explorer as an application because the Web-based application will require Internet Explorer to run properly. Each application can be modified to change the name, icon that it uses, file type associations, and locations for shortcuts on destination computers.

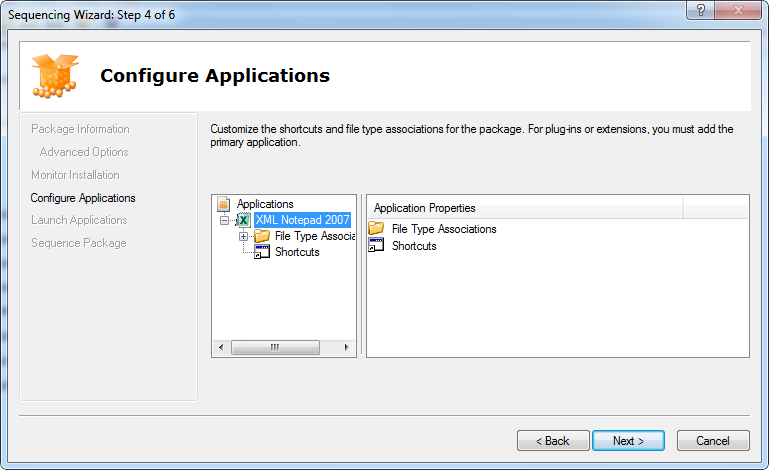


Figure 10: Configure Applications

|  |  |
| --- | --- |
| Configure Apps Component | Description |
| Edit Applications | At this point you can add or remove icons for applications. Additionally, you can edit the file type associations and the icon locations on the client. |

Table 10: Configure Applications Component Description

## Launch Applications

Next, applications can be launched to create the primary feature block. In a package with multiple applications, one, multiple or all applications can be launched to create the primary feature block.

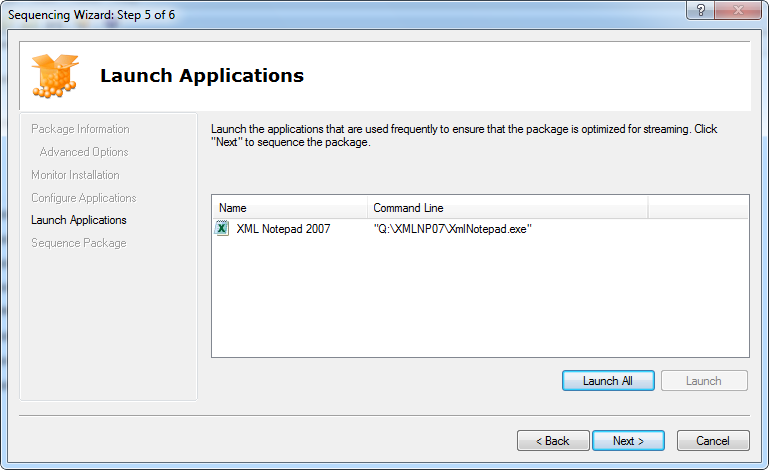


Figure 11: Launch Applications

|  |  |
| --- | --- |
| Launch Applications Component | Description |
| Launch | Here you will launch an application to ensure functionality. Additionally, any files that are executed during this phase will be tagged as primary feature block. Any remaining files will be tagged as secondary feature block. If you don’t launch any application then all files will be a part of the primary feature block. |
| Launch All | Performs the same operation as the launch component, but launches all identified applications in the package. This is useful when trying to create primary and secondary feature blocks for a large suite of applications. |

Table 11: Launch Applications Component Description

## Package Deployment

Upon completion of the sequencing wizard, the package can be further configured. Using the package deployment tab of the Sequencer, the package can be configured for proper deployment prior to saving the package. The following figure and table describe the settings for deployment in detail.

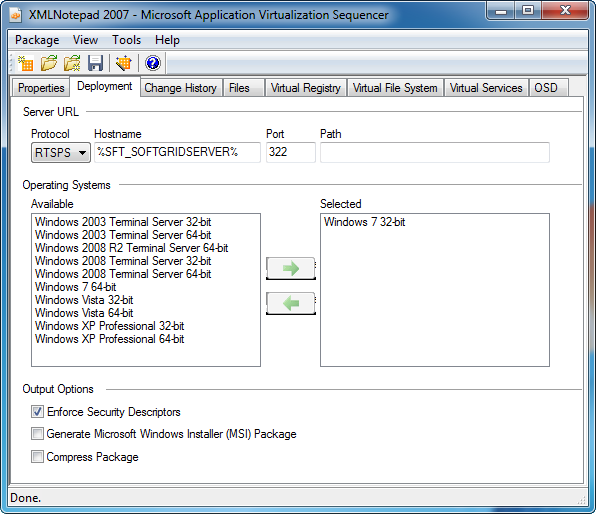


Figure 12: Deployment Tab

|  |  |
| --- | --- |
| Package Deployment | Description |
| Protocol | Select from the following protocols: RTSP, RTSPS, HTTP, HTTPS, and FILE |
| Hostname | Define the hostname of your APP-V server. Leave the default %SFT\_SOFTGRIDSERVER% if you will be defining this variable on your client machines. |
| Port | This will be defined when you select the protocol that you want. |
| Path | This is the relative path of the content of the App-V server. |
| Operating Systems | Select the operating systems that you want the application to run on. |
| Output Options | * Enforce Security Descriptors - When this option is checked, the Sequencer captures security descriptors at packaging time and the client enforces them on the file system drive at runtime. * Generate MSI Package – Select to generate a MSI package for standalone deployment scenarios. |
| Compress Package | Select if you need to compress the size of your package. Note that this only compresses the package itself. After it is streamed to the client it will expand to its full size. |

Table 12: Deployment Tab Component Description

## Saving the Package

The last step in the process of sequencing is to save the package. Saving the package should be done to a different directory than the installation of the application(s).

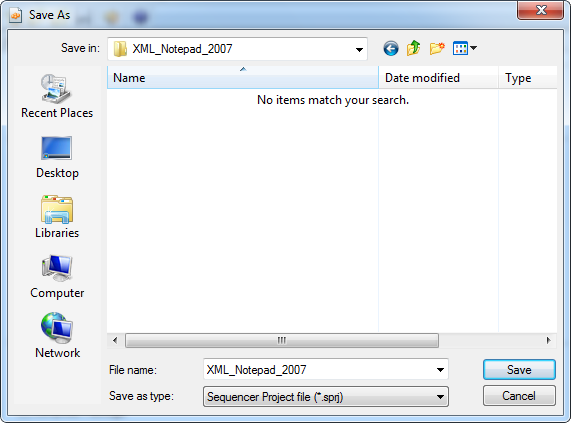


Figure 13: Package Save

|  |  |
| --- | --- |
| Package Save | Description |
| Save | At this stage you’re ready to save your package. One easy method of ensuring that the filename and folder you save in are unique is to use the same PACKAGE NAME that you specified in the Package Information wizard. |

Table 13: Package Save Description

# Application Package Upgrade

Throughout the lifecycle of an application, it will need to be updated. It is important to understand the options associated with updating an application, and when to use them.

## Updating a Package that will Replace the Existing Package

Sometimes, a package needs to be updated and to replace the existing package. Upgrading a package is a necessary part of the application lifecycle when there is a patch, service pack, or other update from the software manufacturer. In this case, follow these steps when sequencing and, subsequently, publishing in production:

* Copy the package to the Sequencer.
* Open the package for Package Upgrade by selecting “Upgrade a Package” from the Welcome Screen.
* After the installation, launch all shortcuts in the same manner as during the original sequence.
* Save the package. You have three options when saving the package. Since we are replacing an existing package you will select option 1:

1. Save - Makes new SFT/OSDs, remove the old files from the directory from where the package was unpacked.
2. Save As - Select a new folder; makes new SFT/OSD.
3. Save As, and select “Save As New Package” - Makes new package including a new GUID & Package Root. This will allow you to publish the application side-by-side with the original.

## Updating a Package that will Run at the Same Time as the Existing Package

Sometimes, a package needs to be updated, but the business community still wants access to the existing package. This could be used to introduce a new pilot version of the application and, eventually, deprecate the previous version after proper testing. In this scenario, follow these steps:

1. Copy the package to the Sequencer.
2. Open the package for Package Upgrade.
3. Select Q: to decode the package to.
4. Run the installation wizard and make the appropriate changes to the package.
5. After the Installation Wizard, run the Application Wizard. Modify either the Name or the Version for all of the shortcuts, as these must be unique for each shortcut. Launch all shortcuts in the same manner as during the original sequence.
6. Select “Save As” to save the package. Check the “Save As New Package” checkbox to create a new GUID and a new package root. Remember, the package root must be 8.3 compliant and unique.

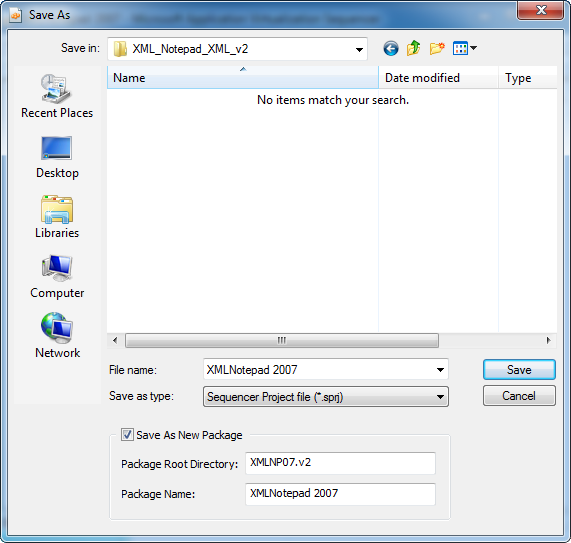


Table 14: Save As Dialogue

# Dynamic Suite Composition

Dynamic Suite Composition (DSC) in Application Virtualization enables you to define an application as being dependent on another application, such as middleware or a plug-in. This enables the application to interact with the middleware or plug-in in the virtual environment, where normally this is prevented. This is useful because a secondary application package can be used with several other applications, referred to as the primary applications, enabling each primary application to reference the same secondary package.

You can use Dynamic Suite Composition when sequencing applications that depend on plug-ins such as ActiveX controls or for applications that depend on middleware such as OLE DB or the Java Runtime Environment (JRE). If each application that used these dependent components needed to be sequenced, including the components, updates to those components would require re-sequencing all the primary applications. By sequencing the primary applications without those components and then sequencing the middleware or plug-in as a secondary package, only the secondary package needs to be updated.

## Advantages

* Reduce the size of the primary packages.
* Provides better control of access permissions on the secondary applications.

**Note: The secondary application can be streamed in the normal way and does not need to be fully cached to run.**

A primary package can have more than one secondary package. However, only one level of dependency is supported, so you cannot define a secondary package as dependent on another secondary package. Also, the secondary application can only be middleware or a plug-in and cannot be another full software product.

If you plan to make several primary applications dependent on a single middleware product, be sure to test this configuration to determine the potential effect on system performance before deploying it.

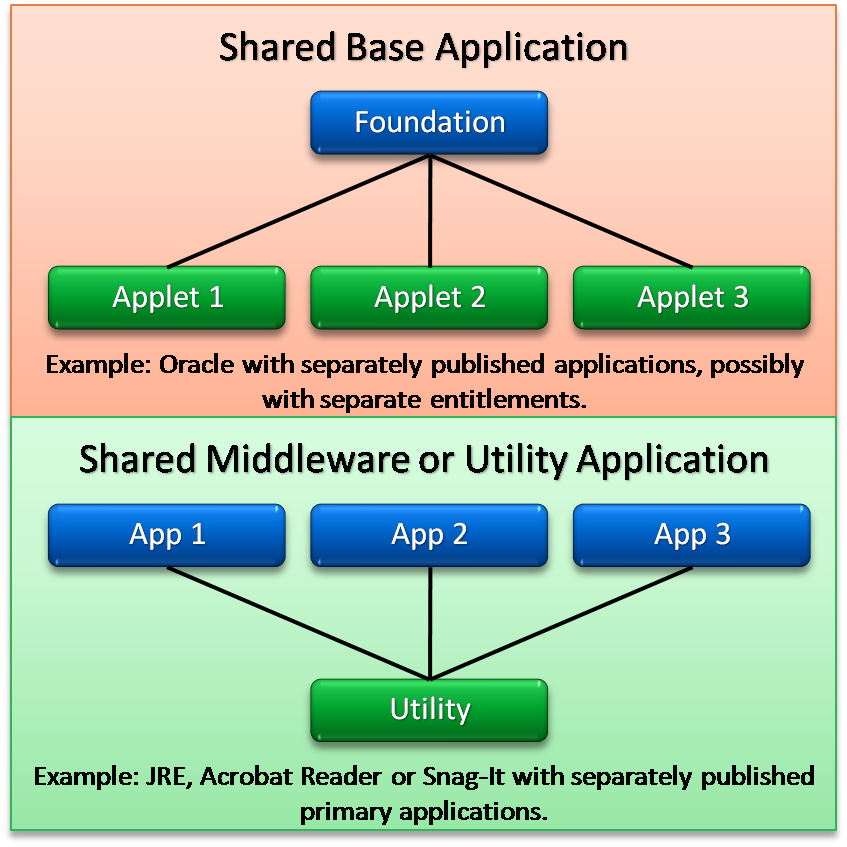


Figure 14: DSC Usage Scenarios

## Create a Secondary Package for Shared Base Application using Dynamic Suite Composition

The following steps describe the proper order to create a secondary package like a line of business application(s) and primary application (such as Oracle Client) in preparation for combining them using DSC. These are the high-level steps for sequencing.

1. On a sequencing computer that is set up with a clean image, install Application Virtualization Sequencer and save the computer state.
2. Install the shared base (middleware) application locally on the sequencing computer, and configure it.
3. Sequence the primary application, and save the package to the Content folder on the server.
4. Restore the sequencing computer to its saved state from step 1.
5. Start the Sequencer to create a new package.
6. Start the Sequencer monitoring phase.
7. Install the middleware application on the sequencing computer, and configure it as in a normal installation.
8. Complete the sequencing process.
9. Save the package to the Content folder on the server.

**Note:** To assist with management of secondary packages, it is recommended that the package name include the term “Secondary package” to emphasize that this is a package that will not function as a standalone application—for example, **[Middleware Name] Secondary package**.

## Create a Secondary Package for a Shared Middleware or Utility using Dynamic Suite Composition

The following steps describe the proper order to create a secondary package like a Microsoft Office plug-in (Open Document Format plug-in) and primary application (Microsoft Office) in preparation for combining them using DSC. These are the high-level steps for sequencing.

1. On a sequencing computer that is set up with a clean image, install Application Virtualization Sequencer and save the computer state.
2. Sequence the primary application, and save the package to the Content folder on the server.
3. Restore the sequencing computer to its saved state from step 1.
4. Install and configure the primary application locally on the sequencing computer.

**Important: You must specify a new package root for the secondary package.**

1. Start the Sequencer monitoring phase.
2. Install the shared middleware or utility application on the sequencing computer and configure it as needed.
3. Open the primary application, and confirm that the plug-in is working properly.
4. In the Sequencer console, create a dummy application to represent the secondary package that will contain the plug-in and select an icon.
5. Save the package to the Content folder on the server.

**Note**: To assist with management of secondary packages, it is recommended that the package name include the term “Secondary package” to emphasize that this is a package that will not function as a standalone application—for example, [Plug In Name] Secondary package.

## Defining the Dependency in the Primary Package

In both above examples, the primary package will need to be modified to incorporate the secondary package(s) for DSC.

1. On the server, open the OSD file of the secondary package for editing. (It is advisable to use an XML editor to make changes to the OSD file; however, you can use Notepad as an alternative.)
2. Copy the **CODEBASE HREF** line from that file.
3. Open the OSD file of the primary package for editing.
4. Insert the **<DEPENDENCIES>** tag after the close of **</ENVLIST>** tag at the end of the **<VIRTUALENV>** section just before the **</VIRTUALENV>** tag.
5. Paste the **CODEBASE HREF** line from the secondary package after the **<DEPENDENCIES>** tag you just created.
6. If the secondary package is a mandatory package, meaning that it must be launched before the primary package is launched, add the **MANDATORY=”TRUE”** property inside the **CODEBASE** tag. If it’s not mandatory, the property can be omitted.
7. Close the **<DEPENDENCIES>** tag by inserting the following:

**</DEPENDENCIES>**

1. Review the changes made to the OSD file, and then save and close the file. The following example shows how the added section should appear. The tag values shown here are for example only.

**<VIRTUALENV>**

**<ENVLIST>**

**…**

**</ENVLIST>**

**<DEPENDENCIES>**

**<CODEBASE HREF="rtsp://virt\_apps/package.1/package.1.sft" GUID="D54C80FA-9DFF-459D-AA33-DD852C9FBFBA" SYSGUARDFILE="package.1\osguard.cp"/>**

**<CODEBASE HREF="rtsp://sample\_apps/package.2/sample.sft" GUID="D54C80FA-9DFF-459D-AA33-DD852C9FBFBA" SYSGUARDFILE="package.2\osguard.cp" MANDATORY="TRUE" />**

**</DEPENDENCIES>**

**</VIRTUALENV>**

## Dynamic Suite Composition Tool

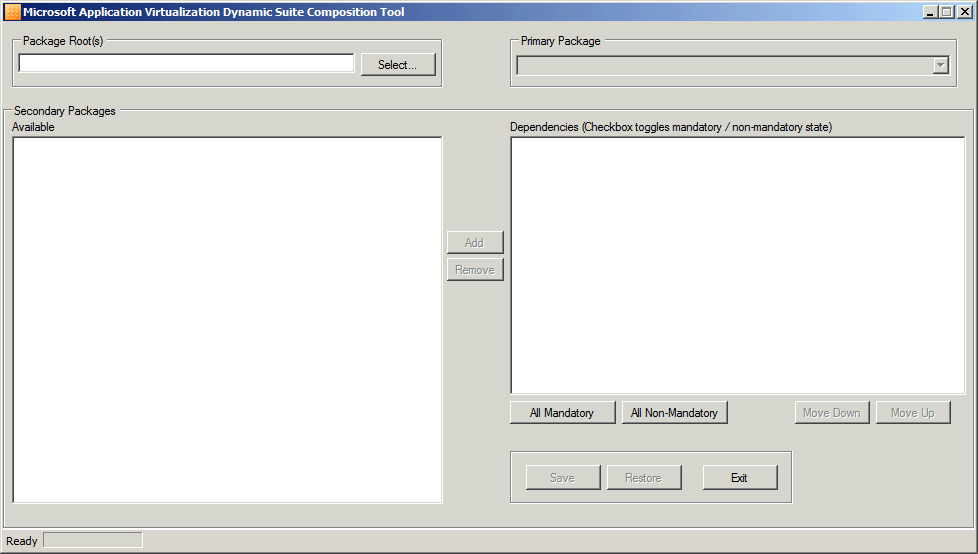
The Dynamic Suite Composition tool provides a GUI interface that makes it easier to configure packages to take advantage of the dynamic suiting feature. This tool also eliminates the possibility of incorrect editing of the OSD files required to take advantage of dynamic suiting. The tool can be installed and used from an App-V Management Server or can be installed remotely. The following outlines the steps required to use the Dynamic Suite Composition Tool.

**Note:** The Dynamic Suite Composition tool requires properly sequenced packages that will support dynamic suiting. The tool facilitates editing the OSD files to support dynamic suiting only.

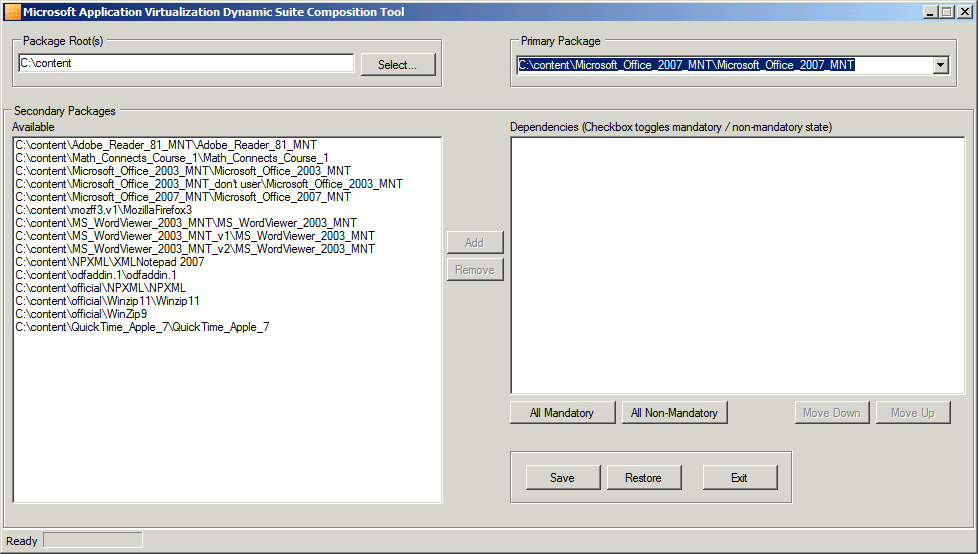
1. Download the Dynamic Suite Composition Tool from: <http://www.microsoft.com/downloads/details.aspx?familyid=DAA898DF-455F-438A-AA2A-421F05894098&displaylang=en>.
2. Run the self-extracting EXE and select a destination directory for the binaries.

**NOTE: The Dynamic Suite Composition tool doesn’t require installation; it is run directly from the directory where it was extracted.**

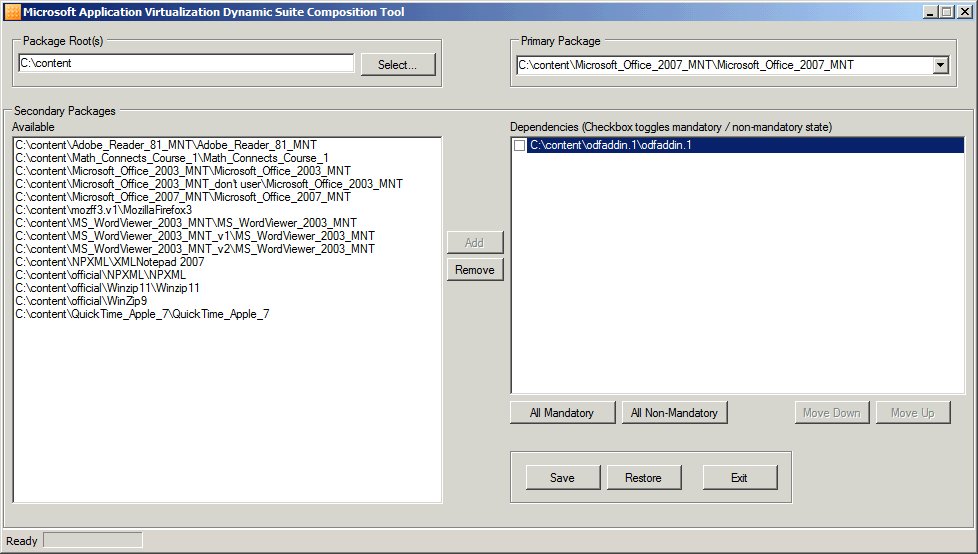
1. Open the Dynamic Suite Composition Tool.



1. Click on **Select** to choose the directory where the App-V packages for dynamic suiting exist.
2. After selecting the package directory, select the **Primary Package** from the drop-down list.

****

1. Select and **Add** the Secondary Package(s) from **Available** list.

****

1. Select the **Checkbox** to the left of any **Dependencies** that are mandatory or use the **All Mandatory / All Non-Mandatory** buttons to toggle the setting for multiple dependencies at one time.
2. Select **Save** to apply the changes to the OSD files to support dynamic suiting.

**Note: The Dynamic Suite Composition Tool creates backup copies of the original OSD files with the extension BAK. These files can be restored using the tool by selecting the primary application and selecting Restore.**

# Compression of the SFT

The package SFT file can be compressed during sequencing to reduce the amount of traffic that will be streamed on the network when loading the package. The compression will impact the end client, as the client PC will need to decompress the SFT and put the data into the client’s cache. The decompression algorithm is CPU-intensive. The default setting is “No Compression” and should be left unless the target LAN for streaming is excessively bandwidth limited.

# Advanced OSD Scripting

In some instances it may be necessary to make modifications to the application to get it to function properly in the App-V environment. One of the most common methods is making modifications to the OSD file to allow certain actions to be performed at a specified time during the execution of the application.

Scripting in the OSD file is an invaluable tool when sequencing / publishing applications. These scripts can be used to setup or alter the virtual environment before an application executes, or can be used to “clean up” the environment after the application terminates. These custom modifications can be made using any text editor or XML editor utility.

It is important to understand the OSD files, and their syntax, prior to using custom scripting. Only experienced sequencing engineers should make custom modifications to the OSD file, and you should always create a backup copy of the original OSD file prior to making any modifications. Placing an incorrect syntax in an OSD file could cause the application to not work properly, or can even prevent the application from appearing at all on the clients.

**OSD file**: Provides information necessary to launch the application, including the name of the Virtual Application Server from which to stream the application, and any Dynamic Suite Composition dependencies. Each application requires an OSD file. The OSD file is written in an XML format. The major sections are detailed below:

* **<SOFTPKG>:** This line displays the package’s Globally Unique IDentifier (GUID), name and version. Each of these fields must be unique across all published applications.
* **<IMPLEMENTATION>:** This section tells the Application Virtualization Client "how" to run the virtual application, including what protocol to use when streaming the application, what server to get the application from, whether or not to "terminate" any child processes when the application closes, etc.
* **<DEPENDENCY>:** This section lists any scripts that must be run prior to launching this application. It also specifies whether a specific version of the Application Virtualization Client must be installed on the client to run this application.
* **<PACKAGE NAME>:** This line lists the name of the package, as specified during sequencing.
* **<ABSTRACT>:** This section displays any comments that were listed during sequencing. You should list as much information here as necessary to describe the sequencing environment.
* **<MGMT\_SHORTCUTLIST>:** This section displays where the shortcuts will be placed on the client. For example, to place a shortcut on the Start\Programs\Virtual\_Applications menu, this would display as "%CSIDL\_PROGRAMS%\Virtual Applications."
* **<MGT\_FILEASSOCIATIONS>:** This section lists all of File Type Associations that will be registered on the client, so an application can be launched by double-clicking a file.

A sample OSD file:

Before we talk about scripting here is a generalized look at a typical OSD file.

<**SOFTPKG** GUID="*application GUID*" NAME="*application name*" VERSION="*version string*">

<**IMPLEMENTATION**>

<**CODEBASE** HREF="*location*“ GUID="*package*" FILENAME="*application*“

PARAMETERS=“*parameters*” SYSGUARDFILE="*state file*"/>

<OS VALUE="*platform*"/>...<OS VALUE="*platform*"/>

<WORKINGDIR>*optional working directory*</WORKINGDIR>

<**VIRTUALENV** *virtual environment configuration options*>

<ENVLIST>*optional variables*</ENVLIST> <REGISTRY>*optional values*</REGISTRY>

<POLICIES>*optional policies*</POLICIES>

<DEPENDENCIES>

<CODEBASE ... MANDATORY="TRUE|FALSE"/>...<CODEBASE ... MANDATORY="TRUE|FALSE"/>

</DEPENDENCIES>

</**VIRTUALENV**>

</**IMPLEMENTATION**>

<**DEPENDENCY**>

<CLIENTVERSION VERSION="4.5.0.0"/> <SCRIPT>*optional scripts*</SCRIPT>

</**DEPENDENCY**>

<**MGMT\_SHORTCUTLIST**><SHORTCUT *definition*>…<SHORTCUT …></**MGMT\_SHORTCUTLIST**>

<**MGMT\_FILEASSOCIATIONS**>

<PROGIDLIST>*shell integration*</PROGIDLIST><FILEEXTENSIONLIST>*FTAs*</FILEEXTENSIONLIST>

</**MGMT\_FILEASSOCIATIONS**>

</SOFTPKG>

## OSD Scripts

All scripts must be placed between the <DEPENDENCY> tags within the OSD files. You can choose to refer to an existing script by its file name or you can enter the exact syntax of the commands in the script section.

When developing scripts, the first step is to decide when the script will run. Using the **SCRIPT TIMING** and **EVENT** tags, the script can be executed at different times during the virtual application launch.

* **Pre Stream:** After the user launches the application, but prior to the application blocks streaming to the client workstation and the virtual environment being set-up. An example of a Pre-Stream script might be starting a necessary VPN connection to connect to a server that the application requires.
* **Post Stream:** After the user launches the application, and after the application blocks stream to the client workstation, but prior to the virtual environment being set-up. An example of a Post-Stream script might be to install a dependent component on the host workstation.
* **Pre Launch:** After the user launches the application, and after the bits stream to the client workstation, and after the virtual environment has been set-up. Whatever commands are called in a Pre-Launch script occur within the virtual environment. An example of a Pre-Launch script might be to ensure a file exists or a registry key gets added to the virtual environment.
* **Post Launch:** After the application launches, but prior to the user getting access to the application’s interface. An example of a Post-Launch script might be calling a script to synchronize a folder inside the virtual environment with one that is outside the virtual environment.
* **Post Shutdown:** After the virtual application has been closed. An example of a Post-Shutdown script might be to reset the workstation back to a clean state to remove any extraneous files that the application itself may have dropped onto the workstation.

**Note:** There is no Pre-Shutdown script as it would be impossible to know the exact moment before the user closes the application.

Next, the script must define where to run the script. The OSD script can be run in one of two locations: either inside the virtual environment or outside, or on the local operating system outside the virtual application virtual environment. The **PROTECT** tag is the only XML tag that defines where the script will run, either inside the virtual environment or outside. The two settings for the PROTECT tag are described below:

* **TRUE**: Allows the script to run within the protected environment. This script placement can be very useful for troubleshooting.
* **FALSE**: Allows the script to run outside the virtual environment. This script placement can be very effective when files need to be copied locally to a client but there is no need to penetrate the virtual environment.

**<SCRIPT TIMING=“PRE” EVENT=“LAUNCH” PROTECT=”TRUE”>**

The last two attributes that affect the script operation are **TIMEOUT** and **WAIT**. These two attributes define how much time the script will need to wait to completely execute. The TIMEOUT attribute, in seconds, defines how long the client will wait. A value of “0” defines the client will wait indefinitely. The WAIT attribute is still supported in version 4.5, for backward compatibility. A value of TRUE defines that the client should wait for the script to complete before starting the next step in the script. A value of FALSE defines that the client can continue onto the next step without waiting for the current step to complete.

**<SCRIPT TIMING=**“**PRE” EVENT=**“**LAUNCH” WAIT=”TRUE” TIMEOUT=”0”>**

**The following sequenced list describes the full application launch and shutdown and when the script events occur. The list also shows whether the virtual environment is loaded and whether or not the script could access the virtual environment.**

1. Virtual Application Launch / Shutdown Progression
   1. User clicks on program shortcut for a virtual application
   2. **Pre-Stream script runs (outside Virtual Environment)**
   3. Page streams FB1
   4. **Post-Stream script runs (outside Virtual Environment)**
   5. Virtual environment (VE) is loaded
   6. **Pre-Launch script runs (inside or outside Virtual Environment**)
   7. The virtual application launches
   8. **Post-Launch script runs (inside or outside Virtual Environment)**
   9. If background streaming is enabled for the application,   
      FB2 will stream into the client cache in the background.
   10. User exits the virtual application
   11. The virtual environment is unloaded
   12. **Post-shutdown script runs (outside Virtual Environment)**

**Note:** The client does not support a **SHUTDOWN** event with Timing of **PRE**.

**Note:** Scripts can be written in any language but their language must be installed locally on the client machine. For example, if they are using PERL, the PERL runtime must be locally installed on the client machine.

## OSD: SCRIPTBODY vs HREF

There are two kinds of scripts **<SCRIPTBODY>** and **<HREF>** which are described in more detail:

1. HREF – Normally used when executing a single command.
2. SCRPITBODY – Used when needing more complex operations and/or multiple commands to be run.

|  |  |
| --- | --- |
| <SCRIPTBODY> | <HREF> |
| Can be used to call an executable by searching for the executable in the client’s Path statement. | **Can be used to call an executable by searching for the executable in the client’s Path statement.** |
| Uses the “\” as an escape character and as such requires an extra “\” to pass the actual character. | **Does not use the “\” as an escape character.** |
| Can be used to pass non executable commands such as the DOS make directory (md). This is actually a part of the command.com. | **Cannot be used to pass non-executable commands such as the DOS make directory (md).** |
| When used, the contents of the script are copied to a temporary .bat file on the client’s drive. | **Does not create a temporary .bat file but instead executes the contents of the script directly on the client.** |
| When used a DOS window appears on the screen during the execution of the script’s .bat file. | **No DOS window appears on the screen unless the process being called is a console application.** |
| <SCRIPTBODY> | **<HREF>** |
| Can be used to call an executable by searching for the executable in the client’s Path statement. | **Can be used to call an executable by searching for the executable in the client’s Path statement.** |

Table 15: OSD Scripting <SCRIPTBODY> vs. <HREF>

## <SCRIPTBODY> Examples

**Example 1 – ABORTRESULT and SUCCESSRESULT**

If ABORTRESULT is specified, the client will check the script's exit code when it completes. If SUCCESSRESULT is specified (and if ABORTRESULT is not or it does not match the script's exit code), the client will check the script's exit code to see if it matches the value in SUCCESSRESULT.

<DEPENDENCY> <SCRIPT TIMING="PRE" EVENT="LAUNCH" PROTECT="TRUE" WAIT="TRUE" TIMEOUT="10" **SUCCESSRESULT**="1" **ABORTRESULT**="0">

<SCRIPTBODY>@ if not %COMPUTERNAME% == “SpecialComputer” exit 1 @ exit 0

</SCRIPTBODY>

</SCRIPT>

</DEPENDENCY>

A more detailed explanation of these commands can be found at:

<http://support.microsoft.com/kb/930973/en-us>

**Example 2 – Pre-Launching Command Prompt**

<DEPENDENCY>

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

**<SCRIPTBODY>cmd.exe</SCRIPTBODY>**

</SCRIPT>

</DEPENDENCY>

**Example 3 – Embedding Batch File Commands within OSD File**

<DEPENDENCY>

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

**<SCRIPTBODY>**

**net use k: \\\\w2k-pdc\\netlogon\n**

**CALL k:\\usr-w2k.cmd\n**

**\\\\sfc-App-V\\shr\editini.exe c:\\word\\word.ini "File Locations" TempPath c:\\temp\n**

**</SCRIPTBODY>**

</SCRIPT>

</DEPENDENCY>

**Example 4 – Pre-Launching EXE from within the Sequence**

<DEPENDENCY>

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

**<SCRIPTBODY>%SFT\_MNT%\\OfficeXP\\Office10\\proflwiz.exe</SCRIPTBODY>**

</SCRIPT>

</DEPENDENCY>

**Example 5 – Pre-Launching a Data File (e.g. AVI file) from within the Sequence**

<DEPENDENCY>

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

**<SCRIPTBODY>%SFT\_MNT%\\App\\Data\\Sequencing.avi</SCRIPTBODY>**

</SCRIPT>

</DEPENDENCY>

**NOTE:** In this example, the sequencing.avi launches the locally installed Media Player.

**NOTE:** A data file (e.g. sequencing.avi) cannot be specified within the FILENAME entry of the CODEBASE tag as shown below.

<CODEBASE

**FILENAME="App\Data\Sequencing.avi" NOT ALLOWED**

SYSGUARDFILE="App\osguard.cp"

HREF="rtsp://App-V:554/app.sft"

GUID="A2A6D95E-8898-41C0-8020-797B23A1E917"

SIZE="33363442"/>

**Example 6 – Pre-Launching a Control Panel Applet within the Sequence**

<DEPENDENCY>

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

**<SCRIPTBODY>%SFT\_MNT%\\App\\CPL\\SYSDM.CPL</SCRIPTBODY>**

</SCRIPT>

</DEPENDENCY>

**Example 6a – Pre-Launching a Control Panel Applet within the Sequence**

<DEPENDENCY>

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

<SCRIPTBODY>

**"C:\\Program Files\\Common Files\\System\\Mapi\\1033\\mlcfg32.cpl"**

</SCRIPTBODY>

</SCRIPT>

</DEPENDENCY>

## <HREF> Examples

**Example 1 – Accessing Local Application**

<DEPENDENCY>

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

**<HREF>c:\winnt\system32\cmd.exe</HREF>**

</SCRIPT>

</DEPENDENCY>

**Example 1a – Accessing Batch File via UNC**

<DEPENDENCY>

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

**<HREF>\\SRV\_NAME\NT\_SHR\PRE\_SCRIPT.CMD</HREF>**

</SCRIPT>

</DEPENDENCY>

**NOTE:** HREF tag cannot be used with %SFT\_MNT% or to launch EXE from within virtual environment.

## <PARAMETERS> Examples

The PARAMETERS attribute is used to provide command line arguments to the FILENAME attribute. The FILENAME attribute points to the location of the program to execute when launching the virtual application. An example is placing filename in the PARAMETERS attribute, like a database file that will be opened when the application is launched.

**NOTE:** The PARAMETERS tag will not support environment variables (e.g. %SFT\_MNT%).

**Example 1 – Generic Example**

<CODEBASE

**FILENAME="\ABCD\EXECUTE\Z\_RUN.EXE"**

**PARAMETERS="FORM=MRF01 LANG=ENG ZENV=X\_POINT\_CHAR"**

SYSGUARDFILE="ACBS\osguard.cp"

HREF="rtsp://hercules:554/ACBS\_V2.sft"

GUID="BD9AF94E-4A8E-415D-B57F-54A454911FAA"

SIZE="648019522"/>

**Example 2 – Launching Data File via a Relative Path**

<CODEBASE

**FILENAME="App\WordEditor\wordpad.exe"**

**PARAMETERS=" ..\Data\cli.doc"**

SYSGUARDFILE="App\osguard.cp"

HREF="rtsp://App-V:554/app.sft"

GUID="A2A6D95E-8898-41C0-8020-797B23A1E917"

SIZE="33363442"/>

**NOTE:** ‘..\Data\cli.doc’ is located in the following location within the sequence:

%SFT\_MNT%\App\Data\cli.doc

## <WORKINGDIR> Examples – Specifying a Working Directory

The WORKINGDIR tag defines where to look for filenames that are not given a full path as in Windows. In the first example below, it is used make the C:\temp directory the default location to look for files if referenced name with no path.

**Example 1**

<IMPLEMENTATION>

**<WORKINGDIR>C:\temp</WORKINGDIR>**

<CODEBASE HREF="rtsp://ds1-soft1:554/np\_ecj.sft?Customer=Sequencer"

FILENAME="np\_ecj\notepad.exe" PARAMETERS="cjm.txt"

GUID="D47C0A23-32CF-11D6-990A-00A0CC6413D5"/>

<OS VALUE="WinXP"/>

</IMPLENENTATION>

**Example 2**

<IMPLEMENTATION>

**<WORKINGDIR>C:\program files\internet explorer</WORKINGDIR>**

<CODEBASE HREF="rtsp://ds1-soft1:554/np\_ecj.sft?Customer=Sequencer"

FILENAME="np\_ecj\notepad.exe" PARAMETERS="cjm.txt"

GUID="D47C0A23-32CF-11D6-990A-00A0CC6413D5"/>

<OS VALUE="WinXP"/>

</IMPLENENTATION>

**Example 3 – Example with Variable**

<IMPLEMENTATION>

**<WORKINGDIR>%SFT\_MNT%\App\Data</WORKINGDIR>**

<CODEBASE HREF="rtsp://ds1-soft1:554/np\_ecj.sft?Customer=Sequencer"

FILENAME="np\_ecj\notepad.exe" PARAMETERS="cjm.txt"

GUID="D47C0A23-32CF-11D6-990A-00A0CC6413D5"/>

<OS VALUE="WinXP"/>

</IMPLENENTATION>

## <REGISTRY> Examples – Registry Customizations

<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>

## Launching Local Applications

Using the FILENAME attribute, the OSD file for an application does not have to open an executable within the virtual environment. In the case of Web add-ins like Microsoft Silverlight, the FILENAME attribute would be pointed to Internet Explorer. There are two examples listed below of launching local applications.

**NOTE:** The PARAMETERS tag will not support environment variables (e.g. %SFT\_MNT%).

**Example 1 – Launching Application that is local (not part of the Sequence)**

<CODEBASE

**FILENAME="C:\Program Files\Internet Explorer\iexplore.exe"**

SYSGUARDFILE="App\osguard.cp"

HREF="rtsp://App-V:554/app.sft"

GUID="A2A6D95E-8898-41C0-8020-797B23A1E917"

SIZE="33363442"/>

<VM VALUE="Win32">

<SUBSYSTEM VALUE=”Windows”/>

</VM>

<CODEBASE

**FILENAME=" C:\Program Files\Windows Media Player\wmplayer.exe"**

**PARAMETERS=" q:\app\data\sequencing.avi"**

SYSGUARDFILE="App\osguard.cp"

HREF="rtsp://App-V:554/app.sft"

GUID="A2A6D95E-8898-41C0-8020-797B23A1E917"

SIZE="33363442"/>

<VM VALUE="Win32">

<SUBSYSTEM VALUE=”Windows”/>

</VM>

**NOTE:** C:\Program Files\Windows Media Player\wmplayer.exe is a local application.

## Setting Environment Variables

Using these steps, an environment variable can be set which could be used by the application in the virtual environment. In this example, the system PATH variable is appended by the Mount path (for eg. Q:) and Visual Studio path. Hence, the application would use this updated PATH environment variable when required.

<VIRTUALENV>

<ENVLIST>

<ENVIRONMENT VARIABLE="PATH">%PATH%;%SFT\_MNT%\vs2005\Microsoft SDK\bin\.;</ENVIRONMENT>

</ENVLIST>

</VIRTUALENV>

## Launching Application via Non-Default Provider Policy

During installation of the App-V Infrastructure, a Default Provider Policy is created and assigned usage permissions. Provider policies can be created and modified to affect the authentication, access permissions of virtual applications, and licensing. Multiple provider policies can be created so that one is used for applications where no license enforcement is needed and another where license enforcement is enabled. In this scenario, applications that need to use a provider policy that is not the default must have the OSD file modified. The following describes the change to the OSD file for a non-default provider policy (in bold), where Sales is the name of the non-default provider policy.

<CODEBASE

FILENAME="Photoshp\Photoshop.exe"

SYSGUARDFILE="Photoshp\osguard.cp"

HREF="rtsp://sgserver:554/adobephotoshop70.sft**?Customer=Sales**"

GUID="FE72DDCD-6E80-4ABE-8E82-8208A134CB34"

SIZE="178299260"/>

## Running an Application as a CONSOLE application

THE SUBSYSTEM tag defines the type of application defined in the FILENAME attribute will be run with two possible elements (**windows or console)** to define the type of application (windows=windows and console=dos) being executed. In this example, the OSD is describing that the application is a 16-bit DOS application. The values for the VMVALUE tag are Win16, Win32, and Win64 and are case sensitive.

<OS VALUE="Win2K"/>

<OS VALUE="Win2KSvr"/>

<OS VALUE="Win2KTS"/>

<OS VALUE="Win2003Svr"/>

<OS VALUE="Win2003TS"/>

<OS VALUE="WinXP"/>

<VIRTUALENV>

<ENVLIST/>

</VIRTUALENV>

**<VM VALUE="Win16">** Can also be “Win32” or “Win64”􀃅 **WARNING! VM Values are case sensitive.**

**<SUBSYSTEM VALUE="console"/>**

**</VM>**

</IMPLEMENTATION>

## OSD XML Syntax Check

To check the XML format of an OSD file, change the .OSD file extension to .XML and drop the .XML file into an Internet Explorer window. If there are any open or missing tags the file will not be displayed.

Note: XML file editors such as XML Notepad can also be used to check XML file format

Example XML file syntax display: <?xml version="1.0" standalone="no" ?>

**-** <SOFTPKG GUID="**26998703-328D-41DE-A256-5AFACD27ED08**" NAME="**WinZip**"

VERSION="**90**">

**-** <IMPLEMENTATION>

<CODEBASE

HREF="**rtsp://ntlabsg01:554/Winzip\_Winzip\_901\_MNT/Winzip\_Winzip\_901\_**

**MNT.sft**" GUID="**A9800C2A-E9ED-4711-B90C-808D6DD0AABC**"

FILENAME="**Winzip.v1\WINZIP32.EXE**" PARAMETERS=""

SYSGUARDFILE="**Winzip.v1\osguard.cp**" SIZE="**6225393**" />

<OS VALUE="**Win2K**" />

<OS VALUE="**Win2KSvr**" />

<OS VALUE="**Win2KTS**" />

<OS VALUE="**Win2003Svr**" />

<OS VALUE="**Win2003TS**" />

<OS VALUE="**WinXP**" />

**-** <VM VALUE="**Win32**">

<SUBSYSTEM VALUE="**windows**" />

</VM>

**-** <VIRTUALENV>

<ENVLIST />

</VIRTUALENV>

</IMPLEMENTATION>

**-** <DEPENDENCY>

<CLIENTVERSION VERSION="**2.0.5.0**" />

</DEPENDENCY>

<SUITE NAME="**Winzip\_Winzip\_901\_MNT**" />

<TITLE />

<ABSTRACT>**Win XP Sp1, IE 6.0 Seq Build 300**</ABSTRACT>

**-** <MGMT\_SHORTCUTLIST>

<SHORTCUT LOCATION="**%CSIDL\_STARTMENU%**" WORKINGDIR=""

DISPLAY="**WinZip**" ICON="**%SFT\_MIME\_SOURCE%/WinZip90.ico**" />

<SHORTCUT LOCATION="**%CSIDL\_DESKTOPDIRECTORY%**" WORKINGDIR=""

DISPLAY="**WinZip**" ICON="**%SFT\_MIME\_SOURCE%/WinZip90.ico**" />

</MGMT\_SHORTCUTLIST>

</SOFTPKG>

# Advanced Sequencing Techniques

This section will briefly cover some sequencing challenges and how to overcome them. It is also recommended to be aware of the logs created when sequencing, a list of the log files related to the sequencer process and purpose.

## Sequencing Applications That Cannot Install to Q:\

At some point while sequencing applications, you will most likely run across an application that will not let you install to Q:\ (PowerPoint Viewer). This by itself does not present a problem when sequencing the application. In fact, only one minor modification will need to be made.

* When you hit the “Start Monitoring” button you will browse to the root of Q:\.
* Make a directory for your application using the 8.3 naming convention and select this directory as the asset directory.
* Select Ok and continue sequencing as normal..

There are pro’s and con’s when sequencing with this method.

## Sequencing Web-Based Applications

Internet Explorer is NOT supported in a sequenced application. However, you can sequence plug-ins or ActiveX controls for Internet Explorer.

Figure 15: Sequencing Web-Based Applications Process

This pulls the locally installed application into the virtual environment with the additions you have specified. In this case it was a Web plug-in. This lets you have a clean and secure Internet Explorer that you can lock down on a user’s machine and then allow them to use the plug-ins that you define.

## Sequencing Applications that Require Access to Local Devices and/or Resources

In many ways, you sequence these applications no different than you would any other. The applications inside the virtual environment can see the local resources on the OS and can access them. Devices are another matter. App-V cannot virtualize drivers. If you can install the driver separately from the application, then the application will fall through to the OS to look for the driver and see that it is installed.

## Sequencer Log Files

* ***Sft-seq-log.txt***  
  Description: The main log file for logging on the Sequencer.  
  Path: ProgramFiles%\Microsoft Application Virtualization Sequencer\Logs\Sft-seq-log.txt
* ***Sftrbt.txt***  
  Description: Stores the actions taken during the reboot processing.  
  Path: %ProgramFiles%\Microsoft Application Virtualization Sequencer \App-V Sequencer\Logs\Sftrbt.txt
* ***SFTCallBack.txt***  
  Description: Records all process starts and stops seen by SystemGuard.  
  Path: %ProgramFiles%\Microsoft Application Virtualization Sequencer \App-V Sequencer\Logs\SFTCallBack.txt
* ***Filter.log***  
  Description: A binary log that stores file activity during sequencing.  
  Path: %ProgramFiles%\Microsoft Application Virtualization Sequencer \App-V Sequencer\Logs\Filter.1.log
* ***Files.txt***  
  Description: Logs all files in the VFS. It must be enabled by setting   
  HKLM\Software\Softricity\App-V  
  Sequencer\CurrentVersion\Configuration\FileManifest to “1”. (REG\_DWORD)  
  Path: %ProgramFiles%\Microsoft Application Virtualization Sequencer\App-V Sequencer\Logs\File.txt

# Finding Additional Information

## Error Codes and the Microsoft Knowledge Base

Microsoft has completed the process of migrating data off of the old Softricity knowledge base. The knowledge base contains useful information regarding planning, implementation and troubleshooting App-V. A link to the Microsoft knowledge base follows:

<http://support.microsoft.com>

*Error Code Formats*: When searching the knowledge base for error codes, be aware that the first 12 digits are unique to the version of the App-V Client software. The ending 8 digits of the error code remain constant between all client versions.

When you search for an error code, enter the last 8 digits such as 000DF004.

e.g., xxxxxx - xxxxxxxx - xxxxxxxx

or

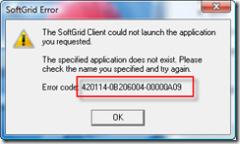
[](http://blogs.technet.com/blogfiles/virtualworld/WindowsLiveWriter/Sowhatdoalltheseerrorsactuallymean_1470B/image_1.png)

Figure 16: Sample Error Message

The first block of 6 characters are around the version build of the platform/client. Basically, when you see these, it’s information based and will vary depending on:

* Major Version
* Minor Version
* Patch Version
* Build

The Next Block of 8 Characters Identify

* File ID
* Line
* Info

The Last Block of 8 Characters show:

* Problem Code

The last 10 digits of the error code are constant between the SoftGrid builds. Because the last 10 digits are consistent we tend to use these for identifying and searching our errors. Regardless of the version, builds, etc., we can use this knowledge to identify problems and issues as these codes are reused between modules.

In addition, Justin Zarb from Premiere Field Engineering in the UK has written an excellent article related to the App-V error codes and what they mean. The article can be found at:

<http://blogs.technet.com/virtualworld/archive/2007/07/27/so-what-do-all-these-errors-actually-mean.aspx>

## App-V Team Blog

This is probably the single greatest repository of App-V related information on the Web. This should be the first stop for anyone interested in learning about App-V and sequencing. The blog is updated frequently so keep checking it for more information. Additionally, look through the archives for valuable information.

* App-V Team Blog - [http://blogs.technet.com/App-V/default.aspx](http://blogs.technet.com/softgrid/default.aspx)
* RSS Feed - [http://blogs.technet.com/App-V/rss.xml](http://blogs.technet.com/softgrid/rss.xml)

## App-V Related Technical Discussion Forums and Web Sites

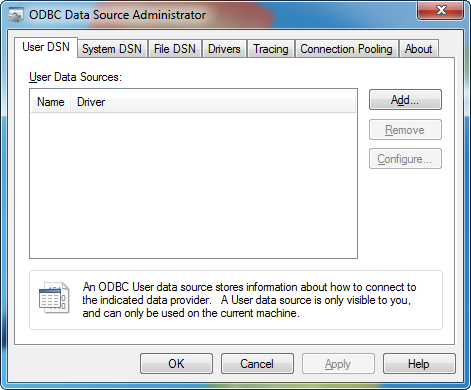
Here is a list of non-Microsoft sites that contain information related to the App-V product and sequencing. Some of these sites include useful App-V related technical discussion forums. While these sites contain valuable information about App-V, it should be noted that these sites are not affiliated with Microsoft in any way and Microsoft cannot verify any of the information contained within.

* <http://www.softgridguru.com/>
* <http://www.brianmadden.com>
* <http://blogs.technet.com/virtualworld>
* <http://forums.microsoft.com/TechNet/default.aspx?ForumGroupID=497&SiteID=17>

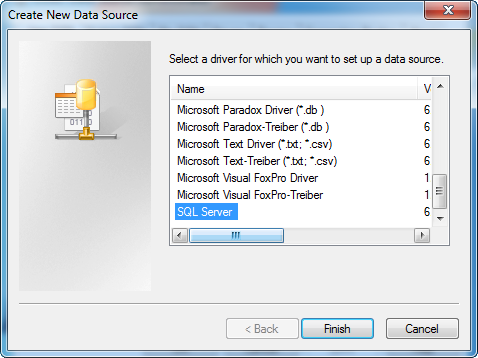
# Appendix A – Creating a Dummy ODBC

Step-by-step instructions for setting up dummy ODBC entries.

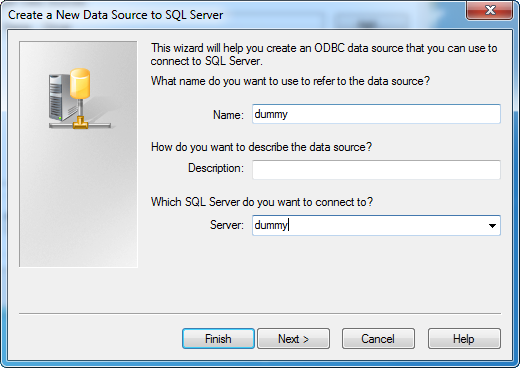
1. Open the Administrative Tools 🡪 Data Sources (ODBC) applet.
2. Select the User DSN tab.



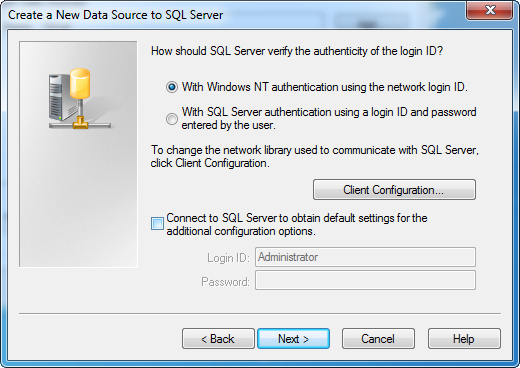
1. Click Add…



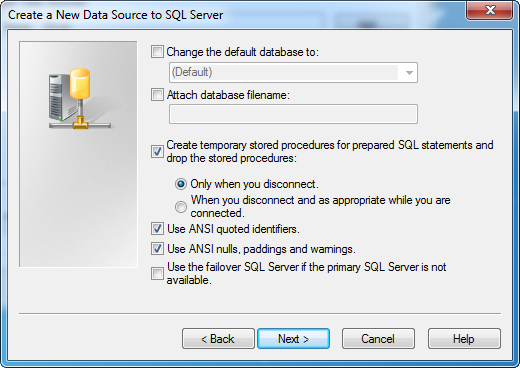
1. For “Select a driver…” select SQL Server and click Finish.



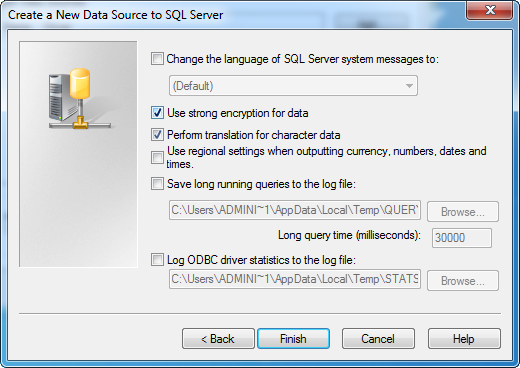
1. For Name, enter “dummy.”
2. For Description, enter “Required for application sequencing.”
3. For Server, enter “dummy.”
4. Click Next.



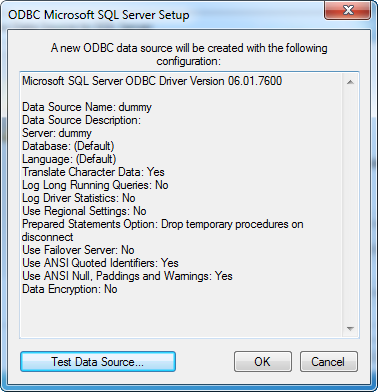
1. Deselect the “Connect to SQL Server to obtain settings…” checkbox.
2. Click Next.



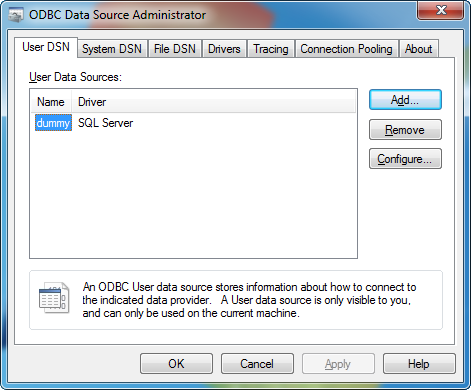
1. Click Next.



1. Click Finish.

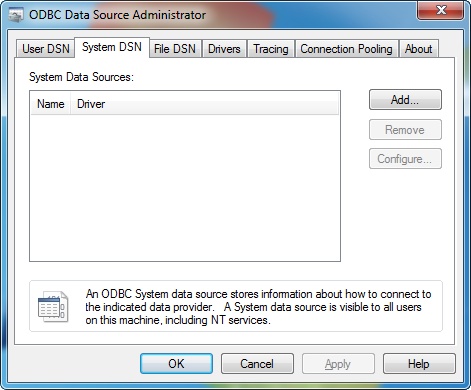


1. Click OK. Note: Do not click “Test Data Source…” since this test would fail.

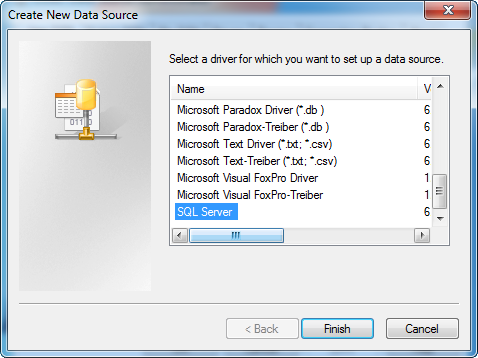


* **Next, create an ODBC System DSN setting as part of the Sequencer base image as follows:**

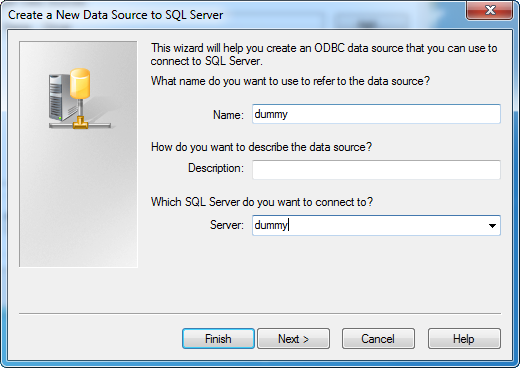
1. Select the System DSN tab.



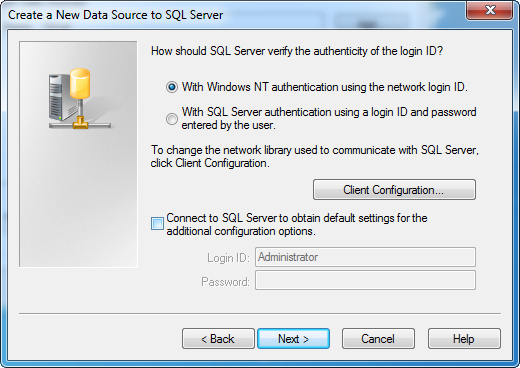
1. Click Add…



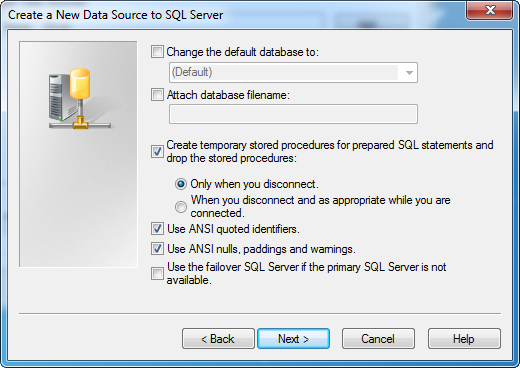
1. For “Select a driver…” select SQL Server and click Finish.



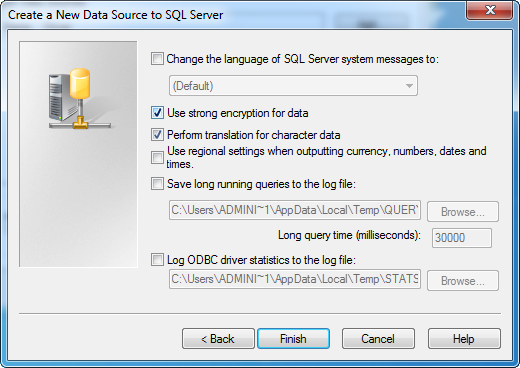
1. For Name, enter “dummy.”
2. For Description, enter “Required for application sequencing.”
3. For Server, enter “dummy.”
4. Click Next.



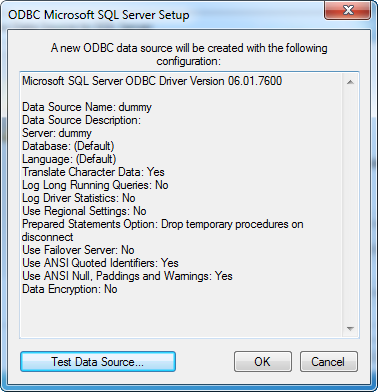
1. Deselect the “Connect to SQL Server to obtain settings…” checkbox   
   and click Next.



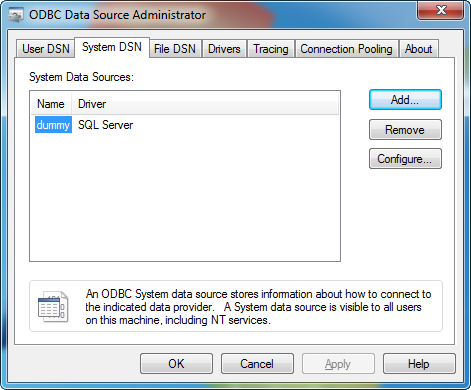
1. Click Next.



1. Click Finish.



1. Click OK. Note: Do not click “Test Data Source…” since this test would fail.

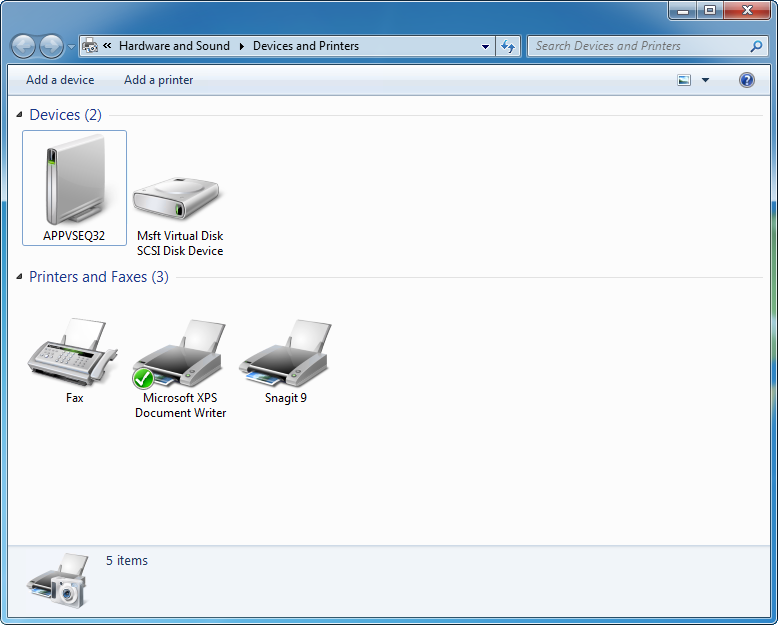


1. Click OK to close the ODBC applet.

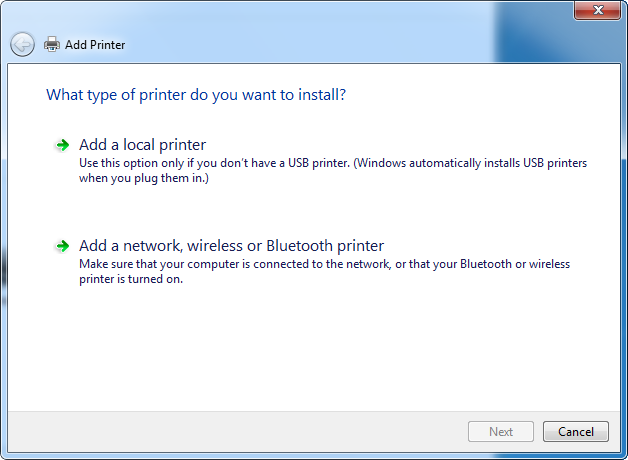
# Appendix B – Creating a Dummy Printer

Step-by-step instructions for setting up a dummy printer.

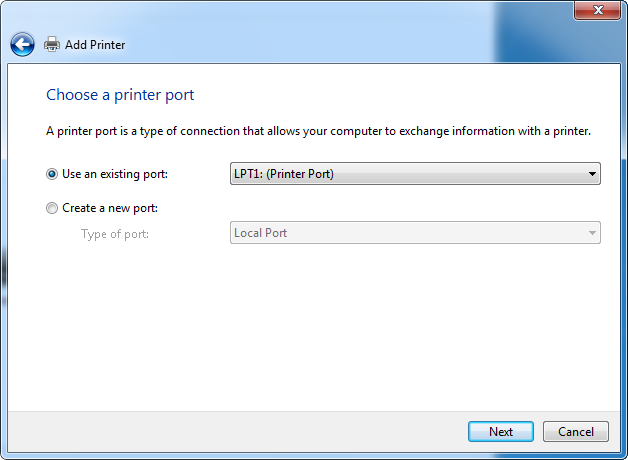
1. Open the Control Panel 🡪 Hardware and Sound 🡪 Devices and Printers applet.



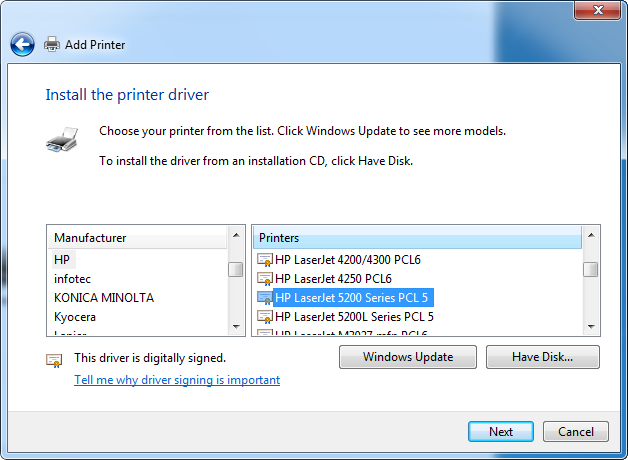
1. Select “Add a Printer.”



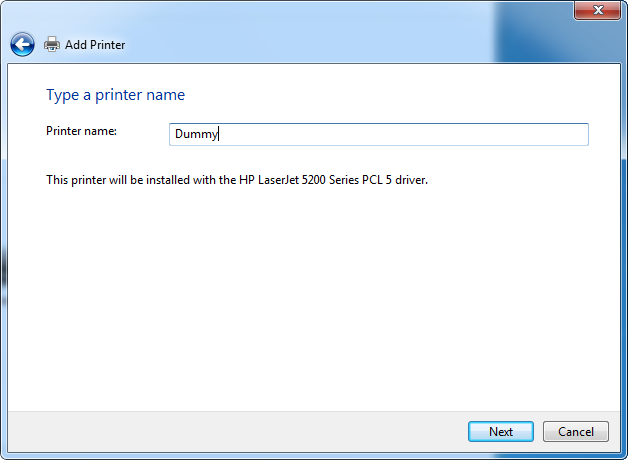
1. Select “Add a local printer.”



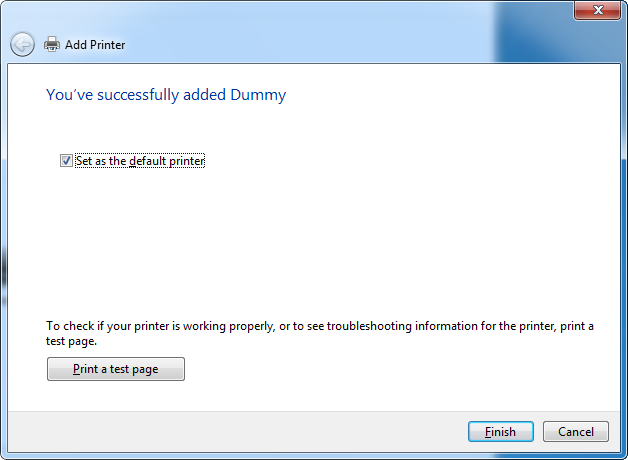
1. Click Next.



1. For Manufacturer, select “HP.”
2. For Printers, “HP LaserJet 5200 Series PCL 5.”
3. Click Next.



1. For Printer Name, enter “Dummy.”
2. Click Next.
3. On the “Printer Sharing” screen click Next.



1. Click Finish.
2. Close Devices and Printers applet.