SFT Encoder

Enterprise & Server Edition

User Guide

V2.1.0 (2010-11-08)

GRID METRIC

http://www.gridmetric.com/

Contents

Overview

SFT Encoder is a multi-purpose Microsoft SoftGrid/App-V package creation and content modification solution, driven from the command-line interface or using powerful processing templates (Enterprise and Server Editions -only). SFT Encoder can be used to create, modify and delete content – files, directories and registry entries – as well as to change associated package metadata such as GUIDs, blocksize, compression, publishing parameters and more.

SFT Encoder is available and licensed in two editions:

- Enterprise Edition, suited for App-V administrators, consultants and packagers doing daily post-processing and content update tasks. Enterprise Edition includes powerful template -based operations for more complex workflow tasks and content modification needs.
- Server Edition, building on the same functionality as Enterprise Edition but from licensing terms aimed at scenarios involving automated processes and service provider –type of usage where non-interactive execution is required.

This documentation provides description of available functionality within SFT Encoder, both with the command-line mode and processing template –format.

System Requirements

- Microsoft Windows XP Service Pack 3, Windows 2003 Server Service Pack 2, Windows Vista Service Pack 2, Windows Server 2008 Service Pack 2, Windows 7 or Windows Server 2008 R2 operating system
- Microsoft .NET Framework runtime 2.0 SP2 or Microsoft .NET Framework 3.51 SP1 installed.

Installing SFT Encoder

SFT Encoder is supplied to you using Windows Installer (MSI) -based installation program, either in 32-bit or 64-bit format depending on the target environment where it is run. You can execute the installer by double-clicking the MSI file or by running following command in the command-prompt:

msiexec.exe /i c:\path\to\installer\SftEncoderEnterprise_X.Y_x86.msi

Note that filename will vary depending on edition of SFT Encoder being installed and according to version being installed.

After installer has finished, you can find shortcut from your Start Menu to a command prompt which will open automatically to directory where the product was installed to along with this documentation if it was installed as part of the setup.

Installing the license

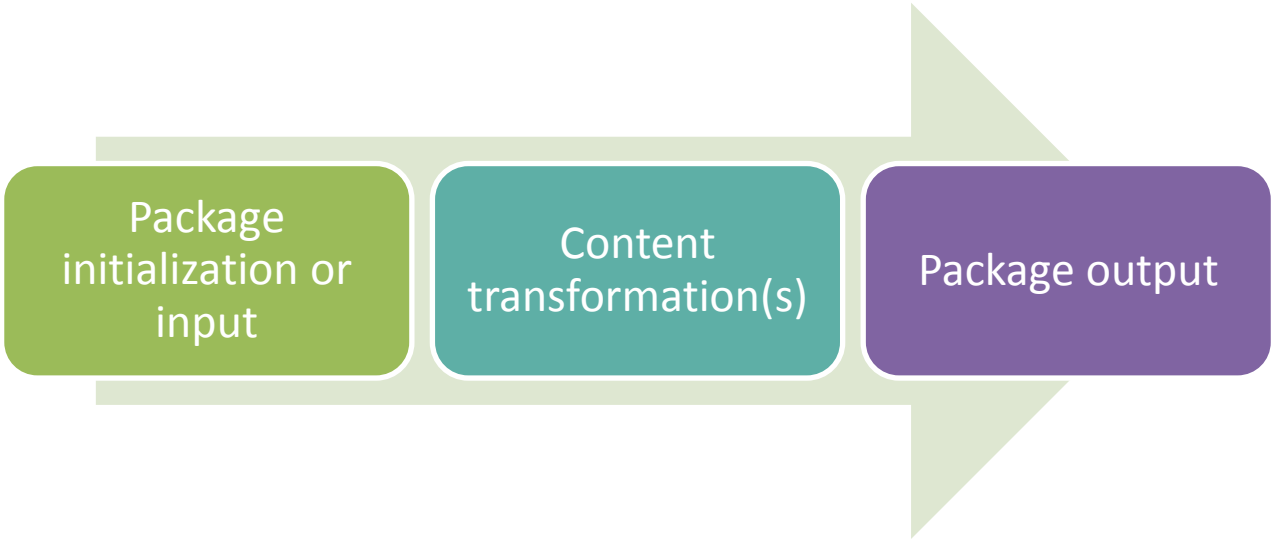
Before you can start using SFT Encoder, you have to supply a valid license key that was provided to you as part of the product purchase. You can perform this by specifying /license -parameter, followed by the license key. Operation has to be done only once as on subsequent usage license key will be remembered by SFT Encoder.

If you don’t have a license key available, please go to SFT Encoder’s product website (http://www.gridmetric.com/products/sftencoder.html) and follow instructions on how to purchase one.

Basic Operations

As SFT Encoder is a console application, its usage is driven non-interactively by providing necessary options as parameters to the main executable, **SftEncoderEnt.exe** (Enterprise Edition) or **SftEncoderSrv.exe** (Server Edition). While the program is executing, its output is displayed on the console screen. If the output needs to be examined later on, it can be piped into external file using normal command interpreter redirection methods.

Basic workflow with SFT Encoder is built around the following process:



As initial step, SFT Encoder creates completely new package from the scratch or opens existing package as base. In case of opening the whole package (identified by SPRJ project file, in contrast to just opening SFT file), SFT Encoder will detect all associated package files and parses those as well. On top of this in-memory representation of the package, all needed content transformation operations are then performed against to, such as adding (importing) new files or directories to the internal file-structure, defining new executables to be launched from package or removing existing registry key(s).

The end-result of these transformation(s) is a package that is ready to be written [back] to the disk. Depending on what was originally opened, SFT Encoder will automatically determine correct file naming for the output files (SFT, SPRJ etc.) which then will be written out. Optionally these file names can be overwritten before SFT Encoder writes files out.

Note: When outputting package files from SFT Encoder, do not specify source (input) directory as output directory since that would result in overwriting of original files. This may leave your package in non-working condition and at minimum cause loss of original source files if encoding process for some reason is interrupted, has non-desired effect or cannot be completed otherwise.

Creating a new package

When creating a new package, at bare minimum you have to define few mandatory things:

- Root-name for package's file system
- Package name
- Output directory

These parameters result an empty package but in order to make it do something, you have to use various import parameters (/encode, /vfsencode, /regencode etc.) and at least one of the launch parameters (/lauch, /vfslaunch or /extlaunch) to publish OSD files.

Example of creating new package:

> *SftEncoderEnt /root "myapp.v1" /name "My Application /out c:\temp /vfsencode "c:\Program Files\MyTool" /vfslaunch "c:\Program Files\MyTool\app.exe"*

Modifying an existing package

When making modifications to existing SoftGrid/App-V packages, at minimum you have to define what to open for modification and where to write it out:

- Either SFT input or SPRJ input
- Output directory

In most of the modification scenarios, you should open the whole package (using SPRJ open), because in that case SFT Encoder detects and opens all additional package files.

This detection happens primarily via SPRJ file, which references both SFT file and all package OSD files. SFT Encoder will additionally scan extra OSD files placed in the same directory as SPRJ file is (package's base-directory) as well as any icons placed in the same directory or under standard icon subdirectory (package name + "Icons", e.g. "Myapp Icons").

Note: If SPRJ references SFT file that does not exist, package cannot be opened. This may be the case, if SFT file has been upgraded at some point and copied to package directory without any associated other files. In that case, reference to SFT is orphaned in SPRJ file and later attempts to open (or publish!) package through it will result an error.

In addition to input and output -related parameters when modifying existing package, various content -related parameters can be defined like when creating a new package.

SFT Encoder parameters

Parameters supported by SFT Encoder are given either as simple switches (parameter without data) or parameters that supply some piece of information as data, like package's name, output file etc. When parameters with data are supplied, any piece of data containing spaces needs to be double-quoted. When double-quoting directory paths, last character cannot be back-slash ('\') so all paths must be supplied in form without it (e.g. "c:\Program Files" instead of "c:\Program Files\"). Parameters can also be supplied in free order.

Example of typical parameters:

> *SftEncoderEnt /in "c:\packages\mypackage.sprj" /compression /out c:\temp*

If executable is invoked without any parameters, SFT Encoder will display list of supported parameters as online help.

All parameters supplied to the executable falls in the four main categories:

1. Input parameters
2. Content parameters
3. Output parameter
4. Other parameters

Input parameters (or lack of) define either of the two supported "modes" in the SFT Encoder, which are *new package creation* and *modifications to an existing package*. When modifications to the existing package is required, appropriate input parameters must be supplied in order to tell SFT Encoder which package or SFT file it needs to open for modification. If input parameters are totally omitted, SFT Encoder will proceed to make create totally new App-V package.

Input parameters are documented in the section called Input parameters.

Content parameters define any and all transformation operations that need to be performed to package's logical content. These modifications can both affect the internal composition of the package (the SFT file) as well as package metadata files related to the modification being made (like updating OSDs' content when changing the internal package root directory).

Note: SFT Encoder does not guarantee execution order of the content modification parameters, making it non-deterministic when it comes to specifying both addition and deletion of the same logical path (registry or filesystem). If strict ordering of content transformation actions is required, usage of *Processing Template* –feature is required.

Content parameters are documented in the section called Content parameters.

Output parameters define to where and how the resulting SFT file and associated package files (OSD, SPRJ etc.) will be saved. There are several encoding parameters, such as fileformat version, making SFT Encoder as powerful tool to perform up- and downgrades of packages.

Output parameters are documented in the section called Output parameters.

Other parameters involves with operations not affecting the actual package processing, such as specifying template file to process (Enterprise and Server Editions only).

Other parameters are documented in the section called Other parameters.

Input parameters

/in *FILE*

Specifies that SPRJ file – and implicitly all associated package files - in path *FILE* is opened for modification. This will put SFT Encoder into modification mode wherein specified package will be the target for content transformations.

Use this parameter if you want to make modifications to existing SoftGrid/App-V package as whole, generating package upgrade or making branched package (depending on the other parameters).

/sftin *FILE*

Specifies that SFT file in path *FILE* is opened for modification. This will put SFT Encoder into modification mode where only the specified SFT file (and not any of the other associated package files, such as SPRJ, OSDs etc.) will be target for content transformations.

Use this parameter if you want to make modifications only to the SFT's content or want to base new whole package only on old one's SFT.

Content parameters

/acl *on|off*

Specifies if internal file/directory ACLs are enforced for the package. Specifying this option sets or clears the flag for the client to either make use of security descriptor information contained in the SFT file or to ignore it (pre-4.5 package behavior).

This option does not have any effect on packages in pre-4.5 format or packages that has been upgraded to 4.5 but do not have any ACLs stored within.

/clearhistory

Specifies that package's sequencing history will be deleted from the package. This option will cause outputted SFT to break version linearity (but not internal version numbering) and as such is mostly suitable to be used together with options /newguids and /root.

/encode[1..n] *DIRECTORY|FILE*

Specifies that physical *DIRECTORY* or *FILE* is imported into the package as new content. This will cause SFT Encoder to perform import of all files and directories from the specified path so that the topmost directory/file will be placed under the package's internal root-directory.

If multiple files or directories need to be imported, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /encode1 "c:\Program Files\Directory One" /encode2 "c:\Program Files\Directory Two"*

/filedelete [1..n] *PATH*

Specifies a *PATH* to a file or directory in the package that needs to be deleted. This will cause SFT Encoder to delete file or directory entry (and all subentries) from the internal directory -structure.

As the root-directory name is different for all the packages, *PATH* is specified in root-relative format where e.g. path of "Myapp\1.0" would in fact refer to a path starting from the package's root directory ("Myapp" would be subdirectory to a root-directory).

If multiple deletions need to be performed, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /filedelete1 VFS\CSIDL_APPDATA /filedelete2 VFS\CSIDL_DESKTOP*

/installercleanup

Removes Windows Installer runtime files (cabinet.dll, msi.dll, msiexec.exe, msihnd.dll and msimsg.dll) from the package if they present in the virtualized system directory.

Sequencers previous to App-V 4.5 version included these files by default to all packages if not explicitly removed and they are a known cause for some packages failing to run on the target machine if present in the sequence.

Note that there isn't matching functionality in the template processing directives; you can achieve same effect by specifying these five files as parameters to FILEDELETE -directive.

/makecopy

Specifies that when import operation for files is running, everything will be first copied to temporary directory. This option will prevent situation where already imported files are modified by some external process, but before SFT package has successfully been encoded to disk. When encoding SFT file, files will be read from the temporary location making sure that no changes or deletions has occurred or new locks has been introduced.

By default SFT Encoder will not create temporary copy of the files during any import operation, but this default behavior may cause failure to create SFT package in some cases. Branching will increase importing time, especially when having large files or great number of files.

/newguids

Specifies that new GUIDs will be generated for the package being modified. This option will make outputted SFT to be independent of the original one as package's uniquely identifying ID is re-generated. Along with the /root -parameter this option makes it possible to create branched packages that can co-exist along with the original one, as in the following example:

> *SftEncoderPro /in c:\old_package\App.sprj /root newroot.v1 /newguids /out c:\new_package*

This option has no effect on new packages.

/override

Specifies that import operation on files and registry will override existing and conflicting content in the package. Setting this option will cause SFT Encoder to override any files or registry values that might exist in the package in the identical path as for one being imported in. If this option is not specified, SFT Encoder will skip conflicting file and registry entries while importing.

/os *VALUE[,VALUE]*

Specifies a list of OS VALUEs that will be added to all OSD files in the package. If multiple values need to be added, list values separated with the comma:

> *SftEncoderPro /in c:\old_package\App.sprj /out c:\new_package /os Win7,Win764*

Using this option requires that the whole package has been opened, not just the SFT file.

/-os *VALUE[,VALUE]*

Specifies a list of OS VALUEs that will be removed from all OSD files in the package. If multiple values need to be removed, list values separated with the comma:

> *SftEncoderPro /in c:\old_package\App.sprj /out c:\new_package /-os Win2K,Win2KTS*

If you want to remove all OS elements, you can specify **ALL** as a parameter to /-os.

Using this option requires that the whole package has been opened, not just the SFT file.

/root *NAME*

Sets package's internal root-directory ("asset directory") name to *NAME*. For new packages (i.e. not based on existing SFT files) this parameter is mandatory and must be supplied in order to define package root. For existing package, using this option makes it possible to perform root-directory rename.

Note that renaming root for packages already in use – without branching package at the same time – may introduce problems at the client. Root renaming will affect virtual registry as well, renaming all references to old root directory name with the new name.

/regencode[1..n] *KEYPATH*

Specifies registry key KEYPATH that is to be imported into the package as new virtual registry content. This will cause SFT Encoder to perform import of specified key and all subkeys/values under it into package's virtual registry. Value-data will be automatically mapped using Intermediate Value mapping process during import (i.e. references to certain volatile names, such as hostname, are encoded as machine-independent format, like %CSIDL_WINDOWS%).

Names of the keys must be specified either with the full hive-name (e.g. HKEY_LOCAL_MACHINE\...) or using abbreviated names of HKLM, HKCU and HKU for HKEY_LOCAL_MACHINE, HKEY_CURRENT_USER and HKEY_USERS, respectively.

If several different, non-nested, keys needs to be imported, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /regencode1 "HKLM\Software\Application One" /regencode2 "HKCU\Software\Application One"*

/regimport [1..n] *FILE*

Specifies registry editor file FILE that is to be imported into the package as new virtual registry content. Value-data will be automatically mapped using Intermediate Value mapping process during import (i.e. references to certain volatile names, such as hostname, are encoded as machine-independent format, like %CSIDL_WINDOWS%).

If multiple registry files needs to be imported, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /regimport1 c:\temp\regfile1.reg /regimport2 c:\temp\regfile2.reg*

/regkeydelete [1..n] *KEYPATH*

Specifies *KEYPATH* to delete from the package's virtual registry. This will cause SFT Encoder to delete specified registry key from the virtual registry, along with all the sub-keys and values. *KEYPATH* is given in the virtual registry -specific format, where paths start with branch identified by MACHINE (for HKEY_LOCAL_MACHINE) and USER (for HKEY_USERS or HKEY_CURRENT_USER if \REGISTRY\USER\%SFT_SID% is given as path).

If multiple deletions need to be performed, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /regkeydelete1 MACHINE\Software\ABC /regkeydelete2 USER\%SFT_SID%\Software\DEF*

/regvaldelete [1..n] *KEYPATH\VALUE*

Specifies virtual registry value to delete from the package's virtual registry, identified by both the *KEYPATH* and name of the *VALUE*. This will cause SFT Encoder to delete specified registry value from the virtual registry. *KEYPATH* is given in the virtual registry -specific format, where paths start with branch identified by MACHINE (for HKEY_LOCAL_MACHINE) and USER (for HKEY_USERS or HKEY_CURRENT_USER if \REGISTRY\USER\%SFT_SID% is given as path).

If multiple deletions need to be performed, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /regvaldelete1 MACHINE\Software\ABC\Username /regdelete2 "USER\%SFT_SID%\Software\DEF\License Data"*

/vfsencode[1..n] *DIRECTORY|FILE*

Specifies that physical *DIRECTORY* or *FILE* is imported into the package as new content. This will cause SFT Encoder to perform import of all files and directories from the specified path as Virtual Filesystem (VFS) content, so that the all imported paths get modified according the Intermediate Value mapping process used in the SFT and mounted under VFS subdirectory within package.

If multiple files or directories need to be imported, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /vfsencode1 "c:\Program Files\Directory One" /vfsencode2 "c:\Program Files\Directory Two"*

Output parameters

/blocksize *SIZE*

Specifies the package block size in kilobytes (kB) into which SFT file's file data content will be split to. By default SFT Encoder will use block size of 32 if creating a new package or block size used in the original package if modifying existing one.

Note that if the block size is changed for existing package, SoftGrid/App-V clients will not stream the updated package if package has been partially or fully cached with the previous block size.

Although officially supported values for block sizes are 4, 16, 32 and 64, SFT Encoder allows usage of any arbitrary block size between 1 and 64 kilobytes, inclusive.

/compression *yes|no*

Specifies if compression scheme is applied to the encoded SFT file. By default, SFT Encoder will not use compression when creating new packages. Compression setting in the original package is used for encoded package if this option is not specified.

If compression is enabled, SFT Encoder will default into using ZLIB compression scheme as it is only supported compression type for App-V 4.5 packages. Older packages with deprecated BZIP2 compression may be used as input to SFT Encoder, but SFT Encoder does not make it possible to encode packages with BZIP2.

/editonly

Specifies that output SFT generated by the SFT Encoder will not contain any trace of update/upgrade being done, apart from the actual content modifications. This effectively means no updates to package -level timestamps, that no new version history entry is generated and that no new version GUID for package. Actual files inside the package (such as osguard.cp) will have their modification timestamps updated, if they have been altered.

This option makes it possible to make modifications to the package without overloading the version history, making encoded SFT to "look like" original one from external view.

/extlaunch[1..n] *FILE*

Specifies any arbitrary external executable by the path *FILE*, which will be published as new application shortcut for the package. This will create a new OSD file containing reference to specified executable (e.g. "\\server\start\ App.exe") in the CODEBASE -element. If the path is a local file system path, additionally Intermediate Value mapping process is applied to it unless package version is for SoftGrid 3.x.

If multiple executables are published as shortcuts, this parameter must be given multiple times along with a numerical suffix, like in the following example:

SftEncoderPro /extlaunch1 "c:\Program Files\Internet Explorer\iexplore.exe" /extlaunch2

/launch[1..n] *FILE*

Specifies that the executable *FILE*, which was imported as part of directory structure using /encode -parameter, will be published as new application shortcut for the package. This will create a new OSD file referring to said executable using internal file system path of the package (e.g. "app.v1\MyApp\App.exe") in the CODEBASE -element.

If multiple imported files are published as shortcuts, this parameter must be given multiple times along with a numerical suffix, like in the following example:

SftEncoderPro /launch1 "c:\Program Files\Directory One\app.exe" /launch2 "c:\Program Files\Directory Two\supergui.exe"

/metadataonly

Specifies that SFT Encoder will re-use the original SFT file when outputting modified package. This mode enables fast(er) editing of parameters affecting only OSD files, such as publishing parameters.

/msi

Specifies that SFT Encoder will save App-V package MSI file along with the other outputted files. This option is only valid for outputting App-V 4.5 or newer packages.

/name *NAME*

Sets Package *NAME*. For new packages (not based on existing SFT files) this parameter is mandatory and must be supplied. SFT Encoder will use this name for SPRJ file and optionally as SFT name if none other is specified using /sftout -parameter, as well as for PACKAGE NAME -element in OSD files.

/out *DIRECTORY*

Specifies the output *DIRECTORY* into which SFT Encoder will save all output files. This option is mandatory for new and modified packages.

Output directory cannot be same as the directory from where SFT or SPRJ file was opened using /sftin or /in –parameter.

/port *PORTNUMBER*

Specifies the port number from where the package is streamed in. Changes port number in all OSD files' HREF attribute to a specified value. When used in new package, using this option makes it possible to set port number correctly for initial publishing.

If no protocol is specified, default of 554 (for RTSP) will be used for any new OSD files generated by SFT Encoder.

/protocol *rtsp|rtsps|http|file*

Specifies the protocol identifier that the package is streamed using with. Changes protocol in all OSD files' HREF attribute to a specified value. When used in new package, using this option makes it possible to set protocol identifier correctly for initial publishing.

If no protocol is specified, default of RTSP will be used for any new OSD files generated by SFT Encoder.

/server *ADDRESS*

Specifies the server name from where the package is streamed from. Changes server name in all OSD files' HREF attribute to a specified value. When used in new package, using this option makes it possible to set server name correctly for initial publishing.

If no name is specified for server, default of %SFT_SOFTGRID_SERVER% will be used for any new OSD files generated by SFT Encoder.

/sftonly

Specifies that SFT Encoder will only output SFT file without any other package files. This option can be used in the situations where modifications only to the SFT file are required, although caution must be observed if /editonly is not specified.

/sftout *NAME*

Specifies explicit *NAME* for the SFT file that SFT Encoder outputs. If this option is specified, SFT Encoder doesn't automatically determine naming for encoded SFT file.

This option specifies only the filename (e.g. "myapp.sft") without full file system path as output location is specified using /out -parameter.

/subdir *DIRECTORY*

Specifies the logical subdirectory path - *DIRECTORY* - into where package files will be placed on the SoftGrid/App-V Server's content directory. By specifying this option, SFT Encoder will modify CODEBASE -element's HREF -attribute accordingly so that when package is imported, relative path to the SFT file will be set correctly. Use empty name ("") if you want to remove subdirectory information.

If this option is not set for new packages, it is assumed to be placed on the root of the content directory.

/utf

Specifies that SFT Encoder will output OSD, SPRJ and Manifest files in UTF-16 encoding instead of default ANSI (also used in Sequencer's output). This will preserve non US characters in shortcut names etc. more reliably as the output won't be dependent on the system codepages.

/version *3/4/4.5*

Specifies the file format for the encoded SFT file. This option can be used to convert older SoftGrid packages to newer format or from newer format to older. When switching versions, some capabilities might be lost (version history, ACLs) that existed in the original package due to differences in capabilities in different SoftGrid/App-V versions.

If no explicit version is specified using this option, encoded SFT file will retain same version level as one that got opened. In the case of new SFT packages, default file format in to which SFT Encoder will encode packages is 4.5.

Note: if downgrading 4.5 packages to 4.x format, some or all of the history entries will be adjusted for Sequencer version information. This is to avoid 4.5 Sequencer refusing to parse sequencing history entries having 4.5 version information but 4.x format.

/vfslaunch[1..n] *FILE*

Specifies that the executable *FILE*, which was imported as part of directory structure using /vfsencode -parameter, will be published as new application shortcut for the package. In effect this will create a new OSD file, referring to said executable using virtualized file system path (e.g. "c:\Program Files\MyApp\App.exe") in the CODEBASE -element.

If multiple VFS imported files are published as shortcuts, this parameter must be given multiple times along with a numerical suffix, like in the following example:

> *SftEncoderPro /vfslaunch1 "c:\Program Files\Directory One\app.exe" /vfslaunch2 "c:\Program Files\Directory Two\supergui.exe"*
>
> *"\\server\apps\myapp\start.exe"*

/warningsaserrors

Causes SFT Encoder to treat possible warning situations as errors, causing all further processing to be stopped.

Other parameters

/license *KEY*

Enters SFT Encoder's license key. SFT Encoder has to be run once with this option in order to supply correct license key. Any subsequent product usage under the same user account (Enterprise Edition) or machine (Server Edition) will use cached registration information.

As Enterprise Edition is licensed per user, serial key needs to be supplied by each distinct user separately. SFT Encoder Server Edition has to have license key only once per machine regardless of the user accounts using it on that same machine.

/template *FILE*

Specifies the Processing Template file that will drive the SFT Encoder execution. By using this option, specially crafted processing template can be processed using SFT Encoder to perform advanced package creation or modification using available processing directives.

Processing Templates

What is Processing Template

In addition of command-line parameter driven usage, SFT Encoder includes a powerful Processing Template –feature that can be used as alternative way of specifying actions that SFT Encoder needs to perform, driving SFT Encoder operations directly. Both new package creation and modifications to existing ones are possible with Processing Templates.

Processing Template –feature is used by specifying all processing instructions (directives) against SFT package creation or modification in specially formatted file, which will then be parsed and processed in sequential manner by SFT Encoder, giving complete control over execution to directives specified in the template.

This sequential processing model makes it possible to make modifications in set order, ensuring for instance that deletions occurs before addition of data when modification is targeted using same logical registry or file system path. In addition to having deterministic model for content transformations, Processing Template includes additional directives that aren't available via command line processing and added functionality to some of features that are also available from command-line parameter –mode.

You can find processing template examples from Appendix A.

Using Processing Templates

You can use SFT Encoder processing templates by running the SFT Encoder executable with the /template –parameter:

SftEncoderEnt /template "c:\templates\packageupdate.template"

When specifying /template as a parameter, you do not need to specify any other command-line parameter as further input is read from the template file itself.

Note that like in normal command-line parameters -based usage, before you can start processing any templates you need for once to specify valid license key by running SFT Encoder with /license -parameter.

Processing Template Format

All processing instructions are written into processing template –file in the following format:

DIRECTIVE DATA

Where *DIRECTIVE* is one of the processing directives supported by the SFT Encoder and documented below and *DATA* is directive –specific parameter or parameters separated from the *DIRECTIVE* by whitespace. Each new line in the processing template –file is interpreted as new directive so parameter data cannot span multiple lines (other than by wrapping automatically in text-editor).

If directive data can contain more than one parameter (either required or optional), each parameter must be enclosed in double-quotes (") and separated by semi-colon, like in the following example:

DIRECTIVE "Data 1";"Data 2";"Data 3"

If data itself contains double-quote as content, it must be escaped by using double double-quotes like in the following example:

DIRECTIVE "Data 1";"Data ""2""";"Data 3"

Data that contains no more than one parameter will not be double-quoted.

You can include comments in the processing file by prefixing processing template line with semicolon (;).

If parameters for directives needs to be further parameterized, usage of processing token variables is supported.

Processing Tokens

SFT Encoder supports the usage of processing tokens, or variables, in the processing templates. These tokens are variables that can be custom-defined inside the processing template with DEFINE –directive and be used in place of hard-coded values for directives. This makes it possible to parameterize processing template to make it more dynamic.

Processing token can be referenced to by enclosing their name in curly brackets, which then will be expanded to token's value by SFT Encoder before processing the actual directive.

Some processing tokens are automatically defined and updated by SFT Encoder itself, usually as a result of some directive, so that they can be used automatically in later directives to reference results of earlier operations.

Currently automatically defined processing tokens are:

{LastPublished}	References to OSD name of the last application published with PUBLISHAPP directive.

Pre-existing environment variables in the system can also readily be read into as processing tokens by using IMPORTTOKENS directive.

Processing Template Directives

ADDOSVALUE

Required parameters: Valid OS VALUE

Example: ADDOSVALUE WinXP64

Specifies an OS VALUE that gets added to all OSD files in the package.

ADDFTA

Required parameters: OSD filename, name for an association, file extension

Examples: ADDFTA "App1.osd";"Compressed file";"zip"

Publishes new file type association for OSD file specified by the OSD file -parameter. File extension is given without leading dot.

ADDSHORTCUT

Required parameters: OSD filename, shortcut location, shortcut name

Examples: ADDSHORTCUT "App1.osd";"%CSIDL_DESKTOP%";"Application"

ADDSHORTCUT "{LastPublished}";"%
%CSIDL_STARTMENU%\Company X";"Launch Menu"

Publishes new application shortcut for OSD file specified by the OSD file -parameter. Shortcut location is usually given by using well-known mapping variables (such as %CSIDL_STARTMENU% etc.) and will be place where App-V client published the shortcut. Shortcut name is the title for shortcut itself, and it cannot contain any characters illegal for filenames (/,\,? etc.).

As a special shortcut, processing token {LastPublished} can be used as reference to OSD name to lastly published application. This token is dynamically updated whenever new PUBLISHAPP directive is encountered in the processing template.

DEFINE

Required parameters: Token name, token value

Examples: DEFINE "RootPath";"%SFT_MNT%\Myroot.v1"

DEFINE "AppVersion";"1.0.0"

Defines special processing token – or variable, which can be used in other directives as a parameter in place of hard-coded values. As an example, if lot of directives in the file reference to application version, rather than hardcoding it in each line, token {AppVersion} could be defined and used in all parameters needing it. SFT Encoder will substitute token with defined value before processing the actual directive.

Tokens that has been defined has to be referenced by enclosing them in curly brackets ({ }) when and where-ever used (eg. {AppName}). In DEFINE directive live, just the name of the token has to be specified without any enclosing brackets (eg. AppName).

DIRECTORYCREATE

Required parameters: Directory to create

Example: DIRECTORYCREATE \Subfolder\App X

Specifies a new directory to create inside the package's directory structure.

Path is defined in the package root-directory –relative form (\ will mark as root-directory itself, \VFS will mark VFS subdirectory under root and so on).

Path can contain any number of intermediate directories that do not readily exist as they will be created automatically.

ENABLEACL

Required parameters: ACL enablement status

Available enablement statuses:

ON:	ACLs are enabled for package
OFF:	ACLs are not enabled for package

Example: ENABLEACL OFF

Specifies if package's internal filesystem ACLs are enforced or not. This directive allows logically switching off the security descriptors in the App-V 4.5 packages, but it does not remove physical ACL entries inside the package.

New 4.5 packages have ACLs enabled by default.

FILEATTRIBUTES

Required parameters: Filesystem -attribute flags separated by comma, Softricity –attribute flags separated by comma, Virtual Filesystem –attribute flag, path to a virtual file or directory

Available filesystem attribute flags:

+R:	sets read-only
-R:	clears read-only
+H:	sets hidden
-H:	clears hidden
+S:	sets system
-S:	clears system

Available Softricity attribute flags:

+OVR: sets override

-OVR: clears override

+PRM: sets permanent

-PRM: clears permanent

UDATA: sets as User Data

ADATA: sets as Application Data

Available Virtual Filesystem attribute flags:

OVR: override local directory

MRG: merge with local directory

Examples: FILEATTRIBUTES "-R,-H,-S";"+OVR,UDATA";"";"\App ABC\config.ini"

FILEATTRIBUTES "";"-PRM";"";"\VFS\CSIDL_APPDATA\ABC\config.xml"

FILEATTRIBUTES "";"";"OVR";"\VFS\CSIDL_APPDATA\ABC"

Specifies filesystem –attributes and Softricity –attributes for file or directory in SFT's internal file-structure and VFS attributes for directories in the VFS structure. If no change is requested for some attribute target, leave flags list as empty.

Path to file or directory is defined in the package root-directory –relative form (\ will mark as root-directory itself, \VFS will mark VFS subdirectory under root and so on).

FILEDELETE

Required parameters: Path to a file or directory

Examples: FILEDELETE \VFS\CSIDL_DESKTOP

FILEDELETE \Application ABC\User Settings\settings.ini

Specifies path to a file or directory in the internal directory structure about which to delete.

Path to file or directory is defined in the package root-directory –relative form (\ will mark as root-directory itself, \VFS will mark VFS subdirectory under root and so on).

FILEMATCHATTRIBUTES

Required parameters: Filesystem -attribute flags separated by comma, Softricity –attribute flags separated by comma, regular expression for filename

Available filesystem attribute flags:

+R:	sets read-only
-R:	clears read-only
+H:	sets hidden
-H:	clears hidden
+S:	sets system
-S:	clears system

Available Softricity attribute flags:

+OVR:	sets override
-OVR:	clears override
+PRM:	sets permanent
-PRM:	clears permanent
UDATA:	sets as User Data
ADATA:	sets as Application Data

Examples: FILEMATCHATTRIBUTES "-R";"UDATA";".*\.dic"

FILEMATCHATTRIBUTES "-S";"";"."

Specifies filesystem -attributes and Softricity -attributes for files and directories in SFT's internal file-structure matched by specified regular expression. This allows wild-card scan of all the files and directories in the internal filesystem tree and setting attributes to matching ones.

FILEWRITE

Required parameters: Path of a file in package to write to

Optional parameters: Path to a file in the real file system from where contents are read-in

Examples: FILEWRITE "\Subfolder\App X\app.lock"

FILEWRITE "\Subfolder\App X\app.ini";"c:\temp\inifile.txt"

Specifies a file in the package that will be overwritten with (optional) content from pre-existing file in the real file system.

If no source file is specified, target file is emptied.

If target file does not exist, new file will be created in the package.

Path to a file in the package is defined in the package root-directory -relative form (\ will mark as root-directory itself, \VFS will mark VFS subdirectory under root and so on).

IMPORTFILESYSTEM

Required parameters: Encode flags separated by comma, source directory or file path

Available encode flags:

VFS : encodes entries into Virtual Filesystem (VFS)

OVR: override any conflicting files

NOACL: do not import security descriptors

COPY: make temporary copy of file(s) during import

Optional parameters: Target mountpoint where imported directory or file is placed under

Examples: IMPORTFILESYSTEM "VFS";"c:\Program Files\Software 1"

IMPORTFILESYSTEM "OVR,NOACL";"c:\Program Files\Software 1";"\"

Specifies directory which will be imported into SFT package and flags that affect how import operation is done. If no flags are given, this parameter is supplied as empty.

Optionally mount-point can be given, where imported content will be place in the package's internal directory structure. Mount-point is defined in the package root-directory –relative form (\ will mark as root-directory itself; \VFS will mark VFS subdirectory under root and so on). If mount-point does not exist, any intermediate directories will be created automatically.

Note that you can only specify explicit mount-point if doing non-VFS import, that is, you cannot redirect files imported as VFS to appear in another VFS location (e.g. importing files with VFS from c:\test cannot appear to be virtualized to c:\Program Files). If you need to do redirection from where the original directories/files were read-in at the originating machine, you have to have existing VFS structure (e.g. CSIDL_PROGRAM_FILES) in place with the correct mapping into which you can do non-VFS import of files and directories as those imported entries will nonetheless inherit VFS mapping information from the VFS-enabled parent directory.

IMPORTREGISTRY

Required parameters: Encode flags separated by comma, path to registry key

Available encode flags:

OVR: override any conflicting values

SINGLE: import only the current key

Examples: IMPORTREGISTRY "";"HKEY_LOCAL_MACHINE\Software\ABC"

IMPORTREGISTRY "SINGLE";"HKLM\Software\Microsoft\Windows"

Specifies registry key which will be imported into SFT package's virtual registry and flags that affect how import operation is done. If no flags are given, this parameter is supplied as empty. By default, all the registry keys and values under given key will be imported unless SINGLE –flag is specified in which case just the specified key and all values within are imported but none of the subkeys.

Hive-part of the registry-key can be given in abbreviated format, where HKLM denotes HKEY_LOCAL_MACHINE, HKCU denotes HKEY_CURRENT_USER and HKU denotes HKEY_USERS.

IMPORTREGISTRYFILE

Required parameters: Encode flags separated by comma, path to registry file

Available encode flags:

OVR: override any conflicting values

Examples: IMPORTREGISTRYFILE "OVR";"c:\exports\registry_1.reg"

Specifies registry file which will be imported into SFT package's virtual registry and flags that affect how import operation is done. If no flags are given, this parameter is supplied as empty.

IMPORTTOKENS

Required parameters: None

Example: IMPORTTOKENS

Imports process environment variables for a running process (SFT Encoder executable) into processing tokens. This makes it possible to make use of existing environment variables in the machine as variables for template directives too.

INCREMENTSFTNAME

Required parameters: None

Example: INCREMENTSFTNAME

Automatically rename SFT -file to include up-to-date version identifier.

METADATAMODE

Required parameters: None

Example: METADATAMODE

Specifies that SFT Encoder will use and copy over the existing SFT file when saving out the package.

Note: Can only be used when processing existing packages and will not allow usage of any other directive that affect configuration of the SFT's internal directory or virtual environment.

NEWPACKAGE

Required parameters: Root directory name, package name

Example: NEWPACKAGE "asset.v1";"My Package"

Specifies that new empty package must be initialized by SFT Encoder using supplied root directory name for SFT's internal directory structure and package name. This directive must appear in the beginning of the Processing Template.

OPENPACKAGE

Required parameters: Path to a SPRJ file

Example: OPENPACKAGE c:\test\myapp.sprj

Specifies full file system –path to package file (SPRJ) that will be opened for modification. This directive must appear in the beginning of the Processing Template.

OPENSFT

Required parameters: Path to a SFT file

Example: OPENSFT c:\test\myapp.sft

Specifies full file system –path to SFT file that will be opened for modification. This directive must appear in the beginning of the Processing Template.

OSDRENAME

Required parameters: Old OSD filename, new OSD filename

Example: OSDRENAME "App1.osd","App1-DSC.osd"

Specifies both the old and new names for the OSD file. SFT Encoder will rename OSD file if one is found from opened or created OSD files by specified old name.

OUTPUT

Required parameters: Directory path, output type

Available output types:

SPRJ : outputs whole package

SFT: outputs SFT file only

Example: OUTPUT "c:\MyApps\","SPRJ"

Instructs SFT Encoder to encode specified package files to output directory. This directive must appear as last one in the Processing Template as processing is stopped and resulting package is saved when OUTPUT is encountered. If output type of "SPRJ" is selected, SFT Encoder will output all package files as specified in the SPRJ file.

PUBLISHAPP

Required parameters: Path to an executable file, parameters to executable

Optional parameters: OSD filename to generate, Application name in OSD, Application version in OSD, Application working directory

Examples: PUBLISHAPP "\Application ABC\app.exe";""

PUBLISHAPP "\Application ABC\app.exe";""; "SuperApp.osd";"Super App";"1.0"

PUBLISHAPP "C:\Program Files\App\App.exe";""; "SuperApp.osd";"Super App";"1.0";" C:\Program Files\App\Data"

Publishes an application (OSD file) from the package or external source by specifying path to executable file and parameters for it. If executable path starts single backslash, it represents the package's internal root directory and will automatically be corrected to reflect correct path. Otherwise path is interpreted to be file system path to somewhere else.

If no optional parameters are given, application name and/or version is detected from the executable being published and OSD is named according to application name.

Note that no shortcuts are initially defined for newly published application and must be added with ADDSHORTCUT directive.

REGENERATEID

Required parameters: None

Example: REGENERATEID

Specifies that SFT Encoder will re-generate package identifiers. This directive is not required for new packages, but for existing packages this causes package to branch from original one along with root-directory renaming.

REGISTRYDELETE

Required parameters: Path to a virtual registry key

Optional parameters: Value name

Examples: REGISTRYDELETE "MACHINE\Control"

REGISTRYDELETE "USER\%SFT_SID%\ABC";"Username"

Specifies virtual registry key which will be deleted from the package's virtual registry along with all the keys and values it contains. Optionally value name can be specified, causing deletion of that value only.

Virtual registry path is given in the virtual registry –specific format, where paths start with branch identified by MACHINE (for HKEY_LOCAL_MACHINE) and USER (for HKEY_USERS or HKEY_CURRENT_USER if \REGISTRY\USER\%SFT_SID% is given as path).

REGISTRYKEYATTRIBUTES

Required parameters: Attributes flags separated by comma, path to a virtual registry key

Available attributes flags:

OVR:	override local key
MRG:	merge with local key
DEL:	set as deleted

Example: REGISTRYKEYATTRIBUTES "OVR,DEL";"USER\%SFT_SID%\ABC"

Specifies what logical attributes to set for virtual registry key. If attribute flag "DEL" is specified, all keys and values under given virtual registry key will be physically deleted, but key itself will stay in virtual registry in deleted form.

Virtual registry path is given in the virtual registry -specific format, where paths start with branch identified by MACHINE (for HKEY_LOCAL_MACHINE) and USER (for HKEY_USERS or HKEY_CURRENT_USER if \REGISTRY\USER\%SFT_SID% is given as path).

REGISTRYVALUEATTRIBUTES

Required parameters: Attributes flags separated by comma, path to a virtual registry key, value name

Available attributes flags:

DEL:	set as deleted

Example: REGISTRYVALUEATTRIBUTES "DEL";"USER\%SFT_SID%\ABC";"Value 1"

Specifies what logical attributes to set for virtual registry value.

Virtual registry path is given in the virtual registry -specific format, where paths start with branch identified by MACHINE (for HKEY_LOCAL_MACHINE) and USER (for HKEY_USERS or HKEY_CURRENT_USER if \REGISTRY\USER\%SFT_SID% is given as path).

REMOVEHISTORY

Required parameters: None

Example: REMOVEHISTORY

Specifies that SFT Encoder will remove all packaging history entries present, effectively removing version linearity information from existing packages. This option is only valid for 4.0 packages and newer.

REMOVEOSVALUE

Required parameters: Valid OS VALUE

Example: REMOVEOSVALUE WinNT

Specifies an OS VALUE that will be removed from all OSD files in the package.

By using value of **ALL** as parameter to this directive, every OS element present in OSD file(s) will be removed making resulting OSD file(s) visible to all operating systems.

REMOVEVARIABLE

Required parameters: Variable name

Example: REMOVEVARIABLE TEMP

Removes virtualized environment variable, specified with variable name parameter, from the package.

Note that the variable name is not case-sensitive.

SAVEMSI

Required parameters: None

Example: SAVEMSI

Specifies that SFT Encoder will generate App-V package MSI file along with the other saved files. This option is only valid for outputting App-V 4.5 or newer packages.

SETBLOCKSIZE

Required parameters: Blocksize in kilobytes

Example: SETBLOCKSIZE 64

Specifies the data block size (i.e. to how small individual pieces data is split inside SFT) to be used for encoded SFT file. Supported range of blocksizes is between 1 and 64, although generally 32 (which is the default) and 64 are recommended settings.

SETCOMPRESSION

Required parameters: Compression type

Available compression types:

NONE: No compression is used

ZLIB: Zlib –compression algorithm is used

Example: SETCOMPRESSION ZLIB

Specifies the compression scheme to be used for encoded SFT file. This directive allows changing compression of existing package to another, new packages are by default uncompressed if not otherwise specified.

SETPUBLISHINGADDRESS

Required parameters: Server-name

Example: SETPUBLISHINGADDRESS myvas

Sets publishing server address that will be changed to all package's OSD files. This information is part of the HREF attribute in OSD file(s) of the package.

SETPUBLISHINGDIRECTORY

Optional parameters: Logical subdirectory

Example: SETPUBLISHINGDIRECTORY myapp.v1

Sets the logical subdirectory for where to package files will be stored on the Content directory. This information is part of the HREF attribute in OSD file(s) of the package. To specify that no subdirectory is used, omit subdirectory parameter.

SETPUBLISHINGPROTOCOL

Required parameters: Protocol identifier

Available protocol identifiers:

RTSP:	Real-time stream protocol
RTSPS:	Secure real-time streaming protocol
HTTP:	HTTP protocol
HTTPS:	HTTPS protocol
FILE:	File-based streaming

Example: SETPUBLISHINGPROTOCOL RTSP

Sets publishing server protocol that will be changed to all package's OSD files. This information is part of the HREF attribute in OSD file(s) of the package.

SETPUBLISHINGPORT

Required parameters: Protocol port number

Example: SETPUBLISHINGPPORT 322

Sets publishing server protocol port that will be changed to all package's OSD files. Use in conjunction with SETPUBLISHINGPROTOCOL -directive to change both the protocol and the port. This information is part of the HREF attribute in OSD file(s) of the package.

SETREGISTRYVALUE

Required parameters: Path to a virtual registry key, value name, value data to set

Optional parameters: Value type

Available value types:

SZ:	String
MZ:	Multi-string
EZ:	Expandable string
DW:	Double-word (32-bit)
QW:	Quad-word (64-bit)
BIN:	Binary

Examples: SETREGISTRYVALUE "USER\%SFT_SID%\ABC";"Value 1";"SFT Encoder"

SETREGISTRYVALUE "MACHINE\Software\App";"Count";"1";"DW"

SETREGISTRYVALUE "MACHINE\Software";"Reg";"1B,00";"BIN"

Writes data into specified virtual registry value. If virtual registry key or value does not exist, they will be created automatically. When creating new value, optional value type can be specified, otherwise data is assumed to be of SZ type.

If writing binary-data to value, data is given as list of comma-separated hexadecimal bytes in string. If writing multi-string values, lines can be separated by using \0 as escape character.

Writing to virtual registry key's default (non-named) value can be achieved by leaving value name as empty.

Virtual registry path is given in the virtual registry -specific format, where paths start with branch identified by MACHINE (for HKEY_LOCAL_MACHINE) and USER (for HKEY_USERS or HKEY_CURRENT_USER if \REGISTRY\USER\%SFT_SID% is given as path).

SETROOTNAME

Required parameters: Filename for the root directory

Optional parameters: Short filename (8+3 format) for the root directory

Examples: SETROOTNAME "appv.v45"

SETROOTNAME "Application.v45";"appv.45"

Specifies the name for the root directory ("asset directory") of the package. For existing package performs renaming of root. Optionally 8+3 formatted short filename can be specified along with the full filename, but as standing recommendation for root-directories is to be in 8+3 format, usage of differing full filename and short name is discouraged.

SETVARIABLE

Required parameters: Variable name, variable data

Example: SETVARIABLE “PATH”;” %PATH%;%SFT_MNT%\app\bin;”

Sets or replaces virtualized environment variable, specified with variable name parameter, to value specified with variable data parameter.

Setting environment variable affects all OSD files in the package and is not case-sensitive for variable name.

SETVERSION

Required parameters: Fileformat version

Available versions:

3.x:	SoftGrid 2.x/3.x package format
4.0:	SoftGrid 4.0 – 4.2 package format
4.5:	App-V 4.5 package format
4.6:	App-V 4.6 package format

Example: SETVERSION 4.6

Specifies the SFT fileformat’s version to be used for encoded SFT file. This directive allows targeting of specific feature set of SFT.

Note: if downgrading 4.5+ packages to 4.x format, some or all of the history entries will be adjusted for Sequencer version information. This is to prevent 4.5 Sequencer from refusing to parse sequencing history entries in unexpected format.

SETVIRTUALDRIVE

Required parameters: Drive letter

Example: SETVIRTUALDRIVE Q

Specifies virtual drive letter that SFT Encoder will use while processing paths during import operations.

This directive make all paths containing specified drive letter be encoded using %SFT_MNT% Intermediate Value. For instance, if importing registry file that has value with data of "Q:\root\file.ini", SFT Encoder encodes this path into the package as "%SFT_MNT%\root\file.ini" if drive letter Q: has been set using this directive, causing App-V client to decode the path according the virtual drive letter used in the client. Otherwise, the path would be stored as-is without any conversion.

Note: Use of this directive at the beginning of template is strongly suggested when the machine does not contain SoftGrid/App-V Client from where the virtual drive designation can be inferred from. Use drive letter that is not used by any local drives present.

SFTRENAME

Required parameters: Name without extension

Example: SFTRENAME My Package

Specifies the new name for the SFT file. Filename is defined without extension which will be applied automatically.

SPRJRENAME

Required parameters: Name without extension

Example: SPRJRENAME My Package

Specifies the new name for the SPRJ file. Filename is defined without extension which will be applied automatically.

STOPONWARNINGS

Required parameters: None

Example: STOPONWARNINGS

Specifies that SFT Encoder will stop all further processing and returns error when any warning condition is encountered.

SUPPRESSHISTORYGENERATION

Required parameters: None

Example: SUPPRESSHISTORYGENERATION

Specifies that SFT Encoder will not generate new history entry and version ID for the encoded package. This option is only valid for 4.0 packages and newer.

SUPPRESSTIMESTAMPINCREMENT

Required parameters: None

Example: SUPPRESSTIMESTAMPINCREMENT

Specifies that SFT Encoder will not set package's modification timestamps to current time of encoding.

SUPPRESSVERSIONINCREMENT

Required parameters: None

Example: SUPPRESSVERSIONINCREMENT

Specifies that SFT Encoder will not increment internal version number of the package.

UTFOUTPUT

Required parameters: None

Example: UTFOUTPUT

Instructs SFT Encoder to output OSD, SPRJ and Manifest files in UTF-16 character encoding instead of the default ANSI codepage which is dependent on system locale.

VFSDIRECTORYCREATE

Required parameters: Attributes flags separated by comma, directory to create

Available attributes flags:

OVR:	set mapped directory as overridden (fully virtualized)
ALL	set all intermediate directories as overridden (used only together with OVR flag)

Examples: VFSDIRECTORYCREATE "";"C:\Windows\System32"

VFSDIRECTORYCREATE "OVR";"C:\Config"

VFSDIRECTORYCREATE "OVR,ALL";"C:\Program Files\MyApp\Data"

Specifies a new VFS –enabled directory to create inside the package's directory structure (under VFS).

Path is defined as it would (and will when package is running at the client) appear in the real file-system (e.g. folder under Program Files, etc.), SFT Encoder will automatically create corresponding directory and any intermediate directory required under VFS structure inside the package.

Flags define the isolation level for newly created directories. If no flags are set, newly create VFS directories will be set to merged state. If OVR flag is set, the final directory in the path is set to overridden. If ALL -flag is set in

conjunction with OVR; all the intermediate directories that gets created are set to override mode.

Note: if path starts with well-known directory (one that gets mapped to variable), this directory is always set to merged regardless of the flags.

Note: If creating VFS directories with merged state, at least one virtual file has to be placed in it and/or subdirectories with override status, otherwise virtualization information is not persisted for newly created VFS directory. This is because of limitation in App-V's VFS implementation.

Appendix A – Processing Template examples

Example 1 – Making offsite copy of active package

Opens existing package from local directory and saves it to remote server with differing publishing parameters and site -specific identification changed into registry for e.g. disaster recovery backup purposes. As new version increment etc. are suppressed, for the clients package still appears to be original one (albeit it has modifications in it).

```
OPENPACKAGE C:\content\reader\reader.sprj
FILEDELETE \bin\user.ini
SETPUBLISHINGADDRESS sitefs
SETPUBLISHINGPROTOCOL http
SETPUBLISHINGPORT 80
SETPUBLISHINGDIRECTORY
SETREGISTRYVALUE
"MACHINE\Software\Reader\Config";"SiteID";"001"
SUPPRESSHISTORYGENERATION
SUPPRESSTIMESTAMPINCREMENT
SUPPRESSVERSIONINCREMENT
OUTPUT "\\sitefs\content";"SPRJ"
```

Example 2 – Building an App-V package as part of software build process

Create a new App-V package as part of the software building process and as last action, saves completed package to local directory.

```
; Initializes new package
NEWPACKAGE "superapp.v01";"Super Application"

; Imports files and directories
IMPORTFILESYSTEM "VFS,NOACL";"C:\Program Files\Common Files\Super Vendor"
IMPORTFILESYSTM "";"C:\Latest Build\SuperApp"
IMPORTFILESYSTM "";"C:\Config\Files";"\SuperApp"
FILEATTRIBUTES "";"";"OVR";"\VFS\CSIDL_PROGRAM_FILES_COMMON\SuperApp"
FILEMATCHATTRIBUTES "+R";"+OVR,ADATA";".*\.lib"
FILEMATCHATTRIBUTES "";"UDATA";".*\.config"
; Creates necessary registry entries
IMPORTREGISTRYFILE "";"C:\Config\UserSettings.reg"
IMPORTREGISTRYFILE "";"C:\Config\ApplicationSettings.reg"
REGISTRYDELETE "USER\%SFT_SID%\Software\Super Vendor\Super Application";"Username"
SETREGISTRYVALUE "MACHINE\Software\Super Vendor\Super Application";"Version";"4.0"

; Publish applications
PUBLISHAPP "\SuperApp\SuperApp.exe";"";"SuperApp.osd"
PUBLISHAPP "\SuperApp\Config.exe";"/virtual";"Configurator.osd"

; Write completed package out
SETVERSION 4.6
SETBLOCKSIZE 64
OUTPUT "c:\Latest Build\App-V";"SPRJ"
```

Appendix B – SFT Encoder Software License Agreement

SFT Encoder (hereinafter "SOFTWARE")

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

SFT Encoder Professional and Enterprise Editions includes a Personal License and SFT Encoder Server Edition includes a Hardware License. Personal License and Hardware License are mutually exclusive.

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. However the license doesn’t allow using the SOFTWARE simultaneously on both computers. The license doesn’t allow using the SOFTWARE as a part of an automated service.

1.3 HARDWARE LICENSE

The Hardware License allows the SOFTWARE to be installed on one computer only. The license does allow using the SOFTWARE as part of a server application, automated service, non-interactive batch process or with other corresponding use.

1.4 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 C – 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 HAVE 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.