Application Virtualization Explorer

User Guide

Version 4.3

gridmetric

Contents

Product Editions

Application Virtualization Explorer now comes in three different editions. Collectively these are referred in this document as "Application Virtualization Explorer".

- Application Virtualization Explorer Express (later "AVE Express") supports viewing but not saving out App-V 5.0 or newer packages
- Application Virtualization Explorer (later "AVE") supports viewing and saving App-V 5.0 or newer packages.
- Application Virtualization Explorer Professional supports viewing and saving both legacy App-V packages (up to App-V 4.6) as well as App-V 5.0 and newer packages. Application Virtualization Explorer Pro comes in two separate binaries, one for legacy App-V package operations ("AVE Pro Classic") and one for App-V 5.0 package operations ("AVE Pro").

System requirements

Application Virtualization Explorer can be run on the following Windows versions:

- Windows 7 RTM or newer
- Windows Server 2008 R2 RTM or newer
- Windows 8 RTM or newer
- Windows Server 2012 or newer
- Windows 10 preview versions (experimental support only)

Application Virtualization Explorer supports operating systems above in both 32-bit and 64-bit editions.

Please note that Windows operating systems from Windows XP SP3 to Windows Vista SP1, while technically capable of running AVE Pro for legacy App-V packages, are now not officially supported by Gridmetric if any problems are encountered when using Application Virtualization Explorer Professional on these platforms.

Additionally, Application Virtualization Explorer requires following component(s) to be installed on each machine it is run at:

- Microsoft .NET Framework version **2.0 SP2** runtime or newer
  or
  Microsoft .NET Framework version **3.51 SP1** runtime or newer
  or
  Microsoft .NET Framework version **4.0 (FULL)** runtime or newer

AVE Express, AVE and AVE Pro (in handling App-V 5.0 and newer packages) all require usage of user account having local administrative rights on the machine or user being able to elevate to as administrator.

AVE Pro Classic in handling legacy App-V packages does not require administrative rights.

Features and benefits

Application Virtualization Explorer provides following features and benefits:

- Support for opening, inspection and export of data contained in all current (AVE Express, AVE and AVE Pro) and past App-V package formats (AVE Pro Classic only)
- Support for opening, inspection and export of data contained in legacy App-V Client's cached files (PKG files) (AVE Pro Classic only)
- Support for easy editing of all aspects App-V package's contents (with the exception of ACL editing) through GUI interface, including importing new content to it
- Support for writing inspected and/or modified App-V packages back to the disk in all past (AVE Pro Classic only) and present (AVE and AVE Pro) App-V package formats
- Converting legacy App-V packages to newer format (AVE Pro only)
- Support for creation of empty App-V packages from scratch for manual content build through Application Virtualization Explorer's UI functionality

Please note that Application Virtualization Explorer does not support creation of new App-V packages using installation monitoring. Furthermore, affecting changes to an existing package using change monitoring – type of operation is not supported.

Support for 64-on-32 and 32-on-64 bitness operations

Even though Application Virtualization Explorer makes it possible to open and also to edit any App-V package when running either in 32-bit or 64-bit Windows, official supportability matrix from Microsoft requires that packages originally created with 64-bit Sequencer should always be edited in 64-bit system and vice versa.

This requirement is mainly in place to make sure that 32-bit and 64-bit specific or dependent directories (such as but not limited to: C:\Program Files, C:\Program Files (x86), c:\Windows\System32 and c:\Windows\SysWOW64) and registry branches will not inadvertently be written using wrong encodings and locations while saving a modified package. Please see more thorough discussion on possible limitations and issues for mixed 32/64 –bit operation in legacy App-V packages Gridmetric's blog at http://blog.gridmetric.com/2011/09/26/possible-caveats-in-mixing-32-bit-and-64-bit-app-v-packages-and-environments/.

Note that to comply with this concern, users of Application Virtualization Explorer are also advised to edit 64-bit originated App-V packages only in 64-bit Windows and 32-bit originated App-V packages only in 32-bit Windows with Application Virtualization Explorer. Any deviation from this recommendation is done on sole responsibility of the user and the resulting package will not be supported by the Gridmetric or Microsoft. Using Application Virtualization Explorer to only view or examine of mixed bitness packages normally doesn't cause problems as such, but resolved mappings (e.g. file-system paths) of registry values etc. might not be fully trusted in this scenario. Examining 64-bit packages on 32-bit operating system can, in rare cases, result of inability to correctly decode the virtual registry due to overlapping path entries as key names.

Getting started

What is Application Virtualization Explorer?

Application Virtualization Explorer is an advanced all-purposes Microsoft App-V virtual application package editor and troubleshooting software.

Application Virtualization Explorer packs number of exiting and advanced features into easy to use and understandable interface, going far beyond the capabilities of package modification using Microsoft –provided built-in tools and what other 3rd party App-V package -capable editors or viewers can offer. Application Virtualization Explorer targets **all** the use cases for App-V packages regardless of their deployment methodologies or infrastructures; be it SCCM –based usage, App-V's native infrastructure, lightweight infrastructure or custom delivery methods using App-V package MSIs.

Application Virtualization Explorer can be used by all App-V administrators, virtual application packagers and support personnel alike; requiring easier, more streamlined and straightforward editing of package's content than what is available with using the other options on the market.

Installing Application Virtualization Explorer

Application Virtualization Explorer is supplied in Windows Installer format (MSI) for easy or/and automated deployment to your computer systems.

Depending on the Windows installation you will have to use either 32-bit or 64-bit installer package, both which are named using the following format:

AVE_Professional_*majorversion.minorversion.patch_architecture*.msi ***or***

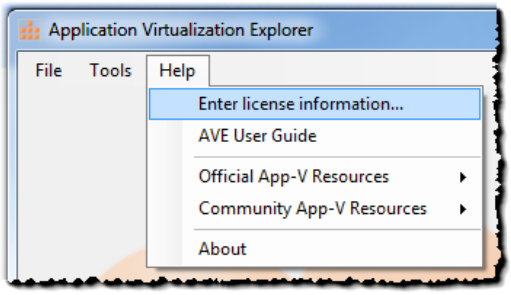
AVE_ *majorversion.minorversion.patch_architecture*.msi ***or***

AVE_Express_*majorversion.minorversion.patch_architecture*.msi

Majorversion, *minorversion* and *patch* will signify version number of the Application Virtualization Explorer (such as 4.2.0), while *architecture* signifies target platform which can be wither x86 for 32-bit systems or x64 for 64-bit systems. Please note that 32-bit install package does not install to 64-bit Windows and vice versa.

If you are upgrading from the previous version of Application Virtualization Explorer, you can just run the MSI package as normal and it will automatically replace existing version present on the system.

Licensing



After Application Virtualization Explorer has been installed and before it can be used, you must supply a valid license key to it.

Depending on the license bought, Application Virtualization Explorer is licensed either per machine and each separate machine requires an individual Application Virtualization Explorer license, or per user in which case each separate user requires and individual Application Virtualization Explorer license.

For multi-user machine scenarios – such as Terminal Servers or Remote Desktop Session Host servers – Application Virtualization Explorer will need a specific multi-user license key when operating in per machine licensing mode.

Application Virtualization Explorer will automatically prompt you for license information upon application startup if it has not been set already. You can also access license registration function from the Help menu (AVE Pro Classic only), using **Enter license information**, if the license information has not yet been supplied.

To supply license for Application Virtualization Explorer, you need to enter license key to the field marked as *License key*. If the license key is per named user type license, then the licensee name needs to be supplied to the *Licensee name* –field which will be enabled. In the case of per machine license, no additional name needs to be used.

Please make sure that you enter both the key, and optionally the name, exactly as you got them from the license purchase confirmation email.

Working with App-V packages in AVE Pro Classic

Application Virtualization Explorer supports loading of App-V packages in two different ways:

- Loading full App-V Package (recommended)
- Loading a SFT -file only

Upon opening the package file(s), Application Virtualization Explorer will decode the contents of all the files into an in-memory representation which means that any and all changes made to the package won't be persisted until the package is saved from Application Virtualization Explorer back to the disk directory.

This feature will enable a safe editing of the packages without the issue of accidentally overwriting original package files.

You'll have several different methods available for opening an App-V package using Application Virtualization Explorer:

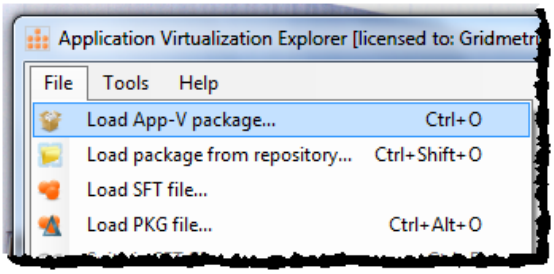
- Using load –functionality in the File –menu
- Drag & drop of package file from Explorer
- Double-clicking SPRJ or SFT file in Windows Explorer
- Accessing recent package list on the program's main screen
- Using package repository browser

Loading package from menu

To load package using menu functionality, select either **Load App-V package** or **Load SFT file** command from the File –menu. This will open a standard Windows file browser which allows you to select either SPRJ file (in case of opening full package) or SFT file (in case of opening just a SFT file) that you want Application Virtualization Explorer to open for you.

Descriptions of the differences of two package types are given below:

Full App-V package



Full App-V package –with both binary contents in SFT file and all associated metadata such as applications being published from package in OSD files – is loaded by using standard App-V SPRJ (project) file as found in the package source directory.

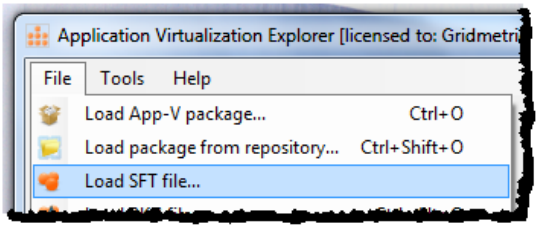
Using this file as a reference, Application Virtualization Explorer is then able to load all the other files making up the complete package. Note that while SPRJ file actually does not list all of the files but only the most important ones (SFT and OSD files), Application Virtualization Explorer will detect the remaining file types (icons, manifest and MSI) from package's directory.

Although the detection mechanism tries to exclude any files that do not actually belong to the package, we recommend that no more than one virtual application package is stored in a given directory as opposed to storing multiple different packages all in the same directory. Failing to do so may cause some of the package data or files be detected incorrectly.

If one or more of the critical package files (SFT and OSD files) cannot be found according to what is listed in the SPRJ file, Application Virtualization Explorer is unable to load the package and you will have to adjust the SPRJ file manually using text editor before being able to open it successfully with Application Virtualization Explorer.

SFT file

In addition to opening up the whole package which is the recommended and normal workflow for handling App-V virtual application packages, Application Virtualization Explorer is able to open up just the SFT file which holds the actual file contents and virtual environment (VE) configuration, such as virtual registry, for the package.



When opening the package in this mode, there will be no possibility for you to make any publishing –related changes to the package. Saving out changed SFT should therefore be limited to scenarios requiring just content viewing or simple in-place modifications of the actual resource contents of the package, i.e. changing things without advancing SFT version number, adding new history entry and so forth.

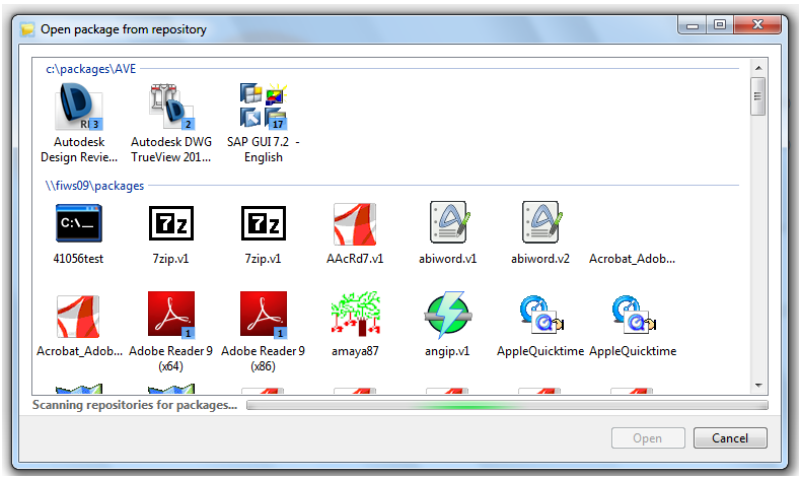
Since none of the original associated package files (like SPRJ or OSD files) will be updated to reflect changed package configuration after saving SFT file only, making changes that change the naming of SFT will cause the whole package to not be correctly associated between various package files.

Accessing recent package list

While working with the packages, Application Virtualization Explorer will store list of most recently opened package into user –specific registry and will display list of those package paths both in the File –menu and on the main program interface when no package is currently open.

This allows quick access to recent package should you need to switch between set of packages very often and simply clicking a package's path will start loading it.

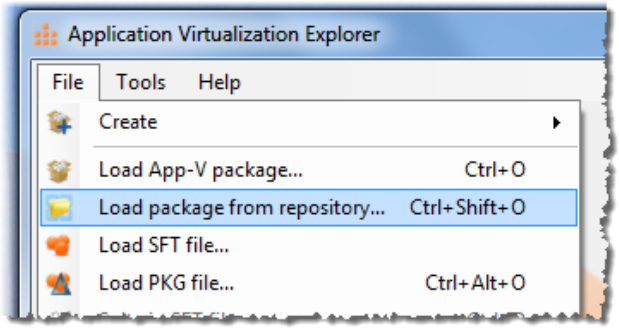
Using package repositories



In addition to specifying manually a folder and file to open, Application Virtualization Explorer supports pre-setting a number of file-system paths which will each represent a logical *package repository*.

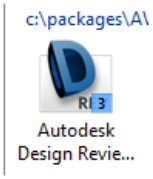
Each package repository is a root of some folder hierarchy – be it App-V Management Server's content – directory or standardized virtual application storage location in the organization's file server – under which multiple packages can be found.

The idea of package repositories is that they will offer a quick and visual way to access a large collection of App-V packages without requiring knowing exact location of each. These locations are then scanned by Application Virtualization Explorer and indexed so that the repository browser can quickly display all available packages once visited for the first time.



Before repositories can be browsed, at least one path to a repository has to be defined. This is done through Repository Settings in the Application Virtualization Explorer's Options –screen. After at least one path is set, repository browser can be started using **Load package from repository** –command in the File – menu.

When repository browser is started, all available packages found from the repository are displayed (and updated in the background if operating over large number of packages and/or behind network path), grouped by the repository.



Application Virtualization Explorer will name the package in the browser list according package name –information found in the OSD file, but placing a mouse cursor over package for a while will show full file-system path to the package in a tooltip for further identification.

Package icons can additionally contain a version number to them (if enabled in the repository options and Manifest XML file is present for package), which allows identification of newest package if repository holds multiple generations of the same individual package.

Double-clicking a package or selecting package and pressing Open will start loading the selected package and repository browser will close.

Performing package suiting / merging

If you already have an App-V package or SFT file open in Application Virtualization Explorer, you can simulate the effects of App-V's Dynamic Suite Composition (DSC) - or dynamic suiting – technology by loading additional SFT files into same view as the primary package.

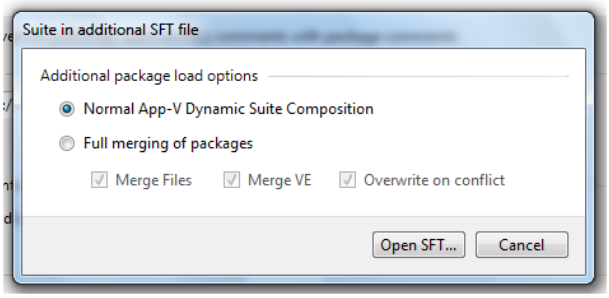
Application Virtualization Explorer is able to perform this suiting operation in two different ways:

- App-V –style dynamic suiting (the default)
- Full merge of data from suited package

Note that regardless of the selected method, only SFT file will be loaded from additional package and any published applications from secondary package will be ignored.

Normal App-V Dynamic Suite Composition

When doing standard App-V -style merging of packages, additional SFT file is loaded side-by-side to the original SFT file so that the package files will appear under one common root (representing root of the virtual drive), and registry data from both (or several) packages will be merged into one single registry.



If package is opened using this mode, resulting package cannot be saved with Save Package functionality as two or more virtual environments are not really merged but rather combined together.

Full merging of packages

Full merge suiting will create one single unified package out of two or more separate virtual environments, which can then be saved out as all additionally loaded data becomes logically part of the primary package. Even in this case, please note that *only* SFT file's contents will be loaded from secondary package(s) and not any of the published applications, virtualized environment variables etc. which may be set in the secondary package. For these, you will have to re-define any missing shortcuts or variables using Application Virtualization Explorer's functionality after merging operation.

With the full merging mode, you have possibility to set following options before additional SFT is loaded to further limit what will be merged in:

- **Merge Files**
  Specifies if any of the files from the additional SFT file will be merged under primary package's directory root.
- **Merge VE**
  Specifies if any of the virtual environment's contents (virtual registry and virtual services) files from the additional SFT file will be merged into primary package's VE.

- **Overwrite on conflict**
  Specifies if data read in from the secondary package will overwrite data in the primary package if there is conflicting entries (files in the same path, registry values in the same key). If this option is not set, original entries won't be overwritten and any such data from additional SFT is discarded.

Working with App-V packages in AVE Express, AVE and AVE Pro

Application Virtualization Explorer supports loading of App-V packages only through the main .APPV file, which loads the full package along with the main package file.

Upon opening the package file(s), Application Virtualization Explorer will decode the contents of all the files into an in-memory representation which means that any and all changes made to the package won't be persisted until the package is saved from Application Virtualization Explorer back to the disk directory.

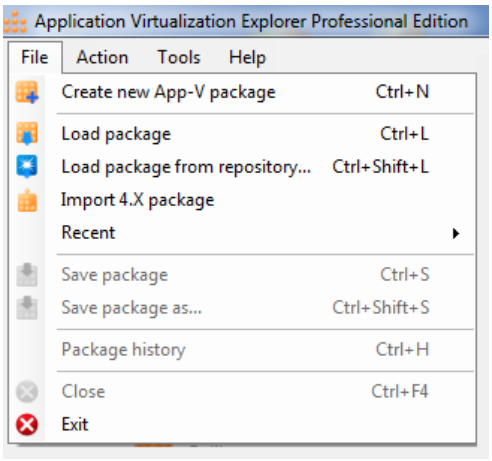
This feature will enable a safe editing of the packages without the issue of accidentally overwriting original package files.

You'll have several different methods available for opening an App-V package using Application Virtualization Explorer:

- Using Load package –functionality in the File –menu
- Double-clicking APPV file in Windows Explorer
- Accessing recent package list on the program's main screen

Loading package from menu

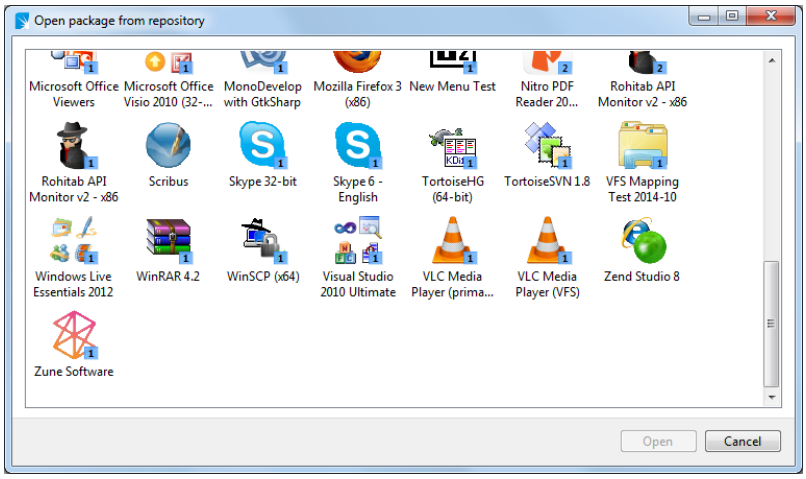
To load package using menu functionality, select the **Load package** command from the File –menu. This will open a standard Windows file browser which allows you to select APPV file that you want Application Virtualization Explorer to open for you.



Both the binary contents in APPV file and all associated dynamic configuration files will be loaded in one go; Application Virtualization Explorer will detect dynamic configuration files in the same directory as the APPV file itself and load them into the package configuration as an alternate configuration contexts.

Although the detection mechanism tries to exclude any files that do not actually belong to the package, we recommend that no more than one virtual application package is stored in a given directory as opposed to storing multiple different packages all in the same directory. Failing to do so may cause some of the package data or files be detected incorrectly.

Using package repositories

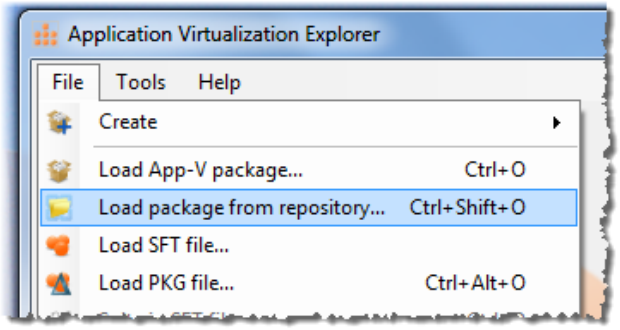


In addition to specifying manually a folder and file to open, Application Virtualization Explorer supports pre-setting a number of file-system paths which will each represent a logical *package repository*.

Each package repository is a root of some folder hierarchy, for example a location in the organization's file-server under which multiple packages can be found. With App-V 5.0 this could be the same repository path as used by the Management Server if published through file-share.

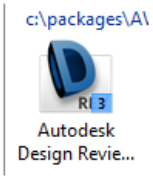
The idea of package repositories is that they will offer a quick and visual way to access a large collection of App-V packages without requiring knowing exact location of each. These locations are then scanned by Application Virtualization Explorer and indexed so that the repository browser can quickly display all available packages once visited for the first time.

Before repositories can be browsed, at least one path to a repository has to be defined. This is done through Repository Settings in the Application Virtualization Explorer's Options –screen.



After setting at least one path, repository browser can be started using **Load package from repository** – command in the File –menu.

When repository browser is started, all available packages found from the repository are displayed (and updated in the background if operating over large number of packages and/or behind network path), grouped by the repository.



Application Virtualization Explorer will name the package in the browser list according package name –information found in the APPV file, and by placing a mouse cursor over package for a while you can get full file-system path to the package in a tooltip for further identification.

Package icons additionally contain a package's version number, which allows identification of newest package if repository holds multiple generations of the same individual package.

Double-clicking a package or selecting package and pressing Open will start loading the selected package and repository browser will close.

Importing 4.X packages

Application Virtualization Explorer Professional edition has the additional capability of importing/converting existing App-V 4.X (or older, provided that the .SPRJ file is available) into App-V 5.0 format.

This allow you to load your existing legacy App-V packages and save them into a new format directly inside AVE, without needing to use package conversion functionality in the Sequencer. As an additional plus, AVE tries to convert package scripting and lift some of the application extensions (but not all) directly from the virtual registry content within the original package.

Importing legacy App-V packages using AVE can be done through **Import 4.X package** –command in the File –menu. For conversion, AVE will need the SPRJ file which has valid references to the all OSD files and SFT file making up the 4.X package.

Please note that it is possible that some legacy App-V packages cannot be mechanically converted by AVE and you should always test the converted package thoroughly after conversion! Especially scripts and integration points may need to be rewritten, and some legacy technologies that worked with App-V 4.X may behave differently with App-V 5.X virtualization.

Accessing recent package list

While working with the packages, Application Virtualization Explorer will store list of most recently opened package into user –specific registry and will display list of those package paths both in the File –menu and on the main program interface when no package is currently open.

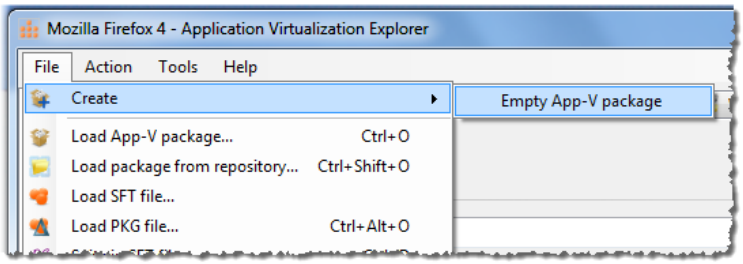
This allows quick access to recent package should you need to switch between set of packages very often and simply clicking a package's path will start loading it.

Creating new App-V packages in AVE Pro Classic

In addition to being able to open any existing App-V package, Application Virtualization Explorer supports creation of new App-V packages directly within Application Virtualization Explorer and without needing to use Sequencer or other tools for the purpose.

Currently the support for new package creation is limited to initially empty packages only (i.e. not performing any installation change monitoring or snapshotting of the system), which will need to be manually constructed further in Application Virtualization Explorer by using content importation functionality such as importing files and directories and importing registry files.

To start new package creation, select **Create -> Empty App-V package** from the File –menu.



Before a new package can be created, you will have to supply few relevant parameters for the package initialization. These are:

- **Package name**
  Unique name for the App-V package.

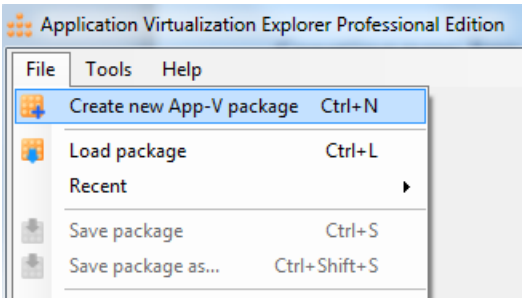
- **Root directory name**
  Name of the root directory for an internal directory structure and one that will be mounted to the Q: drive on a client. By default the same name as the package name will be used for root directory name, but you can customize if you want. If the root directory is longer than 8+3 naming convention, randomly assigned short name will automatically be generated.

- **Package bitness**
  If the resulting package will be 32-bit (default) or 64-bit one.

Creating new App-V packages in AVE Express, AVE and AVE Pro

In addition to being able to open any existing App-V package, Application Virtualization Explorer supports creation of new App-V packages directly within Application Virtualization Explorer and without needing to use Sequencer or other tools for the purpose.

Currently the support for new package creation is limited to initially empty packages only (i.e. not performing any installation change monitoring or snapshotting of the system), which will need to be manually constructed further in Application Virtualization Explorer by using content importation functionality such as importing files and directories and importing registry files.



To start new package creation, select **Create new App-V package** from the File –menu.

Before a new package can be created, you will have to supply few relevant parameters for the package initialization. These are:

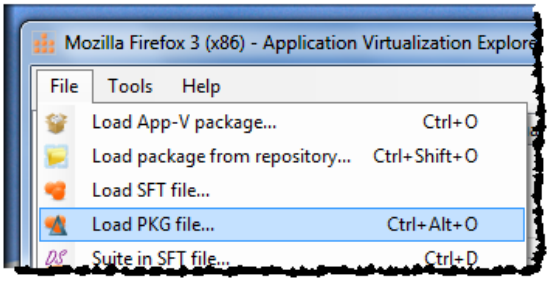
- **Package name**
  Unique name for the App-V package.

- **PVAD path**
  Path of the Primary Virtual Application Directory for the package. This is the path underwhich the package root will be virtualized to on a client, and Microsoft guidelines recommend using the intended installation path of an application for the PVAD (e.g. “C:\Program Files\MyApp”). However, if you do not wish to use the default path that Application Virtualization Explorer suggests, you can click “Generate Random PVAD under system root” link and get a random directory path as PVAD.

- **Package bitness**
  If the resulting package will be 32-bit (default) or 64-bit one.

App-V Client PKG files (AVE Pro Classic only)

Application Virtualization Explorer Professional was the first 3rd party App-V –related tool in the market being able to open up and view PKG files used by the App-V Client to store cached user and machine settings per package basis. To this day, Application Virtualization Explorer is still the most comprehensive tool to examine and export data from the setting files.



Please note that Application Virtualization Explorer does not allow saving of modified PKG files, only viewing them!

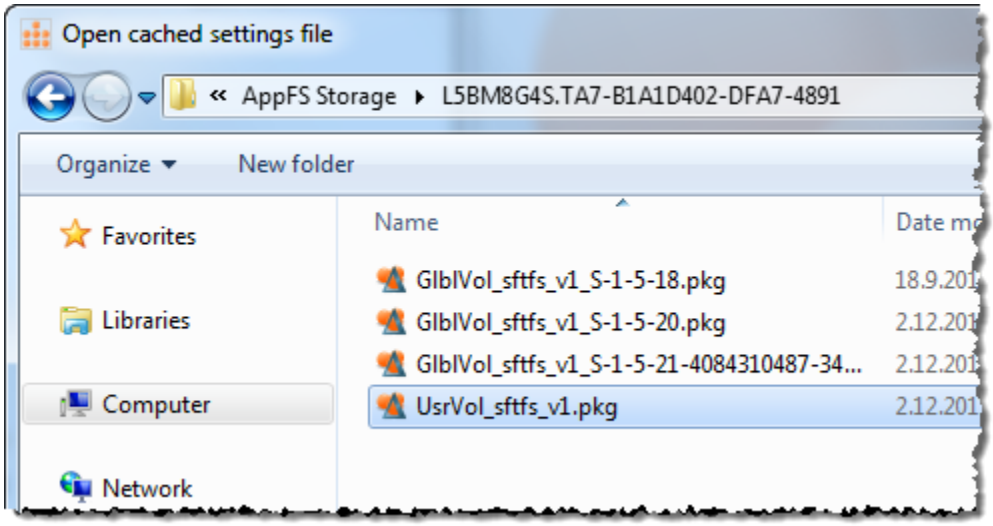
PKG files purpose is to contain changes to package's virtual environment (virtualized files and virtual registry) during App-V package execution on the client and the exact physical location depends on both what type the modified file is and which process effected the change.

Current generation of App-V Clients stores these cached settings in two distinct places in the system, PKGs for each package in separate subdirectory:

- In user's cache directory
- In App-V Client cache directory

In the user's App-V cache directory – by default under roaming application data (**C:\User*account*\AppData\Roaming\SoftGrid Client** in Windows Vista and newer operating systems) – the PKG file is called **UsrVol_sftfs_v1.pkg.** This files stores changes both to User –type files and all changes to a virtual registry done by the processes running in user's context (i.e. applications launched by the user using App-V Client), visible only for that user account. Normally this file is the first of two most interesting ones for examination.

In the App-V Client cache directory, **C:\ProgramData\Microsoft\Application Virtualization Client\SoftGrid Client\AppFS Storage** in Windows 7 and newer operating systems holds the rest of the PKG files applied per each App-V package:



These PKG files are as follows:

- UsrVol_sftfs_v1.pkg
  Changes done to User –type files by select few system processes. These changes are visible only to those selected processes and due to very limited number scenarios involving said processes making changes to App-V package's VE, this file is most likely empty.

- GlblVol_sftfs_v1_S-1-5-20.pkg
  Changes done to Application –type files by system –level processes. These changes are visible only to those system processes, such as virtual services running in the package's context.

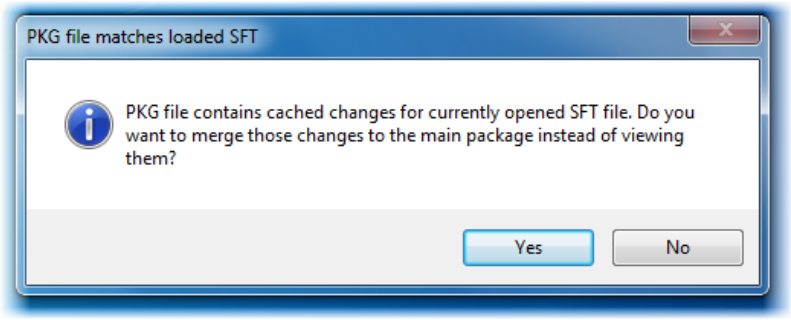
  This file also holds virtual environment configuration for system processes.

- GlblVol_sftfs_v1_*SID*.pkg
  Changes done to Application –type files by user's normal processes. This file is second of two most interesting ones as it likely contains some modifications to file contents.

  Note that since this file hold's reference to SID for user account, there could be multiple of these PKG files per package in Remote Desktop Session Host – or Terminal Server – machine.

Applying PKG to a matching package

In addition to be able to load PKG files just for standalone viewing, Application Virtualization Explorer contains feature to merge changes in PKG file to SFT file it is based on.



You can activate this feature by first loading App-V package or SFT file you intend to merge changes to, and then issuing Load PKG file functionality and selecting any PKG file which is based on currently loaded package. Application Virtualization Explorer will detect (from unique package GUID that all App-V packages carry) that PKG being loaded matches already loaded App-V package and offers to perform merge operation:

By answering Yes, Application Virtualization Explorer will merge all changed files and registry entries from the PKG file to open App-V package, resulting fully combined contents.

By answering No, Application Virtualization Explorer will close underlying package and open PKG file for normal stand-alone viewing.

Note: PKG files can contain lot of extraneous files and registry entries for App-V Client (such as settings.cp files) and Windows itself. It is not advisable to save resulting merged package as-is without

any manual cleanup, as this may severely interfere with client's operation when running package merged this way!

Extra icons for PKG files

When PKG is open in Application Virtualization Explorer, Files –view may display few extra icons from what can be seen when normal App-V package is open in the same view.

These extra folder and file icons are:

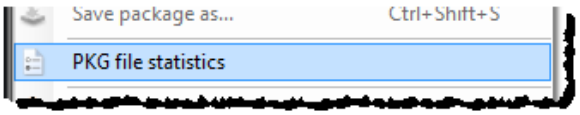
A file reference inside PKG file to original file in associated SFT file. These files are not actually stored in PKG files, but internal references to unmodified files. Application Virtualization Explorer will not by default show these files, but they can be seen by selecting "**Show references to unmodified files from SFT"** from the View –menu

A file that has been created anew in the virtual environment and then subsequently deleted.

A file that is in original SFT file but has then been deleted inside virtual environment.

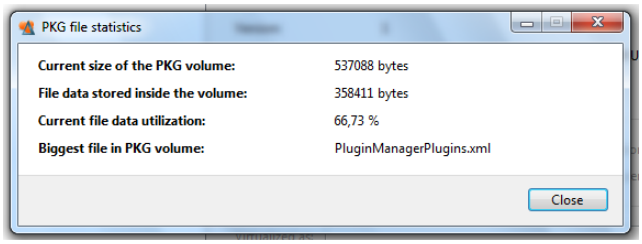
Viewing PKG statistics

When PKG file is loaded and open in Application Virtualization Explorer, basic statistics about it can be viewed by using **PKG file statistics** –command in the File menu.



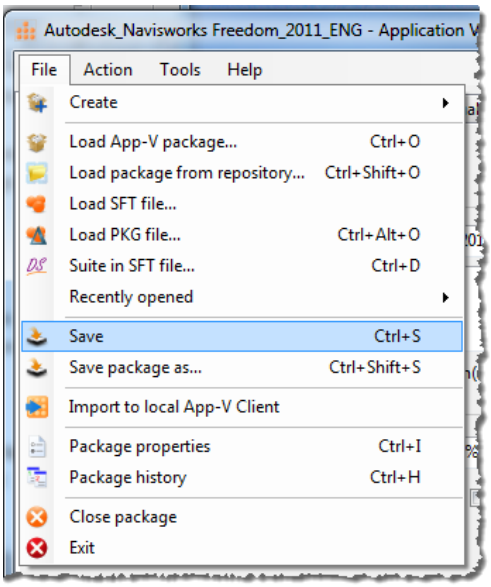
This will open a statistics –screen, having following information displayed about PKG file's contents:

- Current size of the PKG file
- Size of the data currently stored inside the PKG file
- Current data utilization of the whole file
- Largest file stored inside the PKG file



Saving packages in AVE Pro Classic

After making any changes to the package that you want to persist, package needs to be saved out (re-encoded) back to the disk. As a safety mechanism, Application Virtualization Explorer will not make any changes to original files during package editing - only to a memory-based copy of the data - so you are free to change anything inside the package and be able to revert back to clean state if you do not explicitly save changes back.

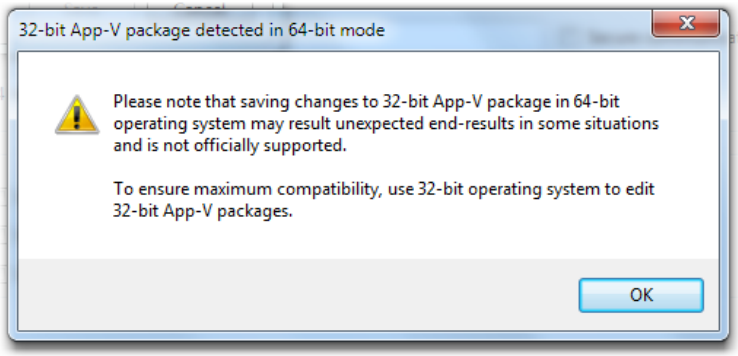


Under those circumstances where it is allowed or desirable to save over an existing package - the original source opened to Application Virtualization Explorer - a simple **Save** –command (or Ctrl+S using keyboard shortcut) can be invoked from the File –menu. This command will save the whole package back to disk using same settings for SFT file generation as was the original source (e.g. compression, SFT file-format etc.) without prompting any additional information.

Depending on the use-case, before saving any packages you might also want to change saving –related general options in the Application Virtualization Explorer's Options –screen first, which will control preservation of the source SFT file after package has been saved successfully out, among others.

Alternative choice is to use **Save package as** –command (or Ctrl+Shift+S using keyboard shortcut) in the File –menu, which will first prompt you for a package –related save options before doing anything, described below.

To prevent making accidental choice of saving 32-bit originated packages on Application Virtualization Explorer running on 64-bit operating system or saving 64-bit originated packages on Application Virtualization Explorer running on 32-bit operating system, opening save options screen will prompt for warning if bitness mismatch is detected.

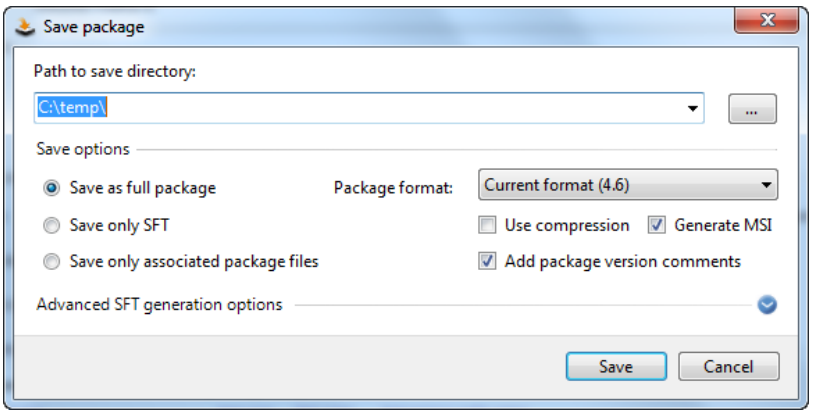


Re-saving package in incorrect bitness may lead to a non-working virtual environment configuration and should be avoided in all production type of scenarios!

Save package options

With save package options, you can control resulting save more detailed instead of just saving back to the originating directory with existing (i.e. source originating) parameters.

More advanced SFT -generation settings are by default hidden and needs to be expanded by clicking blue downward pointing arrow on the right-hand side corner.



Path to save directory

This is the directory where to Application Virtualization Explorer will write modified package files to.

By selecting a directory from which the package was originally opened from, existing package metadata files (OSD(s), SPRJ etc.) will be overwritten and new SFT file is generated, if package save is successful. In order to safekeep your data, Application Virtualization Explorer will not overwrite any of the original files during the progress of the save operation in case of errors occuring at any phase. Only after *all* files has been successfully written will be the original ones replaced, expect for SFT file for which the save behavior can be set in Application Virtualization Explorer's general options.

Save options

Basic options related to save operation.

Package save mode

You can select one of three different modes for the saving:

- **Save as full package**
  All of the package files will be created and written to the disk. This includes both core files (SFT, OSD, ICO and SPRJ) as well as additional files (manifest and MSI) if newer package format (4.5 and up) is selected.

  If package was opened directly from SFT file (and not SPRJ file), this mode is not selectable as Application Virtualization Explorer won't have enough information available to write other files besides modified SFT file.
- **Save only SFT**
  Only SFT file will be created and written to the disk.

- **Save only associated package files**
  All of the associated package files will be created and written to the disk. This includes all files besides SFT file, which makes it possible to edit publishing data only with Application Virtualization Explorer and re-use existing SFT from the source, which will be copied to the destination folder as-is.

  If package was opened directly from SFT file (and not SPRJ file), this mode is not selectable as Application Virtualization Explorer won't have enough information available to write other files besides modified SFT file.

Package format

Specified file format to use for outputted package.

Note that if down converting newer package to an older format, some of the functionalities inside the package dependent on newer format and capabilities can be lost. It is not advisable to downgrade packages, if at all avoidable.

Use compression

If selected, data in the outputted SFT file will be compressed.

Application Virtualization Explorer uses ZLIB compression algorithm for compression in all cases, older BZIP2 algorithm is not supported for saving although Application Virtualization Explorer can still open old packages compressed with that algorithm.

Generate MSI

If selected, package "wrapper" MSI will be generated along with other files. This MSI file will allow importing of App-V package to a standalone –configured client.

This option is not available for pre-4.5 format packages.

Add package version comments

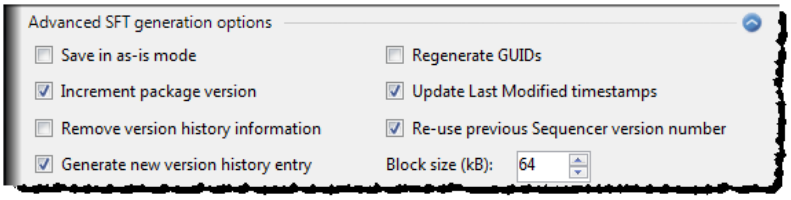
If selected, Application Virtualization Explorer will prompt free-form comment related to this package save and stores it inside the virtual registry as part of the extended history (if enabled). Application Virtualization Explorer's extended history information can be viewed in the package history –screen if history information data is recorded and present inside the package.

Entering version comment is highly advisable as it can create clear change documentation over multiple versions of the same package, something that native App-V package history does not record.

Note that if the overall saving of extended history is turned off in the Application Virtualization Explorer's general options, package version comments cannot be entered in the save screen.

Advanced SFT options

In addition to basic settings, Application Virtualization Explorer supports additional encoding parameters for the SFT file.



Having advanced settings with default states will result in SFT file that is updated in the normal manner. Advanced parameters should not be modified unless having specific reason to do so and clear understanding of the consequences changing any of them:

- **Save in as-is mode**
  Disables all other advanced settings and will result in SFT file which is identical to original SFT other than the logical contents that has been updated.

  This option could be used for situations requiring edits to SFT file which cannot be allowed to gain traditional version increments.

  Enabling as-is options is **not** recommended when saving out packages that supersedes previous versions in any Active Upgrade -related scenario (e.g. using App-V Management Server or Streaming Server).
- **Increment package version**
  Increments internal package version number by one, if selected.
- **Remove version history information**
  Clears all past history entries from the package, if selected.
- **Generate new version history entry**
  Creates new "sequencing" history entry for the latest save, if selected.
- **Regenerate GUIDs**
  Re-creates internal package IDs, if selected. Please note that this will create completely new package from App-V's perspective and if using this option, it is strongly advisable to rename internal root directory as well from Files –screen before saving the package.
- **Update Last Modified timestamps**
  Updates package modification timestamps, if selected.
- **Re-use previous Sequencer version number**
  If set (default) and when saving 4.0+ packages, Application Virtualization Explorer will use the exact same Sequencer version number as is to be found in previous history entry for the package. If not set, Application Virtualization Explorer will default to minimum Sequencer version number appropriate to that App-V package version.

Block size

Allows customization of size of the data blocks that files inside the package will be split to and encoded with.

Note that is not advisable to change block size after package has been originally created, as this will render saved SFT file un-updateable to any clients that has earlier version of the package data in cache unless that cached data is first unloaded.

Saving packages in AVE and AVE Pro

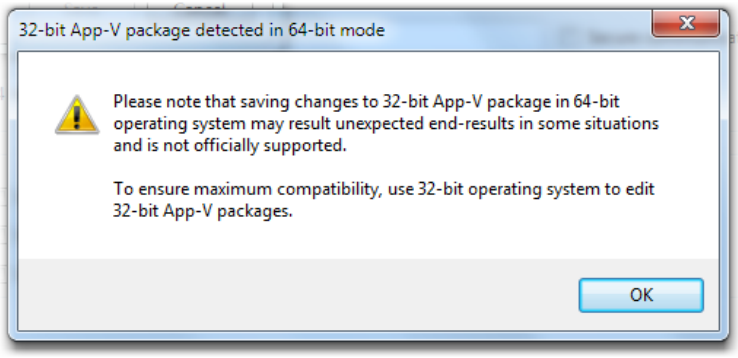
After making any changes to the package that you want to persist, package needs to be saved out to the disk. As a safety mechanism, Application Virtualization Explorer will not make any changes to original files during package editing - only to a memory-based copy of the data - so you are free to change anything inside the package and be able to revert back to clean state if you do not explicitly save changes back.

Under those circumstances where it is allowed or desirable to save over an existing package - the original source opened to Application Virtualization Explorer - a simple **Save** –command (or Ctrl+S using keyboard shortcut) can be invoked from the File –menu. This command will save the whole package back to disk, to the same directory and using same settings for package generation as was used in the original source without prompting any additional information.

Depending on the use-case, you might also want to change saving –related general settings in the Application Virtualization Explorer's Options –screen first, which will control preservation of the source package files and backup after package has been saved successfully out.

If you want to select where to save the package, or branch an existing package to a completely new one, you can use **Save package as** –command (or Ctrl+Shift+S using keyboard shortcut) in the File –menu, which will first prompt you for a package –related save options before doing anything; these are described below.

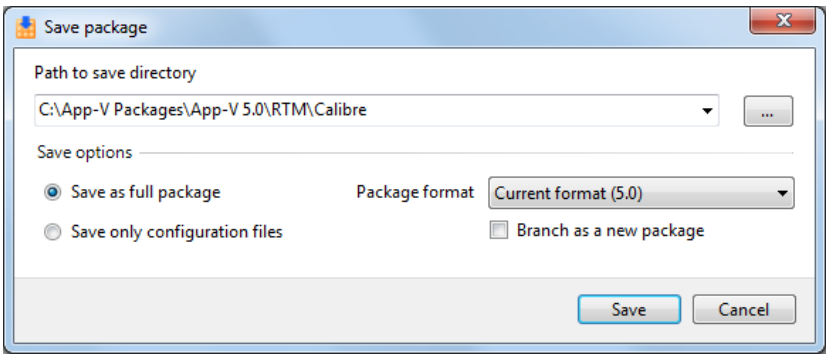
To prevent making accidental choice of saving 32-bit originated packages on Application Virtualization Explorer running on 64-bit operating system or saving 64-bit originated packages on Application Virtualization Explorer running on 32-bit operating system, opening save options screen will prompt for warning if bitness mismatch is detected.



Re-saving package in incorrect bitness may lead to a non-working virtual environment configuration and should be avoided in all production type of scenarios!

Save package options

With save package options, you can control the operation instead of just saving back to the originating directory with existing package's parameters.



Path to save directory

This is the directory where to Application Virtualization Explorer will write modified package files to.

Save options

Basic options related to save operation.

Package save mode

You can select one of three different modes for the saving:

- **Save as full package**
  All of the package files will be created and written to the disk. This includes both core file (APPV) as well as additional files (dynamic configuration files).

- **Save only configuration files**
  All of the associated package dynamic configuration files (both User and Deployment) only will be created and written to the disk. This option skips saving the APPV and MSI files.

Package format

Specified file format to use for outputted package.

Note that if down converting newer package to an older format, some of the functionalities inside the package dependent on newer format and capabilities can be lost. It is not advisable to downgrade packages, if at all avoidable.

Branch as a new package

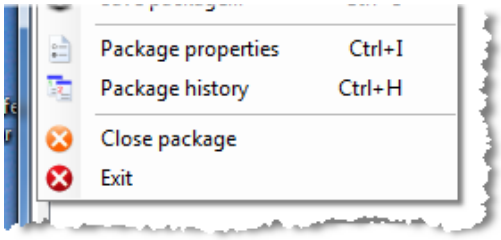
If this option is selected, during the save operation all internal package IDs are re-created and package is branches as new and independent [from the source package] App-V 5 package.

Please note that this will create completely new package from App-V's perspective, allowing old and new package to co-exist side-by-side, and if using this option it is strongly advisable to rename the package itself and/or the virtual applications before saving the package.

Other menu options in AVE Pro Classic

File menu

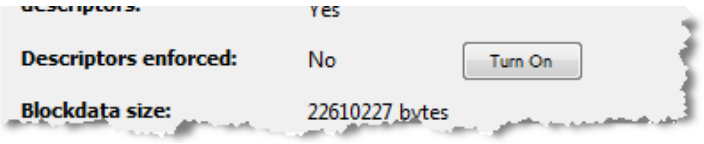
Besides enabling commands for creating a new package, opening up full packages, individual SFT files, additional SFT files (suiting) and PKG files as well as saving out edited packages, File –menu has the following functionalities:



Package properties

Displays statistics and properties of currently opened SFT file.

From the properties screen, you can turn on and off file-system ACL enforcement for App-V 4.5 and newer packages carrying security descriptors for files stored in it.

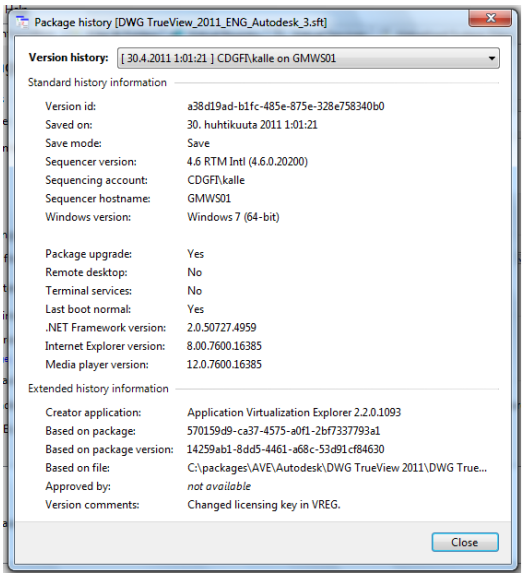


Package history

Displays save (or sequencing) history for the package.

New history entry is automatically generated for all App-V packages in 4.0 or newer format, one per each save. You can use history entries to see in which environment the package was edited/updated, important factor to consider when later on doing updates to a package so that too big environmental changes (like upgrading package in older OS than it was previously upgraded in) won't affect the upgrade.

Package history is not supported for pre SoftGrid 4.0 packages.



Extended history

In addition to standard App-V provided history information, Application Virtualization Explorer can optionally save extended package history which is physically stored using package's virtual registry. Application Virtualization Explorer will record actual name and version of the application that saved the package to overcome Sequencer version –fields limitation to accommodate 3rd party applications, explicit

package lineage (which is normally lost in package branching), source file information (to track actual location used for getting the package) and version comments. Approval field is reserved and not currently used by Application Virtualization Explorer.

Since extended history information is implemented through virtual registry, other manufacturers can add support for Gridmetric's extended App-V package history by adding custom values into appropriate virtual registry branches (REGISTRY\MACHINE\Software\{Extended History}\VID*Package version number*) and/or using Gridmetric–provided values that Application Virtualization Explorer will recognize.

Import to local App-V Client

If you have **Local import of package** –plugin installed, imports currently open App-V package (the version that can be found on the disk) to the locally installed App-V Client (4.5 and newer only).

Close package

Closes any open App-V package, suite of packages or PKG file.

Exit

Exists and closes Application Virtualization Explorer.

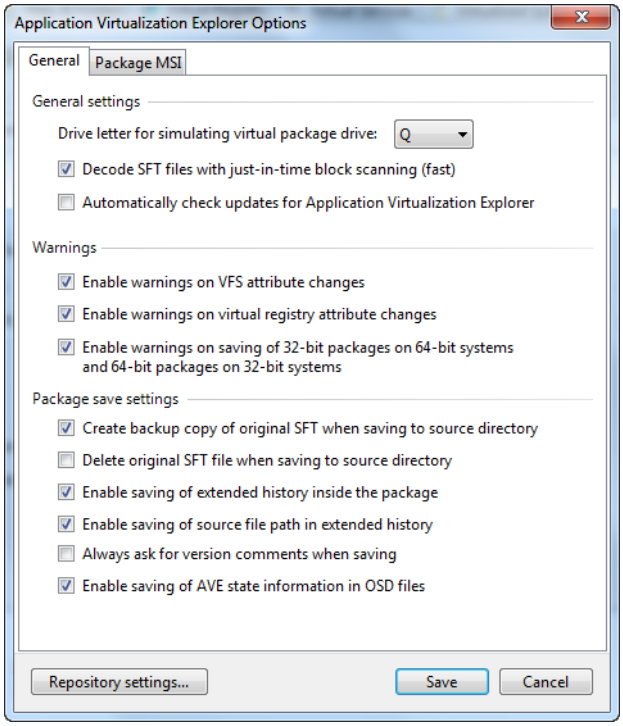
Tools menu

Tools –menu has the following functionalities:

Options

Displays options screen for Application Virtualization Explorer.

You can access both general Application Virtualization Explorer options as well package MSI –related options from the tabs in the options –screen, and repository –related options through **Repository settings** –button.



Drive letter for simulating virtual package drive

If the local machine does not have App-V Client installed, you have to specify virtual drive letter that Application Virtualization Explorer will use internally when doing encoding and decoding of text string in the virtual registry and virtual file-system.

Selecting drive letter has no effect outside Application Virtualization Explorer itself and thus any free drive letter suffices as long as you remember to refer to virtual drive using that letter.

If Application Virtualization Explorer detects that the local machine has App-V Client install, it will be automatically queried and virtual letter in App-V is also used to signify virtual drive letter inside Application Virtualization Explorer.

Decode SFT files with just-in-time block scanning

If set, Application Virtualization Explorer will decode SFT files without scanning through every single DATA block contained inside it.

Reading in SFT files without DATA block scan (default) makes initial SFT opening a fast process, especially over network shares. If data blocks are not scanned during initial load, each access to DATA block later on will once incur slight delay as header needs to be scanned at that time. This is most evident when files are selected in Files & Folders –view and block information needs to be displayed, and when saving a package to disk as Application Virtualization Explorer needs to access all data blocks from the source SFT file (latest) at that point.

Automatically check updates for Application Virtualization Explorer

If set, Application Virtualization Explorer will check for product updates once every two weeks and inform if a new version is available for downloading.

For update check to work, machine has to have active Internet connection so that a HTTP request can be made to Gridmetric's servers. *Note that Application Virtualization Explorer will not update itself automatically under any circumstances, only a version check is performed.*

Create backup copy of original SFT when saving to source directory

If set, Application Virtualization Explorer will rename source SFT file after package has been saved successfully back to the originating directory instead of [default] deletion of the source SFT file. If save operation is not performed back to the source directory but rather an alternative directory path, this setting has no effect. Likewise, if the save operation for some reason does not succeed, source SFT is kept intact.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Delete original SFT file when saving to source directory

If set, Application Virtualization Explorer will delete source SFT file after package has been saved successfully back to the originating directory, unless SFT backup option is selected instead. If save operation is not performed back to the source directory but rather an alternative directory path, this setting has no effect. Likewise, if the save operation for some reason does not succeed, source SFT is kept intact.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Enabled saving of extended history inside the package

If set, Application Virtualization Explorer will save extended package version history inside the package's virtual registry, providing more information about particular saved version in the package's lifespan.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Enable saving of source file path in extended history

If set, Application Virtualization Explorer will store path to a source SFT file as part of the extended history information. You may want to consider disabling this option if not wanting to reveal sensitive environmental factors for future administrators or upgraders for the package.

If extended history saving overall is disabled, this setting has no effect.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Always ask for version comments when saving

If set, Application Virtualization Explorer will always prompt for version specific comments, forming part of the extended history for particular saved version.

These comments are prompted during normal saving and if using save options dialog, appropriate option is automatically selected but can be turned off at that stage. Version comments are good way to document what change went into particular edit with free-form text.

If extended history saving overall is disabled, this setting has no effect.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Enable saving of Application Virtualization Explorer state information in OSD files

If set, Application Virtualization Explorer is allowed to save custom OSD element(s) to hold internal state information for future edits. Currently only feature in Application Virtualization Explorer to support this data is extra information stored for Dynamic Suiting (DSC) links. If the option is not set, Application Virtualization Explorer will not save any extraneous information in the OSD but some Application Virtualization Explorer –specific information about the package may be lost.

To conform to an official OSD schema, Application Virtualization Explorer does not augment pre-existing OSD elements with additional attributes but rather generate completely new – and private – XML elements into OSD to hold state information. These custom elements are preserved and carried on by the Sequencer if package happens to be edited inside it.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

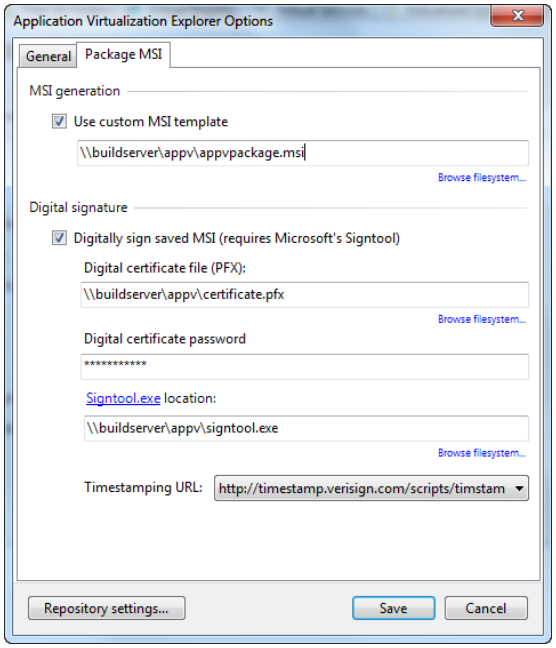
Warnings

Warnings for operations in Files view or Virtual Registry view having possible unintended side-effects can be enabled (default) or disabled at will. Also a warning for bitness mismatch on save dialog can be disabled or enabled (default).

Use custom MSI template

Specifies a path to the custom MSI template, which will be used when Application Virtualization Explorer saves out App-V package MSI file instead of the default Microsoft Sequencer –compatible one.

If you have need to customize the package MSI file (one that imports the App-V package into client) – add a branding or remove/change functionality – you can set Application Virtualization Explorer to use any customized MSI template as long as it derives from the default one because Application Virtualization Explorer makes assumptions on specific properties and files stored in it during Save operation.



After enabling the option, you will need to specify a full path to a MSI template file.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Digitally sign saved MSI (requires Microsoft's Signtool)

Enables support for post-save digital signing of resulting package MSI file. The signing is done by external Microsoft's tool, called Signtool, freely available from Microsoft's website or Windows SDK.

After enabling the option, will need to specify following parameters in order to signing to be successful:

- **Digital certificate file (PFX)**
  A full path to the digital certificate file, in PFX format, that will be used for signing. At this time certificates in the local Windows certificate store cannot be selected, but the certificate has to exist as a file.

- **Digital certificate password**
  A private key password for selected digital certificate file.

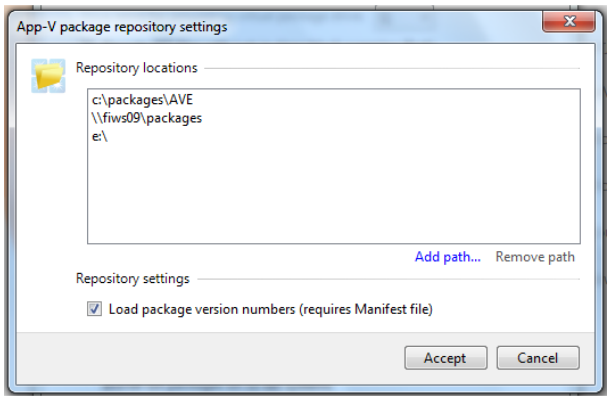
- **Signtool.exe location**
  A full path to the signtool.exe tool.

- **Timestamping URL**
  Timestamping server to use to timestamp digital signature. Either Verisign's or Comodo's server can be used, regardless of the issuer of the digital certificate itself.

When package MSI saving has been done otherwise, Application Virtualization Explorer will try to invoke the Signtool utility from the specified path using the specified parameters.

If the digital signing fails however (due to invalid path to the tool itself or the certificate file, or for any other reasons), package save in whole does not end in error. Please check the resulting MSI file that the Digital Signature is applied by taking properties of the MSI file in Windows Explorer, and verifying that tab called *Digital Signatures* exists and it contains a valid signature.

Repository –related settings



Displays settings related to Application Virtualization Explorer package repositories.

Each repository is a directory path (local or network) pointing to a root of the directory hierarchy containing multiple App-V packages. These repositories can then be used as easy way to access particular App-V package without browsing back and forth in the directories like when opening package through normal Application Virtualization Explorer load interface.

Repository locations

Contains file paths for each separate repository location. You can use **Add path** and **Remove path** -buttons to add or remove locations at will.

When removing a repository path, Application Virtualization Explorer will prompt for deletion of cached data related to that repository. This data contains list of packages last seen in the repository as well as cached icon data, which together will allow Application Virtualization Explorer to quickly pre-populate package list in the repository browser and then update in the background. If you delete the cached data and then later on add back the same repository path, Application Virtualization Explorer has to perform package detection from the scratch – possibly time-consuming operation over network share – instead of making use of previously cached information.

Load package version numbers

If set, Application Virtualization Explorer will detect version number of the package in each founded package location inside repository and overlays package icon with this version number. This has the advantage of quickly displaying which out of possibly many revisions of the same package is the most recent one.

Note that version number is read from the package manifest XML file instead of accessing SFT itself so version number cannot be shown for older SoftGrid packages having no manifest with it.

Copy DSC snippet

If you have **DSC codebase snippet tool** –plugin installed, allows copying of Dynamic Suite Composition – reference (CODEBASE element) of the currently open package. This can then be pasted manually into other App-V packages' OSD files.

Edit object exclusions

If you have **Editor for object exclusion** –plugin installed, allows editing of virtual object exclusions of the currently opened package.

Troubleshooting

If any of the troubleshooting –related plugins have been installed (such as **Viewer for App-V variables** or **Viewer for changed and added files**), they can be found and accessed from the Troubleshooting –submenu.

Help menu

Help –menu has the following functionalities:

Enter license information

Used to supply valid licensee name and license key before Application Virtualization Explorer can be used.

If Application Virtualization Explorer already has valid license loaded, this menu entry is not visible.

Application Virtualization Explorer User guide

Opens this documentation.

About

Displays Application Virtualization Explorer version and copyright notices.

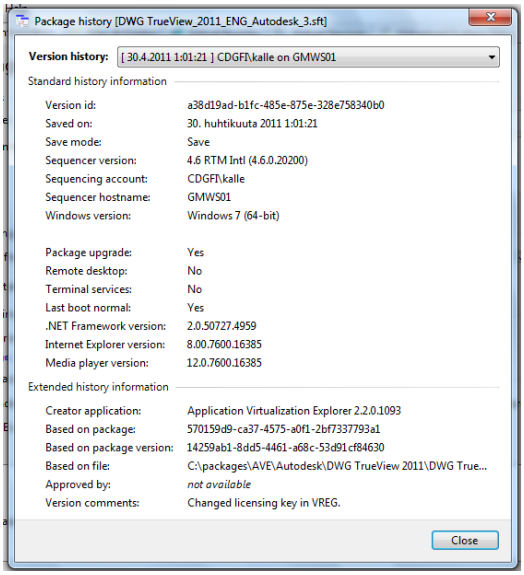
Other menu options in AVE Express, AVE and AVE Pro

File menu

Besides enabling commands for creating new package, opening up existing packages as well as saving out edited packages (AVE and AVE Pro only), File –menu has the following other functionalities:

Package history

Displays save (or sequencing) history for the package.



New history entry is automatically generated for all App-V packages, one per each save. You can use history entries to see in which environment the package was edited/updated, important factor to consider when later on doing updates to a package so that too big environmental changes (like upgrading package in older OS than it was previously upgraded in) won't affect the upgrade.

Note that AVE will identify itself as "Sequencer" packaging engine in history as the App-V Sequencer is enable to show any history entries when custom engine name is used.

Close package

Closes open App-V package.

Exit

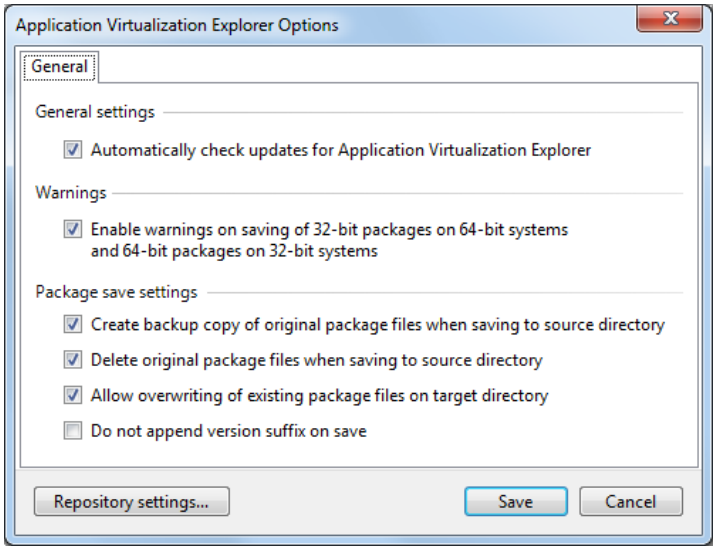
Exists and closes Application Virtualization Explorer.

Tools menu

Tools –menu has the following functionalities:

Options

Displays options screen for Application Virtualization Explorer.



Automatically check updates for Application Virtualization Explorer

If set, Application Virtualization Explorer will check for product updates once every two weeks and inform if a new version is available for downloading.

For update check to work, machine has to have active Internet connection so that a HTTP request can be made to Gridmetric's servers.

Note that Application Virtualization Explorer will not update itself automatically under any circumstances, only a version check is performed.

Create backup copy of original package files when saving to source directory

If set, Application Virtualization Explorer will copy all package files to a subdirectory under source directory, named after the package's name and version, upon successful save. If save operation is not performed back to the source directory but rather an alternative directory path, this setting has no effect.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Delete original package files when saving to source directory

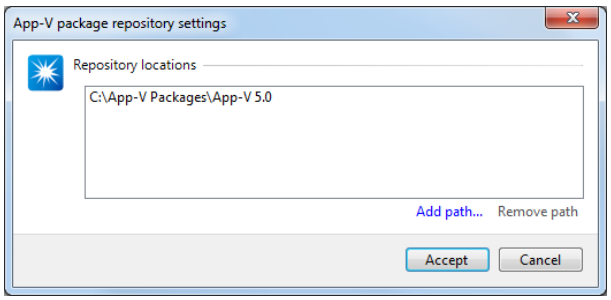
If set, Application Virtualization Explorer will delete source package files after package has been saved successfully back to the originating directory. If save operation is not performed back to the source directory but rather an alternative directory path, this setting has no effect. Likewise, if the save operation for some reason does not succeed, source package files are kept intact.

This setting can be overridden using Application Virtualization Explorer's Group Policy Administrative Template.

Warnings

Warning for bitness mismatch on save dialog can be disabled or enabled (default).

Repository –related settings



Displays settings related to Application Virtualization Explorer package repositories.

Each repository is a directory path (local or network) pointing to a root of the directory hierarchy containing multiple App-V packages.

These repositories can then be used as easy way to access particular App-V package without browsing back and forth in the directories like when opening package through normal Application Virtualization Explorer load interface.

Repository locations

Contains file paths for each separate repository location. You can use **Add path** and **Remove path** -buttons to add or remove locations at will.

When removing a repository path, Application Virtualization Explorer will prompt for deletion of cached data related to that repository. This data contains list of packages last seen in the repository as well as cached icon data, which together will allow Application Virtualization Explorer to quickly pre-populate package list in the repository browser and then update in the background. If you delete the cached data and then later on add back the same repository path, Application Virtualization Explorer has to perform package detection from the scratch – possibly time-consuming operation over network share – instead of making use of previously cached information.

Help menu

Help –menu has the following functionalities:

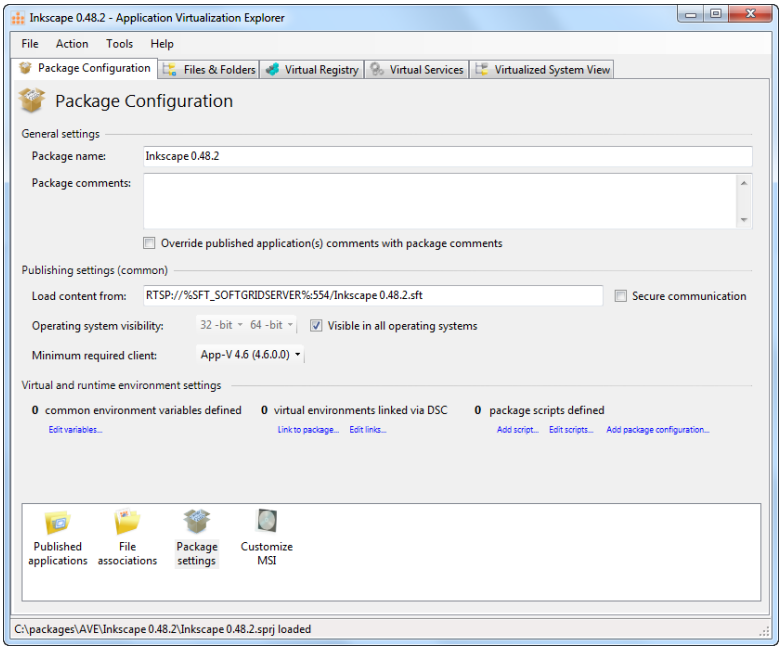
About

Displays Application Virtualization Explorer version and copyright notices.

Package screens in AVE Pro Classic

Package configuration view

When you open up a full App-V package using SPRJ file, Package Configuration screen will show most common parameters related to package's overall configuration.



On the bottom of the screen, you can navigate between package settings and MSI customization (if the source package has MSI file associated with it) screens, or switch into published application and file association views using appropriate folder links (**Published applications** and **File associations**).

By default, package configuration view displays package settings -screen.

Package Configuration Action -menu options

When in Package Configuration -screens, the Action menu contains following options in addition to possible selection specific options:

- **Rename core package files**
  Allows renaming of package files (SPRJ, Manifest and SFT files). If the names are customized, Application Virtualization Explorer will use these names during package save. This functionality is generally needed when doing package branching.

- **Change Sequencer defaults**
  Sets or clears settings that relate to App-V Sequencer's options for the opened package. These settings do not affect client operations or Application Virtualization Explorer, but rather the situation if the package is later on updated using the Sequencer.

Package settings

These are the overall settings you can set per package basis.

Changing any of the settings in this screen will not only affect default OSD data (which is the template for a newly published applications for the package) contained in the SPRJ file – if available, but also for each of the associated OSD files that has already been published from the package (i.e. each separate application in the package).

Package settings screen is divided into general settings, settings related to publishing and client visibility and settings related to virtual environment and runtime.

Package name

Overall name for the package.

Package comments

Free-form comments for the package.

If *Override published application(s) comments with package comments* –option is enabled below the comment field, changing package comments will automatically replace comments for each published application. When enabling this option, Application Virtualization Explorer will also prompt if you want to copy package comments immediately over to all published applications even before the comments are edited.

Load content from

Streaming path for a SFT file if package is to be run at the App-V client based and SFT's content is accessed on-demand basis (typically, but not limited to traditional App-V infrastructure or lightweight infrastructure). By toggling *Secure Communication* –option on to the right, streaming path is automatically modified between non-secure (RTSP and HTTP) and secure (RTSPS and HTTPS) protocols and port numbers.

This setting is not relevant if package is to be used with any standalone method (e.g. with associated MSI file) and not streamed. Also in the case of SCCM integrated use or if ASR settings are used at the client, streaming path is overridden from what is set in OSD files.

Operating system visibility

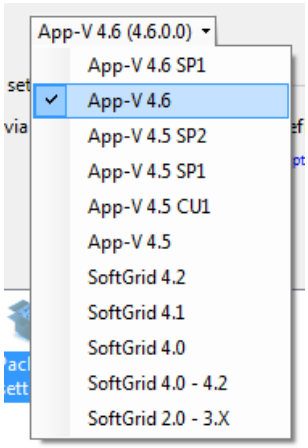
Controls which operating systems package will be visible in. Published applications from the package won't show up in any of the operating systems not selected here.

If you wish to make package visible to any possible target operating system, enable *Visible in all operating systems* –option.

Note that customizing operating system –based visibility on a package -level will overwrite matching setting in all published applications.

Minimum required client

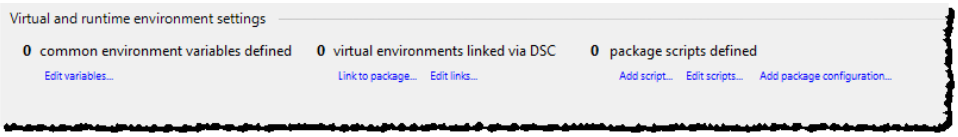
Controls which App-V Client versions will load the package in.



Note that customizing client version –based visibility on a package –level will overwrite settings in all published applications.

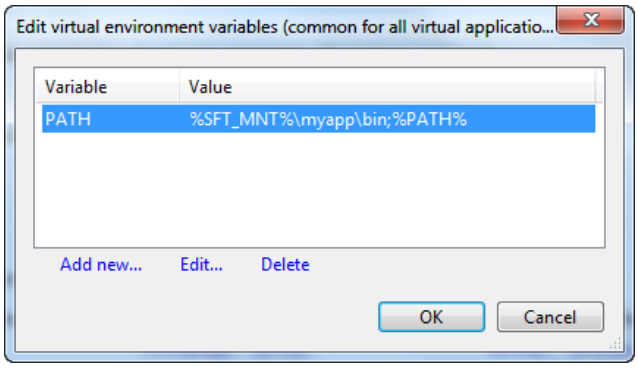
Virtual and environment runtime settings

Virtual environment settings –section allows customization of virtual environment as it pertains to the whole package, instead of individual published virtual application.



Common virtual environment variables

Virtual environment variables, which needs to be present for all published applications from the package (i.e. common variables), can be customized by pressing *Edit variables* –link.



When edited through the package settings instead of per application -basis, Application Virtualization Explorer will only show those environment variables that are found to be identical in each and every published application inside the package.

In the virtual environment variable editor, variables can be added (*Add new* –link), edited (*Edit* –link) and deleted (*Delete* -link) in the manner similar to Windows' own environment variable editor.

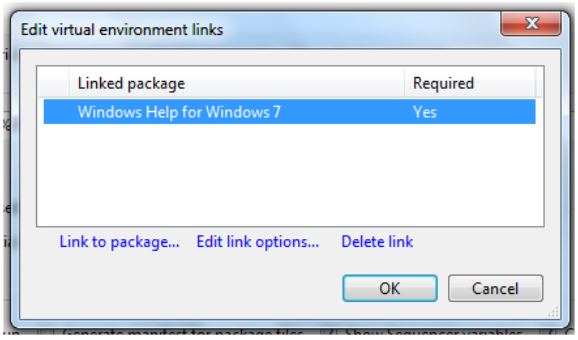
Virtual environment DSC linking

As App-V 4.5 and newer supports Dynamic Suite Composition - or linking packages to packages - Application Virtualization Explorer has also the capability of defining those links from the primary (i.e. one that needs functionality from one or more auxiliary package) package.

Application Virtualization Explorer automatically displays current number of links currently set for the package, and although the setting can be found under package –related settings, these links are actually created in all OSD files separately. For setting up and maintaining links, *Link to package* and *Edit links* –commands can be found under link count listing.

As the functionality for creating a new link can also be accessed from the link editing interface, the process is described below.

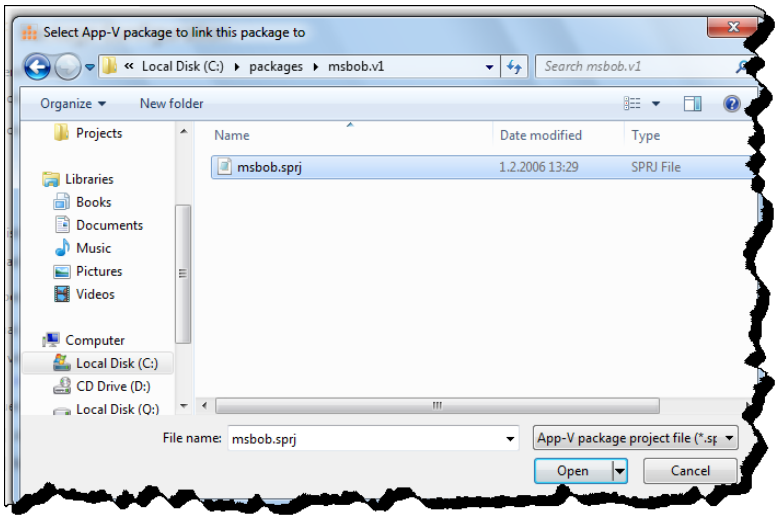
Opening up DSC link editor using *Edit* link –command display currently defined links for the package. If link has previously been done with Application Virtualization Explorer's assistance the actual package name is shown for the linked package, otherwise name of the SFT file is shown.



Also shown is the information if the linking is required or not in order to application from primary package to launch, this matches the MANDATORY attribute in OSD filed. If package link is *not* required, then App-V Client does try to load the linked package but upon failure continues application startup anyways.

Link to package

When invoking linking to a package, Application Virtualization Explorer will first prompt for a SPRJ or alternatively OSD file for the secondary package against which the link will be created:



After reading through the selected secondary package for metadata (including scanning all OSD files), Application Virtualization Explorer prompts for package linkage options.

- **Linked package must be available when starting this package**
  Controls if the secondary package needs to be available for primary package to continue launching at the App-V Client.

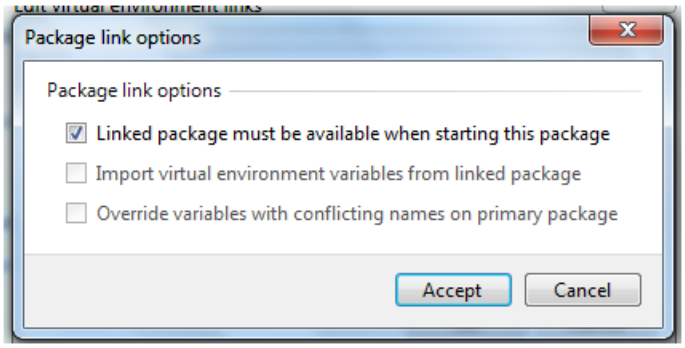
- **Import virtual environment variables from linked package**
  Instructs Application Virtualization Explorer to collect and read in all virtualized environment variables from secondary package and integrate them into all OSD files in the primary package. One of the limitations in App-V's DSC is that virtual environment variables from linked packages are not available to the primary package's merged environment (coming from OSD file rather than SFT which is the only file accessed from other packages in linking scenario) and so Application Virtualization Explorer is able to pre-import them to the primary package.

- **Override variables with conflicting names on primary package**
  If both primary package and linked package has the exact same virtualized environment variable, controls if variable imported from linked package replaces one already on primary package.

After setting the options, Application Virtualization Explorer performs necessary task(s) to set up the link to the secondary package using Dynamic Suiting technology.

Edit link options

Allows changing linking options for the selected secondary package.



Note that the option to import virtualized environment variables is not available after link has initially been created since the DSC link does not carry information about the location of OSD files required for this operation.

Delete link

Removes selected Dynamic Suiting link. Removal operation removes the linking from all applications in the package.

Package scripts

Creating package level scripts causes Application Virtualization Explorer to create an additional application (OSD) into a package. This so-called package application is set to run at user logon through Start Menu Startup –group and triggers all package level scripts set in it.

The underlying idea of package level scripting is to provide single logical point to set-up of resources and/or environment common to all applications in the virtual application package and independently of any one application [inside the package], like OSD scripting would normally be done. Also it is to overcome the situation where packager cannot reliably predict which one application, out of the possibly many, in the suite of applications in one App-V package might first be started by the user and critical [to the correct functioning of the application inside] operations needs to performed in the scripting. Of course, this does not always work correctly if applications are distributed on-demand and users do not log off and back on in-between, so careful consideration needs to be done if package level scripting is something that can effectively be used.

Overall scripting –related features and options are covered later on in this manual as they are the same for both package level script and per application scripting.

MSI customization

MSI customization screen allows you to change parameters for MSI file that App-V generates as method for easy package adding to the standalone clients.

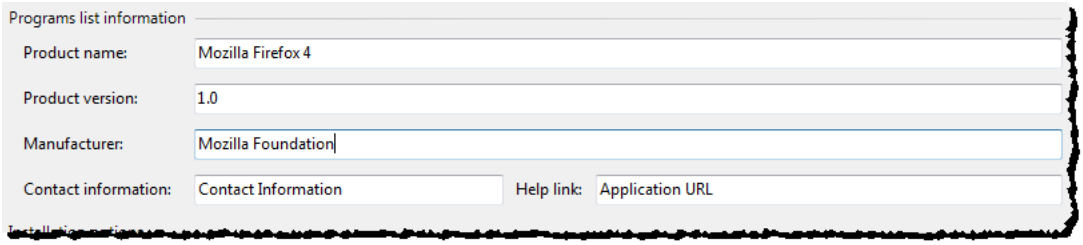
Some of the settings cannot normally be altered at all (at least without need for MSI table editor) while some of the settings can also be set using MSI properties during package execution. When customizing MSI –related settings with Application Virtualization Explorer you have the possibility for deploying resulting MSI file without need to set properties at runtime.

Note that MSI customization screen is only available for packages already having MSI file at the source from where Application Virtualization Explorer opened the package. For packages without MSI, you first have to save the package with Application Virtualization Explorer (in App-V 4.5 or newer format), which will result in MSI file generated by Application Virtualization Explorer, and then re-opening that package for further editing.

MSI customization screen has the following settings for Add/Remove Program data and for installation options:

Product name, Product version, Manufacturer, Contact information and Help link

Information about application for Windows' Add/Remove Programs list / Programs and Features list.



By default App-V generated MSI files will inherit package name as product name and package save version as product version, and generic texts for Manufacturer, Contact and Help link.

You can customize these fields to enable more informative data for end-user by enabling *Customize displayed package information* –option first.

Source for package's content (SFT)

Defines from where the SFT file will be loaded from, either when application is first started (streaming mode MSI) or when MSI is run at the client (standalone mode). Customizable paths in this section equals with *OVERRIDEURL* and *SFTPATH* MSI -properties for App-V MSIs.

You can set streaming or import path without exact SFT filename at the end as Application Virtualization Explorer will automatically complete the path upon saving MSI file with current SFT filename that was saved as part of the package.

Published applications

Published applications folder contains all applications you publish out of the App-V package. Each of these applications represents one OSD file when package is saved to a disk.

Published applications Action -menu options

When in Published applications -screen, the Action menu contains following application –specific options (accessible also from the context menu):

- **Add new application**
  Creates a new published application into package. You can either select any arbitrary executable from the local file-system or from the list of commonly-used Windows utilities. If needing to publish file from within the package itself, you can use Publish this file –functionality in the Files & Folders –screen.

- **Duplicate application**
  Creates a copy of the select application.

- **Delete application**
  Removes selected application from the package.

- **Rename OSD file**
  Allows customization of the OSD file name for the published application.

Application view options

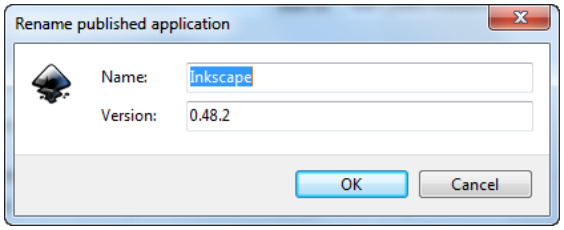
When in viewing published applications, the View menu contains following options:

- **Show all applications**
  Allows showing package level scripting application, which is by default hidden by Application Virtualization Explorer if one exists on the package.

Application screen is divided into: application settings, settings related to publishing and client visibility and settings related to virtual environment.

Application name and version

Displays a full name for the published application.



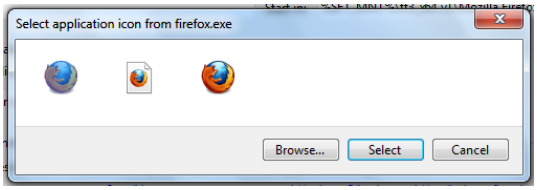
You can rename application by clicking *Edit name* –link beneath the application name.

When renaming application, Application Virtualization Explorer will prompt about renaming all associated [Start menu / desktop] shortcuts. This renaming will be based only on the information in Name –field as that is

what is used for a shortcut naming; App-V Client itself recognized published application by its full name (Name and Version fields together).

If you want to customize comments for published application, which by default are inherited from the package level comments, you can do so with using *Edit comments* –link.

If you want to change the icon associated with the application, you can press *Change icon* –icon link. This will open an icon selection screen where all icons found in target executable are listed and selecting any of them and pressing **Select** –button will change the icon. If you want to browse for an arbitrary icon-containing file (EXE, DLL and ICO files), you can press **Browse** –button.



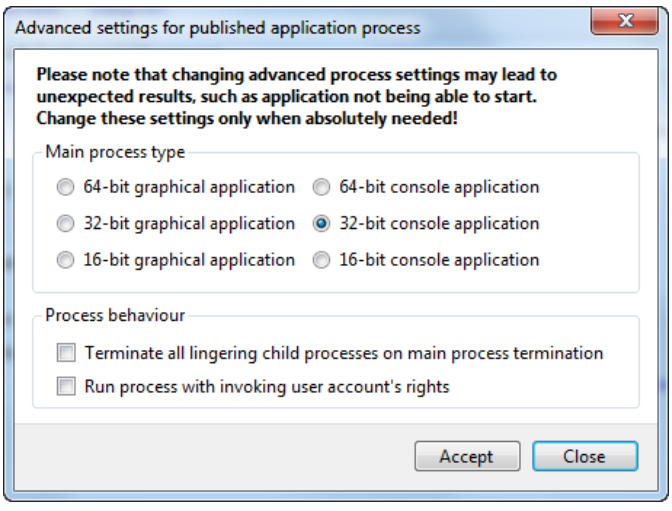
Start program

Path to an executable or other file, which App-V Client will instruct Windows to start when published application is executed on the client.

This path can point to internal to the package file or any valid external files, such as EXE file on the network share etc.

Path can be defined without preceding drive letter, in which case it is interpreted as being path to an internal directory of the package. For easy access to executables inside the package, *Browse Package* and *Browse filesystem* –links beneath file path can be clicked after which it is possible to select executable file using integrated package files browser.

Although Application Virtualization Explorer (and Sequencer) will try to detect automatically from the published application which type of process it is, sometimes a manual change to this information is needed.



A sample case could be graphical Java –based application that is launched through java.exe which is a console process; telling that published application is graphical in this case might lead to a situation wherein App-V Client tries to monitor java.exe process for window messages (a mechanism in Windows for GUI applications), which it won't be able to do as console processes do not handle have such messages. This may lead to situation where client will deem that application did not start correctly even though it did, and show an error message.

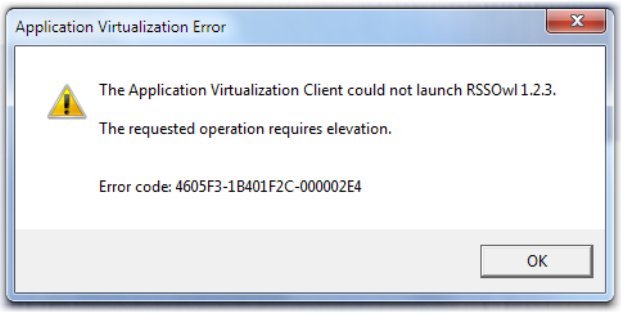
Application Virtualization Explorer supports changing of process type for published application by pressing *Advanced process settings* –link beneath a Start program -field. This will bring up a screen in which the process type can be changed to match the type of program in question.

Terminate all lingering child processes on main process termination

Enables or disables (default) the setting that instructs App-V Client to automatically terminate subordinate processes started by the main process (i.e. one that Start program field defines) when the main process exists. In normal situations those processes should close by themselves but in some rare cases this does not happen and forced termination could be used.

Run process with invoking user account rights

Enables or disables (default) special virtualized environment variable that instructs Windows to use user rights of the account invoking the application.



This can be set to avoid situation wherein Windows tells that application being started requires elevation and the user either already has the administrator rights or application in question in fact does not need administrative rights at all. This setting has no effect on applications already containing manifest telling what rights are needed, but rather on (old) applications without any embedded manifest information.

Please note that changing any of the settings in this screen may cause unintended behavior if used incorrectly!

Startup parameters

Any startup parameters that will be given to executable in Start program –field.

Start in

A working directory for the executable in the Start program –field.

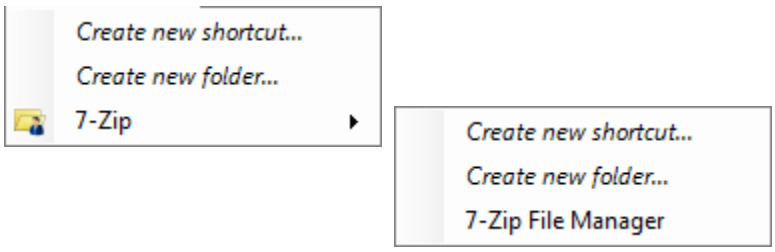
Operating system visibility

Same visibility settings as in the overall package settings screen, but if set here it affects just this published application independently of other possible applications in the same package.

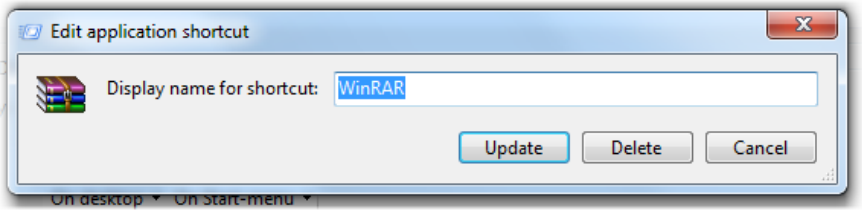
Show shortcuts

Allows customizing where to shortcuts will be placed on the client system when published application is added to App-V Client.

App-V supports placement of shortcuts for same published application to multiple locations, and Application Virtualization Explorer supports customizing these selections for both Start Menu and desktop locations using menu hierarchy modeling both locations:



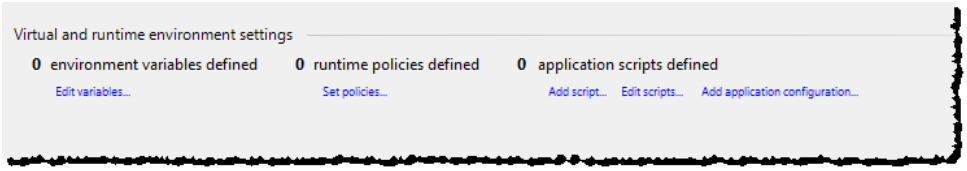
If you want to rename or delete shortcut, click on it from the menu and application shortcut editing screen will be displayed.



After editing the name of the shortcut, press *Update* –button to apply changes. Press *Delete* –button to remove shortcut from the menu location completely.

Virtual and runtime environment settings

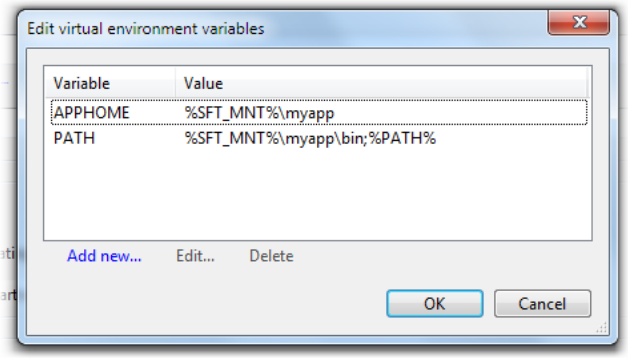
Virtual environment settings –section allows customization of virtual environment as it pertains to published application. While package's virtual registry (VREG) and virtual filesystem (VFS) is shared by all published applications from that package, changing virtual environment settings for published application affects only that application.



Virtual environment variables

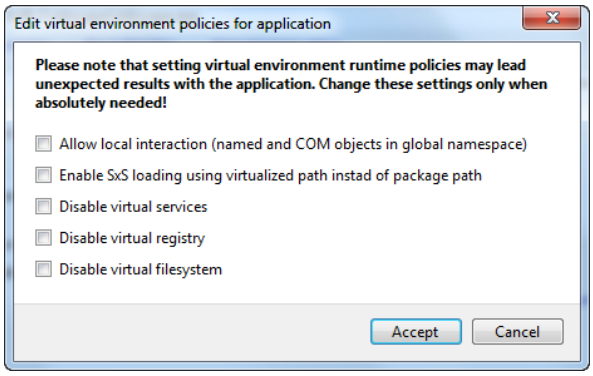
Published application's environment variables can be customized by pressing *Edit variables* –link.

In the virtual environment variable editor, variables can be added (*Add new* –link), edited (*Edit* –link) and deleted (*Delete* -link) in the manner similar to Windows' own environment variable editor.



Runtime policies

In some very rare cases – or while debugging a package – special runtime policy settings could be set for published applications. These policy settings are accessible using *Set policies* –link.



In policy settings screen, following options can be set:

- **Allow local interaction**
  Causes runtime -instantiated COM objects and other named objects (from virtual application) to be created outside the isolated name-space for the package, making them effectively visible to other processes on the system. Furthermore, behavior of in which order COM requests from virtual applications are routed is changed; making local system first to be searched and resources in the virtual environment a second.

  This option is commonly used to solve interaction –related issues between virtual application and local applications/system. If policy is set for one published application from the package and found resolving a interaction issue, it should be set for all other published applications from the same package as well as to not introduce any inconsistencies.

- **Enable SxS loading using virtualized path instead of package path**
  Causes side-by-side (SxS) DLL loads to be presented to application using resolved mapped path (e.g. c:\Windows\SxS) instead of real, physical, path (e.g. Q:\app\VFS\CSIDL_WINDOWS\SxS) if the file is in virtual filesystem.

- **Disable virtual services**
  If package contains virtual services, setting this option will cause them not to be started (or visible) to virtual application.

- **Disable virtual registry**
  Setting this option will cause virtual registry entries from the package to not be visible to virtual application.

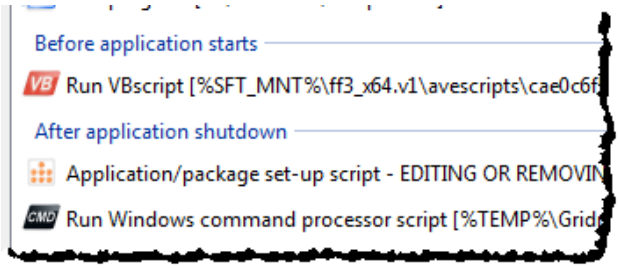
- **Disable virtual filesystem**
  Setting this option will cause virtual filesystem (VFS) entries from the package to not be mapped according their virtual paths.

Please note that changing any of the settings in the policy screen may cause unintended behavior if used incorrectly!

Application scripts

Shows a number of application scripts set for the package. Applications scripts and script editing is explained in the following section.

Application scripts



App-V packages have a powerful mechanism available for performing client-based operations, which cannot be added to the virtual environment directly or need to affect local system directly. This mechanism is scripting, which in this context means both running an actual Windows command interpreter script or invoking any executable on the system or in the package.

While scripting in OSD is good way to extend what needs to be done at the runtime, Sequencer has basically zero support for managing them as they need to be added manually through OSD editing function. Built-in scripts also have some limitations into them, most notably that they always execute on virtual application -launching user's security context.

Application Virtualization Explorer aims to solve the hardness of adding, editing and maintaining virtual application scripts, and to some extent to overcome some of the built-in scripting functionality's limitations, with dedicated script –editing interface and script execution enhancements.

While OSD file does not really make an difference between simple program execution and actual scripting file (with the exception of SCRIPTBODY scripting), Application Virtualization Explorer has built-in support for detection and intelligent handling of Windows command-script, VBscript script, Powershell script and actual program invocation. This makes it very easy to distinguish between different types of scripts set for the package and manage actual script's contents without needing to know or care about how to call out the script in OSD's syntax.

Application script interface is both accessible for individual applications but also for package level scripting, in which case a package –specific extra application will be created into package that is hosting the scripts set to execute on package level. You can access the list of currently present scripts with *Edit scripts* –link or start new script creation (also accessible from script list interface) using *Add script* –link.

Script list

When in main list of scripts, new script can be added using *Add script to run* –link. Alternative you can use *Add program to run* –link to specify any arbitrary executable you want to run "as script". The distinction is that explicitly selecting adding new script causes Application Virtualization Explorer to handle script addition as being one of the three supported script languages while adding a program makes no assumptions on the content.

While a script is selected in the screen, its execution settings can be altered using *Edit execution settings* – link and its contents can be edited using *Edit contents*- link.

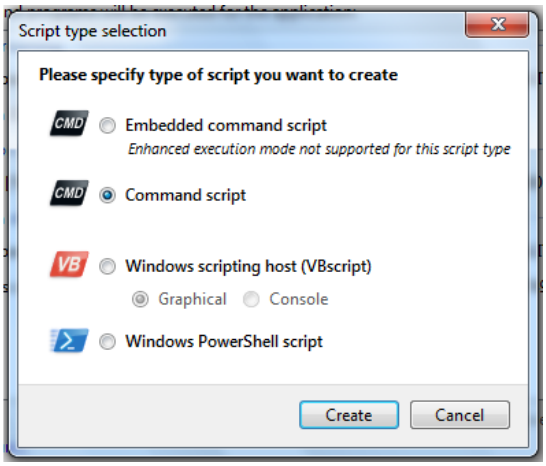
If wanting to remove script from the list (and thus from the resulting OSD file), *Remove script or program* – link can be chosen.

To annotate your script, you can use *Edit description* –link to give description for the scripts purpose and Application Virtualization Explorer will show that description in the script-list instead of the actual path to the script file.

Since OSD scripts are by nature bound by the virtual application they have been created / defined against, setting the same script to multiple virtual applications would normally require creation of the same script over and over again for each published application. To help setting the same script to multiple virtual applications in the same package, *Copy script to* –link can be used to make a copy of the selected script into a different virtual application from the one it's currently defined in.

Script type selection

When adding an actual script, first screen that is shown is the selection of script type.



Script type can be one of the following:

- **Embedded command script**

  Windows command interpreter script which is directly embedded inside OSD files and has no separate script file anywhere.

- **Command script**

  Windows command interpreter script which is executed from within or outside the package.

- **Windows scripting host**

  VBscript script which is executed from within or outside the package. Application Virtualization Explorer supports both Graphical interface and command-prompt interface scripts, executed with wscript.exe and cscript.exe respectively.

- **Windows PowerShell script**

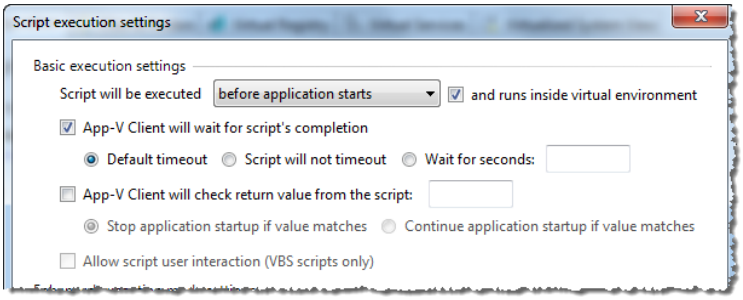
  PowerShell script which is executed from within or outside the package. If creating a PowerShell script, all target machines need to have PowerShell installed into them.

Except the **Embedded command script** type, all other scripting types support all Application Virtualization Explorer's scripting features, including enhanced settings. Embedded command script means OSD's SCRIPTBODY –script and it has added limitations from Application Virtualization Explorer's point of view since its execution is handled more directly by App-V Client itself rather than triggering external scripting host.

After creating new script of selected type, script execution and content screens are shown next.

Script execution settings

Execution setting screen allows control of execution timing and other parameters related to running a script or program. In addition to standard App-V supported execution settings ("Basic execution settings"), Application Virtualization Explorer provides enhanced mode settings that extend script control capabilities normally available for App-V packagers.



For enhanced execution mode settings, Application Virtualization Explorer replaces target script or program invocation with Application Virtualization Explorer –provided launcher application (*ScriptRunner.exe*) which in turn executes the actual script or application. This way Application Virtualization Explorer can inject itself in-between the App-V Client and script and provide more execution control over what App-V scripting natively supports. Because OSD's SCRIPTBODY –type scripting does not support this injection method, enhanced settings are disabled for all embedded scripts.

Since Application Virtualization Explorer will store its own script launcher application directly inside the package's file-system, enhanced execution settings cannot be set for any script or application that runs before streaming as it is not possible to have access to this launcher application from the package at that stage.

Important note: You are licensed to use and distribute Application Virtualization Explorer's script launcher application only in conjunction with your Application Virtualization Explorer usage and not in scenarios involving with adding it manually to packages outside Application Virtualization Explorer.

Script will be executed

Control the timing of when App-V Client executes the script. This can happen before and after streaming operation (or very early on in the phase if package is fully cached and/or no streaming delivery is used), before or after virtual application is launched or after virtual application is shut down.

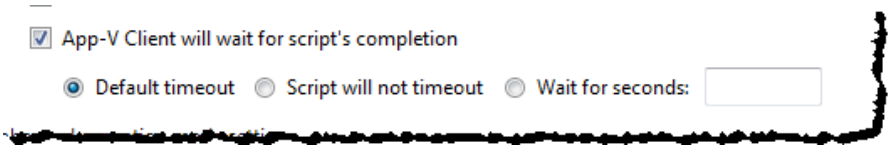
And runs inside virtual environment

If set, specifies that script will be running inside the package's virtual environment and has access to the virtual registry and package's file-system. If this selection is cleared, script or program is started normally outside any App-V context.

Please note that you cannot run script inside virtual environment before streaming as virtual environment is not available for App-V at that phase.

If script is set to run outside the virtual environment and script file is physically stored inside the package, Application Virtualization Explorer will automatically add additional logic for copying out the script file to temporary location on the real system so that it can be executed by scripting host (which would not have access to Q: drive).

App-V Client will wait for script's completion



If set, specifies that App-V Client will run script synchronously i.e. waits until script or program returns before continuing to the next script or to the next phase in the launch order. If the script is run asynchronously, App-V Client will start specified script and continue forward without checking if execution succeeds.

If synchronous execution is enabled, you can set the timeout value (in seconds) for the script execution. Having script to timeout at some point is useful if there is possibility that script or program execution will be blocked or gets stuck in some scenarios since App-V application launch/shutdown will be delayed with it. If, however, it is unknown how long the execution lasts then you can enable *Script will not timeout* –option to allow script to run indefinitely. When timeout occurs at the client, application launch process is aborted and virtual application does not launch either.

App-V Client will check return value from the script

If set, specified that App-V Client will check for return value (also known as ERRORLEVEL value) from the executed script, and will either stop the virtual application launch process if the return value matches or does not match the value specified in the numeric field. The control of the behavior what will happen depends on if the *Stop application startup if value matches* –option is selected or if *Continue application startup if value matches* –option is selected.

App-V Client will check return value from the script

If set, Application Virtualization Explorer will not try to run VBscript script in a non-interactive mode which is the default state for VBscript scripts created from Application Virtualization Explorer. This is useful for scripts that will need some user-input from the currently logged on user on a client machine.

This option is not available for other script types.

Run script only once

If set, specifies that Application Virtualization Explorer will execute the script's contents only once instead of each and every time application/package is run.

For additional fine-grained control, you can specify if the script is run once per machine, once per user and persisting run-once state over App-V Client's package repair operation(s) or once per user so that script is executed again if package is repaired.

Hide script execution

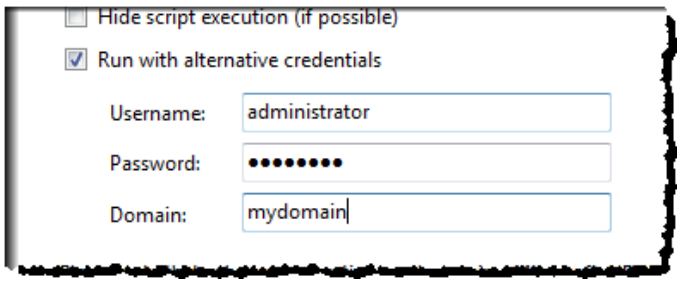
If set, specifies that Application Virtualization Explorer will try to hide script's output if it is technically possible. This option is especially useful for hiding command script's execution and no cmd.exe's window will be shown in that case.

Please be aware that if execution is hidden and script prompts for input from user or displays message boxes, the script execution will be stuck since user is not able to see and acknowledge those events.

Run with alternate credentials

If set, runs script or program using alternate credentials from what the current user is using who is launching the application. This option is especially useful for running actions via scripting that needs e.g. local administrative rights and would not otherwise be possible with OSD scripting.

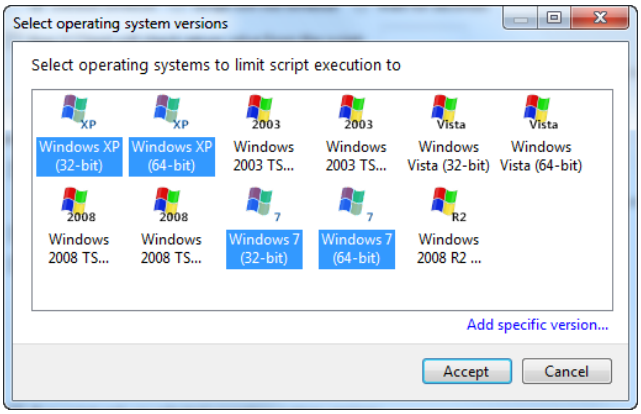
When running a program or script with alternate credentials, please be aware that it is not possible to access contents of the package from virtual drive since process being run is not running in user's context and App-V explicitly denies access to virtual drive in that case.



Note that when set, alternative credentials are stored encrypted and cannot be viewed by the packager or end users outside script editing interface of Application Virtualization Explorer itself.

Run script only on selected operating system versions

If set, allows limitation of script execution based on Windows operating system version on a client machine. By default OSD scripts will run on all machines where the package itself has been successfully added to, but if there is a need to limit scripting's scope based on operating system, Application Virtualization Explorer's enhanced scripting option can be used for the purpose.



After the option is enabled, pre-populated list of operating systems is presented from which multiple versions can be selected. All the selected systems will be in the list of allowed operating system for script execution.

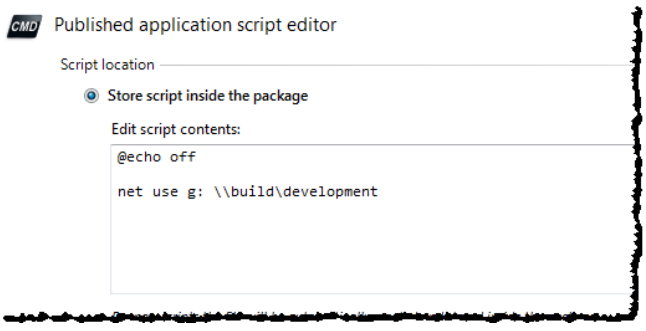
If there is a need for a fine-grained system version selection, *Add specific version* –link can be used to specify operating system version, service pack –level and the bitness, after which the custom version is added to the main list for selection.

Script content editor

To facilitate easy editing of script's contents, Application Virtualization Explorer has an integrated script content editor. Content editor interface will differ for supported script types and plain program executions (or scripts that Application Virtualization Explorer does not directly support).

If the script file is physically inside the package and of supported type, Application Virtualization Explorer detects this script and displays contents in an editor window. Changing any of the text will automatically update the script file inside the package without need to explicitly do so, after pressing OK -button.
You also have a possibility to open the script for editing in your preferred editor by pressing *Edit externally* –button. Application Virtualization Explorer automatically opens program registered as editor for that file type (for example, powershell_ise.exe for PowerShell) and waits until that program closes, after which changes to the file is scanned and put back into package.

Scripts created with Application Virtualization Explorer will automatically be named and placed inside the package, without packager needing to do any manual steps to that effect.



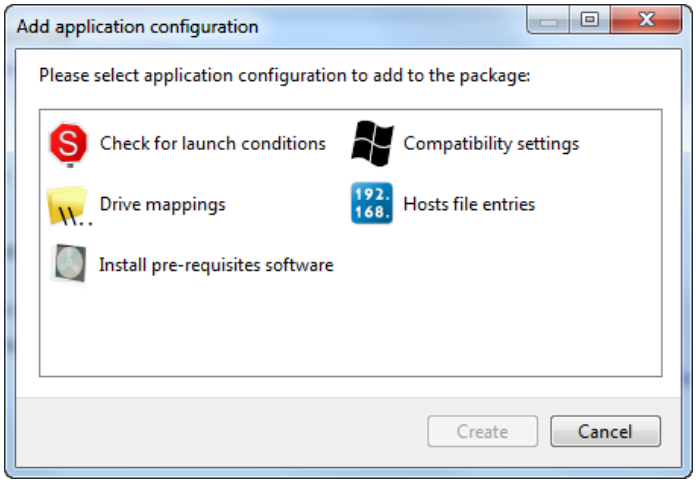
If script path points to outside the package's structure or you want to manually specify script path, choosing *Execute script from outside the package* –option allows you to do just that. Note that this option is disabled for embedded command scripts.

If the script requires some parameters passed to it (to the script, not scripting engine), you can set those in script parameters –section. Note that this option is disabled for embedded command scripts.

For non-script type application scripts, Application Virtualization Explorer will show actual command to run as a script instead of integrated editing interface.

If raw access to command to invoke (i.e. HREF's contents) is required, pressing *Advanced script edit* –link will change content editor into mode where that direct invocation can be freely edited and customized.

Application configuration scripts



In addition of making it possible for packager to add any custom scripts – executed from inside or outside the package's VE, Application Virtualization Explorer ships with number of pre-created configuration scripts for commonly needed actions that cannot be done with native App-V package capabilities and would therefore require packager to write own custom script.

Adding a new configuration script can be done with *Add application configuration* –link, when in either Package or Application view. This will launch a configuration script selector screen, listing all built-in configuration scripts. Please note that some of the configuration scripts can only be selected when accessing this screen from individual application and then some only from package –level; this is because those operations are only meaningful for one application or overall package.

Each application script contains a configuration screen, allowing packager to customize and configure script's operation. Once application configuration script addition is run against the package, configuration cannot be done again since scripts will appear as normal application scripts. If required, added configuration script can be removed and then re-configured by re-running the configuration script screen.

Important note: Since pre-made configuration scripts are implemented through VBscript scripts, Windows Scripting Host 5.7 or newer has to be installed on the target machine. Pre-made scripts are also provided as-is and may not work on all target environments and configurations.

Important note: You are licensed to use, customize and distribute these pre-made scripts only in conjunction with your Application Virtualization Explorer usage and not with adding them manually to packages.

Application Virtualization Explorer ships currently with the following application configuration scripts:

- **Check for launch conditions**

  Checks for existence of entries from the filesystem (files and directories), or from the registry (keys and values, including the data). If the matching entry is found, either stops or allows continuation of the launch process.
  This script can be used for checking some environmental conditions required by a virtual

application, and preventing the start of it is they are not met.

- **Compatibility settings**

  Sets application compatibility settings for a virtual application.
  Configuration includes the same compatibility mode settings as normally available from Windows' Compatibility –tab for a normal application.

- **Drive mappings**

  Performs network and local drive mappings, including optional disconnect of mapped drives.

- **Hosts file entries**

  Adds entries to Windows HOSTS –file.
  This script supports setting alternative credentials that has modify -rights to the HOSTS –file on all target machine since the file is normally not writable with standard user rights.

- **Install pre-requisites software**

  Performs software installation to the target machine outside the virtual environment, using simple software package file stored inside the package itself or executed from the external location.
  This script supports setting alternative credentials for installation purposes.

  When selecting MSI file for installation, you don't need to specify any unattended parameters as they will be added automatically.

- **Protocol handlers**

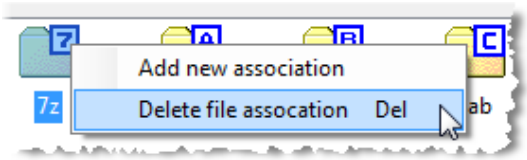
  Registers simple protocol handlers to the current user's context.
  This script additionally performs detection of protocol handler entries in the package's virtual registry and pre-fills protocol handler list with any protocols present inside VE.

  Please note that registered handlers are **not** automatically removed from the user's registry if virtual application is removed. This is due to limitation of App-V wherein it does not provide any removal hooking for scripts.

File associations

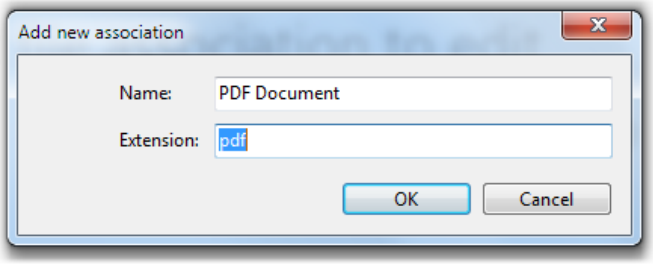
File associations folder contains all file type associations (FTAs) you publish out of the App-V package. FTAs are entries stored in the OSD file as each extension is associated with some published application.

Each file extension association supports following actions; accessible either from mouse context-menu, keyboard command or through **Action** –menu:



- **Add new association**
  Creates a new file association for the package.

  When creating a new association, **Name** and **Extension** for that association is prompted.



  The extension part is defined without preceding dot (e.g. “pdf” instead of “.pdf”).

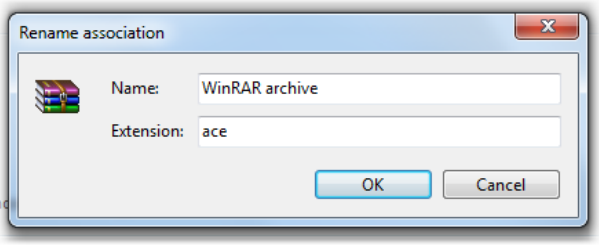
  As all associations have to be linked to some existing published application in the package, Application Virtualization Explorer will also prompt for published application which will handle specified file extension.

- **Delete file association**
  Deletes file association from the package.

Association name and extension

Description and extension for file association

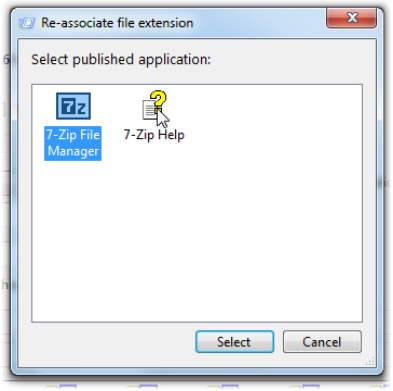
You can rename association description and extension by clicking *Edit* –link beneath the association description.



If you want to change the icon associated with the extension, you can press *Change icon* –icon link.

This will open an icon selection screen where all icons found in target executable file extension is pointing to are listed and selecting any of them and pressing **Select** –button will change the icon. If you want to browse for an arbitrary icon-containing file (EXE, DLL and ICO files), you can press **Browse** –button.

Associated to application



Published application which will be launched when file with this extension is opened.

You can move file extension to some other application published from the package by clicking *Reassociate* –link beneath the name of the application. This will display application selection window displaying all published applications for package, from which new application can be selected.

Explorer shell behavior

Options on how files having specified extension will be handled by and in Windows Explorer.

Commands

Specifies list of command verbs specified for the extension.

These commands will show up in Windows Explorer's context menu when matching file has been selected.

Existing commands can be renamed using *Rename command* –link beneath a selected command, and commands can also be removed using *Remove* –link.

Register as default action for the extension

Assuming multiple commands exists for a given association; one of them can be designated to be a default command. This makes the command to be a default action when user double-clicks the file having the associated extension in the Windows explorer.

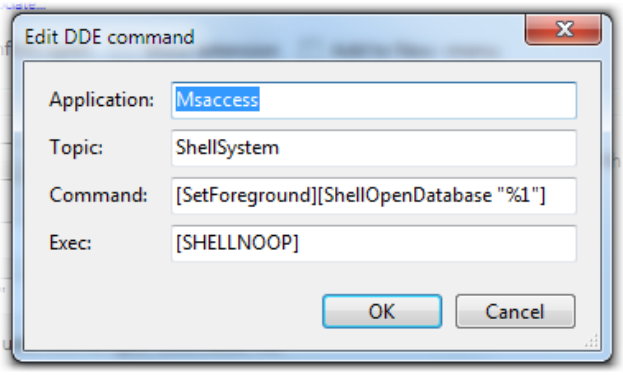
Display name

User –displayed name for the selected command.

Parameters to application

Command-line parameters passed to the published application when selected command is invoked.

This command uses DDE to open associated file



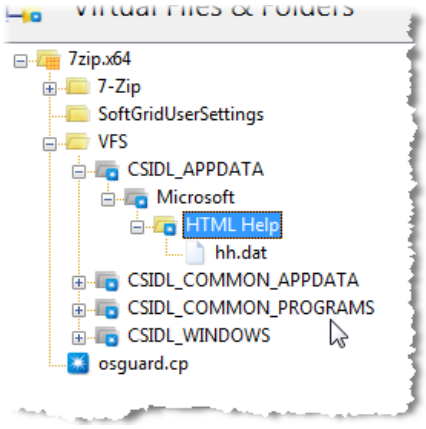
If extension has to be opened using DDE, checking DDE –checkbox enables DDE for it.

To define DDE parameters, *Edit DDE parameters* –link can be used to open parameters screen.

In the parameters screen, **Application**, **Topic**, **Command** and **Exec** –fields needs to be filled with application –specific information that DDE passes to the application.

Virtual files & folders view

Internal directory structure



On a left-hand side of the screen is a tree view representing the internal directory with package root (or asset) directory as a topmost node, under which all other package files and directories are displayed in.

You can add new virtual files or directories, delete them and make modifications to contents and properties.

Special icons for the directory tree are:

 A directory which is virtualized fully through VFS.

 A directory which is virtualized as merged through VFS.

 A file in the VFS configuration that has at least one target mapping which is marked as deleted.

Following files and directories carry a special meaning to App-V, and should not be edited or renamed:

- **VFS**
  Subdirectory that holds all directories and files mapped through Virtual Filesystem (VFS) functionality over other partitions on the App-V client machine.

  In normal circumstances, VFS directory holds directories representing common Windows locations (such as CSIDL_WINDOWS for Windows –directory), and other directories or files that needs to be virtualized over that location.

- **SoftGridUserSettings**
  Subdirectory that holds user-specific cached settings file (itself physically cached in the PKG file) when package has been run at the client.

- **osguard.cp**
  A virtual environment configuration file, which is re-created each time package is saved out.

View options

When in the Virtual Files & folders view, the View menu contains following options:

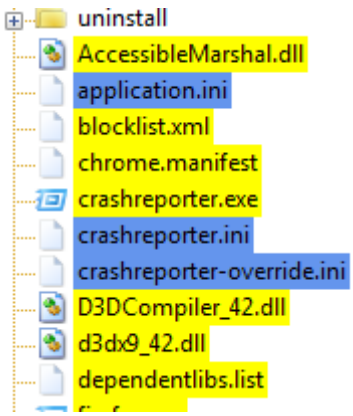
- **Filter view**
  Allows filtering of displayed virtual files using the following characteristics:

  - **Displaying files in 8+3**
    Switches tree view into mode that shows virtual files and directories in 8+3 format instead of the usual long name.

  - **Display only files in FB1**
    Filters out all files that are only part of the FB2 (i.e. data that is not streamed in initially if SFT streaming is used for App-V Clients).

    Note that enabling this option will cause all DATA blocks to be scanned for all files, which might incur delay when applying filter.

- **Highlight files**
  Allows highlighting (using background coloring) of virtual files having the following attributes:

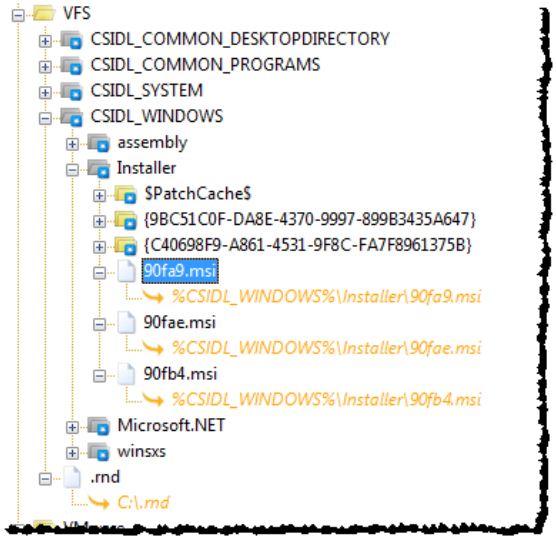
  - **With override status**
    Highlights all the files that have the Override –attribute set using a yellow color.

  - **With user data status**
    Highlights all the files that have the User Data or User Configuration –attribute set using a blue color.

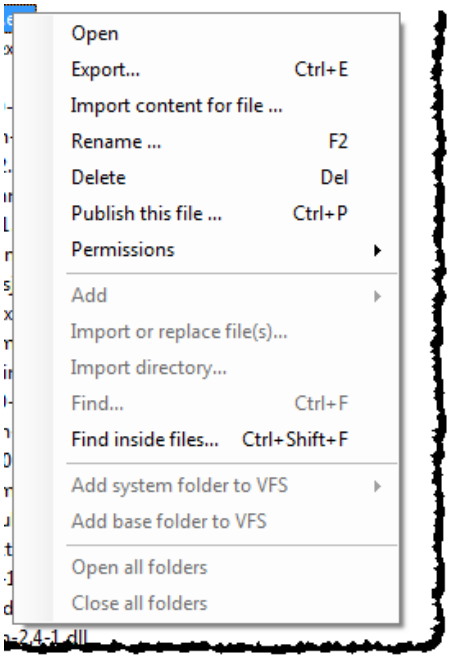
- **Resolve VFS targets**
  If enabled, displays VFS mapping target paths converted to a currently/locally valid directory locations. Otherwise encoded paths are shown as virtualization targets (e.g. "%CSIDL_SYSTEM%\app.dll").

- **Show VFS targets as subnodes**
  If enabled, displays VFS mapping targets paths directly in the package files tree, underneath the virtual files located in the VFS folder structure.

- **Enable deprecated features**
  Enables usage of attributes that has been deprecated in the newer App-V versions.



Available actions for virtual files and directories



Each file or folder in the files view supports subset of the following actions; accessible either from mouse context-menu, keyboard command or through **Action** –menu:

- **Open**
  Extracts specified virtual file to a temporary location on machine and open it through Windows.

  This action is not available for directories.

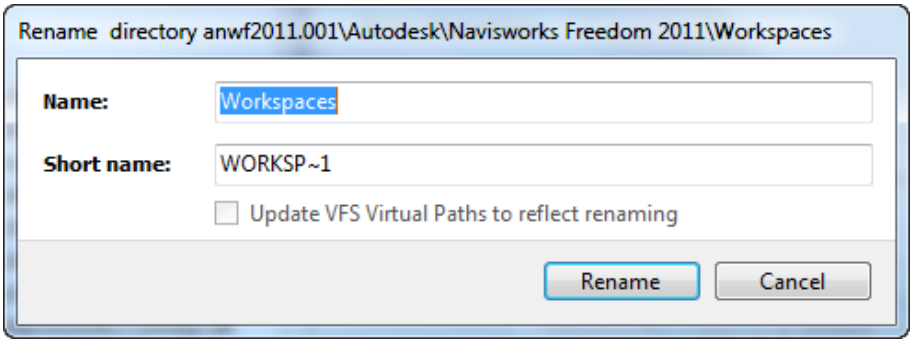
- **Export**
  Exports specified virtual file or contents of a directory to a target location on the local system.

- **Import content for file**
  Overwrites contents of specified virtual file with the contents from selectable file on the local system. Can be used to replace/update file in the package with new content.

  This action is not available for directories.

- **Rename**
  Change the name of the selected virtual file or directory. Root directory should never be renamed unless intending to save out completely new, independent, package which would also require regeneration of GUIDs for the package.

  Invoking this action displays file/directory renaming screen.



This screen will allow changing both normal file name, as well as short name (8+3) assigned for the directory entry if the long name exceeds the 8+3 format.

If the entry to be renamed is part of the VFS configuration, by default Application Virtualization Explorer will also rename all VFS mapping targets as it is conceivable that virtualized paths need to reflect new directory or file name, unless directory being renamed is one of the well-known directories. VFS renaming can be disabled by clearing out *Update VFS Virtual Paths to reflect*

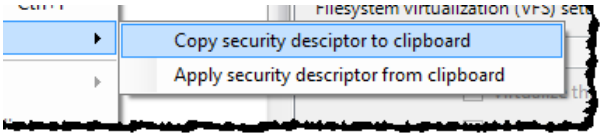
renaming –option.

- **Delete**
  Deletes virtual file or directory from the package.

- **Publish this file**
  Creates a new published application out of the specified virtual file.

  This action is not available for directories and when only SFT has been loaded instead of the full package.

- **Permissions**
  Allows copying and applying security descriptors for virtual files and directories using SDDL (Security Descriptor Definition Language) formatted string.



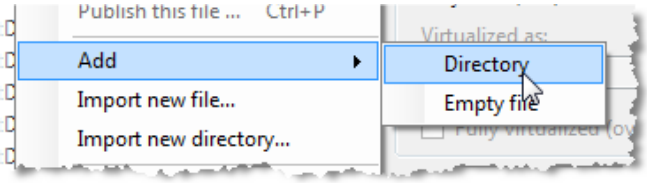
  This option is only available for packages and files that store security descriptors (App-V 4.5 and newer). Furthermore, manually assigning explicit security desciptors to virtual directories won't affect security descriptors for any directories or files underneath them; this may cause unintended side-effects with NTFS permission inheritance as the package is being executed on the client.

  - **Copy security descriptor to clipboard**
    Copies security descriptor string from the file or directory into clipboard.

  - **Apply security descriptor from clipboard**
    Replaces security descriptor for the file or directory with one read from the clipboard. Please note that if the SDDL string in the clipboard is invalid, the security descriptor cannot be updated.

- **Add – Directory**
  Creates a new empty directory underneath specified virtual directory.

  This action is not available for files.



- **Add – Empty file**
  Creates a new empty file underneath specified virtual directory. You can then use *Import content for file* –action for newly created file to add contents to it.

  This action is not available for files.

- **Import or replace file(s)**
  Imports file(s) from a local filesystem to a specified virtual directory. Can be used to add new files to the package or to replace existing ones. If name of the one or more files selected matches names

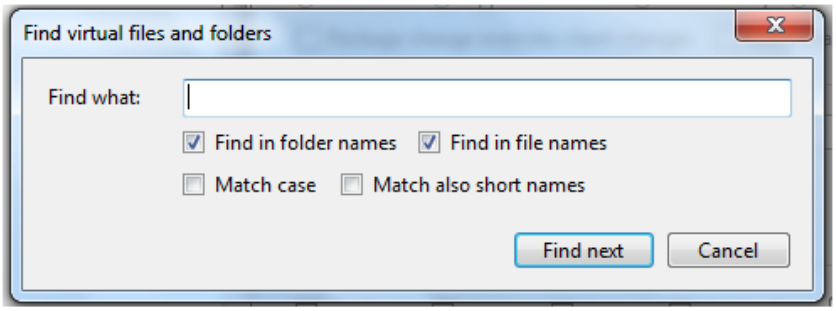
of the virtual files in the specified virtual directory, only replaces contents of those virtual files.

This action is not available for files.

- **Import directory**
  Imports contents of a directory (directories and files) from a local filesystem to a specified virtual directory. Can be used to add existing directory structures to the package.

  This action is not available for files.

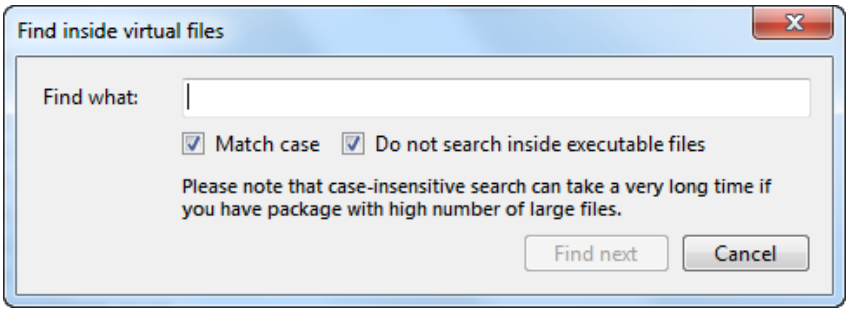
- **Find**
  Opens a files and folders find dialog:



  Find dialog allows searching files and folder with the matching name. You can set additional options in the dialog to perform case-sensitive search and optionally include 8+3 filenames in search.

  This action is not available for files.

- **Find inside files**
  Opens a find inside virtual files dialog:

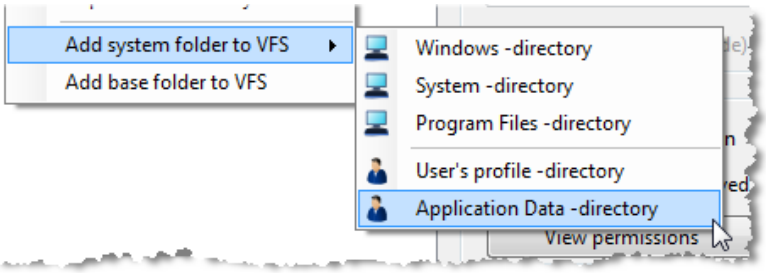


  Find inside files –dialog allows searching for text strings inside virtual files in the package. The search will be performed against specified search text using three different encoding variations of ASCII, UTF8 and UTF-16 (Unicode).

  You can optionally disable the case-sensitive searching (which is the default), but choosing case-insensitive search can considerably increase the search time. By default Application Virtualization Explorer will not search inside files that end with .exe, .dll or .com; but those file-types can be

included in the search by switching off the option to skip executable files.

- **Add system folder to VFS**
  Create pre-defined Windows or user-specific directory to a VFS configuration. Can be used to add commonly needed VFS folders.



  By default folder created using this action will be virtualized as merged.

  This action is only available for VFS directory.

- **Add base folder to VFS**
  Create any arbitrary VFS mapped folder to a VFS configuration. Can be used to add custom folders or uncommon Windows locations.

  Selecting this option will open Virtual Filesystem mapping editor for the newly created VFS folder. By default folder created using this action will be fully virtualized.

  This action is only available for VFS directory.

- **Open all folder**
  Expands selected virtual directory and all virtual directories under it in the tree view.

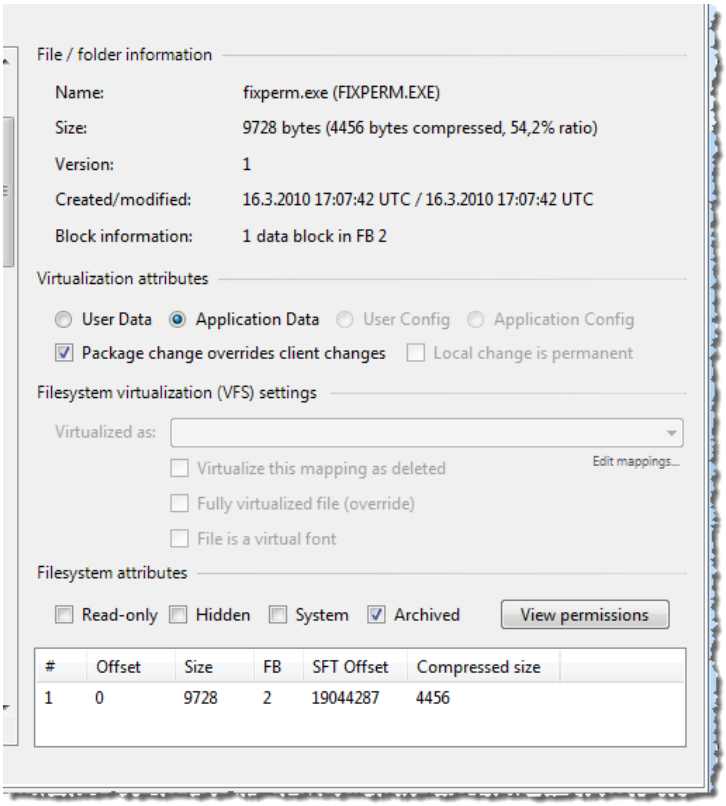
- **Close all folder**
  Closes selected virtual directory and all virtual directories under it in the tree view.

Please note that when PKG file has been opened for viewing, only Open and Export actions are supported for files and directories.

Directory or file properties

On a right-hand side of the screen, information for the currently displayed file or directory is shown.

Information screen is divided into sections of general file information, virtualization attributes, VFS virtualization related information and file-system attributes.



Name

Displays both long and short (8+3) names for the file or directory selected.

Size

Displays size of the file.

If the package is in compressed format, additionally displays compressed size and compression ratio for the file.

Version

Displays internal version number for the file or directory.

Each time a file in the App-V package gets updated, its version number is incremented to match overall package version number. App-V uses this information to track which files has been changed in which version of the SFT file.

Creation and modification timestamp

Displays creation and last modified timestamps for the file or directory.

Block information

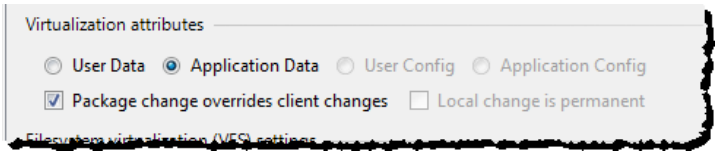
Displays information about DATA blocks (storing file's data inside SFT file) file is divided into.

For new virtual files added through Application Virtualization Explorer, and before package is next time saved out, does not display anything as DATA blocks are calculated and created during SFT encoding operation.

For files cached in a PKG does not display anything as PKG does not store files divided into App-V DATA blocks or have a concept of Feature Blocks.

Virtualization attributes

Displays App-V -specific file attributes.



These attributes control file's behavior when it changes on the client or has been updated centrally in SFT file.
Some of the attributes (User Config, Application Config and Permanent) has been deprecated in the newer versions of App-V, and so Application Virtualization Explorer will by default disable those options in the interface unless explicitly enabled from the View -menu.

- **User Data**
  If virtual file is changed on the client, the change will cached specific to that user and other users won't see file as changed.

- **Application Data**
  If virtual file is changed on the client, the change will be cached specific to that machine and other users will see file as changed.

- **User Config and Application Config**
  Semantically works the same way as User Data and Application Data.

- **Package change overrides client changes**
  If file has been modified on the client and subsequently cached (in PKG files), updated version from the SFT file will overwrite it at the event of conflict.

  Usage of this attribute makes sure that centrally applied update to a file will always take precedence over client-changed one. By default, App-V Sequencer marks all executable files with override –attribute.

- **Local change is permanent**
  Mechanism in older SoftGrid versions to override already set override -attribute i.e. reverse the meaning of override without actually un-setting override attribute.

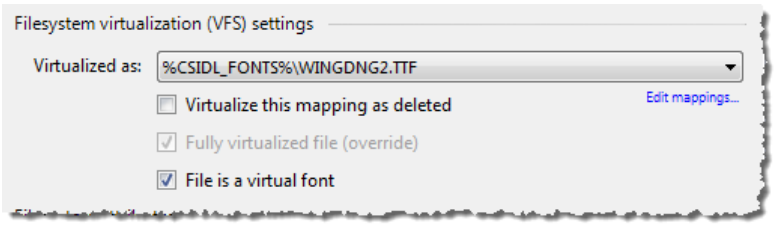
  This attribute has been deprecated and is not supported by newer App-V Clients.

Virtual Filesystem (VFS) settings

If selected file or directory is part of the package's VFS configuration (i.e. underneath VFS folder), Filesystem (VFS) Virtualization sections shows attributes related to the VFS mapping for that entry.

Virtualized as

List of all currently defined targets for the file mapping.



These mappings represents information by which App-V Client will show VFS –enabled entries overlaid over normal partitions (such as C: drive) to the running virtual application, while keeping the actual files and directories stored inside the package's directory structure. When application requests access to such file or directory, I/O request is seamlessly forwarded (i.e. path altered) to the virtual drive and package's own structure.

While in normal circumstances a directory or file is only ever mapped to one target location (for instance, **\package\VFS\CSIDL_SYSTEM\file.dll** to **%CSIDL_SYSTEM%\file.dll** which is the further expanded through App-V parse items (or intermediate value mappings) to physical path of **C:\Windows\System32\file.dll**), technically same source file or directory could appear in multiple target locations. This can sometimes happen when App-V Sequencer records copy of some temporary file (such as application source MSI from temporary location to %CSIDL_WINDOWS%\Installer), resulting file mapped to multiple places. Native App-V tools won't show this in VFS mappings, but Application Virtualization Explorer is able to tell and differentiate these multiple targets in the dropdown list.

You can edit VFS mapping list by pressing *Edi mappings* –link which will open VFS Mapping Editor.

Virtualize this mapping as deleted

Enables or disables logical deletion for the target path.

Enabling this attribute will cause directory or file appear as deleted (effectively not appearing at all, masking also directory or file with the same path). Deletion mapping attribute is specific to the target path and so if multiple target paths exist for the same virtual file/directory, some of them could be logically deleted while others are not.

This option is mapping –specific and can also be set in the VFS Mapping Editor.

Fully virtualized folder (override)

Set a virtualization level for the VFS enabled directory.

VFS mapped directory can be virtualized in either fully to the target file system or merged with the contents of the target. If directory is fully virtualized, application running on the client will only ever see contents coming from the package's VFS configuration for the directory (i.e. subdirectories and files in it) and possible additions done at runtime and subsequently cached in the PKG file(s). It is advisable NOT to make any common Windows directories mapped as fully virtualized as this may prevent virtual application from functioning properly.

If directory is mapped as merged, client will see both contents in the local, equivalent, directory and entries added through VFS. Problem with the merged directories is their dependency on the local matching directory, this means that if no such directory exists on the target path (as set in VFS mappings), nothing from the App-V package will be shown unlike with fully virtualized directories. Merged directories also cannot exist as sole entries in the VFS configuration i.e. they must always contain at least one virtual file or fully virtualized directory. This is a limitation of how VFS mapping are physically stored in the App-V package. This implicitly also means that merged directories cannot have independent target mapping information on their own; although Application Virtualization Explorer will allow creation of merged directories with such and such target mapping, this information is lost on save but preserved in the contained – fully virtualized - entries.

This setting is not supported for virtual files, which by nature are always fully virtualized. For virtual folders, this setting affects all mappings for directory.

File is a virtual font

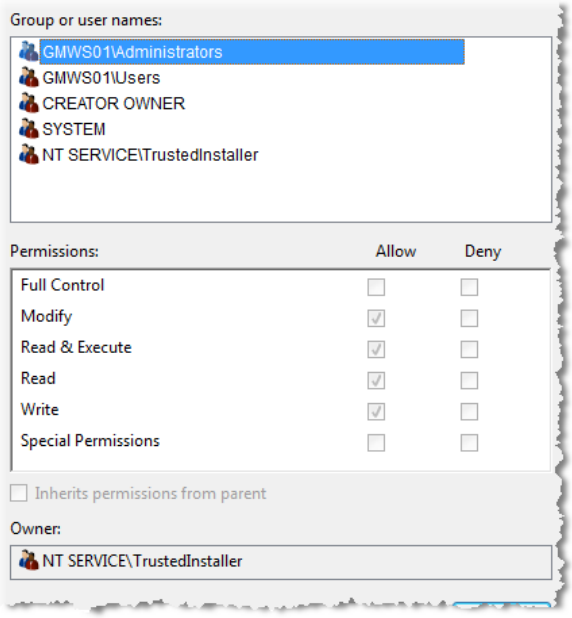
Marks a virtual file as a font file.

This will cause App-V Client to do special handling for the file so that it will appear in the list of available fonts for application querying system fonts. Only files under VFS folder structure and inside CSIDL_FONTS – folder should ever be marked as font files. If the virtual file does not have recognized font file extensions (such as .ttf), Application Virtualization Explorer will warn about enabling virtual font information before allowing flagging the file as such.

Filesystem attributes

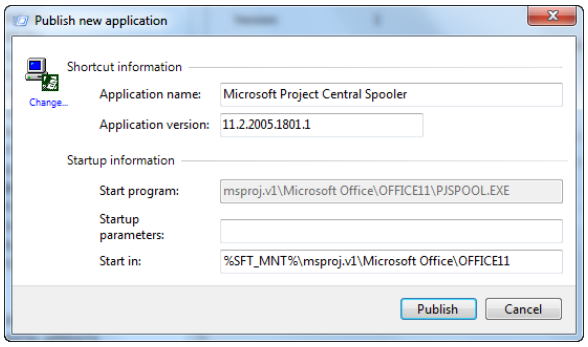
Displays normal file-system related attributes active for the file or directory.

If package was created with App-V 4.5 or newer, or later upgraded to that format, you can see Access Control List entries applied for the directory entry by clicking *View Permissions* –button:



If package has some ACLs present, but file or directory that is selected does not have any applied to it, *View Permissions* -button will be inactive.

Publishing new applications



You can publish any application file (mainly executable files i.e. files with .EXE extension, although technically it does not need to be so) from the internal directory as a new published application by invoking *Publish this file* –action.

This will open a screen wherein base parameters for new published application can be set before application creation is done. After clicking **Publish**, new application will appear under Package configuration view.

If you want to change application icon, you can do so by clicking *Change* –link or change it later on in the Application settings –screen.

Application name and version

Application name and version information is automatically taken from the executable's header.

It is recommended that these two fields are manually double-checked for sensible values, as not all executables carry meaningful information for this purpose and if publishing files other than EXE, such information cannot even be read in at all from the file itself.

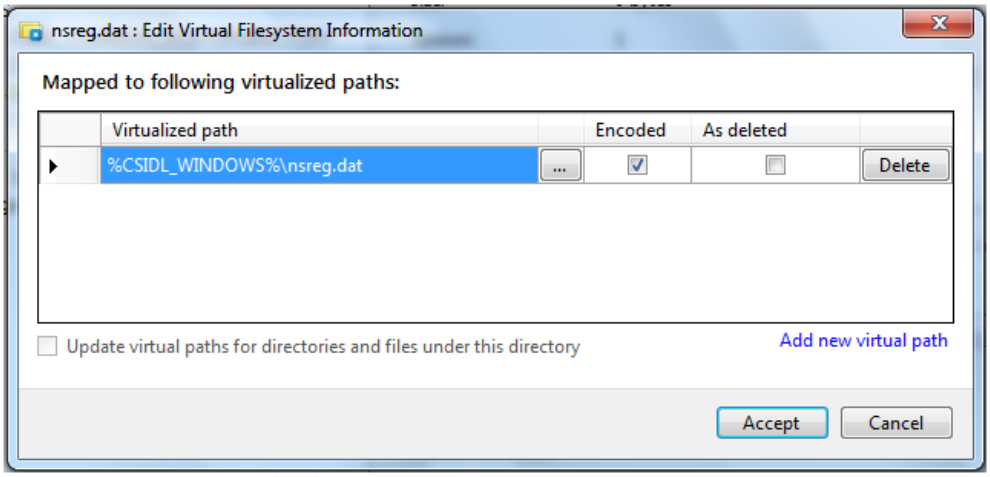
Startup information

Contains parameters and working directory for the published application.

If an application needs any startup parameters, those need to be defined manually in respective field. Working directory is automatically filled to be a directory containing file being published which in most cases should be the correct default.

Virtual Filesystem (VFS) mapping editor

When editing VFS mappings for virtual files and directories, Application Virtualization Explorer contains simple but powerful VFS mapping editor for that purpose.



Mapping editor allows creation of multiple mapping targets (i.e. files or directories virtual directory entry will be made look like for the virtual application) or *virtualized paths*, and each mapping target contains two attributes specific for that target path.

If you want to add new path, simply click *Add new virtual path* –button, which will create new row in the mapping list. You can either write the path manually or use … -button to browse some existing file or directory. If writing manual path, press Enter –key when the path is complete and VFS editor will automatically encode the line if the path is contained inside well-known Windows directory (e.g. **c:\Users*account*\AppData\Roaming\My Application** becomes **%CSIDL_APPDATA%\Firefox**), making it machine and user neutral for purpose of successful virtualization.

Encoded column will be checked automatically if path defined is in encoded form, but you can deselect Encoding check box and press Enter to transform path back to decoded one, which is how the path will then be saved in VFS configuration. This is very unusual need, as generally well-known paths need to be in encoded form as to not cause problems in the client.

If path needs to appear in logically deleted state, that can be controlled via *As deleted* -checkbox. Same attribute is also settable in the main Files –view using *Virtualize as deleted* –setting.

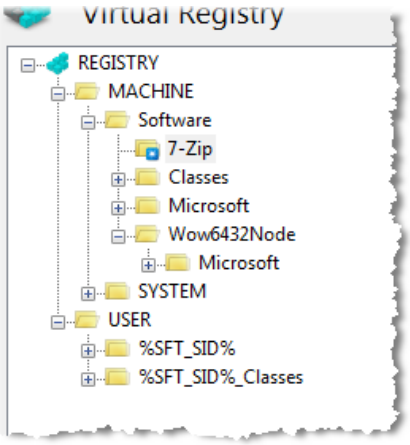
If entry being edited is virtual directory, by default Application Virtualization Explorer will modify all mapping paths under that directory to match changed virtualized path unless *Update virtual paths for directories and files under this directory* –setting is turned off. It is recommended to leave the setting on, as not modifying mapping information for sub-directories and files can have unintended side-effects when package is run at the client.

When paths are as required, VFS mapping information for the directory entry can be saved with *Accept changes* –button and to return main display.

Note that for merged folders paths more than one cannot be added, deletion attribute is not available and path updating is not available.

Virtual Registry view

Virtual registry keys



On a left-hand side of the screen is a tree view representing the virtual registry tree stored in the package's virtual environment, with registry root as a topmost node.

Registry root holds two special keys, MACHINE and USER, which represents HKEY_LOCAL_MACHINE and HKEY_USERS branches, respectively.

It is recommended not to delete either MACHINE or USER keys and Application Virtualization Explorer prevents renaming of REGISTRY, MACHINE or USER nodes.

Special icons for the registry keys are:


A key which is virtualized as overwritten in the registry.


A key which is virtualized as logically deleted in the registry. This key cannot hold any sub-keys or values.

View options

When in the Virtual Registry view, the View menu contains following options:

- **Resolve IM values in key names**
  Displays decoded mapped values in registry key names, instead of displaying them in encoded form.

- **Resolve IM values in value names**
  Displays decoded mapped values in registry value names, instead of displaying them in encoded form.

- **Resolve IM values in value data**
  Displays decoded mapped values in registry value data (in case of string –based registry values), instead of displaying them in encoded form.

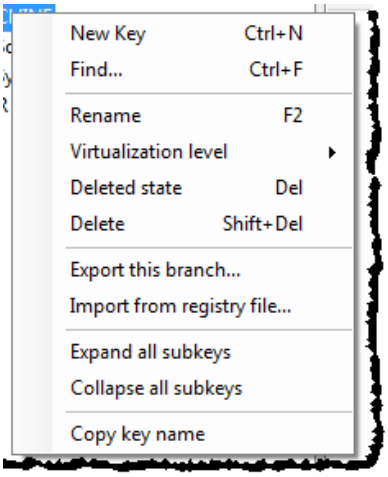
- **Show key/value attributes**
  Displays information about to which virtual registries registry keys and values belong to.

  Each virtual registry key and value can internally be part of the three different virtual registries:

main registry, user-specific registry and system registry (Sec9). Enabling this option marks each key and value with **U** or **S** or both, designating user registry and system registry respectively. This can be useful information in knowing what parts of the virtual registry will be visible for system processes, like virtual services. Normal user processes will always see all entries of main registry which contains all entries regardless of the designation.

Available actions for registry keys

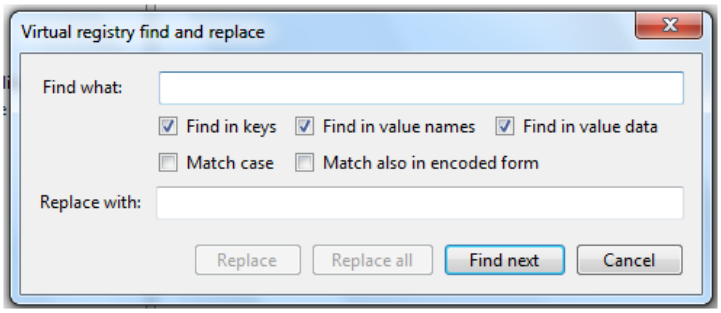
Each registry key supports subset of the following actions; accessible either from mouse context-menu, keyboard command or through **Action** –menu:



- **New Key**
  Create a new virtual registry key beneath a selected one.

  This action is not available for root REGISTRY key.

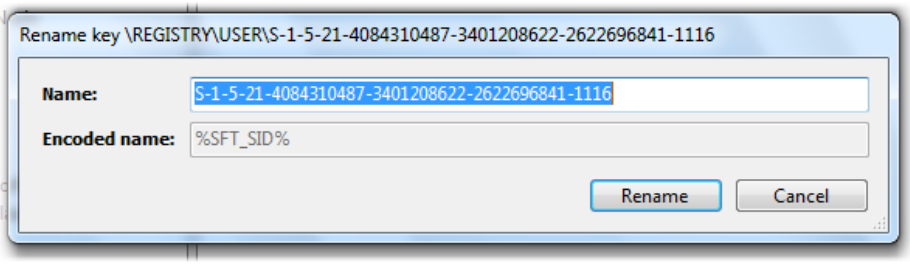
- **Find**
  Opens a registry find and replace dialog:



  Find and replace dialog allows searching key, values and value data, and optionally making replacements.
  You can set additional options in the dialog to limit search only to certain items and to include encoded key or value name or data.

- **Rename**
  Change the name of the selected virtual registry key.

  Invoking this action displays key renaming screen:

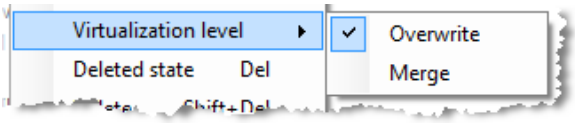


This screen will allow changing of registry key name. Renaming screen will also display name as it would be in encoded format, which is how virtual registry will store the name internally and which enables machine and user account independent operation for registry entries.

Please note that App-V does not currently support encoded key names inside other encoded names, and thus Application Virtualization Explorer will not encode any key names – even if containing data that would otherwise be encoded – if any of the keys in the path already has encoded values stored within.

This action is not available for root REGISTRY key.

- **Virtualization level - Overwrite**
  Changes virtualization level for the registry key to overwritten (fully virtualized).



Any keys with overwrite –status will mask all keys in the same path in local – physical – registry. Only keys that belong solely to the virtual application itself should be marked as fully virtualized as masking other parts of the registry may cause unintended side-effects when application is run at the client.

This action is not available for root REGISTRY key.

- **Virtualization level – Merge**
  Changes virtualization level for the registry key to merged (partly virtualized).

Any keys with merge –status will mask only those virtual keys inside it that has overwrite attribute set and all virtual values stored in or under it.

This action is not available for root REGISTRY key.

- **Deleted state**
  Sets or removes logically deleted state for the virtual registry key.

Any keys set to logically deleted has the effect of fully virtualized state in that it will mask all equivalent local keys and everything underneath it in the local registry, but the key itself won't show up either, causing virtual application to consider key as being non-existent.

Changing virtual registry key to deleted state in Application Virtualization Explorer will remove all sub keys and values stored in it, as logically deleted key cannot hold anything inside it. This removal cannot be reversed and so caution must be observed before setting key as deleted.

This action is not available for root REGISTRY key.

- **Delete**
  Physically deletes specified virtual registry key.

  Deleting key (as opposed to setting it logically deleted) will remove virtual registry key and all keys and values beneath it physically from virtual environment configuration.

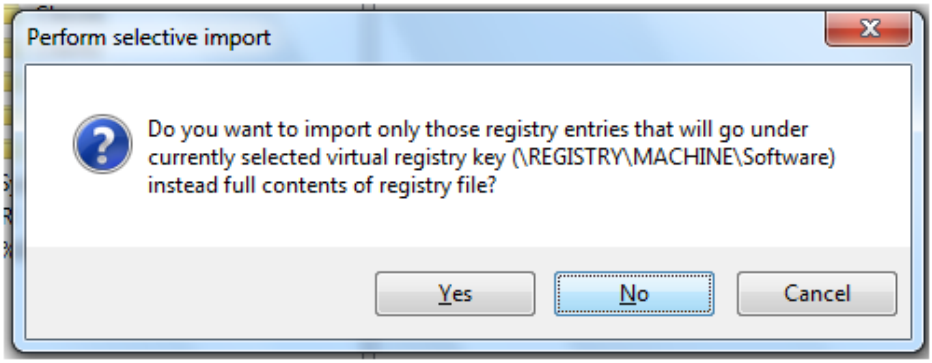
  This action is not available for root REGISTRY key.

- **Export this branch**
  Exports contents of the specified key and everything beneath it.

  Exporting a virtual registry key will create a Windows registry editor compatible .REG file holding exported contents.

- **Import from registry file**
  Performs registry key and value import from Windows Registry Editor file.

  When selecting non-root node for importation, Application Virtualization Explorer will ask if the import will only read in those registry entries that would normally go under specified key. This limitation of scope is convenient in the situations wherein the registry file contains large number of entries and only entries going under selected branch are necessary.



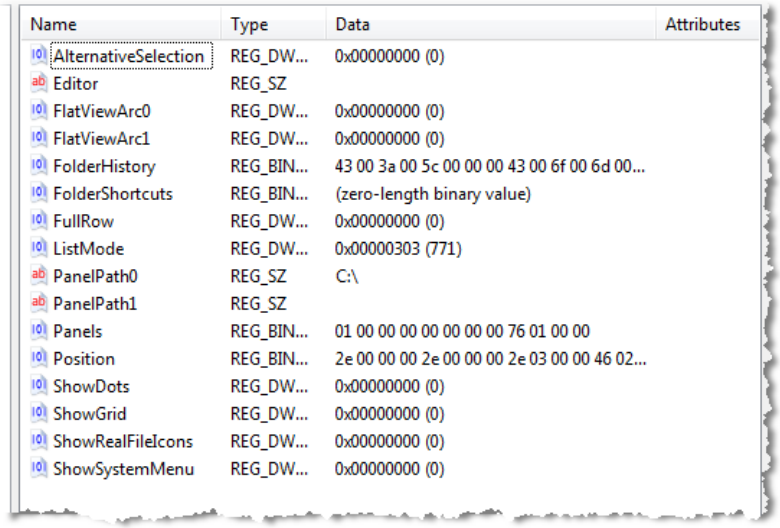
- **Expand all subkeys**
  Expands selected virtual registry key and all sub keys in the tree view.

- **Collapse all subkeys**
  Closes selected virtual registry key and all sub keys in the tree view.

- **Copy key name**
  Copies path to a selected registry key into Windows clipboard.

Virtual registry values

After selecting virtual registry key from the tree view, right-hand side of the virtual registry view is filled with virtual registry values associated with the key.



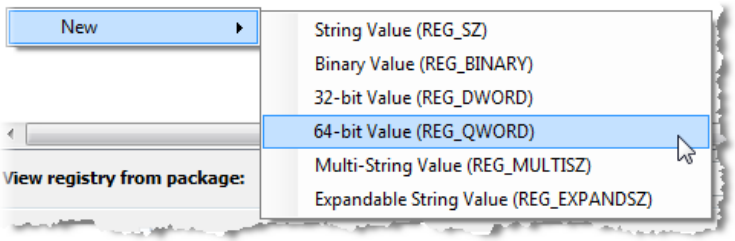
Name	Type	Data	Attributes
AlternativeSelection	REG_DW...	0x00000000 (0)	
Editor	REG_SZ		
FlatViewArc0	REG_DW...	0x00000000 (0)	
FlatViewArc1	REG_DW...	0x00000000 (0)	
FolderHistory	REG_BIN...	43 00 3a 00 5c 00 00 00 43 00 6f 00 6d 00...	
FolderShortcuts	REG_BIN...	(zero-length binary value)	
FullRow	REG_DW...	0x00000000 (0)	
ListMode	REG_DW...	0x00000303 (771)	
PanelPath0	REG_SZ	C:\	
PanelPath1	REG_SZ		
Panels	REG_BIN...	01 00 00 00 00 00 00 00 76 01 00 00	
Position	REG_BIN...	2e 00 00 00 2e 00 00 00 2e 03 00 00 46 02...	
ShowDots	REG_DW...	0x00000000 (0)	
ShowGrid	REG_DW...	0x00000000 (0)	
ShowRealFileIcons	REG_DW...	0x00000000 (0)	
ShowSystemMenu	REG_DW...	0x00000000 (0)	

Special icons for the registry values are:

A value which is virtualized as logically deleted in the registry.

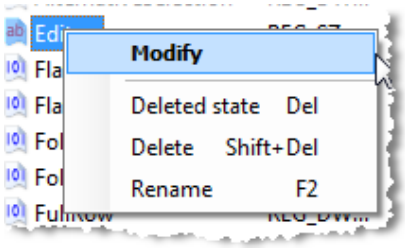
Available actions for registry values

When no values are selected in the value list, following actions are accessible either from mouse context-menu or through **Action** –menu:



- **New – String Value**
  Create a new REG_SZ type registry value (simple string).
- **New – Binary Value**
  Create a new REG_BINARY type registry value (binary data).
- **New – 32-bit Value**
  Create a new REG_DWORD type registry value (32-bit integer/double word).
- **New – 64bit Value**
  Create a new REG_QWORD type registry value (64-bit integer/quadruple word).
- **New – Multi-String Value**
  Create a new REG_MULTISZ type registry value (multi-string).
- **New – Expandable String Value**
  Create a new REG_EXPANDSZ type registry value (expandable string).

Each registry value – when selected - supports following actions; accessible either from mouse context-menu, keyboard command or through **Action** –menu:



- **Modify**
  Opens registry value data for editing.

- **Deleted state**
  Sets or removes logically deleted state for the virtual registry value.

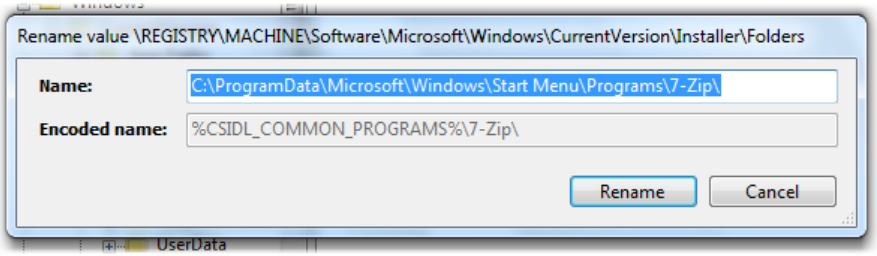
  Any value set to logically deleted has the effect of making virtual application to consider value as being non-existent, even if local equivalent value exists in the local registry (and containing virtual key is set to merged).

  Changing virtual registry value to deleted state in Application Virtualization Explorer will remove value data, as logically deleted value cannot hold anything inside it. This removal cannot be reversed and so caution must be observed before setting value as deleted.

- **Delete**
  Physically deletes specified virtual registry value from containing key.

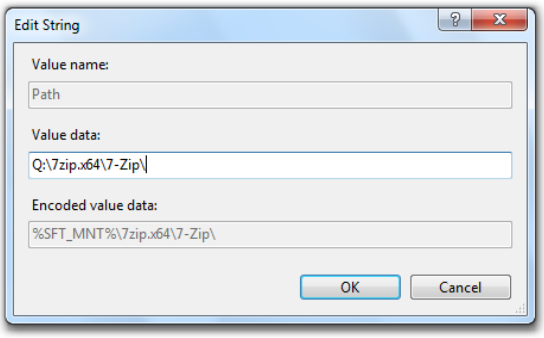
- **Rename**
  Change the name of the selected virtual registry value.

  Invoking this action displays value renaming screen:



  This screen will allow changing of registry value name. Renaming screen will also display name as it would be in encoded format, which is how virtual registry will store the name internally and which enables machine and user account independent operation for registry entries.

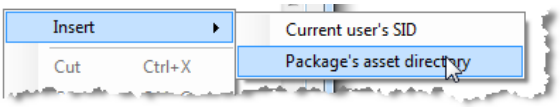
Editing simple and multi-string values



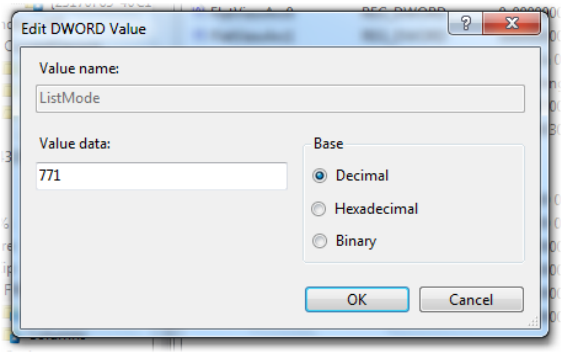
When editing simple string data (REG_SZ, REG_EXPANDSZ and REG_MULTISZ), string editor screen will automatically convert text entered in *Value Data* –field into encoded format, which is how the data is physically stored inside virtual environment.

This means that you need to enter e.g. well-known paths how they would appear on the local machine, such as **C:\Windows** or **C:\ProgramData\Application**.

In order to assist in entering more hard to come-by values, Application Virtualization Explorer can fill-in those special values when selecting appropriate entry from the context-menu (accessible through alternative mouse click):

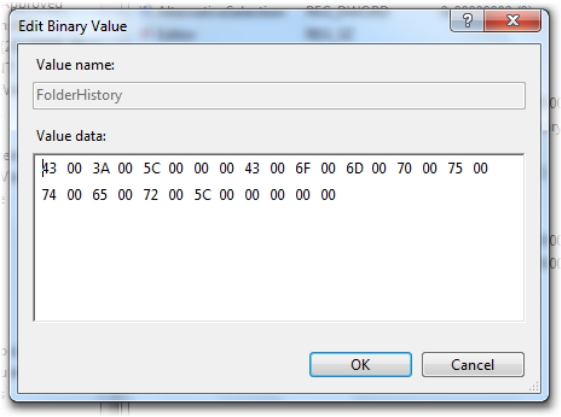


Editing integer values



When editing integer – or numerical - values (REG_DWORD and REG_QWORD), you can switch between different number-bases using *Decimal*, *Hexadecimal* and *Binary* –selections on the edit screen.

Editing binary values

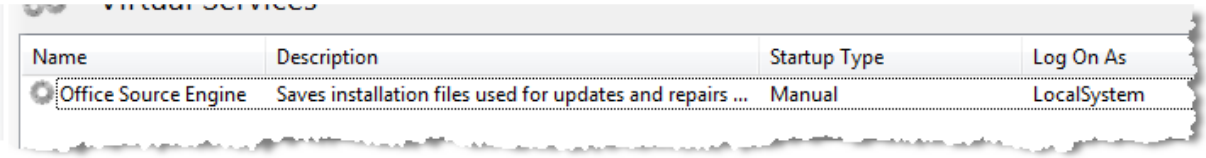


Editing binary values (REG_BINARY, or any of the unsupported registry types) allows adding, modification and deleting of individual bytes.

Because of internal limitation of the App-V's virtual registry implementation, Application Virtualization Explorer is also only able to support up to 512 kilobytes of binary data per one registry value.

Virtual Services view

Virtual Service view displays all virtualized services contained in the package.

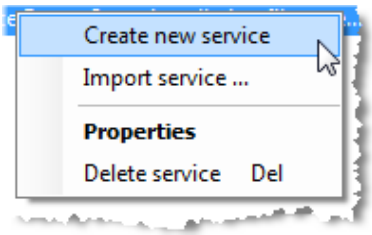


Using this screen, virtual services can be added, edited and removed. Virtual services do not support some of the parameters available for normal service, like specifying an account under which the virtual service is executed but rather all virtual services are run in LocalSystem context.

Virtual services are not very common in App-V packages and need specific type of executable that can be run under Windows' service control manager (virtualized or not). This means that any arbitrary executable from the package cannot and should not be added as new virtual service.

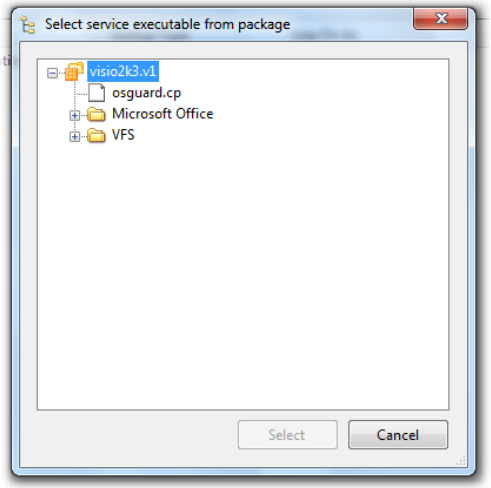
Available actions for virtual services

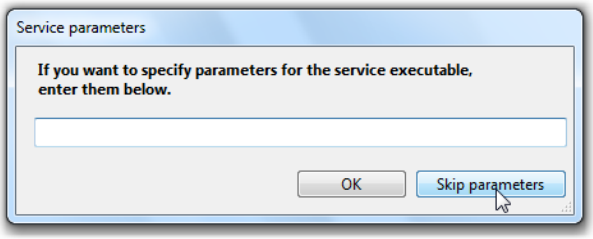
Virtual services supports following actions; accessible either from mouse context-menu, keyboard command or through **Action** –menu:



- **Create new service**
  Create a new virtual service using specified parameters.

  When this action is invoked, Application Virtualization Explorer will display file selector which can be used to select service executable file from within the package's internal directory.





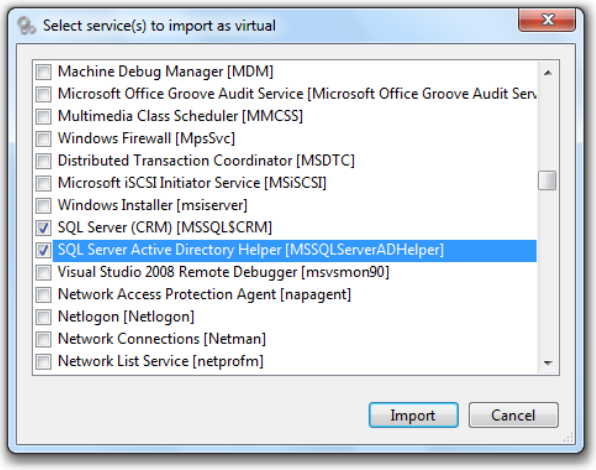
If selected service executable needs any startup parameters they can be entered or can be skipped using *Skip parameters* –button in parameter query screen.

This will create new virtual service to the services list, which can then be opened for modification

or be renamed.

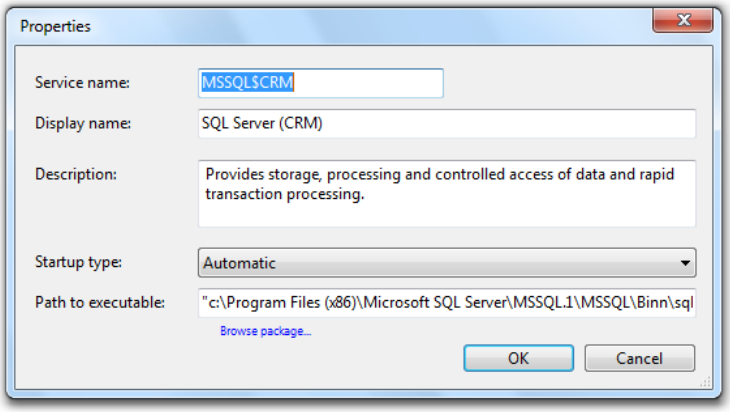
- **Import service**
  Imports virtual service(s) based on locally present services.

  Import service functionality can be used to create new virtual service, or multiple of services, based on physical services available on the local machine.



  Invoking this command will display list of available services in the screen from which one or more service can be selected. Application Virtualization Explorer will read in service parameters based on selection(s) and create equivalent virtual services.

- **Properties**
  Shows service properties for editing.



  From the properties screen, basic names and options for services can be changed, but no advanced options normally found in Windows service properties are available as virtual services do not support complex services. Usually only *Startup type* needs to be altered to Manual or Disabled for unnecessary services that cause long startup delay for virtual application.

  If service executable needs alteration, you can use *Browse package* –link to show package directory

browser for easy executable picking.

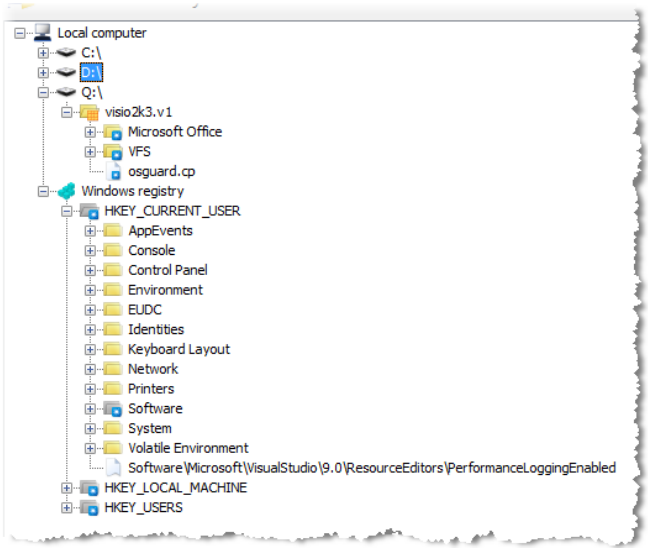
Note that this action is only available when only one virtual service has been selected from the list.

- **Delete service**
  Deletes virtual service(s).

  Note that this action is only available when one or more virtual services has been selected from the list.

Virtualized System View

Application Virtualization Explorer includes a unique feature for App-V package simulation so that you can see how the open package would be look like when deployed to the local machine.



Virtualized system view displays all locally present drives and registry In a tree view, combined with all the entries defined in the Virtual Files & Folders view and Virtual Registry. This will give a idea how the virtual application that is run from the package would see its surroundings; especially important when fully virtualized VFS directories are defined or registry keys that overwrite locally present ones.

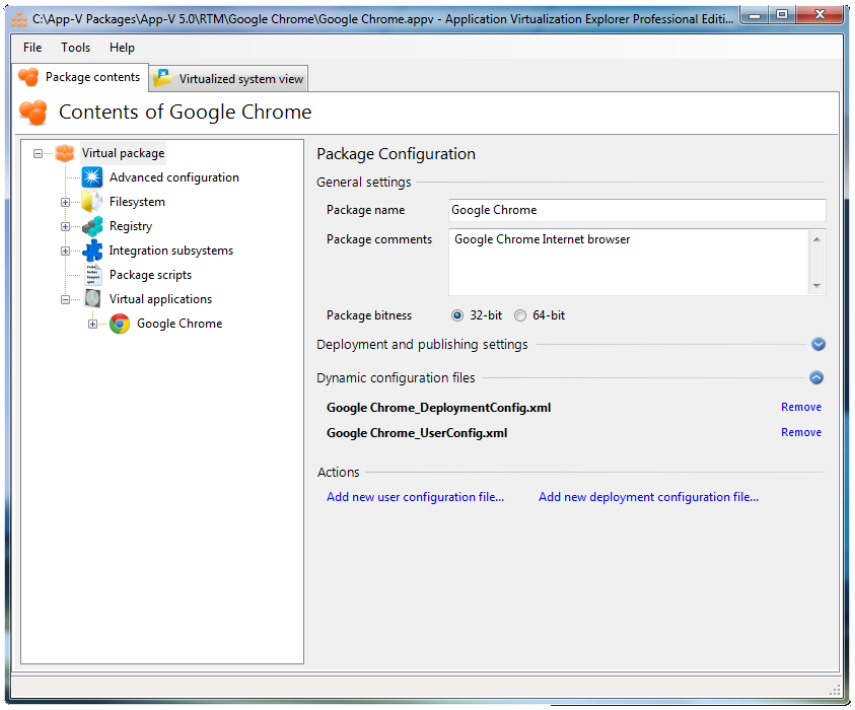
When selecting any entry from the tree, you can use mouse context menu, **Action** menu or keyboard shortcut to jump into that entry in other screens, or in the case of physical files, in the real system. This enables easy way to see from where each entry is gathered to the virtualized system view.

If local machine has App-V Client installed, you can also browse to the virtual drive and see current package's directory mounted to the root of it. Note that directories and files shown under virtual drive are not coming from App-V Client itself, but merely simulation based on the opened package just like any other virtualized entry in it.

Package screen in AVE Express, AVE and AVE Pro

Package contents view

When you open up an App-V package, Package Package contents screen will show all of the content in one unified content tree.



Different main level nodes correspond to various parts of the package, such as the internal filesystem, registry, application integration subsystems and scripting; described below.

You can also open additional views of selected parts of the content tree by right-clicking the node you want to open, and then choosing “Open in new tab”. This way you will be able to examine the package’s content in parallel views and slice the content the way you want, such as concentrating on one specific registry key in one tab.

Virtual package level

Virtual package level allows you to change main package settings, such as the name of the package and operating system visibility.

General settings

Package name

Overall name for the package.

Package comments

Free-form comments for the package.

Package bitness

Allows changing of existing package’s internal bitness setting.

Please note that already existing package to a different bitness may cause unintended effects on the client, or render the package invalid, and as such the resulting package will be unsupported by Gridmetric and/or Microsoft.

Deployment and publishing settings

Operating system visibility

Controls on which operating systems the resulting package will be shown. For any OS not listed, App-V Client will refuse to publishing the package.

Dynamic configuration files

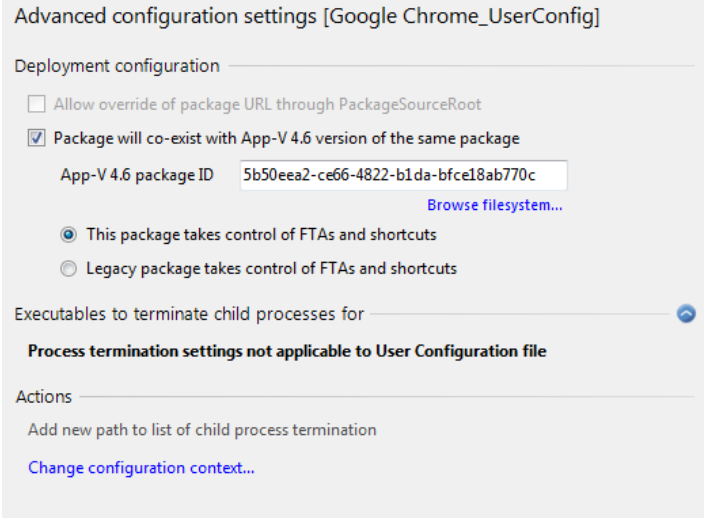
Lists all loaded dynamic user and deployment configuration files associated with the package. You can remove one or more of the configuration files from the package configuration by using **Remove** link.

Actions

Actions in the package configuration screen allows you to add new User or Deployment configuration files to the package configuration.

Advanced configuration settings level

Advanced configuration settings level allows you to make advanced settings changes for the package. All of these settings only apply to the configuration files context of the package, and you cannot change internal configuration using these settings.



Deployment configuration settings

Allow override of package URL through PackageSourceRoot

If enabled, App-V Client's PackageSourceRoot registry setting can override this package's streaming URL.

Package will co-exist with App-V 4.6 version of the same package

If enabled, there is a legacy App-V package on the same client side-by-side, which has the same (i.e. conflicting) file-type association and/or shortcuts as this package. When enabling this setting, you need to define the package ID for the legacy App-V package in order for 5.0 Client to integrate with 4.6 Client properly for package co-existence.

You can also control which one of the package will take the responsibility for FTAs and shortcuts (if conflicting).

Executables to terminate child processes for

Lists all the executables that, if running, will be terminated by App-V Client upon package's virtual environment unload.

Actions

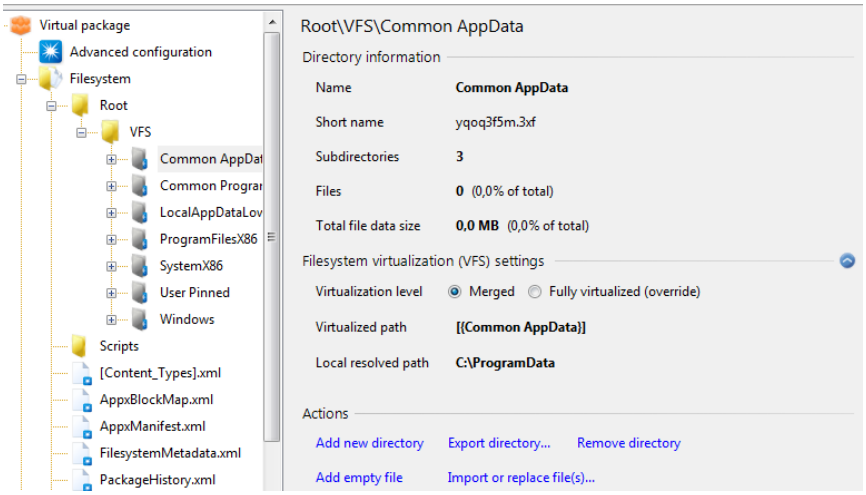
Actions in the advanced configuration settings screen allow you to add new process termination paths or change currently active configuration context.

Filesystem level

Filesystem level allows you to access package's internal file structure. App-V 5.0 packages will have all user-defined content under directory called **Root** in the root of the filesystem itself. **Root** directory will be mapped to package's primary virtual application directory (PVAD) path when the package is executing on the client.

For carrying scripting in the manner that App-V Sequencer exposes it, Application Virtualization Explorer allows creation of scripts –folder underneath the root of the filesystem using "Add script directory" functionality from the Filesystem –node directory. This directory can then be referenced in the script settings using path of "[{AppvPackageRoot}]\..\scripts\".

Additional files under filesystem root are App-V package's internal configuration files, and Application Virtualization Explorer does not enable changing of those files as they are automatically generated upon package save.



Underneath **Root** directory, special directory called **VFS** exists. This directory contains well-known system directories that will be mapped using Virtual Filesystem (VFS) technology on top of the existing filesystem tree on a client.

From the main Filesystem –level node, you can examine statistics of the package's internal filesystem and see what the current PVAD for the package is. Additionally, you can add special Scripts –directory to the root of the filesystem, which can be used to store package scripts.

Drilling down to the actual directories and files, you can see the basic file attributes for the files, along with the VFS mapping information for those files and directories residing under VFS structure.

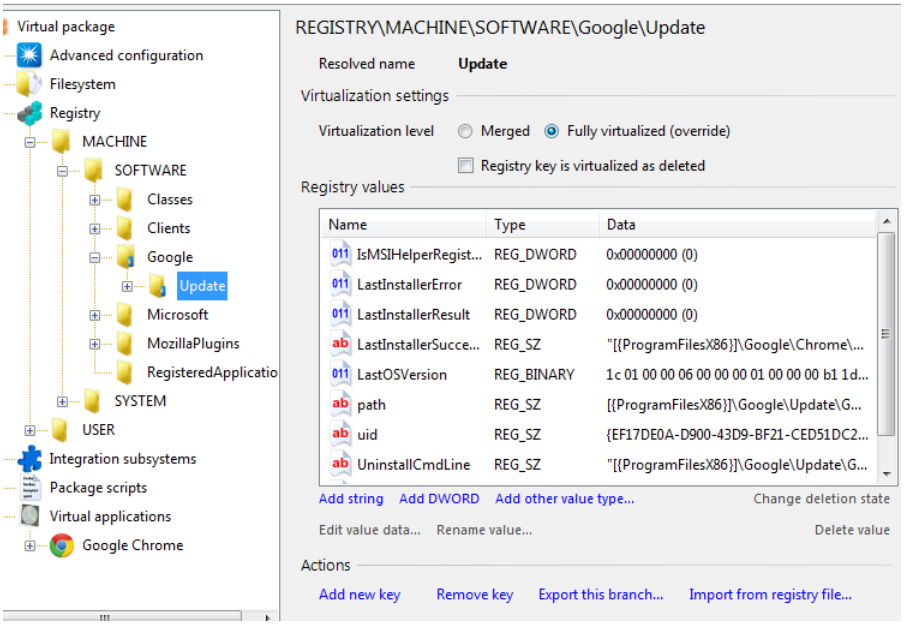
Actions

Actions at the directory level allow you to add and remove new directories and files, import and export content for the files and import and export directories.

For virtualized executable files, you can additionally publish them as new virtual applications.

Registry level

Registry level allows you to access package's virtual registry.



From the main Registry –level node, you can examine statistics of the package's registry and export or import registry files regarding the whole virtual registry's contents.

Drilling down to the actual registry keys, you can see the virtualization level settings and virtualization attributes for keys, along with the list of registry values associated with the key.

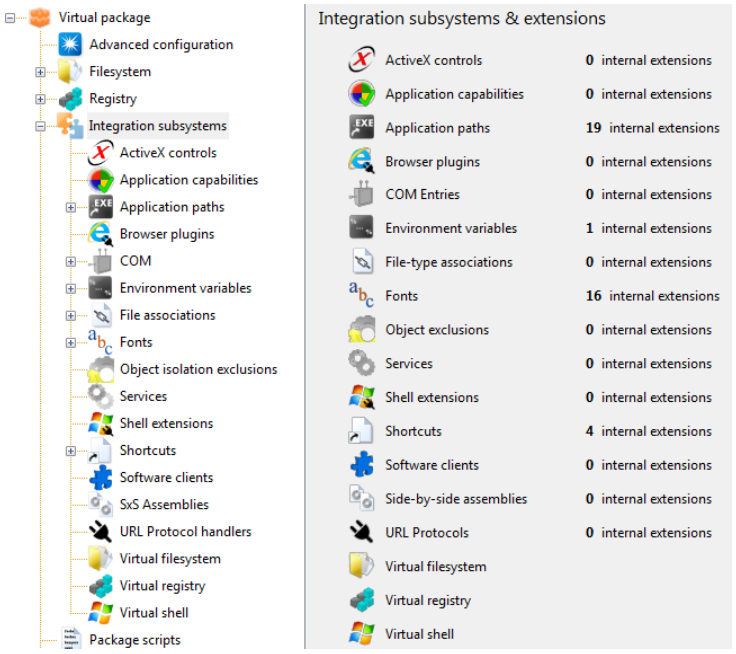
For registry values, you can create new values, edit, rename and delete existing ones and change the logical deletion state (i.e. value appear as deleted masking local value by the same name).

Actions

Actions at the registry key level allow you to add and remove new keys, and import and export content for the selected registry branch.

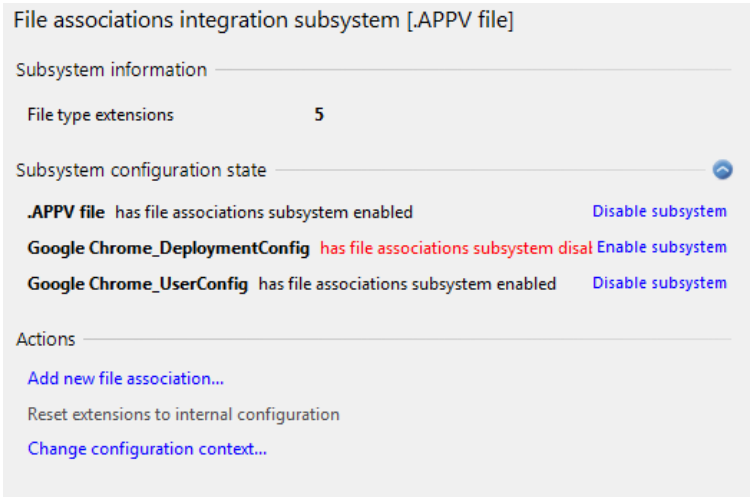
Integration subsystems level

Integration subsystems level allows you to add, change and configure various application integration subsystems and related application extensions supported by App-V 5.0 packages.



Integration subsystems, and associated application extensions (if supported by the subsystem) are all displayed in the Application Virtualization Explorer based on the configuration context selected.
This allows you to make customizations to a specific context (whether internal APPV file context or some dynamic configuration file).

Note that not all of the integration subsystems necessarily support all the functionality in all contexts, but you can individually enable and disable the whole subsystem in specific configuration context using the integration subsystem's main level.



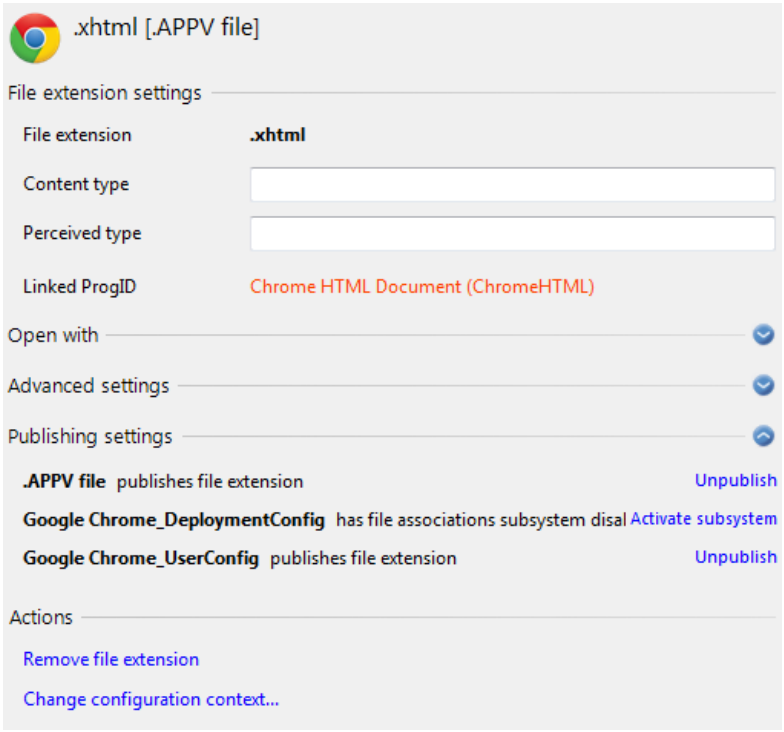
Depending on subsystem, you can also add new application extensions to a specific context. Please note that Application Virtualization Explorer does not currently support adding or editing of all types of application extensions supported by App-V 5.0 packages.

In case of adding new extension to the internal configuration (i.e. to the APPV file directly), the new extension will automatically be copied to the all loaded configuration files as well.

If you edit application extensions, edits made while in internal configuration context will be reflected to all other configuration contexts that have the same extension, while changes done to an application extension in context of configuration file will only be visible to that context.
This allows you to customize configuration files without reflecting changes into any other configuration file, but still having possibility to make changes globally to all configuration contexts.

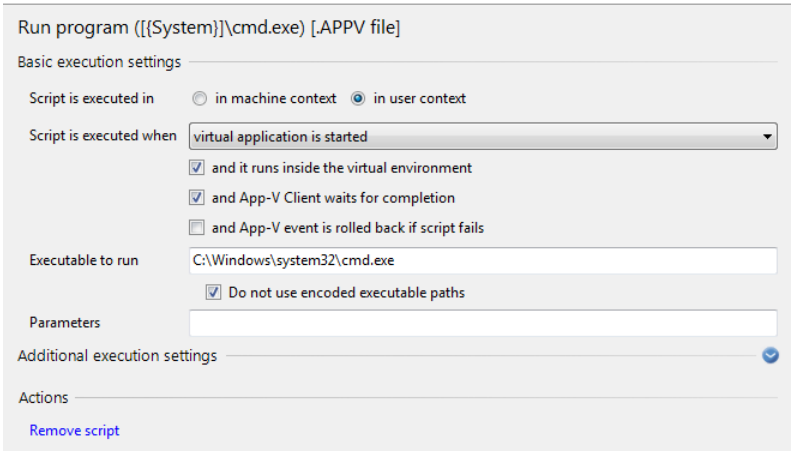
While having a specific extension selected, you can also add or remove that same extension across all configuration contexts by using the Publishing settings for the extension.

Publishing the extension to another configuration context simply means adding the extension, and unpublishing it means removing the extension. In the extension configuration screen, you can see in which contexts the currently selected extension is active by looking at the list under Publishing settings – section.



Package scripts level

Package scripts allows you to add, edit and remove App-V package scripts either inside the APPV file or inside a specific configuration file.



While having the script selected, you can change various scripting parameters.

App-V 5.0 package scripts are run either in the machine or user context, and depending on the selected context, not all of the timing triggers are possible to select.

Furthermore, only when virtual environment or application is started it is possible to run anything inside the virtual environment.

For selected timing triggers, under **Additional execution settings,** it is possible to specify virtual application published from the package that the script relates to and App-V Client won't execute the script unless it is that specific application that is being started or stopped.

Please note: due to technical restrictions of scripting in App-V 5, only one kind of each execution trigger can be had in one individual package (i.e. one script for package publishing, one script for virtual environment start etc.)!

Please note: due to technical restrictions of scripting in App-V 5, user-context scripts will only be executed for domain accounts and not for local accounts!

Actions

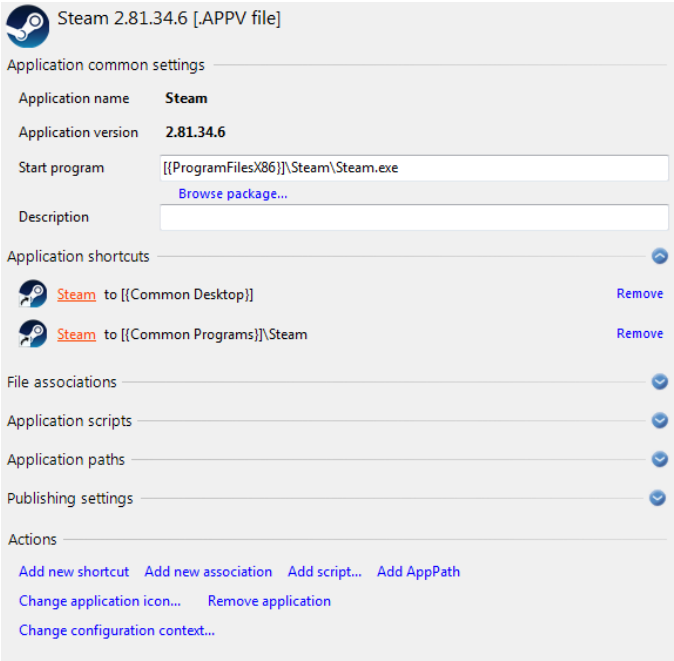
Actions at the package scripts level allows you to add new internal scripts or external (configuration file – based) scripts.

Actions at the script level allow you to remove script from the package.

Virtual applications level

Virtual application level lists all the virtual applications that are published from the package. Selecting an individual application allows you to change its settings.

For virtual applications, the ability to change some settings depends on the configuration context in use. For example, the executable being started can only be set in the package's internal context, while the display name for the application can be customized in configuration file context or in internal context both.



To rename an application, or its version, click on the name and version fields directly to enable editing.

Unlike for most application extensions, configuration files cannot define new applications or remove existing virtual applications directly. But you can enable and disable pre-existing virtual applications defined in the internal context using configuration file context.

Viewing individual virtual application also lists all those application extensions which are directly linked with the application in the currently selected configuration context. For those extensions, you can remove extensions directly from the virtual application view and this has the same effect as going into extension itself, and removing it from there.

Actions

Actions at the application level allow you to change application's icon, add new shortcuts, file-extensions, AppPath -settings and scripts for the application.

You can also remove virtual application and change the currently active configuration context.

Other functionality in AVE and AVE Pro

Processing automation

Using processing automation, you can execute pre-defined processing automation files against currently opened package in Application Virtualization Explorer.

These automation files (written in plain-text) can make content modifications to the package, allowing to for example cleanup of known artifacts, adding corporate-standard new content for each package etc. Please see the example processing automation files supplied with your installation for syntax of the files.

You can access processing automation from Tools –menu, using **Run processing automation** –command.

Expanding package to local system

Just like the Sequencer, Application Virtualization Explorer is able to expand currently opened package to the local system. Expanding in this context means placing all virtual registry and virtual filesystem contents to the locations they would appear in when virtual package is executing on the client. This allows you to extract pre-requisites packages to the packaging machine as if they were originally created on that machine.

Unlike the Sequencer, with Application Virtualization Explorer's package expansion you have some degree of control over how the resource expansion is handled. An example of this is allowing filtering of target directories and registry branches so that any content destined to those location will be skipped during extraction, allowing you to extract content selectively, or for electing to translate 32-bit registry entries to Wow64 branch of the registry if expanding 32-bit packages on 64-bit system.

You can access processing automation from Tools –menu, using **Expand package to local system** –command.

Please note that package expansion feature at this time does not re-create application shortcuts to the system (unless they have been captured directly as .lnk files inside the package). This feature also does NOT guarantee that the expanded application(s) will run successfully after expansion; there is no attempt of trying to do actual virtualization that running the package under App-V Client would perform to ensure portability.

Warning: Expanding package content without any target path filtering, or using options to clear out target directories/registry keys if Overwrite –attribute is set for virtual resources, can render the local system in unexpected state if the App-V package carries extra content that is related to Windows' and/or its components operation!

Automatic App-V package creation for AVE

From the Help –menu, you can automatically generate App-V package for the Application Virtualization Explorer itself.

This package has special configuration applies to it to ensure AVE can run inside virtual environment, namely the virtual registry is disabled by the policy so you do need to use either Dynamic Deployment or User Configuration files supplied.

Please note that the resulting package will not have any licensing information embedded and you will need to normally enter license key for AVE on each individual machine the package is run on.

Plugin functionality in AVE Pro Classic

Application Virtualization Explorer ships with additional functionality that is implemented as plugins and can be installed or left out from installation at will.

At the time of this release, following plugins ship with Application Virtualization Explorer:

Dynamic Suiting Code Snipping Tool

This plugin adds functionality into Tools –menu, which enables easy copying of CODEBASE –element needed when package is linked using App-V's Dynamic Suiting (DSC).

Virtual object exclusion editor

This plugin adds functionality into Tools –menu, which enables editing of virtual object exclusion entries stored directly inside the package.

Local import of package

This plugin adds functionality into File –menu, which enables local importation of the opened App-V package into locally installed App-V Client. Please note that the setting for App-V Client must allow addition of new virtual applications and import operation of the SFT file, at minimum, as well as have virtual application usage without central authorization enabled.

Viewer for changed and added virtual files

This plugin adds functionality into Tools –menu, which enables examination of all virtual files last added or changed during specific version of the package save history.

Resources on the web

This plugin adds additional entries into Help –menu collecting popular Microsoft –official and community – based resources (forums, documentation, blogs and tools) which can be launched directly from the Application Virtualization Explorer.

Variable viewer

This plugin adds variable viewer into Tools –menu, which enables inspection of intermediate value mappings as currently in effect for Application Virtualization Explorer on the present machine (e.g. %SFT_SID% translating to current user's SID).

Package version comment log viewer

This plugin adds version comment log viewer into Tools –menu, which enables review of all the comments made during package saving from Application Virtualization Explorer. Note that in order to package to include any save version comments, you must enable saving of extended history –option in Application Virtualization Explorer as well as supplying version comments during the package save operation.

Local expansion of the package

This plugin adds functionality into Tools –menu, which enables expansion of the currently loaded package to the local machine. Package expand –functionality is similar to that which can be found from the new App-V Sequencer versions.

The plugin will extract all virtual files and virtual registry entries to the local system, in the respective locations like they would appear if run virtualized. While the expansion plugin will try to map all location and other references correctly for the local machine during expansion process, it should be noted that expansion's end result does not necessarily run (at all) or appear correctly installed in all possible scenarios.

Please note that is **not** recommended to expand packages on primary production machines, expanded entries can overwrite important system files or registry entries if the virtual application package contains any such resources and they are not locked by operating system or other applications.

For best possible expansion outcome, please run Application Virtualization Explorer as an administrator to allow expansion process to have full control to various file system and registry locations.

Administrative templates

Application Virtualization Explorer ships with administrative templates (ADM and AMDX versions) that can be installed or left out from installation at will.

These administrative templates can be used with Active Directory's Group Policies to centrally control overall options in Application Virtualization Explorer as well as disable functionality, such as saving the packages.

Both traditional ADM –style template and newer ADMX/ADML –style templates contains true managed policy settings which Application Virtualization Explorer will observe over possible settings in the user's registry. Both machine and user –based targeting is supported, machine policies taking precedence over user policies.

Troubleshooting

Technical support and product upgrades

Application Virtualization Explorer Professional license price includes one year maintenance period, which will start from the day of license purchase.

During this period, you will have access to email based technical support at support@gridmetric.com for any questions or issues you may encounter for your product usage.

As a licensed user, you have access to Gridmetric Customer Portal at https://secure.gridmetric.com/ . This portal is the distribution point for possible product upgrades, which you can freely access during maintenance period. After the period has ended, you will continue to have access to most recent version of Application Virtualization Explorer released at the time maintenance expired.

Appendix A - License

Application Virtualization Explorer

SOFTWARE LICENSE AGREEMENT

Application Virtualization Explorer Express, Application Virtualization Explorer and Application Virtualization Explorer Professional (collectively hereinafter "SOFTWARE")

This is a legal agreement between the buyer (hereinafter "Licensee") and GridMetric Oy (hereinafter "GridMetric", "Gridmetric" or "Licensor").

1. GRANT OF LICENSE

1.1 GENERAL

The Licensor hereby grants to the Licensee a non-exclusive and non-assignable license subject to the terms and condition hereafter set forth to use the SOFTWARE.

The Licensee acknowledges that the SOFTWARE and thereto related technical information and know how provided hereunder are valuable and proprietary to the Licensor.

The Licensee has no right to use the technical information or know how for any purpose other than purposes permitted under this Agreement.

The Licensee shall have no right to grant any sublicenses in respect of the rights granted under this Agreement without the prior written consent of the Licensor.

The Licensee shall not assign its rights or obligations under this Agreement to a third party or otherwise dispose of or deal with those rights or obligations except to the extent permitted under this Agreement.

SOFTWARE includes a Personal License.

1.2 PERSONAL LICENSE

The Personal License is for one person only and is non-transferable. If the license is purchased by a legal entity, the Personal license must be assigned to one person within that entity. The license allows installing of the SOFTWARE on two computers. The license doesn't allow using the SOFTWARE as a part of an automated service.

1.3 HARDWARE LICENSE

The Hardware License allows installing of the SOFTWARE on two computers. However the license doesn't allow using the SOFTWARE simultaneously on both computers. The license doesn't allow SOFTWARE to be used in a manner wherein concurrent usage of the SOFTWARE is possible by a multiple persons such as, but not limited to, Terminal Server computer. The license doesn't allow using the SOFTWARE as a part of an automated service.

1.4 MULTI-USER HARDWARE LICENSE

The Multi-User Hardware License allows the SOFTWARE to be installed on one computer only and is non-transferable. The license allows concurrent usage of the SOFTWARE in the computer shared by multiple simultaneous user sessions such as, but not limited to, Terminal Server or Remote Desktop Session Host server. The license doesn't allow using the SOFTWARE as a part of an automated service.

1.5 OTHER LICENSES

GridMetric may grant also other types of licenses upon GridMetric's discretion.

2. COPYRIGHT

The SOFTWARE is owned by GridMetric and is proprietary in nature. The SOFTWARE is protected by Finnish copyright law as well as copyright laws of United States and international treaty provisions.

3. OTHER RESTRICTIONS

You may not rent, lease, sublicense, loan, copy, modify, adapt, merge or translate the SOFTWARE unless expressly allowed according to the license acquired. You may not reverse engineer, decompile or disassemble the SOFTWARE unless expressly allowed according to the license acquired.

4. LIMITED WARRANTY

GridMetric may grant a warranty in the SOFTWARE if it is expressly stated so. Otherwise the SOFTWARE has no warranty.

To the maximum extent permitted by applicable law, GridMetric and its distributors exclude express or implied warranties of any kind, including without limitation, merchantability or fitness for a particular purpose with regard to the SOFTWARE and/or the accompanying written materials.

5. LIMITATION OF LIABILITY

In no event shall GridMetric or its distributors be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the product of GridMetric, even if GridMetric has been advised of the possibility of such damages.

GridMetric's cumulative liability to you or any other party for any loss or damages resulting from any claims, demands, or actions arising out of or relating to this software license agreement shall not exceed the license fee paid to GridMetric for the use of the SOFTWARE.

6. CUSTOMER REMEDIES

GridMetric and its distributors' entire liability and your exclusive remedy shall be, at GridMetric's option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet GridMetric's Limited Warranty and that is returned to GridMetric's distributor with a copy of the receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication.

7. TERM

This software license agreement shall be effective until you terminate it by destroying the SOFTWARE and its documentation together with all copies. It shall also terminate if you fail to abide its terms. Upon termination you agree to destroy all copies of the SOFTWARE and its documentation including any SOFTWARE stored on the hard disk of any computer under your control.

8. APPLICABLE LAW

This software license agreement shall be governed by Finnish law and the sole legal venue shall be Helsinki, Finland.

Appendix B – 3rd party licenses

WiX Deployment Tools Foundation

Common Public License Version 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and

b) in the case of each subsequent Contributor:

i) changes to the Program, and

ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HApplication Virtualization Explorer ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.