

Gridmetric Lib-V

A complete App-V package Software Development Kit (SDK)

API Documentation

V6.12

Copyright © 2008 – 2019 Gridmetric Oy

Portions copyright © Mike Krueger

Portions copyright © Microsoft Corporation

“Lib-V” is a trademark of Gridmetric Oy.

Windows, SoftGrid and App-V are registered trademarks of Microsoft Corporation.

All other trademarks and copyrights referred to are the property of their respective owners.

THIS DOCUMENTATION IS PROVIDED «AS IS» AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Contents

Overview	11
System Requirements	12
General information on Lib-V	13
Assemblies	13
Lib-V Assembly dependency chart for legacy App-V packages	15
Quick start into using Lib-V.....	16
Entering Lib-V licensing.....	16
For legacy App-V package processing.....	16
For App-V 5.0+ package processing	16
SFT binary package file.....	17
Associated package files (OSD, SPRJ, Manifest, MSI)	17
APPV binary package file	18
Lib-V Core, Encoding and Decoding assemblies API reference.....	19
ClientVersion class.....	19
Properties.....	19
EncodingAdditionalParameters class.....	22
Properties.....	22
HistoryEntry class	24
Properties.....	24
HistoryEntry45 class.....	29
Properties.....	29
IntermediateValueManager class.....	31
Static methods.....	31
Static properties.....	32
Methods.....	34
Properties.....	36
Lib-V class.....	37
Static methods.....	37
Static fields	38
ParserItem class.....	39
Properties.....	39
ProcessorInfo class.....	41

Properties.....	41
SequencerVersion class.....	43
Properties.....	43
SftDecoder class	46
Constructors	46
Methods.....	46
Events.....	49
SftDecoder.SftDecodingEventArgs class	51
Properties.....	51
SftEncoder class.....	53
Constructors	53
Methods.....	53
Properties.....	59
Events.....	60
SftEncoder.SftEncodingEventArgs class	61
Properties.....	61
SftPackageCreationInfo class	63
Properties.....	63
SftPackageAdvancedCreationInfo class	65
Properties.....	65
SftPackage class.....	67
Static methods.....	67
Methods.....	68
Properties.....	73
Events.....	77
SftPackage.FilesystemVirtualizationEventArgs class	80
Properties.....	80
SftPackage.RegistryVirtualizationEventArgs class.....	81
Properties.....	81
SftPackageProperties class	82
Properties.....	82
SftPackageSecurity class.....	87
Properties.....	87
SftPackageFilesystem class.....	89

Properties.....	89
VfsMappingTarget class.....	94
Properties.....	94
VirtualDirectory class.....	97
Constructors	97
Methods.....	98
Properties.....	103
VirtualDirectoryEntry class	106
Methods.....	106
Properties.....	106
VirtualEnvironmentConfiguration class	116
Methods.....	116
Properties.....	117
VirtualFile class.....	121
Constructors	121
Methods.....	122
Properties.....	125
VirtualFileStream class.....	129
Methods.....	129
Properties.....	134
VirtualFileBlock class.....	137
Properties.....	137
VirtualRegistryEntry class.....	140
Properties.....	140
VirtualRegistryKey class.....	146
Constructors	146
Methods.....	147
Properties.....	156
VirtualRegistryValue class.....	160
Constructors	160
Properties.....	161
VirtualService class	165
Constructors	165
Properties.....	167

WindowsInfo class.....	172
Methods.....	172
Properties.....	173
Enumerations.....	181
NtSuiteMasks.....	182
NtSystemType.....	183
PackageSaveMode	184
ProcessorFamily.....	185
ServiceErrorControl	186
ServiceStartupType	187
ServiceType.....	188
SftCompressionType.....	189
SftDecodingOptions.....	190
SftDecodingPhase.....	192
SftEncodingOptions.....	193
SftEncodingPhase.....	195
SftFileformatType.....	196
SftFileformatVersion.....	197
SoftricityFileAttributes.....	198
VirtualFilesystemAttributes.....	199
VirtualFilesystemMappingAttributes	200
VirtualFilesystemMode.....	201
VirtualRegistryAttributes	202
VirtualRegistryIMStatusAttributes	204
Lib-V Metadata and serialization assemblies API reference	205
ManifestAppElement class	206
Constructors	206
Properties.....	207
ManifestFile class.....	211
Constructors	211
Methods.....	212
ManifestFileTypeElement class.....	214
Constructors	214
Properties.....	215

ManifestSerializer class	218
Static methods	218
MetadataFileBase	224
Properties	224
MetadataSerializer class	225
MsiSerializer class	226
Static methods	226
MsiWrapperPackage class	230
Constructors	230
Properties	231
NameOsdFileCollection class	240
Constructors	240
Methods	241
Properties	243
OsdCodebaseElement class	245
Constructor	245
Properties	246
OsdCodebaseHref class	249
Constructor	249
Methods	249
Properties	250
OsdDscCodebaseElement class	253
Constructors	253
Properties	254
OsdDdeExecElement class	256
Constructor	256
Properties	257
OsdDependenciesElement class	259
Constructor	259
Properties	259
OsdDependencyElement class	260
Constructor	260
Properties	260
OsdElementBase class	262

Properties.....	262
OsdEnvironmentElement class.....	264
Constructor.....	264
Properties.....	264
OsdFile class.....	266
Constructors	266
Methods.....	268
Properties.....	271
OsdFileAssociationsElement class	278
Constructor.....	278
Properties.....	278
OsdFileExtensionElement class	280
Constructor.....	280
Properties.....	281
OsdImplementationElement class.....	284
Constructors	284
Properties.....	285
OsdPoliciesElement class.....	288
Constructor.....	288
Properties.....	288
OsdProgIdElement class.....	292
Constructor.....	292
Properties.....	293
OsdRegistryElement class.....	297
Constructor.....	297
Properties.....	297
OsdRegkeyElement class	298
Constructor.....	298
Properties.....	299
OsdRegvalueElement class	301
Constructor.....	301
Properties.....	302
OsdScriptElement class	304
Constructor.....	304

Properties.....	305
OsdSerializer class.....	310
Static methods.....	310
OsdShellCmdElement class.....	317
Constructor.....	317
Properties.....	318
OsdShellCommandListElement class	321
Constructor.....	321
Properties.....	321
OsdShortcutElement class	323
Constructors	323
Properties.....	325
OsdVirtualenvElement class	329
Constructors	329
Properties.....	329
OsdVmElement class.....	332
Constructor.....	332
Properties.....	332
SprjDefaultElement class.....	334
Constructor.....	334
Methods.....	334
Properties.....	335
SprjDeletedFileExtension.....	336
Constructor.....	336
Properties.....	336
SprjDeletedFileType.....	338
Constructor.....	338
Properties.....	338
SprjDeletedObjectElement	340
Properties.....	340
SprjDeletedShortcut.....	341
Constructor.....	341
Properties.....	341
SprjExclusionElement.....	343

Constructor.....	343
Methods.....	344
Properties.....	344
SprjFile class.....	345
Constructors	345
Methods.....	346
Properties.....	347
SprjParserItemElement	353
Properties.....	353
SprjSerializer class.....	355
Static methods.....	355
SprjSubstitutionElement.....	361
Constructor.....	361
Methods.....	362
Properties.....	362
Enumerations	364
MsiInstallMode	365
OsdOperatingSystemTypes	366
OsdRegistryValueType.....	368
OsdScriptConditional.....	369
OsdScriptEvent.....	370
OsdScriptTiming.....	371
OsdScriptType.....	372
OsdVmType	373
OsdSubsystemType	374
SprjDeletedObjectType.....	375
SprjParserItemContext.....	376
SprjParserItemType	377
Appendix A – Lib-V Software License Agreement	379
Appendix B – 3rd party licenses	382
WiX Deployment Tools Foundation.....	382

Overview

Gridmetric Lib-V is a .NET library component for providing programmatic access to legacy Microsoft App-V package file format (including .SFT, .OSD etc. files), as well as latest App-V 5.X and Windows 10 –specific App-V package formats (including .APPV, .MSI and .XML Dynamic Configuration files).

Purpose of the Lib-V is to provide necessary programmatic abstraction around aforementioned file types, making it possible to create, use, update and save content in App-V package in object-oriented manner.

Lib-V provides the following main functionalities for legacy 4.X packages:

- Access to SFT package via [SftPackage](#) –object.
- Decoding existing physical SFT file into [SftPackage](#) –object using [SftDecoder](#) class.
- Encoding [SftPackage](#) to physical SFT file using [SftEncoder](#) class.
- Access to App-V package’s metadata files via [OsdFile](#), [SprjFile](#), [ManifestFile](#) and [MsiWrapperPackage](#) –objects.
- Parsing and serializing physical OSD file from and into [OsdFile](#) –object using [OsdSerializer](#) –class.
- Parsing and serializing physical SPRJ file from and into [SprjFile](#) –object using [SprjSerializer](#) –class.
- Parsing and serializing physical Manifest file from and into [ManifestFile](#) –object using [ManifestSerializer](#) –class.
- Parsing and serializing physical MSI file from and into [MsiWrapperPackage](#) –object using [MsiSerializer](#) –class.

Lib-V provides the following main functionalities for 5.X/Windows 10 packages:

- Access to unified package view via AppVPackage –object, providing Save() and Load() methods for decoding and encoding the package and its files.

For reporting errors, omissions or general feedback for this documentation, please contact support@gridmetric.com.

System Requirements

- Microsoft Visual Studio 2005 or newer, or other compatible .NET development environment
- [Microsoft .NET Framework runtime 2.0 Service Pack 2](#) (or newer) or [Microsoft.NET Framework runtime 3.51 Service Pack 1](#) (or newer) installed.
- Windows 7, Service Pack 1 or newer is recommended.
- For App-V 5.0+ package operations, local account having SE_BACKUP and SE_RESTORE privileges is required.

General information on Lib-V

Assemblies

Gridmetric Lib-V is shipped as six separate .NET assemblies:

- **Gridmetric.Lib-V.V5.Core.dll**

Implements all functionality and object models for the App-V 5 packages. This assembly is always required for all code utilizing Lib-V functionality for 5.X package handling.

- **GridMetric.LibV.Core.dll**

Implements core functionality and object model for the SFT file. This assembly is always required for all code utilizing Lib-V functionality for legacy packages handling.

- **GridMetric.LibV.Decoder.dll**

Implements decoding functionality for the SFT file. This assembly is required for all code decoding SFT files into internal object representation.

- **GridMetric.LibV.Encoder.dll**

Implements encoding functionality for the SFT file. This assembly is required for all code encoding internal object representation into physical SFT files.

- **GridMetric.LibV.Metadata.dll**

Implements core functionality and object model for all other filetypes besides SFT file. This assembly is required for all code accessing and manipulating metadata file types.

- **GridMetric.LibV.Metadata.Serialization.dll**

Implements serialization functionality for all other filetypes besides SFT file. This

assembly is required for all code reading or writing metadata file types from or to disk.

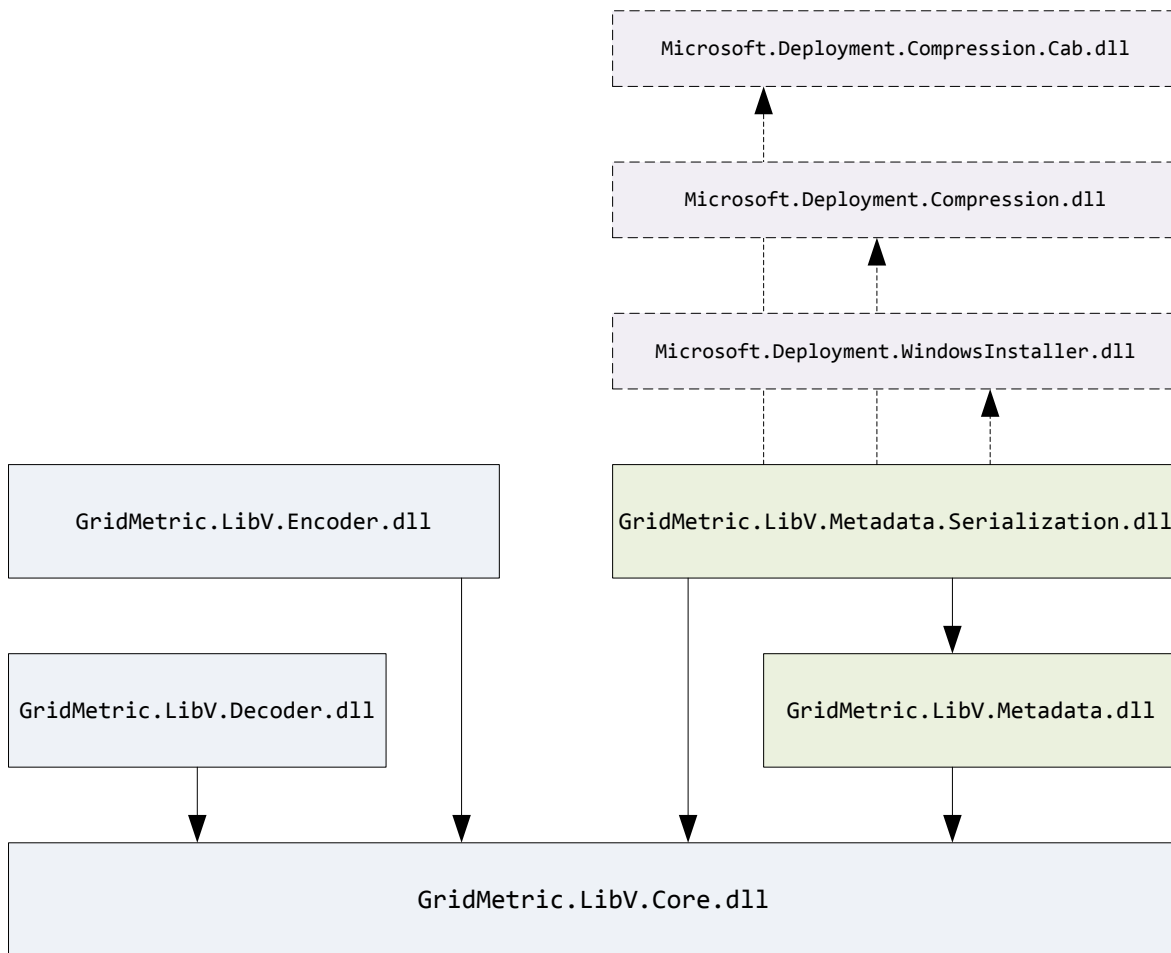
Lib-V ships with two separate sets of metadata serialization DLLs, as it links to selected DTF assemblies from WiX toolset (<http://wix.codeplex.com/>) for the purpose of reading and writing MSI files (namely, *Microsoft.Deployment.WindowsInstaller.dll*, *Microsoft.Deployment.Compression.dll* and *Microsoft.Deployment.Compression.Cab.dll*).

One version of the serialization DLL is linked together with these WiX libraries so that calling code won't have external dependencies to carry other assemblies besides Lib-V, while the second version of serialization assembly dynamically links to WiX libraries.

It is the responsibility of the caller to select which DLL is suitable if App-V MSI (de)serialization capabilities are needed and link against it. Functionally both libraries are the same.

Note: Unless otherwise stated, all methods and properties documented in this document are not thread-safe.

Lib-V Assembly dependency chart for legacy App-V packages



Quick start into using Lib-V

Please note that for the rest of the chapters, this document covers only the API specification of Lib-V for legacy (4.X and pre) package handling. App-V 5.0+ package API specification can be found from a separate [Lib-V API Documentation.chm](#) file.

Entering Lib-V licensing

In order to access both SFT decoding and encoding functionality and metadata file (de)serialization functions in the Lib-V, a valid license key has to be supplied to it. This should be done in the calling main assembly's SetLicense() method and it needs to be done only once in code.

For legacy App-V package processing

To supply license key for App-V 4.X engine, use [LibV.SupplyLicense\(\)](#) static method.

For App-V 5.0+ package processing

To supply license key for App-V 5.X engine, use LibV.SupplyLicense() static method.

SFT binary package file

All actual access to SFT file's content happens through [SftPackage](#) –class, which can either be created from the scratch using static method on the class, [Create\(\)](#), or using [SftDecoder](#) class for decode existing SFT file to this in-memory representation.

[SftPackage](#) contains two main parts to it; [internal filestructure](#) to which all the directories and files needs to be mounted to under one package root directory (this is the directory mounted to App-V client's virtual drive's root) and [virtual registry](#) which will be overlaid on top of the existing Windows registry at the client.

You can either make explicit modification to these by instantiating – and mounting – new directories, files or registry key and value, or utilize helper methods in the main package class, [VirtualizeFromFilesystem](#) which can be used to import pre-existing data from the local file system or [VirtualizeFromRegistry](#) which can be used to import pre-existing data from the local registry.

When necessary modifications are done to package's content, it has to be serialized by using [SftEncoder](#) class. This creates a fully functioning SFT file, which then can be utilized by the SoftGrid/App-V systems.

Associated package files (OSD, SPRJ, Manifest, MSI)

For the purpose of reading and writing associated metadata files for App-V package, Lib-V exposes main class for each of the file types, namely [OsdFile](#), [SprjFile](#), [ManifestFile](#), [MsiWrapperPackage](#) which can be instantiated anew using the class' constructor or read-in from the disk using file-specific serializer class.

Each metadata file object closely mirrors the structure they have in XML representation format (e.g. having property –hierarchy similar to element nesting) and in case of MSI file, exposing App-V specific MSI properties as .NET properties.

To maintain compatibility to .NET version 2.0, Lib-V's serializer classes for metadata files internally re-implements `ExtensionAttribute` from `System.Runtime.CompilerServices` for the purpose of making extension methods available for `OsdFile`, `SprjFile`, `ManifestFile` and

MsiWrapperFile. This enables these classes to offer Save() method when serialization assembly is referenced.

APPV binary package file

All access to APPV file (main binary package file for 5.0 packages) and its content happens through AppVPackage –class, which provides unified content model for base package contents (coming from .APPV file) as well all associated dynamic configuration files (files ending in .XML and typically in the same directory as .APPV file). These packages can be created from the scratch using Lib-V, and subsequently be saved to the disk, or they can be loaded from disk for modification and re-saving.

AppVPackage contains two main parts to it; a virtual environment carrying internal filestructure and virtual registry which will be overlaid on top of the existing Windows registry at the client, as well as Integration Subsystems for defining application extensions, such as file extensions or shortcuts.

For the filesystem, all user-generated files go under directory called “Root” under package’s filesystem root as this is the directory that the App-V Client will mount as Primary Virtual Application Directory (PVAD) on a client. The real root of the filesystem is reserved for App-V package’s internal files only. A special “VFS” directory underneath the “Root” carries all VFS mapped directories.

For virtual registry, there is already pre-created “MACHINE” and “USER” keys after initializing an empty package. These keys correspond to the “HKEY_LOCAL_MACHINE” and “HKEY_USERS” branches on a real registry, respectively. Additionally pre-made key under “USER” key exists, which will appear as root of “HKEY_CURRENT_USER” on a client machine.

When necessary modifications are done to package's content, it has to be serialized to disk by using AppVPackage’s Save() –method. This will create a fully functioning APPV file, and possible related configuration files, which then can be utilized by the App-V systems.

Lib-V Core, Encoding and Decoding assemblies API reference

ClientVersion class

Provides version information for [RequiredClient](#) property in [VirtualEnvironmentConfiguration](#) -class.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class ClientVersion
```

Properties

Build

Gets the build version number.

Syntax:

```
public UInt16 Build { get; }
```

Property value:

Type: System.UInt16

Build version.

Major

Gets the major version number.

Syntax:

```
public UInt16 Major { get; }
```

Property value:

Type: System.UInt16

Major version.

Minor

Gets the minor version number.

Syntax:

```
public UInt16 Minor { get; }
```

Property value:

Type: System.UInt16

Minor version.

Patch

Gets the patch version number.

Syntax:

```
public UInt16 Patch { get; }
```

Property value:

Type: System.UInt16

Patch version.

Version

Gets the revision full version number (*major.minor.patch.build*) as string.

Syntax:

```
public string Version { get; }
```

Property value:

Type: System.String

Full version number.

EncodingAdditionalParameters class

Specifies extended information for package encoding when passed to [SftEncoder.InitializeEncoder\(\)](#).

Namespace:

GridMetric.LibV.Encoder

Syntax:

```
public class EncodingAdditionalParameters
```

Properties

HistoryEntryAccountName

Overrides detected sequencer/packager account's name in the generated history entry with free-form text.

Property value:

Type: System.String

Text to use for account name information in new SFT version history entry.

Syntax:

```
public string HistoryEntryAccountName { get; set; }
```

HistoryEntryMachineName

Overrides detected hostname in the generated history entry with free-form text.

Property value:

Type: System.String

Text to use for machine name information in new SFT version history entry.

Syntax:

```
public string HistoryEntryMachineName { get; set; }
```

HistoryEntry class

Defines package sequencing history information for SFT files version 4.0 and upwards.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class HistoryEntry
```

Note:

Version history information for versions saved in 4.5 format is provided by inherited class [HistoryEntry45](#). When package has originally been in the different format, history information will contain instances of both classes.

All properties in this class are read-only as [SftEncoder](#) creates new history entry automatically during package encoding unless [NO_HISTORY_GENERATION](#) –flag has been passed to encoding engine.

Properties

Account

Gets the account (user) name under which the package was saved in.

Syntax:

```
public string Account { get; }
```

Property value:

Type: System.String

Windows login account name.

Hostname

Gets the computer name where the package was saved in.

Syntax:

```
public string Hostname { get; }
```

Property value:

Type: System.String

Computer name.

OperatingSystem

Gets the operating system version information for computer where the package was saved in.

Syntax:

```
public WindowsInfo OperatingSystem { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.WindowsInfo](#)

Operating system environment information.

Processor

Gets the processor information of the computer where the package was saved in.

Syntax:

```
public ProcessorInfo Processor { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.ProcessorInfo](#)

Computer processor information.

SaveMode

Gets the mode in which the package version was saved in.

Syntax:

```
public PackageSaveMode SaveMode { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.PackageSaveMode](#)

Save mode of the package version.

Sequencer

Gets the Sequencer version under which the package was saved in.

Syntax:

```
public SequencerVersion SaveMode { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.PackageSaveMode](#)

Sequencer version information.

Note:

For packages versions encoded by Lib-V, this information is set to 4.5.0.1485 (RTM of Microsoft App-V 4.5) for version 4.5 packages and 4.0.0.0 for version 4.x packages.

SlowMachine

Gets the information if the computer where the package was saved in is categorized as slow by the Windows.

Syntax:

```
public bool SlowMachine { get; }
```

Property value:

Type: System.Boolean

true if the machine is slow, otherwise **false**.

Timestamp

Gets the timestamp for the history entry.

Syntax:

```
public DateTime Timestamp { get; }
```

Property value:

Type: System.DateTime

Timestamp of when the package version was saved.

Upgraded

Gets the information if the package version was an upgrade to an existing version.

Syntax:

```
public bool Upgraded { get; }
```

Property value:

Type: System.Boolean

true if the package was upgrade of the existing one, **false** for new packages.

Version

Gets the then-current version of package when it was saved in.

Syntax:

```
public UInt32 Version { get; }
```

Property value:

Type: System.UInt32

Package's version.

VersionId

Gets the unique version identifier for the package.

Syntax:

```
public Guid VersionId { get; }
```

Property value:

Type: System.Guid

Package's version identifier (GUID).

Note:

SoftGrid/App-V uses this information to track version lineage for upgraded packages.

HistoryEntry45 class

Defines package sequencing history information for SFT files version 4.5 and upwards.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class HistoryEntry45 : HistoryEntry
```

Note:

Version history information for versions saved in pre-4.5 format is provided by base class [HistoryEntry](#). When package has originally been in the different format, history information will contain instances of both classes.

All properties in this class are read-only as [SftEncoder](#) creates new history entry automatically during package encoding unless [NO_HISTORY_GENERATION](#) –flag has been passed to encoding engine.

Properties

DotNetVersion

Gets the .NET Framework version installed on the machine where the package was saved in.

Syntax:

```
public string DotNetVersion { get; }
```

Property value:

Type: System.String

.NET Framework version number, otherwise **null**.

Note:

If multiple versions are installed on the machine, this version reflects the most recent one.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

IEVersion

Gets the Windows Internet Explorer version installed on the machine where the package was saved in.

Syntax:

```
public string IEVersion { get; }
```

Property value:

Type: System.String

Windows Internet Explorer version number, otherwise **null**.

MediaPlayerVersion

Gets the Windows Media Player version installed on the machine where the package was saved in.

Syntax:

```
public string MediaPlayerVersion { get; }
```

Property value:

Type: System.String

Windows Media Player version number, otherwise **null**.

IntermediateValueManager class

Provides mapping functionality between normal values and values encoded with Intermediate Values (such as %SFT_MNT%, %CSIDL_WINDOWS%) as used in virtual registry and virtual filesystem.

Access to encoding and decoding methods goes through SFT version –specific instances returned by static method `Version()`, as specific supported mappings vary between different generations of SFTs.

Namespace:

GridMetric.LibV

Syntax:

```
public class IntermediateValueManager
```

Static methods

Version

Return SFT version specific instance of `IntermediateValueManager` class.

Syntax:

```
public static IntermediateValueManager Version(  
    SftFileformatVersion version  
)
```

Parameters:

version

Type: [GridMetric.LibV.Structures.SftFileformatVersion](#)

SFT package format against which the mappings are used .

Return value:

Type: [GridMetric.LibV.IntermediateValueManager](#)

Version –specific instance.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Static properties

AllUsersProfileDirectory

Gets path to a profile directory shared by all users.

Syntax:

```
public static string AllUsersProfileDirectory { get; }
```

Property value:

Type: System.String

File system path shared profile directory root.

CurrentSid

Gets string representation of the SID for current user account.

Syntax:

```
public static string CurrentSid { get; }
```

Property value:

Type: System.String

String representation of Windows' SID (Security Identifier) of current user.

IsX64

Returns information if code is executing in 64-bit system/context.

Syntax:

```
public static bool IsX64 { get; }
```

Property value:

Type: System.Boolean

true if environment is 64-bit, false if 32-bit.

ProfilesDirectory

Gets path to a Windows' profile directory.

Syntax:

```
public static string ProfilesDirectory { get; }
```

Property value:

Type: System.String

File system path to Windows profile directory root.

UserProfileDirectory

Gets path to a current user's profile directory.

Syntax:

```
public static string UserProfileDirectory { get; }
```

Property value:

Type: System.String

File system path to profile directory root of current user.

VirtualDrive

Gets or sets drive letter of Virtual Drive. This drive letter represents the drive that SoftGrid/App-V client creates and which Intermediate Value %SFT_MNT% points to.

Syntax:

```
public static string VirtualDrive { get; set; }
```

Property value:

Type: System.String

Drive letter used by the Virtual Drive.

Note:

Although Lib-V will automatically detect locally present virtual drive, this property provides setter for setting virtual drive explicitly if local machine does not have SoftGrid/App-V client installed. If no virtual drive has been defined, Intermediate Value of %SFT_MNT% does not resolve to anything.

Methods

DecodeString

Decodes (converts) string value that contains IM values into form which represents those values in expanded form. For example, %SFT_SID% will be expanded to current user's SID value, %CSIDL_PROGRAM_FILES% to Program Files –directory path on the present computer and so forth.

Syntax:

```
public string DecodeString(  
    string encodedString  
)
```

Parameters:

encodedString

Type: System.String

String value that contains zero or more Intermediate Values in it.

Return value:

Type: System.String

String value that has all Intermediate Values expanded to their local equivalence.

Note:

Since decoded Intermediate Values are representation on how paths, names etc. are on the computer under user account executing the code, these decoded values are not portable across users and computers.

EncodeString

Encodes (converts) string value that contains physical paths, names etc. into form where those values are replaced by IM values. For example, current user's SID will be encoded to %SFT_SID%, local Windows directory will be encoded to %CSIDL_WINDOWS% and so forth.

Syntax:

```
public string EncodeString(  
    string stringToEncode  
)
```

Parameters:

stringToEncode

Type: System.String

String value that needs to be converted to IM representation form.

Return value:

Type: System.String

String value that has all applicable values encoded into Intermediate Values representation.

GetParserItems

Returns list of currently active and defined intermediate value items, or parser items, in the format used by App-V SPRJ files. Each item in list is encapsulated in [ParserItem](#) –object.

Syntax:

```
public void GetParserItems(  
    out ParserItem[] parserItems  
)
```

Parameters:

parserItems

Type: [GridMetric.LibV.IntermediateValueManager.ParserItem\[\]](#)

Array of parser items currently active for specified instance of IntermediateValueManager.

Properties

ActiveMappings

Gets Intermediate Value mappings currently active in the present context.

Syntax:

```
public IDictionary<string,string> ActiveMappings { get; }
```

Property value:

Type: System.Collections.Generic.IDictionary<TKey,TValue>

Mapping relationship between Intermediate Values and equivalent decoded value, where collection's key represent IM value (e.g. %SFT_MNT%) and collection's value represent resolved form.

Note:

Returned list is a copy of actual mappings used internally by the class and any change to it is not reflected on the internal operations.

Lib-V class

Namespace:

GridMetric.LibV

Syntax:

```
public class LibV
```

Static methods

Dispose

Cleans temporary files and folders used by the Lib-V core assembly. This method should be called at latest during shutdown of the program utilizing Lib-V functionality to ensure disk-space is not consumed by temporary data left behind.

Syntax:

```
public static void Dispose()
```

SupplyLicense

Initialize package engine with valid license key.

Syntax:

```
public static void SupplyLicense(  
    string licenseKey  
)
```

Parameters:

licenseKey

Type: System.String

A valid Lib-V license key string.

Note:

If license key is not valid or it has not been entered, encoding/decoding functionality cannot be used.

Static fields

VERSION

Gets the product version number for the Lib-V.

Syntax:

```
public const string VERSION
```

Property value:

Type: System.String

Version number (major.minor.patch) for the Lib-V product.

FULLVERSION

Gets the complete product version number for the Lib-V.

Syntax:

```
public const string FULLVERSION
```

Property value:

Type: System.String

Full version number (major.minor.patch.build) for the Lib-V product.

ParserItem class

Data represented in this class can directly be transformed and written to associated SPRJ file for the package.

Namespace:

GridMetric.LibV.IntermediateValueManager

Syntax:

```
public class ParserItem
```

Properties

Context

Gets context in which this parser item is valid.

Syntax:

```
public string Context { get; }
```

Property value:

Type: System.String

Context for the parser item.

Pattern

Gets pattern (i.e. decoded value) for the parser item.

Syntax:

```
public string Pattern { get; }
```

Property value:

Type: System.String

Pattern for the parser item.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Replacement

Gets replacement (i.e. encoded value) for the parser item pattern.

Syntax:

```
public string Replacement { get; }
```

Property value:

Type: System.String

Replacement string for parser item.

Type

Gets mapping type for the parser item.

Syntax:

```
public string Type { get; }
```

Property value:

Type: System.String

Mapping type for the parser item.

ProcessorInfo class

Defines processor information for sequencing history entries implemented in class [HistoryEntry](#).

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class ProcessorInfo
```

Properties

Count

Gets the number of processors installed on the machine under which the package was saved in.

Syntax:

```
public UInt32 Count { get; }
```

Property value:

Type: System.UInt32

Number of processors in the machine.

Family

Gets the processor family of processors installed on the machine under which the package was saved in.

Syntax:

```
public ProcessorFamily Family { get; }
```

Property value:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Type: [GridMetric.LibV.Structures.ProcessorFamily](#)

Family of the processors.

Note:

For all current SoftGrid/App-V packages, this property always return processor family of [Pentium](#) for all 32-bit processors.

Level

Gets the stepping level of processors installed on the machine under which the package was saved in.

Syntax:

```
public UInt32 Level { get; }
```

Property value:

Type: System.UInt32

Processor stepping level.

Revision

Gets the revision of processors installed on the machine under which the package was saved in.

Syntax:

```
public UInt32 Revision { get; }
```

Property value:

Type: System.UInt32

Processor revision.

SequencerVersion class

Provides version information for [Sequencer](#) -property in [VirtualEnvironmentConfiguration](#) -class and [HistoryEntry](#) -class.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class SequencerVersion
```

Properties

Build

Gets the build version number.

Syntax:

```
public UInt16 Build { get; }
```

Property value:

Type: System.UInt16

Build version.

Major

Gets the major version number.

Syntax:

```
public UInt16 Major { get; }
```

Property value:

Type: System.UInt16

Major version.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Minor

Gets the minor version number.

Syntax:

```
public UInt16 Minor { get; }
```

Property value:

Type: System.UInt16

Minor version.

Patch

Gets the patch version number.

Syntax:

```
public UInt16 Patch { get; }
```

Property value:

Type: System.UInt16

Patch version.

Version

Gets the revision full version number (*major.minor.patch.build*) as string.

Syntax:

```
public string Version { get; }
```

Property value:

Type: System.String

Full version number.

SftDecoder class

Decodes serialized SFT file (i.e. SFT file in disk) into in-memory representation of [SftPackage](#) –object

Namespace:

GridMetric.LibV.Decoder

Syntax:

```
public class SftDecoder
```

Constructors

Initializes a new instance of the [SftDecoder](#) class.

Syntax:

```
public SftDecoder()
```

Methods

Decode

Decodes serialized SFT file into [SftPackage](#) object using specified options.

Syntax:

```
public SftPackage Decode(  
    SftDecodingOptions decodingOptions  
)
```

Parameters:

decodingOptions

Type: [GridMetric.LibV.Structures.SftDecodingOptions](#)

Specifies decoding flags to use.

Return value:

Type: [GridMetric.LibV.SftPackage](#)

Reference to a decoded package instance.

Note:

Unless otherwise specified with [SKIP_BLOCKSCAN](#) –flag, decoding process will scan each of the DATA –section blocks during decode process. For large files and/or files residing in network shares, this may increase decoding time significantly. If data block scanning is skipped, most of the block information (see: [VirtualFileBlock](#) –class) for files remains unknown until those blocks are accessed for the first time (effectively a Just-In-Time scanning).

After decoding, [SftPackage](#) object returned will hold reference to an stream used as source and closing the stream in other manners than using [Close\(\)](#) method on the package will cause it to throw exceptions when content is accessed.

Dispose

Disposes the decoder instance.

Syntax:

```
public void Dispose()
```

InitializeDecoder

Initializes a new decoder instance. This method must be called before calling [Decode\(\)](#).

Overload List:

Initializes decoder to use specified Stream as where from to read serialized SFT content.

```
public void InitializeDecoder(  
    Stream fileContent  
)
```

Initializes decoder to use specified Stream as where from to read serialized SFT content and specifies file system directory where package files are placed in during decoding operation if [DECODE FILESTRUCTURE](#) flag is passed to decoding engine.

```
public void InitializeDecoder(  
    Stream fileContent,  
    string decodeDirectory  
)
```

Initializes decoder to use specified path to a file as where from to read serialized SFT content.

```
public void InitializeDecoder(  
    string pathToSftFile  
)
```

Initializes decoder to use specified path to a file as where from to read serialized SFT content and specifies file system directory where package files are placed in during decoding operation if [DECODE FILESTRUCTURE](#) flag is passed to decoding engine.

```
public void InitializeDecoder(  
    string pathToSftFile,  
    string decodeDirectory  
)
```

Parameters:

fileContent

Type: System.IO.Stream

Stream to use as a source from where serialized SFT content is read in and decoded.

decodeDirectory

Type: System.String

Path to an existing directory into which decoding engine uses as temporary storage for files and directories decoded from the package. Decoding engine will not place any directories or files under this path unless [DECODE FILESTRUCTURE](#) flag is passed to it using Decode() method.

pathToSftFile

Type: System.String

Path to an SFT file from where serialized SFT content is read in and decoded.

Note:

InitializeDecoder() method just specifies necessary parameters to decoding engine, but it will not do actual decoding which is handled by [Decode\(\)](#) method.

Events

PackageLoadEnd

Occurs when decoding engine has finished package decoding/loading.

Syntax:

```
public event EventHandler PackageLoadEnd
```

PackageLoadProgress

Occurs periodically during package decoding, providing status indication on current decoding phase via [SftDecodingEventArgs](#).

Syntax:

```
public event EventHandler<SftDecoder.SftDecodingEventArgs>  
PackageLoadProgress
```

PackageLoadStart

Occurs when decoding engine has started package decoding/loading.

Syntax:

```
public event EventHandler PackageLoadStart
```


SftDecoder.SftDecodingEventArgs class

Provides status information on package's decoding phase. This class is implemented as an inner class for [SftDecoder](#) class.

Namespace:

GridMetric.LibV.Decoder

Syntax:

```
public class SftDecodingEventArgs
```

Properties

CurrentPhase

Gets the current decoding phase.

Syntax:

```
public SftDecodingPhase CurrentPhase { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftDecodingPhase](#)

Current phase.

Note:

Since SFT files contain several different virtual registries, [VirtualRegistryParse](#) –phase will appear multiple times one for each virtual registry instance parsed.

Percentage

Gets the completion percentage for phase specified by [CurrentPhase](#).

Syntax:

```
public byte Percentage { get; }
```

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Property value:

Type: System.Byte

Percentage.

SftEncoder class

Encodes in-memory [SftPackage](#) –object into serialized SFT file content.

Namespace:

GridMetric.LibV.Encoder

Syntax:

```
public class SftEncoder
```

Constructors

Initializes a new instance of the [SftEncoder](#) class.

Syntax:

```
public SftEncoder()
```

Methods

Encode

Serializes [SftPackage](#) object into SFT file using specified options and returns instance of new version of package.

Overload List:

Encodes package into specified stream.

```
public SftPackage Encode(  
    Stream encodeToStream,  
    SftEncodingOptions encodingOptions  
)
```

Encodes package into specified path.

```
public SftPackage Encode(  

```

```
        string encodeToFile,  
        SftEncodingOptions encodingOptions  
    )
```

Parameters:

encodeToStream

Type: System.IO.Stream

Stream to use as a target where serialized SFT content is written to. This stream must support writing.

encodeToFile

Type: System.String

Path to a file which will be written by encoding engine when package serializes .

encodingOption

Type: [GridMetric.LibV.Structures.SftEncodingOptions](#)

Specifies encoding flags to use.

Return value:

Type: [GridMetric.LibV.SftPackage](#)

Reference to a package that was created during serialization.

Note:

Encode method returns reference to a new package – which is otherwise an exact copy of the original one passed to [InitializeEncoder\(\)](#) method – because during encoding process all file-blocks will be re-placed in the SFT file, some package properties will change and internal reference to serialized file changes. This way, the old package reference that was passed in to the encoding engine will still retain all values as before encoding whereas returned package represents the new version of the package.

InitializeEncoder

Initializes a new decoder instance with encoding parameters. This method must be called before calling [Encode\(\)](#).

Overload List:

Initializes encoder to use specified package object and to use encoding parameters based on original package's properties.

```
public void InitializeEncoder(  
    SftPackage packageToEncode  
)
```

Initializes encoder to use specified package object and compression type and to use other encoding parameters based on original package's properties.

```
public void InitializeEncoder(  
    SftPackage packageToEncode,  
    SftCompressionType compressionToUse  
)
```

Initializes encoder to use specified package object and SFT file version and to use other encoding parameters based on original package's properties.

```
public void InitializeEncoder(  
    SftPackage packageToEncode,  
    SftFileformatVersion encodeAsVersion  
)
```

Initializes encoder to use specified package object, SFT file version and compression type and to use other encoding parameters based on original package's properties.

```
public void InitializeEncoder(  
    SftPackage packageToEncode,  
    SftFileformatVersion encodeAsVersion,  
    SftCompressionType compressionToUse  
)
```

Initializes encoder to use specified package object, SFT file version and block size and to use other encoding parameters based on original package's properties.

```
public void InitializeEncoder(  

```

```

        SftPackage packageToEncode,
        SftFileformatVersion encodeAsVersion,
        uint blockSizeToUse
    )

```

Initializes encoder to use specified package object, SFT file version, block size and compression type.

```

public void InitializeEncoder(
    SftPackage packageToEncode,
    SftFileformatVersion encodeAsVersion,
    uint blockSizeToUse,
    SftCompressionType compressionToUse
)

```

Initializes encoder to use specified package object and block size and to use other encoding parameters based on original package's properties.

```

public void InitializeEncoder(
    SftPackage packageToEncode,
    uint blockSizeToUse
)

```

Initializes encoder to use specified package object and additional encoding parameters, and to use other encoding parameters based on original package's properties.

```

public void InitializeEncoder(
    SftPackage packageToEncode,
    EncodingAdditionalParameters additionalParameters
)

```

Initializes encoder to use specified package object, SFT file version and additional encoding parameters, and to use other encoding parameters based on original package's properties.

```

public void InitializeEncoder(
    SftPackage packageToEncode,

```

```

        SftFileformatVersion encodeAsVersion,
        EncodingAdditionalParameters additionalParameters
    )

```

Initializes encoder to use specified package object, block size and additional encoding parameters, and to use other encoding parameters based on original package's properties.

```

public void InitializeEncoder(
    SftPackage packageToEncode,
    uint blockSizeToUse,
    EncodingAdditionalParameters additionalParameters
)

```

Initializes encoder to use specified package object, compression type and additional encoding parameters, and to use other encoding parameters based on original package's properties.

```

public void InitializeEncoder(
    SftPackage packageToEncode,
    SftCompressionType compressionToUse,
    EncodingAdditionalParameters additionalParameters
)

```

Initializes encoder to use specified package object, SFT file version, block size and additional encoding parameters, and to use other encoding parameters based on original package's properties.

```

public void InitializeEncoder(
    SftPackage packageToEncode,
    SftFileformatVersion encodeAsVersion,
    uint blockSizeToUse,
    EncodingAdditionalParameters additionalParameters
)

```

Initializes encoder to use specified package object, SFT file version, compression type and additional encoding parameters, and to use other encoding parameters based on original package's properties.

```
public void InitializeEncoder(  
    SftPackage packageToEncode,  
    SftFileformatVersion encodeAsVersion,  
    SftCompressionType compressionToUse,  
    EncodingAdditionalParameters additionalParameters  
)
```

Parameters:

packageToEncode

Type: [GridMetric.LibV.SftPackage](#)

Package object which to serialize.

encodeAsVersion

Type: [GridMetric.LibV.Structures.SftFileformatVersion](#)

SFT package format to use for serialized file .

blockSizeToUse

Type: System.UInt32

Block size setting to use for serialized package's DATA block. Valid values for block size are 1024 – 65536 bytes (1 to 64 kilobytes).

compressionToUse

Type: [GridMetric.LibV.Structures.SftCompressionType](#)

Compression type to apply for package. While technically Lib-V will allow for it, for 4.5 packages compression method of BZIP2 has been deprecated and might not be supported at all by Microsoft Application Virtualization Sequencer or client in the future.

additionalParameters

Type: [GridMetric.LibV.Encoder.EncodingAdditionalParameters](#)

Additional encoding parameters.

Note:

InitializeEncoder() method supplies necessary parameters for encoding engine, but it will not start actual encoding which is handled by [Encode\(\)](#) method.

Properties

EncodingBlocksize

Gets the block size set for encoding.

Syntax:

```
public uint EncodingBlocksize { get; }
```

Property value:

Type: System.UInt32

Block size setting for serialization.

EncodingCompression

Gets the compression type set for encoding.

Syntax:

```
public SftCompressionType EncodingCompression { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftCompressionType](#)

Compression setting for serialization.

EncodingFormat

Gets the SFT file format set for encoding.

Syntax:

```
public SftFileformatVersion EncodingFormat { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftFileformatVersion](#)

SFT version setting for serialization.

Events

PackageEncodingEnd

Occurs when encoding engine has finished package encoding.

Syntax:

public event EventHandler PackageEncodingEnd

PackageEncodingProgress

Occurs periodically during package encoding, providing status indication on current encoding phase via [SftEncodingEventArgs](#).

Syntax:

public event EventHandler<[SftEncoder.SftEncodingEventArgs](#)>
PackageEncodingProgress

PackageEncodingStart

Occurs when encoding engine has started package encoding.

Syntax:

public event EventHandler PackageEncodingStart

SftEncoder.SftEncodingEventArgs class

Provides status information on package's encoding phase. This class is implemented as an inner class for [SftEncoder](#) class.

Namespace:

GridMetric.LibV.Encoder

Syntax:

```
public class SftEncodingEventArgs
```

Properties

CurrentPhase

Gets the current encoding phase.

Syntax:

```
public SftEncodingPhase CurrentPhase { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftEncodingPhase](#)

Current phase.

Note:

Since SFT files contain several different virtual registries, [VirtualRegistryEncoding](#) –phase will appear multiple times; one for each virtual registry instance encoded.

Percentage

Gets the completion percentage for phase specified by [CurrentPhase](#).

Syntax:

```
public byte Percentage { get; }
```

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Property value:

Type: System.Byte

Percentage.

SftPackageCreationInfo class

Specifies necessary information for new package creation when passed to [SftPackage.Create\(\)](#). At minimum, [FileformatVersion](#) and [RootDirectoryName](#) has to be specified.

Namespace:

GridMetric.LibV

Syntax:

```
public class SftPackageCreationInfo
```

Properties

FileformatVersion

Specifies file format version to use for the new package.

Syntax:

```
public SftFileformatVersion FileformatVersion { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftFileformatVersion](#)

SFT's file format version.

RootDirectoryName

Specifies long directory name to use for the package's internal root directory. To set explicit short name for root directory, use [SftPackageAdvancedCreationInfo](#) class instead.

Syntax:

```
public string RootDirectoryName { get; set; }
```

Property value:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Type: System.String

Long name for root directory.

RootDirectoryPath

Specifies existing directory to use as source for root directory. This will take names and security attributes from specified directory but will not traverse contents for inclusion into package.

Syntax:

```
public string RootDirectoryPath { get; set; }
```

Property value:

Type: System.String

Existing directory path to use as source.

ScratchRoot

Specifies existing directory to use as scratch directory for package when it is used in-memory. This scratch directory will hold all temporary files created against package before files are encoded to SFT file.

Syntax:

```
public string ScratchRoot { get; set; }
```

Property value:

Type: System.String

Existing directory path to use as scratch directory.

SftPackageAdvancedCreationInfo class

Specifies extended information for new package creation when passed to [SftPackage.Create\(\)](#) instead of using [SftPackageCreationInfo](#) class.

Namespace:

GridMetric.LibV

Syntax:

```
public class SftPackageAdvancedCreationInfo : SftPackageCreationInfo
```

Properties

PackageId

Specifies explicit unique identifier to use for new SFT package. Otherwise GUID is randomly created when creating new package.

Property value:

Type: System.Guid

GUID value to use as package unique ID.

Syntax:

```
public Guid PackageId { get; set; }
```

RootDirectoryShortName

Specifies explicit short name (8+3 DOS format) for root directory. Otherwise automatically generated short name will be used.

Syntax:

```
public string RootDirectoryShortName { get; set; }
```

Property value:

Type: System.String

Short name for root directory.

SftPackage class

Provides abstraction on SFT file's content i.e. SoftGrid/App-V logical package.

Namespace:

GridMetric.LibV

Syntax:

```
public class SftPackage
```

Static methods

Create

Creates and initializes new empty SoftGrid/App-V package object.

Syntax:

```
public static SftPackage Create(  
    SftPackageCreationInfo creationInfo  
)
```

Parameters:

creationInfo

Type: [GridMetric.LibV.SftPackageCreationInfo](#)

Specifies necessary information to initialize new package.

Return value:

Type: [GridMetric.LibV.SftPackage](#)

An empty SoftGrid/App-V package object.

Methods

Close

Closes a package, frees all resources (such as temporary files) related to that package and disposes it. This method should be called as a last operation for all [SftPackage](#) objects used.

Syntax:

```
public void Close()
```

CreateClone

Clones SftPackage instance to a new independent package object by deep copying all objects referenced in the package. This method can be used for making snapshot of the existing SftPackage.

Syntax:

```
public static SftPackage CreateClone()
```

Return value:

Type: [GridMetric.LibV.SftPackage](#)

Package copy of the original.

Note:

While cloning creates an independent copy otherwise, all temporary files that are referenced by branched/new files by the original will be used for the cloned package. Therefore, calling [Close\(\)](#) on either of the packages will invalidate those references to other package as well, causing possible exception when trying to continue working with such content.

Dispose

This method should not be called directly, you can achieve same end-result by calling [Close\(\)](#).

Syntax:

```
public void Dispose()
```

SetDefaultDirectorySecurity

Assigns default security descriptors as per App-V 4.5 file format to the certain standard files and directories in package's internal directory structure.

Syntax:

```
public void SetDefaultDirectorySecurity()
```

Note:

This method does not need to be called when creating 4.5 packages as [Create](#) method will take care of that. When converting older packages to 4.5 file format, packages won't by default have security for other directories or files either so this method does not need to be called.

VirtualizeFromFilesystem

Performs import operation on real files and directories on the file system so that they will be virtualized into package either as normal mounted directories or alternatively as Virtual Filesystem (VFS) directories.

Overload List:

Virtualizes directory structure or file from file system and mounts it under existing virtual directory in the package.

```
public void VirtualizeFromFilesystem(  
    string pathToDirectoryOrFile,  
    VirtualDirectory packageFilesystemMountPoint,  
    bool getSecurity,  
    bool skipInaccessible,  
    bool createCopy,
```

```

        bool overrideExisting
    )

```

Virtualizes directory structure or file from file system and mounts it under existing virtual directory in the package, with the optional import of the top level directory.

```

public void VirtualizeFromFilesystem(
    string pathToDirectoryOrFile,
    VirtualDirectory packageFilesystemMountPoint,
    bool getSecurity,
    bool skipInaccessible,
    bool createCopy,
    bool overrideExisting
    bool importTopDirectory
)

```

Virtualizes directory structure or file from file system as Virtual Filesystem (VFS) entry and creates necessary mappings during import.

```

public void VirtualizeFromFilesystem(
    string pathToDirectoryOrFile,
    bool getSecurity,
    bool skipInaccessible,
    bool createCopy,
    bool overrideExisting
)

```

Parameters:

pathToDirectoryOrFile

Type: System.String

Specifies path to existing directory or file that is to be virtualized.

packageFilesystemMountPoint

Type: [GridMetric.LibV.Structures.VirtualDirectory](#)

Reference to existing virtual directory in the package under which imported directories or file will be place to.

getSecurity

Type: System.Boolean

true to read in and include security descriptors, otherwise **false**.

skipInaccessible

Type: System.Boolean

true to silently skip all files and directories which could not be accessed, otherwise **false** to generate exception when unvirtualizable file or directory is encountered.

createCopy

Type: System.Boolean

true to make temporary copy of all files that are getting included in to, otherwise **false** to use original file path as source. If original paths are used, possible changes in virtualized files – after VirtualizeFromFilesystem has been run and before SFT package is encoded – may result failures during encoding as files' actual content are accessed and read in at that point.

overrideExisting

Type: System.Boolean

true to override possibly conflicting files already in the package using the same path, otherwise **false** to preserve existing file.

importTopDirectory

Type: System.Boolean

true to import the top level directory to the specified mount point along with the contents, **false** to import only the contents but not the top level directory itself. This setting has no effect when importing individual files.

Note:

While making temporary copies of files during virtualization prevents issues at the package encoding phase, doing so will increase time to execute this method as all of the content of files has to be copied on-the-fly to the temporary files. On the other hand, making temporary copies will prevent access denied –type of errors when some other process has locked the file in-between (skipping inaccessible files affects only execution of this method

and not encoding phase) or when content of the file has been modified by external process.

This method has two overloads for two different scenarios: when existing file/directory structure needs to be put under arbitrary package path or when virtualized files and directories has to appear as virtualized over the real file system when package is executed on client.

If files and directories are virtualized using VFS, any intermediate directories (such as CSIDL_PROGRAM_FILES) are automatically created under VFS subdirectory in the package's internal directory-structure and marked as non-virtualized (merge) whereas all imported directories are assigned to fully virtualized (override) status.

Note that if you virtualize directories and files with explicit mount point and if that mount point happens to be under any existing directory under VFS structure, directories and files will inherit mapping information from the parent directory (e.g. files virtualized to mount point of ...\\VFS\\CSIDL_PROGRAM_FILES\\ will also be mapped to the Program Files directory).

VirtualizeFromRegistry

Performs import operation on keys and values from real registry into virtual registry.

Overload List:

Virtualizes specified registry key and all the subkeys into package.

```
public void VirtualizeFromRegistry(  
    RegistryKey keyToVirtualize,  
    bool virtualizeInaccessible,  
    bool overrideExisting  
)
```

Virtualizes specified registry key with option to control subkey recursion.

```
public void VirtualizeFromRegistry(  
    RegistryKey keyToVirtualize,  
    bool virtualizeInaccessible,  
    bool overrideExisting,  
    bool recursion
```

)

Parameters:

keyToVirtualize

Type: Microsoft.Win32.RegistryKey

Specifies path to existing directory or file that is to be virtualized.

virtualizeInaccessible

Type: System.Boolean

true to indicate that those keys that cannot be accessed are included as empty keys into package, otherwise **false** to generate exception when inaccessible registry key is encountered.

overrideExisting

Type: System.Boolean

true to override possibly conflicting (by name) keys or values already in the package's virtual registry, otherwise **false** to preserve existing key or value.

recursion

Type: System.Boolean

true to virtualize (import) the whole branch underneath specified registry key, otherwise **false** to virtualize just the specified key and all the values in stores.

Properties

FileformatType

Specifies file format type used by the package.

Syntax:

```
public SftFileformatType FileformatType { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftFileformatType](#)

File format type information.

FileformatVersion

Specifies file format version used by the package.

Syntax:

```
public SftFileformatVersion FileformatVersion { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftFileformatVersion](#)

File format version information.

Filesystem

Provides access to package's internal directory structure and file system –related properties.

Syntax:

```
public SftPackageFilesystem Filesystem { get; }
```

Property value:

Type: [GridMetric.LibV.SftPackageFilesystem](#)

[SftPackageFilesystem](#) object instance.

IsDecodedToFilesystem

Gets a value that determines if the package was decoded to real directory structure on disk by the [SftDecoder](#).

Syntax:

```
public bool IsDecodedToFilesystem { get; }
```

Property value:

Type: System.Boolean

true if package's internal directory structure was decoded in entirety to disk, otherwise **false**.

Properties

Provides access to package's overall properties.

Syntax:

```
public SftPackageProperties Properties { get; }
```

Property value:

Type: [GridMetric.LibV.SftPackageProperties](#)

[SftPackageProperties](#) object instance.

ScratchRoot

Gets path to current scratch directory for package.

Syntax:

```
public string Properties { get; }
```

Property value:

Type: System.String

Absolute path to temporary directory (scratch directory) used by the package.

Security

Provides access to package's security -related properties.

Syntax:

```
public SftPackageSecurity Security { get; }
```

Property value:

Type: [GridMetric.LibV.SftPackageSecurity](#)

[SftPackageSecurity](#) object instance.

Tag

Gets or sets the object that contains data about the SFT package.

Syntax:

```
public object Tag { get; set; }
```

Property value:

Type: System.Object

Object to associate with SFT package.

Note: Lib-V does not use this property and it will not be copied by Lib-V if package object is cloned.

VersionHistory

Provides access to sequencing/packaging history of the package. Applies to version 4.0 and newer packages.

Syntax:

```
public IList<HistoryEntry> VersionHistory { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of history entries representing each past save of the package.

VirtualEnvironment

Provides access to package's virtual environment, such as virtual registry.

Syntax:

```
public VirtualEnvironmentConfiguration VirtualEnvironment { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualEnvironmentConfiguration](#)

VirtualEnvironmentConfiguration object instance.

Note:

If package does not contain virtual environment, or it couldn't be parsed successfully, this property might return **null**.

Events

FileImported

Occurs when file has been added into package as a result of [VirtualizeFromFilesystem](#) operation.

Syntax:

```
public event EventHandler<FilesystemVirtualizationEventArgs>  
FileImported
```

FileImportFailed

Occurs when file couldn't be added into package as a result of [VirtualizeFromFilesystem](#) operation.

Syntax:

```
public event EventHandler<FilesystemVirtualizationEventArgs>  
FileImportFailed
```

FileImportSkipped

Occurs when file was skipped as a result of [VirtualizeFromFilesystem](#) operation.

Syntax:

```
public event EventHandler<FilesystemVirtualizationEventArgs>  
FileImportSkipped
```

FilesystemImportBegin

Occurs when [VirtualizeFromFilesystem](#) method is starting directory/file importing.

Syntax:

```
public event EventHandler FilesystemImportBegin
```

FilesystemImportEnd

Occurs when [VirtualizeFromFilesystem](#) method has completed directory/file importing.

Syntax:

```
public event EventHandler FilesystemImportEnd
```

RegistryKeyImported

Occurs when registry key has been added into package as a result of [VirtualizeFromRegistry](#) operation.

Syntax:

```
public event EventHandler<RegistryVirtualizationEventArgs>  
RegistryKeyImported
```

RegistryKeyImportFailed

Occurs when registry key couldn't be added into package as a result of [VirtualizeFromRegistry](#) operation.

Syntax:

```
public event EventHandler<RegistryVirtualizationEventArgs>  
RegistryKeyImportFailed
```

RegistryImportBegin

Occurs when [VirtualizeFromRegistry](#) method is starting registry importing.

Syntax:

```
public event EventHandler RegistryImportBegin
```

RegistryImportEnd

Occurs when [VirtualizeFromRegistry](#) method has completed registry importing.

Syntax:

```
public event EventHandler RegistryImportEnd
```

SftPackage.FilesystemVirtualizationEventArgs class

Provides status information on file/directory import operation. This class is implemented as an inner class for [SftPackage](#) class.

Namespace:

GridMetric.LibV

Syntax:

```
public class FilesystemVirtualizationEventArgs : EventArgs
```

Properties

FileName

Gets the file or directory event is related to.

Syntax:

```
public string FileName { get; }
```

Property value:

Type: System.String

Name of the entry.

IsDirectory

Is entry event relates to a directory.

Syntax:

```
public bool IsDirectory { get; }
```

Property value:

Type: System.Boolean

True if entry is a directory, false if file.

SftPackage.RegistryVirtualizationEventArgs class

Provides status information on registry import operation. This class is implemented as an inner class for [SftPackage](#) class.

Namespace:

GridMetric.LibV

Syntax:

```
public class RegistryVirtualizationEventArgs : EventArgs
```

Properties

Path

Gets the registry key path event is related to.

Syntax:

```
public string Path { get; }
```

Property value:

Type: System.String

Path to a registry key.

SftPackageProperties class

Provides access to [SftPackage](#)'s general properties.

Namespace:

GridMetric.LibV

Syntax:

```
public class SftPackageProperties
```

Properties

BitRate

Gets bitrate used by the package.

Syntax:

```
public UInt32 BitRate { get; }
```

Property value:

Type: System.UInt32

Bitrate used by the package.

Note:

This value is not used by current SoftGrid/App-V systems and access to it is provided only for backwards compatibility reasons.

BlockCount

Gets number of DATA blocks contained in the package.

Syntax:

```
public UInt32 BlockCount { get; }
```

Property value:

Type: System.UInt32

Total number of DATA blocks in the package.

Note:

This value is updated only during package decoding and it is not updated real-time and/or during package encoding.

BlockSize

Gets the data block size used by the package. This value specifies on how small chunks (or blocks) the file data contained in SFT file is divided to.

Syntax:

```
public UInt32 BlockSize { get; }
```

Property value:

Type: System.UInt32

Block size used by the package.

Note:

You cannot set package's block size by using this property but rather by passing desired block size to [InitializeEncoder\(\)](#) method.

Compression

Gets the compression type used in the package.

Syntax:

```
public SftCompressionType Compression { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftCompressionType](#)

Compression type used.

Note:

You cannot set package's compression type by using this property but rather by passing desired compression type to [InitializeEncoder\(\)](#) method.

CreationTimeUtc

Gets the original creation time (in UTC timezone) of the package.

Syntax:

```
public DateTime CreationTimeUtc { get; }
```

Property value:

Type: System.DateTime

Creation timestamp for package.

ModificationTimeUtc

Gets the latest modification time (in UTC timezone) of the package.

Syntax:

```
public DateTime ModificationTimeUtc { get; }
```

Property value:

Type: System.DateTime

Modification timestamp for package.

Note:

This property does not provide setter for setting timestamp manually. [SftEncoder](#) will increase the last modification timestamp at the time of encoding, unless

[NO_TIMESTAMP_INCREMENT](#) flag is passed into encoder engine in which case modification timestamp is not automatically updated.

PackageId

Gets the unique identifier (GUID) for the package. This value differentiates SFT packages from each other from SoftGrid/App-V system's standpoint.

Syntax:

```
public Guid PackageId { get; }
```

Property value:

Type: System.Guid

Package's unique ID.

StreamId

Gets the identification number for package stream.

Syntax:

```
public UInt32 StreamId { get; }
```

Property value:

Type: System.UInt32

Stream identification number.

Note:

This value is not used by current or past SoftGrid/App-V systems and access to it is provided only for information purposes.

Version

Gets the version number for the package.

Syntax:

```
public UInt32 Version { get; }
```

Property value:

Type: System.UInt32

Current package version.

Note:

For newly created package this value is initially set to zero and is increased to 1 by the [SftEncoder](#) when encoding the package. Version number is automatically incremented on each encoding/save unless [NO_VERSION_INCREMENT](#) flag is passed into encoder engine.

VersionId

Gets the version identifier for the current version of the package. This value differentiates different revisions of the same SFT package from each other from SoftGrid/App-V system's standpoint.

Syntax:

```
public Guid? VersionId { get; }
```

Property value:

Type: System.Nullable<T>

Current package version. For 3.x packages this property returns **null**.

Note:

Version identifier is automatically incremented on each encoding/save unless [NO_HISTORY_GENERATION](#) flag is passed into encoder engine.

SftPackageSecurity class

Provides access to [SftPackage](#)'s security –related properties.

Namespace:

GridMetric.LibV

Syntax:

```
public class SftPackageSecurity
```

Properties

IsAclsEnforced

Gets or sets enforcement status of file system ACLs for App-V 4.5 packages.

Syntax:

```
public bool IsAclsEnforced { get; set; }
```

Property value:

Type: System.Boolean

true to enforce security descriptors present in the package on the client, otherwise **false** to make client ignore them.

Note:

This setting has no effect on pre-4.5 packages.

IsAclsPresent

Gets the information whether or not package contains security descriptors.

Syntax:

```
public bool IsAclsPresent { get; }
```

Property value:

Type: System.Boolean

true if package has been saved with (some) security descriptors (4.5 packages), otherwise **false** if no security descriptors are present.

SftPackageFilesystem class

Provides access to [SftPackage](#)'s filesystem-related properties, such as internal directory structure.

Namespace:

GridMetric.LibV

Syntax:

```
public class SftPackageFilesystem
```

Properties

CreationTimeUtc

Gets the original creation time (in UTC timezone) of the package's file system.

Syntax:

```
public DateTime CreationTimeUtc { get; }
```

Property value:

Type: System.DateTime

Creation timestamp for the file system.

DirectoryRoot

Gets the root directory of the package's internal directory structure. This is an entry point for accessing all the files and directories contained within the package.

Syntax:

```
public VirtualDirectory DirectoryRoot { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualDirectory](#)

Virtual directory representing the package's root directory (sometimes referred to as asset directory).

DirectorySize

Gets the size of package's directory map (i.e. directory entries). This map size is sometimes referred to as "maximum block size", although it bears no relation to actual file data blocks used in the package.

Syntax:

```
public UInt32 DirectorySize { get; }
```

Property value:

Type: System.UInt32

Directory map size in bytes.

Note:

This value is not updated in real-time during modification operations against the [SftPackage](#). It is only updated during encoding/decoding operation.

LaunchDataSize

Gets the initial launch size for the package. This value is amount of data contained in the Feature Block 1 (FB1) part of the package.

Syntax:

```
public UInt32 LaunchDataSize { get; }
```

Property value:

Type: System.UInt32

Size of the Feature Block 1.

Note:

This value is not updated in real-time during modification operations against the [SftPackage](#). It is only updated during encoding/decoding operation.

ModificationTimeUtc

Gets the latest modification time (in UTC timezone) of the package's file system.

Syntax:

```
public DateTime ModificationTimeUtc { get; }
```

Property value:

Type: System.DateTime

Modification timestamp for the file system.

Note:

This property does not provide setter for setting timestamp manually. [SftEncoder](#) will increase the last modification timestamp at the time of encoding, unless [NO_TIMESTAMP_INCREMENT](#) flag is passed into encoder engine in which case modification timestamp is not automatically updated.

PackageCompressedDataSize

Gets the compressed size of the all file data contained in the package.

Syntax:

```
public UInt32 PackageCompressedDataSize { get; }
```

Property value:

Type: System.UInt32

Compressed size of the file data in package

Note:

This value is not updated in real-time during modification operations against the [SftPackage](#). It is only updated during decoding operation.

For packages not having compression (ZLIB or deprecated BZIP2) applied to it, this size equals [PackageDataSize](#) property.

PackageDataSize

Gets the size of the all file data contained in the package.

Syntax:

```
public UInt32 PackageDataSize { get; }
```

Property value:

Type: System.UInt32

Size of the file data in package

Note:

This value is not updated in real-time during modification operations against the [SftPackage](#). It is only updated during encoding/decoding operation.

Version

Gets the version number for the package file system.

Syntax:

```
public UInt32 Version { get; }
```

Property value:

Type: System.UInt32

Current package file system version.

Note:

For newly created package this value is initially set to zero and is increased to 1 by the [SftEncoder](#) when encoding the package. Version number is automatically incremented on each encoding/save unless [NO_VERSION_INCREMENT](#) flag is passed into encoder engine.

VfsMappingTarget class

Provides mapping information for entries defined in Virtual Filesystem (VFS). This mapping information provides information to the SoftGrid/App-V on how the entry in the package's internal directory structure should be mapped on top of the real file system present on the client.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VfsMappingTarget
```

Properties

Attributes

Gets or sets Virtual Filesystem –specific attributes for the mapping entry.

Syntax:

```
public VirtualFilesystemAttributes Attributes { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualFilesystemAttributes](#)

Attributes describing the general characteristics of the mapping entry.

Note:

All VFS entries should set [Entry](#) –attribute at minimum, and all VFS directories should set [Directory](#) –attribute in addition. If you want to make entry appear as deleted (i.e. not appearing in VE and also masking any local equivalent file/directory), [Deleted](#) –attribute should be set.

MappingAttributes

Gets or sets attributes related to file system mapping.

Syntax:

```
public VirtualFilesystemMappingAttributes MappingAttributes { get;  
set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualFilesystemMappingAttributes](#)

Attributes describing the characteristics of the mapping operation.

RawVirtualPath

Gets the target path where entry will be mapped to in raw format.

Syntax:

```
public string RawVirtualPath { get; }
```

Property value:

Type: System.String

Fully qualified target path in raw format.

Note:

If VirtualPath contains intermediate values (such as, "%CSIDL_WINDOWS%\my.dll"), [MappingAttributes](#) -property has to contain [TargetHasIntermediateValue](#) -attribute in it in order to make client resolve the path correctly.

VirtualPath

Gets or sets the target path where entry will be mapped to.

Syntax:

```
public string VirtualPath { get; set; }
```

Property value:

Type: System.String

Fully qualified target path.

Note:

If VirtualPath contains intermediate values (such as, "%CSIDL_WINDOWS%\my.dll"), [MappingAttributes](#) -property has to contain [TargetHasIntermediateValue](#) -attribute in it in order to make client resolve the path correctly.

This property will automatically add and remove trailing \??\ in the path to make it as valid raw path.

VirtualDirectory class

Represents a directory in the package's internal directory structure.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualDirectory : VirtualDirectoryEntry, IDisposable
```

Constructors

Overload List:

Creates a new virtual directory using the specified name and initializes the directory entry with default attributes.

```
public VirtualDirectory(  
    string directoryName,  
    SftPackage parentPackage  
)
```

Creates a new virtual directory using the specified long and short names and initializes the directory entry with default attributes.

```
public VirtualDirectory(  
    string directoryName,  
    string shortName,  
    SftPackage parentPackage  
)
```

Creates a new virtual directory using the specified file system path to an existing directory as source for the names and attributes.

```
public VirtualDirectory(  
    string physicalDirectoryPath,  
    SftPackage parentPackage,
```

```
        bool getSecurity  
    )
```

Parameters:

directoryName

Type: System.String

Specifies a name for the virtual directory.

shortName

Type: System.String

Specified an explicit short name for the virtual directory.

parentPackage

Type: [GridMetric.LibV.SftPackage](#)

Package instance which will host this virtual directory.

physicalDirectoryPath

Type: System.String

Path to an existing physical directory from where to read directory entry attributes from.

getSecurity

Type: System.Boolean

true to read in and include security descriptors from the specified existing physical directory, otherwise **false**.

Note:

After instantiating new [VirtualDirectory](#), it has to be mounted in the package's directory structure by calling [Add\(\)](#) method on suitable [VirtualDirectory](#) already present in the package.

Methods

Add

Adds a new file or directory to another directory.

Syntax:

```
public void Add(  
    VirtualDirectoryEntry entryToAdd  
)
```

Parameters:

entryToAdd

Type: [GridMetric.LibV.Structures.VirtualDirectoryEntry](#)

New [VirtualDirectory](#) or [VirtualFile](#) entry to add to this directory.

Note:

When adding new virtual directory or file to a directory that contains VFS mapping information, Lib-V tries automatically to update VFS information on that newly added entry. For instance, VFS mode is updated accordingly and VFS mappings are created to reflect parent directory's VFS mappings.

This method is thread-safe.

Dispose

Disposes resources reserved by the virtual directory, including all virtual directories and files beneath this directory. This method does not need to be called directly on directories mounted in the package as closing the package will automatically dispose all virtual directories. If the virtual directory is not mounted, it should be explicitly disposed to ensure all resources are released.

Syntax:

```
public void Dispose()
```

GetDirectory

Gets virtual directory by the specified name or path from the current directory.

Syntax:

```
public VirtualDirectory GetDirectory(  
    string directoryName  
)
```

Parameters:

directoryName

Type: System.String

Directory name or path to a file. If path is specified (e.g. "subdir\another"), the last directory in the path is returned.

Return value:

Type: [GridMetric.LibV.Structures.VirtualDirectory](#)

Virtual directory if found by the specified name or path, otherwise returns **null**.

Note:

Directory is searched in case-insensitive manner.

GetFile

Gets virtual file by the specified name or path from the current virtual directory.

Syntax:

```
public VirtualFile GetFile(  
    string fileName  
)
```

Parameters:

fileName

Type: System.String

File name or path to a file.

Return value:

Type: [GridMetric.LibV.Structures.VirtualFile](#)

Virtual file if found by the specified name or path, otherwise returns **null**.

Note:

File is searched in case-insensitive manner.

HasDirectory

Checks if the virtual directory by the specified name exists under the current virtual directory.

Syntax:

```
public bool HasDirectory(  
    string directoryName  
)
```

Parameters:

directoryName

Type: System.String

Directory name.

Return value:

Type: System.Boolean

true if directory by the specified name is found, otherwise **false**.

Note:

Name is searched in case-insensitive manner.

HasFile

Checks if the virtual file by the specified name exists under the current virtual directory.

Syntax:

```
public bool HasFile(  
    string fileName  
)
```

Parameters:

fileName

Type: System.String

File name.

Return value:

Type: System.Boolean

true if file by the specified name is found, otherwise **false**.

Note:

Name is searched in case-insensitive manner.

Remove

Removes (deletes) directory or file by the specified name from the current virtual directory.

Syntax:

```
public void Remove (  
    string fileOrDirectoryName  
)
```

Parameters:

fileOrDirectoryName

Type: System.String

Name of the file or directory to remove.

Note:

Name is matched in case-insensitive manner.

This method is thread-safe.

Properties

Directories

Gets read-only list of virtual directories this directory currently holds.

Syntax:

```
public IList<VirtualDirectory> Directories { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of virtual directories entries.

Note:

This property returns read-only copy of the list of subdirectories. If additions or deletions to subdirectories is needed, use [Add\(\)](#) and [Remove\(\)](#) methods.

Files

Gets read-only list of virtual files this directory currently holds.

Syntax:

```
public IList<VirtualFile> Files { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of virtual file entries.

Note:

This property returns read-only copy of the list of files. If additions or deletions to files is needed, use [Add\(\)](#) and [Remove\(\)](#) methods.

Name

Gets or sets the long filename for virtual directory.

Syntax:

```
public new string Name { get; set; }
```

Property value:

Type: System.String

Long filename for virtual directory.

ShortName

Gets or sets the short filename (8+3 DOS format) for virtual directory.

Syntax:

```
public new string ShortName { get; set; }
```

Property value:

Type: System.String

Short filename for virtual directory.

VfsMode

Gets or sets the Virtual Filesystem (VFS) mode for the virtual directory.

Syntax:

```
public new VirtualFilesystemMode VfsMode { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualFilesystemMode](#)

Mode describing how the entry is mapped according to VFS functionality

Note:

Only virtual directories that have mode of [Override](#) will actually be written to virtual environment configuration file, see notes on [MapsTo](#) property.

All directories that needs to be mapped using Virtual Filesystem has to be placed under VFS subdirectory under package's root directory. It's on caller's responsibility to make necessary adjustments both to VfsMode property and to mapping list in order to virtualize entries with VFS. Alternatively, you can use [VirtualizeFromFilesystem \(\)](#) method of the [SftPackage](#) object to import entries from the real file system and have an automatic VFS entry creation as a result.

VfsMode property on the virtual directory will perform propagation of VFS mode change to all subdirectories. If virtual directory is changed to [Merge](#), parent all parent and grandparent directories having [Override](#) mode are changed to [Merge](#) as overridden directories cannot contain merged directories.

VirtualDirectoryEntry class

Provides common abstract base for both VirtualFile and VirtualDirectory classes. All files and directories contained within [SftPackage](#) and accessible from [DirectoryRoot](#) –property are defined through these three classes.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public abstract class VirtualDirectoryEntry
```

Methods

GetDecodedMappingPaths

Processes directory entry's VFS mappings and returns list of target paths into which this entry gets virtualized to. Target paths are decoded if they contain any Intermediate Values within.

Syntax:

```
public IList<string> GetDecodedMappingPaths()
```

Return value:

Type: System.Collections.Generic.IList<T>

List of file system paths that are valid on the current machine and where to this entry would map to.

Properties

CreationTimeUtc

Gets the original creation timestamp in UTC format for the directory entry.

Syntax:

```
public DateTime CreationTimeUtc { get; }
```

Property value:

Type: System.DateTime

Creation date and time.

FilesystemAttributes

Gets or sets the file system attributes for the directory entry.

Syntax:

```
public FileAttributes FilesystemAttributes { get; set; }
```

Property value:

Type: System.IO.FileAttributes

File attributes.

Id

Gets the unique identifier that SoftGrid/App-V uses to track individual directory entries inside the package.

Syntax:

```
public Guid Id { get; }
```

Property value:

Type: System.Guid

Unique identifier (GUID) for the directory entry.

Note:

This identifier is automatically generated for all new files and directories and remains the same even if the package is branched as new one by regenerating package-level identifiers.

IsModified

Gets the information if the directory entry has been modified during the time the package has been open.

Syntax:

```
public bool IsModified { get; }
```

Property value:

Type: System.Boolean

true if the entry has been modified, otherwise **false**.

Note:

This property, as defined on the base class, tracks if any of the file's properties has been modified. Inherited classes may expand on this behaviour.

IsNew

Gets the information if the directory entry has been created as new during the time the package has been open.

Syntax:

```
public bool IsNew { get; }
```

Property value:

Type: System.Boolean

true if the entry is a new one, otherwise **false**.

LastAccessTimeUtc

Gets the last access timestamp in UTC format for the directory entry.

Syntax:

```
public DateTime LastAccessTimeUtc { get; }
```

Property value:

Type: System.DateTime

Last access date and time.

Note:

This property is not automatically updated in the real-time during operations done to the open package, but it is updated to reflect the current timestamp when package is encoded and if the entry has been modified.

LastWriteTimeUtc

Gets the last modification timestamp in UTC format for the directory entry.

Syntax:

```
public DateTime LastWriteTimeUtc { get; }
```

Property value:

Type: System.DateTime

Last modification date and time.

Note:

This property is not updated automatically by the Lib-V and it is responsibility of the caller to update modification timestamp if needed.

MapsTo

Gets the list of Virtual Filesystem (VFS) mapping entries.

Syntax:

```
public List<VfsMappingTarget> MapsTo { get; }
```

Property value:

Type: System.Collections.Generic.List<T>

List of [VfsMappingTarget](#) entries defined for the directory entry.

Note:

If [VfsMode](#) for the directory or file is set to [None](#), this property returns **null** and mappings cannot be added to it. Before adding any new mappings, [VfsMode](#) has to be changed either to [Override](#) or [Merge](#).

These mapping entries are effectively an instructions to a SoftGrid/App-V client based on which it knows how to represent the directory or file physically stored in the package's own path on top of the other drives. Some of those entries can be in inactive state (as determined by mapping entry's [Active](#) –flag) in which case the mapping to that target path is not performed.

Only level of [Override](#) can in actuality have mapping entries defined for it as SFT file format cannot store any mappings for merged paths. This means that if any paths are defined by a caller via MapsTo property, and [VfsMode](#) is set to [Merge](#), those paths will be lost when encoding the SFT file. [SftDecoder](#) does assign merged status for some entries and also generates one target mapping path per each such entry, but this is provided for display/informative purposes only and the path information for any such entries is deducted from the overridden entries beneath merged entries.

Name

Gets or sets the long filename for directory entry.

Syntax:

```
public string Name { get; set; }
```

Property value:

Type: System.String

Long filename.

Parent

Gets the parent [VirtualDirectory](#) for this entry.

Syntax:

```
public VirtualDirectory Parent { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualDirectory](#)

Parent virtual directory containing this directory entry. Returns **null** for the root directory.

Path

Gets the logical path for the directory entry.

Syntax:

```
public string Path { get; }
```

Property value:

Type: System.String

Path for the directory entry.

Note:

This path is representation of the internal directory structure, so it always begins with the root directory name.

Security

Gets or sets the security descriptor for the directory entry.

Syntax:

```
public RawSecurityDescriptor Security { get; set; }
```

Property value:

Type: System.Security.AccessControl.RawSecurityDescriptor

Security descriptor defined assigned to a directory entry.

Note:

Lib-V does not offer full security information propagation or inheritance facilities. This means that it is on the caller's responsibility to update any security descriptors beneath the directory entry if modifying descriptors for [VirtualDirectory](#).

ShortName

Gets or sets the short filename (8+3 DOS format) for directory entry.

Syntax:

```
public string ShortName { get; set; }
```

Property value:

Type: System.String

Short filename.

ShortPath

Gets the logical path for the directory entry in the short filename format.

Syntax:

```
public string ShortPath { get; }
```

Property value:

Type: System.String

Path for the directory entry in shortname format.

Note:

This path is representation of the internal directory structure, so it always begins with the root directory name.

SoftricityAttributes

Gets or sets the Softricity attributes for the directory entry.

Syntax:

```
public SoftricityFileAttributes SoftricityAttributes { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.SoftricityFileAttributes](#)

Attributes describing directory entry's virtualization –specific characteristics.

Note:

These attributes control the branching and persistence related aspects of files and directories when they are stored in the SoftGrid/App-V Client.

SystemguardPath

Gets the absolute path for the directory entry in the format used by the SystemGuard.

Syntax:

```
public string SystemguardPath { get; }
```

Property value:

Type: System.String

Absolute path for the directory entry.

Note:

This path is representation of the file's or directory's path as executing on the client and as such it begins with \\?\\%SFT_MNT%\\ which in turn maps to a client's virtual drive.

Version

Gets the version number for the directory entry.

Syntax:

```
public UInt32 Version { get; }
```

Property value:

Type: System.UInt32

Version number of the directory entry.

Note:

This version number is incremented automatically by the encoding engine for all modified files when writing out the package. It is not updated automatically when the package is open and file and directories are being accessed.

VfsMode

Gets or sets the Virtual Filesystem (VFS) mode for the directory entry.

Syntax:

```
public VirtualFilesystemMode VfsMode { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualFilesystemMode](#)

Mode describing how the entry is mapped according to VFS functionality

Note:

Only entries that have mode of [Override](#) will actually be written to virtual environment configuration file, see notes on [MapsTo](#) property.

All entries that are needed to be mapped using Virtual Filesystem has to be placed under VFS subdirectory under package's root directory. It's on caller's responsibility to make necessary adjustments both to VfsMode property and to mapping list in order to virtualize entries with VFS. Alternatively, you can use [VirtualizeFromFilesystem \(\)](#) method of the [SftPackage](#) object to import entries from the real file system and have an automatic VFS entry creation as a result.

VirtualEnvironmentConfiguration class

Contains configuration information for the virtual environment, including Virtual Registry and Virtual Services.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualEnvironmentConfiguration : IDisposable
```

Methods

Dispose

Disposes resources reserved by the [VirtualEnvironmentConfiguration](#) class. This method does not need to be called directly as closing the package will automatically dispose the virtual environment.

Syntax:

```
public void Dispose()
```

SetDefaultObjectExclusions

Initializes list of object exclusions to a default list used in App-V 4.5.

Syntax:

```
public void SetDefaultObjectExclusions ()
```

Note:

When package is decoded from the existing SFT file, object exclusions are read in from it, but new package created using [Create\(\)](#) method won't contain any exclusions by default. If you want to include the default set of exclusions, use this method from the new package's virtual environment.

Properties

HasDecodingErrors

If set, indicates that the virtual environment configuration could not be successfully decoded.

Syntax:

```
public bool HasDecodingErrors { get; }
```

Property value:

Type: System.Boolean

true if virtual environment decoding encountered errors, **false** if no errors were found.

Note:

When decoding SFT file and specifying [IGNORE VE ERRORS](#) –flag for the decoder, decoding does not throw exceptions if structural errors are identified in the VE configuration. In that scenario, after decoding has completed, this flag is set to true instead notifying caller that all or some of the contents could not be decoded and may be missing.

If no errors happened during decoding, this flag is set to false and package contents can be trusted.

Id

Gets the unique identifier for the virtual environment.

Syntax:

```
public Guid Id { get; }
```

Property value:

Type: System.Guid

Package's virtual environment unique identifier (GUID).

Note:

When encoding package and specifying [REGENERATE IDENTIFIERS](#) -flag for the encoder, this identifier will be regenerated for the new package along with the [package's unique identifier](#).

Is64BitPackage

Gets or sets flag indicating if the package is 32-bit (default) or 64-bit.

Syntax:

```
public bool Id { get; set; }
```

Property value:

Type: System.Boolean

true if package contains 64-bit registry entries and/or components, **false** otherwise.

Note:

Clearing this flag indicates to x64 App-V Client that all registry entries in the virtual registry needs to be redirected under Wow6432Node so that 32-bit processes can see them. Having the flag disabled redirection by the App-V Client.

Do not set this flag for purely 32-bit package.

This flag has no effect on encoding packages previous to App-V 4.6.

ObjectExclusions

Gets the list of object exclusions set for the package's virtual environment. This list is used by the client to exempt specified NT objects from isolation.

Syntax:

```
public ICollection<string> ObjectExclusions { get; }
```

Property value:

Type: System.Collections.Generic.ICollection<T>

List of object names or partial names to exclude.

RequiredClient

Gets the version information for the client required to run this virtual environment configuration.

Syntax:

```
public ClientVersion RequiredClient { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.ClientVersion](#)

Client version required to run the VE.

Note:

Version in this value is not related to CLIENT VERSION –information in the OSD file.

Sequencer

Gets the version information for the Sequencer used to create this virtual environment configuration.

Syntax:

```
public SequencerVersion Sequencer { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SequencerVersion](#)

Sequencer version that created the VE.

VirtualRegistry

Gets the root entry of the package's virtual registry.

Syntax:

```
public VirtualRegistryKey VirtualRegistry { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryKey](#)

Root registry key of the virtual registry, representing \REGISTRY –node.

VirtualServices

Gets the list of virtual services defined for the package. These virtual services are represented by [VirtualService](#) class instances.

Syntax:

```
public ICollection<VirtualService> VirtualServices { get; }
```

Property value:

Type: System.Collection.Generic.ICollection<T>

List of virtual services.

Note:

For new packages, this list will be initialized to empty list (i.e. no virtual services defined).

VirtualFile class

Represents a file in the package's internal directory structure.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualFile : VirtualDirectoryEntry, IDisposable
```

Constructors

Overload List:

Creates a new virtual file using the specified name and initializes the file entry with default attributes.

```
public VirtualFile(  
    string fileName,  
    SftPackage parentPackage  
)
```

Creates a new virtual file using the specified long and short names and initializes the file entry with default attributes.

```
public VirtualFile(  
    string fileName,  
    string shortName,  
    SftPackage parentPackage  
)
```

Creates a new virtual file using specified file system path to an existing file as source for the names and attributes.

```
public VirtualFile(  
    string physicalFilePath,  
    SftPackage parentPackage,
```

```
        bool getSecurity  
    )
```

Parameters:

fileName

Type: System.String

Specifies a name for the virtual file.

shortName

Type: System.String

Specified an explicit short name for the virtual file.

parentPackage

Type: [GridMetric.LibV.SftPackage](#)

Package instance which will host this virtual file.

physicalFilePath

Type: System.String

Path to an existing physical file from where to read file entry attributes from.

getSecurity

Type: System.Boolean

true to read in and include security descriptors from the specified existing physical file, otherwise **false**.

Note:

After instantiating new [VirtualFile](#), it has to be mounted in the package's directory structure by calling [Add\(\)](#) method on suitable [VirtualDirectory](#) already present in the package.

Methods

Dispose

Disposes resources reserved by the virtual file, including deleting possible temporary file created as backing file for content modification. This method does not need to be called directly for files mounted to package's internal directory structure as closing the package

will automatically dispose all virtual files. If the virtual file is not mounted, it should be explicitly disposed to ensure all resources are released.

Syntax:

```
public void Dispose()
```

GetBytes

Allocates and returns an array of bytes holding the virtual file's full contents. This method can be called as alternative to [GetStream\(\)](#) if content does not need to be modified.

Syntax:

```
public byte[] GetBytes()
```

Return value:

Type: System.Byte[]

Virtual file's full contents.

Note:

This method is suitable only for files containing small enough length as it may cause large memory consumption due to byte array returned. For larger files, stream –based usage via [GetStream\(\)](#) is recommended.

Byte array returned is state of the virtual file's contents at the moment when method was called and is not possible changes either to file itself or returned byte array is not reflected between the two.

GetStream

Opens new stream for accessing virtual file's contents.

Overload List:

Opens stream for read/write access.

```
public VirtualFileStream GetStream()
```

Opens stream for specified access mode.

```
public VirtualFileStream GetStream(  
    bool readOnlyAccess  
)
```

Parameters:

readOnlyAccess

Type: System.Boolean

true if virtual stream is opened only for reading, otherwise **false** for read/write access.

Return value:

Type: [GridMetric.LibV.Structures.VirtualFileStream](#)

Stream representing the file's content.

Reset

Resets the file into its original state by discarding any modifications made to file's contents. This method can only be called on virtual files that has been decoded from the existing SFT file as resetting loses branched backing file for it.

Syntax:

```
public void Reset()
```

Note:

If virtual file has VirtualFileStream open for it, calling Reset will throw exception.

Properties

Blocks

Gets the list of DATA blocks associated with the virtual file. These blocks are represented by the instances of [VirtualFileBlock](#) class.

Syntax:

```
public IList<VirtualFileBlock> Blocks { get; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of blocks for file.

Note:

This list is valid only for the virtual files decoded from the existing SFT file. All new virtual files or files having content modifications return **null** since up-to-date block list for files is generated by the encoding engine during SFT encoding.

IsModified

Gets the information if the virtual file has been modified during the time the package has been open.

Syntax:

```
public new bool IsModified { get; }
```

Property value:

Type: System.Boolean

true if the virtual file has been modified, otherwise **false**.

Note:

This property tracks if any of the file's properties or its contents has been modified.

IsVirtualFont

Gets or sets the flag if the virtual file is virtualized as font.

Syntax:

```
public bool IsVirtualFont { get; set; }
```

Property value:

Type: System.Boolean

true if the virtual file is font, otherwise **false**.

Note:

This property can only be set for files located under Virtual Filesystem (VFS) directory and having [VfsMode](#) set.

This property should only be set to **true** if file is actually a font.

Length

Gets the length of the virtual file.

Syntax:

```
public UInt32 Length { get; }
```

Property value:

Type: System.UInt32

Current length of the file.

Note:

This property does not allow changing file's length, use [SetLength\(\)](#) method from the stream associated with the virtual file.

Name

Gets or sets the long filename for virtual file.

Syntax:

```
public new string Name { get; set; }
```

Property value:

Type: System.String

Long filename.

PreferredFeatureBlock

Gets or sets the preferred feature block number for the virtual file's blocks.

Syntax:

```
public UInt16 PreferredFeatureBlock { get; set; }
```

Property value:

Type: System.UInt16

Preferred feature block number, allowed values are 1 or 2.

Note:

This property allows setting feature block number to which DATA blocks for file will be allocated to if the content is modified. By default, virtual files in decoded SFT file will have feature block number associated with it per block basis, but if encoding engine has to regenerate this block list because content was modified, this property's value will dictate feature block for file's all blocks.

For existing files, this property will return 1 if at least one block of the file is placed in feature block 1, otherwise it returns 2.

ShortName

Gets or sets the short filename (8+3 DOS format) for virtual file.

Syntax:

```
public new string ShortName { get; set; }
```

Property value:

Type: System.String

Short filename for virtual file.

VirtualFileStream class

Provides a stream object to access [VirtualFile](#)'s contents.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public sealed class VirtualFileStream : Stream, IDisposable
```

Note:

Although [VirtualFileStream](#) is inherited from Stream class, it does not implement all of the Stream's methods and properties, like the asynchronous versions of read and write. Only methods and properties documented below are safe to use if [VirtualFileStream](#) is cast to Stream.

Methods

Close

Closes the virtual stream and updates status of the [VirtualFile](#) the stream was created from.

Syntax:

```
public override void Close()
```

Dispose

Disposes resources reserved by the stream. This method does not need to be called directly as calling [Close\(\)](#) will perform disposing automatically.

Syntax:

```
public new void Dispose()
```

Flush

Clears all buffers for stream and causes any buffered data to be written to the underlying file.

Syntax:

```
public override void Flush()
```

Read

Reads a sequence of bytes from virtual file stream and advances the position within the stream by the number of bytes read.

Syntax:

```
public override int Read(  
    byte[] buffer,  
    int offset,  
    int count  
)
```

Parameters:

buffer

Type: System.Byte[]

An array of bytes. When this method returns, the buffer contains the specified byte array with the values between offset and (offset + count - 1) replaced by the bytes read from the current source.

offset

Type: System.Int32

The zero-based byte offset in buffer at which to begin storing the data read from the stream.

count

Type: System.Int32

The maximum number of bytes to be read from the stream.

Return value:

Type: System.Int32

The total number of bytes read into the buffer. This can be less than the number of bytes requested if that many bytes are not currently available, or zero (0) if the end of the stream has been reached.

ReadByte

Reads a byte from the stream and advances the position within the stream by one byte or returns -1 if at the end of the stream.

Syntax:

```
public override int ReadByte()
```

Return value:

Type: System.Int32

The unsigned byte cast to an Int32, or -1 if at the end of the stream.

Seek

Sets the position within the stream.

Syntax:

```
public override long Seek(  
    long offset,  
    SeekOrigin origin  
)
```

Parameters:

offset

Type: System.Int64

A byte offset relative to the origin parameter.

origin

Type: System.IO.SeekOrigin

A value indicating the reference point used to obtain the new position.

Return value:

Type: System.Int64

The new position within the current stream.

Note:

Although Seek method specifies offsets with 64-bit integer value because of inheritance from Stream class, SFT file format currently do not support files bigger than 4 GB (32-bit maximum value) and as such, limits this method's usable length to maximum of 4 GB.

SetLength

Sets the length of the stream.

Syntax:

```
public override void SetLength(  
    long value,  
)
```

Parameters:

value

Type: System.Int64

The desired length of the stream in bytes.

Note:

Although SetLength method specifies length with 64-bit integer value because of inheritance from Stream class, SFT file format currently do not support files bigger than 4

GB (32-bit maximum value) and as such, limits this method's usable length to maximum of 4 GB.

Write

Writes a sequence of bytes to the stream and advances the current position within this stream by the number of bytes written.

Syntax:

```
public override void Write(  
    byte[] buffer,  
    int offset,  
    int count  
)
```

Parameters:

buffer

Type: System.Byte[]

An array of bytes. This method copies count bytes from buffer to the current stream.

offset

Type: System.Int32

The zero-based byte offset in buffer at which to begin copying bytes to the stream.

count

Type: System.Int32

The number of bytes to be written to the stream.

WriteByte

Writes a byte to the current position in the stream and advances the position within the stream by one byte.

Syntax:

```
public override void WriteByte(  
    byte value  
)
```

Parameters:

value

Type: System.Byte

The byte to write to the stream.

Properties

CanRead

Gets a value indicating whether the stream supports reading.

Syntax:

```
public override bool CanRead { get; }
```

Property value:

Type: System.Boolean

true if the stream supports reading; otherwise, **false**.

CanSeek

Gets a value indicating whether the stream supports seeking.

Syntax:

```
public override bool CanSeek { get; }
```

Property value:

Type: System.Boolean

true if the stream supports seeking; otherwise, **false**.

CanWrite

Gets a value indicating whether the stream supports writing.

Syntax:

```
public override bool CanWrite { get; }
```

Property value:

Type: System.Boolean

true if the stream supports writing; otherwise, **false**.

Length

Gets the length in bytes of the stream.

Syntax:

```
public override long Length { get; }
```

Property value:

Type: System.Int64

A long value representing the length of the stream in bytes.

Note:

Although Length property returns 64-bit integer value because of inheritance from Stream class, SFT file format currently do not support files bigger than 4 GB (32-bit maximum value) and as such, limits this property's usable value to maximum of 4 GB.

Position

Gets or sets the position within the stream.

Syntax:

```
public override long Position { get; set; }
```

Property value:

Type: System.Int64

The current position within the stream.

Note:

Although Position property sets or returns 64-bit integer value because of inheritance from Stream class, SFT file format currently do not support files bigger than 4 GB (32-bit maximum value) and as such, limits this property's usable value to maximum of 4 GB.

VirtualFileBlock class

Represents an individual DATA block in the virtual file.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualFileBlock
```

Properties

CompressedLength

Gets the length of the block in when it is in compressed state, in bytes. If the package block is not compressed, this length equals length returned by the [Length](#) –property.

Syntax:

```
public UInt32 CompressedLength { get; }
```

Property value:

Type: System.UInt32

Compressed block's length.

CompressionMethod

Gets the method data block is compressed with.

Syntax:

```
public SftCompressionType CompressionMethod { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.SftCompressionType](#)

Compression method used.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

FeatureBlockNum

Gets or sets the feature block number assigned for the data.

Syntax:

```
public UInt16 FeatureBlockNum { get; set; }
```

Property value:

Type: System.UInt16

Feature block number assigned for block.

Length

Gets the length of the block, in bytes.

Syntax:

```
public UInt32 Length { get; }
```

Property value:

Type: System.UInt32

Block's length.

Offset

Gets the offset of the block's data in the [VirtualFile](#) containing the block.

Syntax:

```
public UInt32 Offset { get; }
```

Property value:

Type: System.UInt32

Block's offset in containing file.

SftOffset

Gets the relative offset of the block in the SFT file.

Syntax:

```
public UInt32 SftOffset { get; }
```

Property value:

Type: System.UInt32

Block's offset in SFT file.

Note:

Offset returned is relative to a start of Feature Block 1 in the SFT file. Therefore, for the first block in SFT file this property returns 0 and so forth.

VirtualRegistryEntry class

Provides common abstract base for both VirtualRegistryKey and VirtualRegistryValue classes. All virtual registry entries contained within [SftPackage](#) and accessible from [VirtualRegistry](#) –property are defined through these three classes.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public abstract class VirtualRegistryEntry
```

Properties

InExplicitSharedRegistry

Specifies if the virtual registry entry explicitly belongs to shared part of the registry.

Syntax:

```
public bool InExplicitSharedRegistry { get; }
```

Property value:

Type: System.Boolean

true if entry belongs to shared registry, otherwise **false**.

InExplicitUserRegistry

Specifies if the virtual registry entry explicitly belongs to user-specific part of the registry.

Syntax:

```
public bool InExplicitUserRegistry { get; }
```

Property value:

Type: System.Boolean

true if entry belongs to user-specific registry, otherwise **false**.

InImplicitSharedRegistry

Specifies if the virtual registry entry implicitly belongs to shared part of the registry.

Syntax:

```
public bool InImplicitSharedRegistry { get; }
```

Property value:

Type: System.Boolean

true if entry implicitly belongs to shared registry, otherwise **false**.

InImplicitUserRegistry

Specifies if the virtual registry entry implicitly belongs to user-specific part of the registry.

Syntax:

```
public bool InImplicitUserRegistry { get; }
```

Property value:

Type: System.Boolean

true if entry implicitly belongs to user-specific registry, otherwise **false**.

InMainRegistry

Specifies if the virtual registry entry belongs to common part of the registry.

Syntax:

```
public bool InMainRegistry { get; }
```

Property value:

Type: System.Boolean

true if entry belongs to common registry, otherwise **false**.

IntermediateValueEncodingStatus

Gets registry entry's Intermediate Value encoding status.

Syntax:

```
public VirtualRegistryIMStatusAttributes  
IntermediateValueEncodingStatusAttributes { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryIMStatusAttributes](#)

Intermediate Value encoding status.

LastWritten

Gets last-written timestamp for the registry entry.

Syntax:

```
public DateTime LastWritten { get; }
```

Property value:

Type: System.DateTime

Timestamp of last write to registry entry.

Note:

Lib-V does not currently update this timestamp when modifying either VirtualRegistryKey or VirtualRegistryValue.

Name

Gets the decoded name for the registry entry.

Syntax:

```
public string Name { get; }
```

Property value:

Type: System.String

Registry entry's name in decoded form.

Note:

This property is overridden in the inherited classes providing setter along with the getter.

This property returns Intermediate Value decoded name for the registry entry, use [RawName](#) –property to get [possibly] encoded form of the name.

Parent

Gets the parent virtual registry key for the registry entry.

Syntax:

```
public VirtualRegistryKey Parent { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryKey](#)

Registry entry's parent key.

Path

Gets the decoded full path for the registry entry.

Syntax:

```
public string Path { get; }
```

Property value:

Type: System.String

Registry entry's full path (e.g. \REGISTRY\ USER\S-1-5-21-4084310487-3401208622-2622696841-1116\Software) in decoded form.

Note:

This property returns Intermediate Value decoded path for the registry entry, use [RawPath](#) –property to get [possibly] encoded form of the path.

RawName

Gets the name for the registry entry.

Syntax:

```
public string RawName { get; }
```

Property value:

Type: System.String

Registry entry's name in raw form.

Note:

This property returns raw name for the registry entry without automatic decoding of Intermediate Values, use [Name](#) –property to get decoded form of the name.

RawPath

Gets the full path for the registry entry.

Syntax:

```
public string RawPath { get; }
```

Property value:

Type: System.String

Registry entry's full path (e.g. \REGISTRY\USER\%SFT_SID%\Software) in raw form.

Note:

This property returns raw path for the registry entry without automatic decoding of Intermediate Values, use [Path](#) –property to get decoded form of the path.

RegistryAttributes

Gets or sets attributes for the virtual registry entry. This property has replaced both Attributes and ValueAttributes –properties found in earlier Lib-V versions.

Syntax:

```
public VirtualRegistryAttributes RegistryAttributes { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryAttributes](#)

Virtual registry entry attributes.

VirtualRegistryKey class

Represents a key in the package's virtual registry.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualRegistryKey : VirtualRegistryEntry,  
IComparable<VirtualRegistryKey>
```

Constructors

Overload List:

Creates a new virtual registry key using the specified physical registry key present in the local registry and initializes it with attributes read from that key.

```
public VirtualRegistryKey(  
    RegistryKey physicalKey  
)
```

Note:

Does not virtualize values (apart from the default value) or keys beneath the physical key specified.

Creates a new virtual registry key using the specified name and initializes it with the default attributes.

```
public VirtualRegistryKey(  
    string keyName  
)
```

Creates a new virtual registry key using the specified name and supplies default value for the key.

```
public VirtualRegistryKey(  
    string keyName,
```

[VirtualRegistryValue](#) *defaultValue*
)

Parameters:

registryKey

Type: Microsoft.Win32.RegistryKey

Physical registry key to initialize virtual registry key from.

keyName

Type: System.String

Specifies a name for the virtual registry key.

defaultValue

Type: [GridMetric.LibV.Structures.VirtualRegistryValue](#)

Default value to set for the newly created key.

Note:

Virtual registry key name has to be specified using the real (i.e. not Intermediate Value encoded variable) name as encoding of raw name happens automatically.

After instantiating new key, it has to be mounted in the package's virtual registry structure by calling [Add\(\)](#) method on suitable [VirtualRegistryKey](#) already mounted in the package's virtual registry.

Methods

Add

Adds a new file or directory to another directory.

Syntax:

```
public void Add(  
    VirtualRegistryEntry keyOrValueToAdd  
)
```

Parameters:

keyOrValueToAdd

Type: [GridMetric.LibV.Structures.VirtualRegistryEntry](#)

New [VirtualRegistryKey](#) or [VirtualRegistryValue](#) entry to add to this key. If changing key's default value, use [Default](#) –property instead.

Note:

When adding new virtual registry key or value to an existing virtual registry, Lib-V will automatically update [Overridden](#) –status based on parent key's status.

This method is thread-safe.

ContainsKey

Looks for if the key by the specified name exists.

Syntax:

```
public bool ContainsKey(  
    string keyName  
)
```

Parameters:

keyName

Type: System.String

Name of the key to look for.

Return value:

Type: System.Boolean

true if the key by the name of *keyName* exists, otherwise **false**.

Note:

Performs case-insensitive matching of name.

This method handles names in decoded form, if you need to search using raw name, use [ContainsKeyByRawName\(\)](#) method instead.

ContainsKeyByRawName

Looks for if the key by the specified raw name exists.

Syntax:

```
public bool ContainsKeyByRawName(  
    string rawKeyName  
)
```

Parameters:

rawKeyName

Type: System.String

Raw name of the key to look for.

Return value:

Type: System.Boolean

true if the key by the name of *rawKeyName* exists, otherwise **false**.

Note:

Performs case-insensitive matching of name.

This method handles names in raw form, if you need to search using decoded name, use [ContainsKey \(\)](#) method instead.

ContainsValue

Looks for if the value by the specified name exists.

Syntax:

```
public bool ContainsValue(  

```

```
        string valueName
    )
```

Parameters:

valueName

Type: System.String

Name of the value to look for.

Return value:

Type: System.Boolean

true if the value by the name of *valueName* exists, otherwise **false**.

Note:

Performs case-insensitive matching of name.

This method handles names in decoded form, if you need to search using raw name, use [ContainsValueByRawName\(\)](#) method instead.

ContainsValueByRawName

Looks for if the value by the specified raw name exists.

Syntax:

```
public bool ContainsValueByRawName(
    string rawValueName
)
```

Parameters:

rawValueName

Type: System.String

Raw name of the value to look for.

Return value:

Type: System.Boolean

true if the value by the name of *rawValueName* exists, otherwise **false**.

Note:

Performs case-insensitive matching of name.

This method handles names in raw form, if you need to search using decoded name, use [ContainsValue\(\)](#) method instead.

GetKey

Returns key by the specified name.

Syntax:

```
public VirtualRegistryKey GetKey(  
    string keyName  
)
```

Parameters:

keyName

Type: System.String

Name of the key to get.

Return value:

Type: [GridMetric.LibV.Structures.VirtualRegistryKey](#)

Virtual registry key or **null** if no such key found.

Note:

Performs case-insensitive matching of name.

This method handles names in decoded form, if you need to get key using raw name, use [GetKeyByRawName\(\)](#) method instead.

GetKeyByRawName

Returns key by the specified raw name.

Syntax:

```
public VirtualRegistryKey GetKeyByRawName(  
    string rawKeyName  
)
```

Parameters:

rawKeyName

Type: System.String

Raw name of the key to get.

Return value:

Type: [GridMetric.LibV.Structures.VirtualRegistryKey](#)

Virtual registry key or **null** if no such key found.

Note:

Performs case-insensitive matching of name.

This method handles names in raw form, if you need to get key using decoded name, use [GetKey\(\)](#) method instead.

GetValue

Returns value by the specified name.

Syntax:

```
public VirtualRegistryValue GetValue(  
    string valueName  
)
```

Parameters:

valueName

Type: System.String

Name of the value to get.

Return value:

Type: [GridMetric.LibV.Structures.VirtualRegistryValue](#)

Virtual registry value or **null** if no such value found.

Note:

Performs case-insensitive matching of name.

This method handles names in decoded form, if you need to get value using raw name, use [GetValueByRawName\(\)](#) method instead.

GetValueByRawName

Returns value by the specified raw name.

Syntax:

```
public VirtualRegistryValue GetValueByRawName(  
    string rawValueName  
)
```

Parameters:

rawValueName

Type: System.String

Raw name of the value to get.

Return value:

Type: [GridMetric.LibV.Structures.VirtualRegistryValue](#)

Virtual registry value or **null** if no such value found.

Note:

Performs case-insensitive matching of name.

This method handles names in raw form, if you need to get value using decoded name, use [GetValue\(\)](#) method instead.

RemoveKey

Removes (deletes) virtual registry key by the specified name from the current key.

Syntax:

```
public void RemoveKey(  
    string keyName  
)
```

Parameters:

keyName

Type: System.String

Name of the key to remove.

Note:

Performs case-insensitive matching of name.

This method handles names in decoded form, if you need to remove key using its raw name, use [RemoveKeyByRawName\(\)](#) method instead.

This method is thread-safe.

RemoveKeyByRawName

Removes (deletes) virtual registry key by the specified raw name from the current key.

Syntax:

```
public void RemoveKeyByRawName(  
    string rawKeyName  
)
```

Parameters:

rawKeyName

Type: System.String

Raw name of the key to remove.

Note:

Performs case-insensitive matching of name.

This method handles names in raw form, if you need to remove key using its decoded name, use [RemoveKey\(\)](#) method instead.

This method is thread-safe.

RemoveValue

Removes (deletes) virtual registry value by the specified name from the current key.

Syntax:

```
public void RemoveValue(  
    string valueName  
)
```

Parameters:

valueName

Type: System.String

Name of the value to remove.

Note:

Performs case-insensitive matching of name.

This method handles names in decoded form, if you need to remove value using its raw name, use [RemoveValueByRawName\(\)](#) method instead.

If you need to remove key's default value, set key's [Default](#) –property to **null**.

This method is thread-safe.

RemoveValueByRawName

Removes (deletes) virtual registry value by the specified raw name from the current key.

Syntax:

```
public void RemoveValueByRawName(  
    string rawValueName  
)
```

Parameters:

rawValueName

Type: System.String

Raw name of the value to remove.

Note:

Performs case-insensitive matching of name.

This method handles names in raw form, if you need to remove value using its decoded name, use [RemoveValue\(\)](#) method instead.

If you need to remove key's default value, set key's [Default](#) –property to **null**.

This method is thread-safe.

Properties

Default

Gets or sets virtual registry key's default value.

Syntax:

public [VirtualRegistryValue](#) Default { **get**; **set**; }

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryValue](#)

Default value for the key.

HasKeys

Specifies if the current virtual registry key has keys.

Syntax:

public bool HasKeys { **get**; }

Property value:

Type: System.Boolean

true if key contains other keys, otherwise **false**.

HasValues

Specifies if the current virtual registry key has values.

Syntax:

public bool HasValues { **get**; }

Property value:

Type: System.Boolean

true if key contains values, otherwise **false**.

Note:

This property does not count possible default value for the key, which can be tested with [Default](#) –property against **null** value.

Keys

Gets list of keys current key contains.

Syntax:

```
public IList<VirtualRegistryKey> Keys { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

Read-only list of keys.

Note:

Returned list is read-only. If you need to make modification to the list of keys virtual registry key holds, use [Add\(\)](#) and [RemoveKey\(\)](#) methods.

Name

Gets or sets the name for the registry key.

Syntax:

```
public new string Name { get; set; }
```

Property value:

Type: System.String

Registry key name.

Note:

This property accesses key's name in Intermediate Value decoded form, which is automatically encoded for the raw form if needed to.

RegistryAttributes

Gets or sets attributes for the virtual registry key. This property has replaced both Attributes and ValueAttributes –properties found in earlier Lib-V versions.

Syntax:

```
public new VirtualRegistryAttributes RegistryAttributes { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryAttributes](#)

Virtual registry key attributes.

Values

Gets list of values current key contains.

Syntax:

```
public IList<VirtualRegistryValue> Value { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

Read-only list of values.

Note:

Returned list is read-only. If you need to make modification to the list of values virtual registry key holds, use [Add\(\)](#) and [RemoveValue\(\)](#) methods.

VirtualRegistryValue class

Represents a value in the package's virtual registry.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualRegistryValue : VirtualRegistryEntry,  
IComparable<VirtualRegistryKey>
```

Constructors

Overload List:

Creates a new virtual registry value using the specified physical registry key and value present in the local registry and initializes it with attributes read from that value.

```
public VirtualRegistryValue(  
    RegistryKey parentKeyInRegistry,  
    string valueName  
)
```

Creates a new virtual registry value using the specified name and type and initializes it with the default attributes.

```
public VirtualRegistryKey(  
    string valueName,  
    RegistryValueKind valueKind  
)
```

Creates a new virtual registry value using the specified name, type and value data and initializes it with the default attributes.

```
public VirtualRegistryKey(  
    string valueName,  
    RegistryValueKind valueKind,  
    object valueData
```

)

Parameters:

parentKeyInRegistry

Type: Microsoft.Win32.RegistryKey

Physical registry key from which to look for value named *valueName*.

valueName

Type: System.String

Specifies a name for the virtual registry value.

If creating a value to be used as default value in [VirtualRegistryKey](#), you can set name to null.

valueKind

Type: Microsoft.Win32.RegistryValueKind

Data type for the value.

valueData

Type: System.Object

Data to assign to the value.

Note:

Virtual registry value name has to be specified using the real (i.e. not Intermediate Value encoded variable) name as encoding of raw name happens automatically.

After instantiating new value, it has to be mounted to existing key by calling [Add\(\)](#) method on suitable [VirtualRegistryKey](#) object.

Properties

Data

Gets or sets data for the virtual registry value.

Syntax:

```
public object Data { get; set; }
```

Property value:

Type: System.Object

Value data.

Note:

This property returns data in decoded form (in case of strings, multi-strings and expandable strings). To get value data in raw form, use [RawData](#) –property.

This property automatically encodes data if needed to, and hence string-based data should not be Intermediate Value encoded before storing it through this property.

IntermediateOffsets

Gets list of intermediate value offsets for data stored in value.

Syntax:

```
public IList<Int32> IntermediateOffsets { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of byte-offsets or **null** if data does not contain Intermediate Values.

Name

Gets or sets the name for the registry value.

Syntax:

```
public new string Name { get; set; }
```

Property value:

Type: System.String

Registry value name.

Note:

This property accesses value's name in Intermediate Value decoded form, which is automatically encoded for the raw form if needed to.

RawData

Gets data for the virtual registry value in raw form.

Syntax:

```
public object RawData { get; }
```

Property value:

Type: System.Object

Value data in raw form.

Note:

This property returns data in encoded form (in case of strings, multi-strings and expandable strings). To get value data in automatically decoded form, use [Data](#) –property.

RegistryAttributes

Gets or sets attributes for the virtual registry value. This property has replaced both Attributes and ValueAttributes –properties found in earlier Lib-V versions.

Syntax:

```
public new VirtualRegistryAttributes RegistryAttributes { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.VirtualRegistryAttributes](#)

Virtual registry value attributes.

ValueType

Gets type of the virtual registry value.

Syntax:

```
public RegistryValueKind ValueType { get; }
```

Property value:

Type: Microsoft.Win32.RegistryValueKind

Value type.

VirtualService class

Represents a virtualized service.

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class VirtualService
```

Constructors

Syntax:

Creates new virtual service using specified service parameters.

```
public VirtualService(  
    string name,  
    string displayName,  
    string description,  
    string path,  
    string LogOnAccount,  
    ServiceStartupType startupType,  
    ServiceErrorControl errorControl,  
    ServiceType type,  
    string[] dependOnService,  
    string group,  
    string[] dependOnGroup,  
    UInt32? tag  
)
```

Parameters:

name

Type: System.String

Specifies a service name. This value is the internal service name, user visible name is defined through displayName.

displayName

Type: System.String

Specifies a display name for the service. If this value is left **null**, name specified by name parameter is used for display.

description

Type: System.String

Specifies a description for the service. This parameter can be **null**.

path

Type: System.String

Specifies path to service's executable.

logOnAccount

Type: System.String

Specifies user account under which the service is run. SoftGrid/App-V currently supports only virtual services executing under LocalSystem, for which logOnAccount is left unspecified with **null** value.

startupType

Type: [GridMetric.LibV.Structure.ServiceStartupType](#)

Startup type for the service. Currently supported values are [Automatic](#), [Manual](#) or [Disabled](#).

errorControl

Type: [GridMetric.LibV.Structure.ServiceErrorControl](#)

Error control for the service.

type

Type: [GridMetric.LibV.Structure.ServiceType](#)

Type of the virtual service. Currently only supported value is [Service](#).

dependOnService

Type: System.String[]

Specifies list of services this service depends on. This value can be **null**.

group

Type: System.String

Group to which service belongs to. This value can be **null**.

dependOnGroup

Type: System.String[]

Specifies list of groups this service depends on. This value can be **null**.

tag

Type: System.Nullable<T>

Free form tag for service. This value can be **null**.

Properties

DependOnGroup

Gets or sets list of groups this service depends on.

Syntax:

```
public string[] DependOnGroup { get; set; }
```

Property value:

Type: System.String[]

List of groups or **null** if none specified.

DependOnService

Gets or sets list of services this service depends on.

Syntax:

```
public string[] DependOnService { get; set; }
```

Property value:

Type: System.String[]

List of service names or **null** if none specified.

Description

Gets or sets a description for the service.

Syntax:

```
public string Description { get; set; }
```

Property value:

Type: System.String

Description or **null** if none specified.

DisplayName

Gets or sets a display name for the service.

Syntax:

```
public string DisplayName { get; set; }
```

Property value:

Type: System.String

Display name or **null** if none specified.

ErrorControl

Gets or sets error control type for the service.

Syntax:

```
public ServiceErrorControl ErrorControl { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.ServiceErrorControl](#)

Error control type.

Group

Gets or sets a service's group name.

Syntax:

```
public string Group { get; set; }
```

Property value:

Type: System.String

Group name or **null** if none specified.

ImagePath

Gets or sets a path to an executable hosting the service.

Syntax:

```
public string ImagePath { get; set; }
```

Property value:

Type: System.String

Path to service's executable.

Name

Gets or sets a path to service name.

Syntax:

```
public string Name { get; set; }
```

Property value:

Type: System.String

Name of a service.

ObjectName

Gets or sets name of an account under which the service run.

Syntax:

```
public string ObjectName { get; set; }
```

Property value:

Type: System.String

Logon account name or **null** if none specified explicitly.

ServiceType

Gets or sets type of a service.

Syntax:

```
public ServiceType ServiceType { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.ServiceType](#)

Type of the service. Currently only supported value is [Service](#).

StartupType

Gets or sets start up type of a service.

Syntax:

```
public ServiceStartupType StartupType { get; set; }
```

Property value:

Type: [GridMetric.LibV.Structures.ServiceStartupType](#)

Start up type of the service.

Tag

Gets or sets optional tag for the service

Syntax:

```
public UInt32? Tag { get; set; }
```

Property value:

Type: System.Nullable<T>

Tag or **null** if none specified.

WindowsInfo class

Defines Windows operating system information for sequencing history entries implemented in class [HistoryEntry](#).

Namespace:

GridMetric.LibV.Structures

Syntax:

```
public class WindowsInfo
```

Methods

ToString

Provides full textual version information string of the Windows operating system version and service pack level as defined in the version properties in containing class.

Syntax:

```
public override string ToString()
```

Return value:

Type: System.String

Textual representation of the version information.

Note:

This method provides expanded version representation instead of just version numbers, for example: "Windows Server 2003 Enterprise Edition Service Pack 1".

Properties

Build

Gets the build number of Windows operating system under which the package was saved in.

Syntax:

```
public UInt32 Build { get; }
```

Property value:

Type: System.UInt32

Windows build number.

DBCSUser32

Gets the information whether User32.dll supports DBCS (Double-byte Character Set) on the operating system under which the package was saved in.

Syntax:

```
public bool DBCSUser32 { get; }
```

Property value:

Type: System.Boolean

true if DBCS is supported, otherwise **false**.

DebugUserExe

Gets the information whether debug version of User.exe is installed on the operating system under which the package was saved in.

Syntax:

```
public bool DebugUserExe { get; }
```

Property value:

Type: System.Boolean

true if debug version is installed, otherwise **false**.

LastBootNormal

Gets the information if the last boot was normal on the operating system under which the package was saved in.

Syntax:

```
public bool LastBootNormal { get; }
```

Property value:

Type: System.Boolean

true if last boot was normal, otherwise **false**.

Major

Gets the major version number of Windows operating system under which the package was saved in.

Syntax:

```
public UInt32 Major { get; }
```

Property value:

Type: System.UInt32

Windows major version number.

MaximumMemoryAddress

Gets the highest memory address available for applications on operating system under which the package was saved in.

Syntax:

```
public UInt32 MaximumMemoryAddress { get; }
```

Property value:

Type: System.UInt32

Maximum address.

Note:

Since this value is 32-bit integer, packages encoded by Lib-V on 64-systems has this value set to 0 as 64-bit address space limits cannot be represented by this property.

MinimumMemoryAddress

Gets the lowest memory address available for applications on operating system under which the package was saved in.

Syntax:

```
public UInt32 MinimumMemoryAddress { get; }
```

Property value:

Type: System.UInt32

Minimum address.

Minor

Gets the minor version number of Windows operating system under which the package was saved in.

Syntax:

```
public UInt32 Minor { get; }
```

Property value:

Type: System.UInt32

Windows minor version number.

PageSize

Gets the memory allocation page granularity on operating system under which the package was saved in.

Syntax:

```
public UInt32 PageSize { get; }
```

Property value:

Type: System.UInt32

The page size and granularity.

PlatformId

Gets the platform id for the operating system under which the package was saved in.

Syntax:

```
public UInt32 PlatformId { get; }
```

Property value:

Type: System.UInt32

Platform id. This is always initialized to value of 2 (VER_PLATFORM_WIN32_NT).

ProcessorMask

Gets the mask for processors that are configured on the under which the package was saved in.

Syntax:

```
public UInt32 ProcessorMask { get; }
```

Property value:

Type: System.UInt32

Processor mask.

Note:

Although this property has to do with processors, it's a operating system dependent value of allowed processors by the Windows operating system edition.

RemoteControl

Gets the information whether the session, under which the package was saved in, was remotely controlled in the Terminal Services environment.

Syntax:

```
public bool RemoteControl { get; }
```

Property value:

Type: System.Boolean

true if the session was under remote control, otherwise **false**.

ServicePackMajor

Gets the major version number of Service Pack installed on the Windows operating system under which the package was saved in.

Syntax:

```
public UInt32 ServicePackMajor { get; }
```

Property value:

Type: System.UInt32

Windows Service Pack major version number.

ServicePackMinor

Gets the minor version number of Service Pack installed on the Windows operating system under which the package was saved in.

Syntax:

```
public UInt32 ServicePackMinor { get; }
```

Property value:

Type: System.UInt32

Windows Service Pack minor version number.

Suite

Gets the operating system suite information of Windows operating system under which the package was saved in.

Syntax:

```
public NtSuiteMasks Suite { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.NtSuiteMasks](#)

Suite mask for the Windows Operating System.

SystemDirectory

Gets the system directory path on the operating system under which the package was saved in.

Syntax:

```
public string SystemDirectory { get; }
```

Property value:

Type: System.String

Path to a system directory.

SystemType

Gets the operating system type information of Windows operating system under which the package was saved in.

Syntax:

```
public NtSystemType SystemType { get; }
```

Property value:

Type: [GridMetric.LibV.Structures.NtSystemType](#)

System type for the Windows Operating System.

TerminalServices

Gets the information whether the session, under which the package was saved in, was a Terminal Services/Remote Desktop session.

Syntax:

```
public bool TerminalServices { get; }
```

Property value:

Type: System.Boolean

true if the session was a Terminal Services/Remote Desktop session, otherwise **false**.

UserWindowsDirectory

Gets the user account's Windows directory path on the operating system under which the package was saved in.

Syntax:

```
public string UserWindowsDirectory { get; }
```

Property value:

Type: System.String

Path to a Windows directory as seen by the user account.

Version

Gets the full version information (*major.minor.build*) of the operating system under which the package was saved in.

Syntax:

```
public string Version { get; }
```

Property value:

Type: System.String

Full version.

WindowsDirectory

Gets the Windows directory path on the operating system under which the package was saved in.

Syntax:

```
public string WindowsDirectory { get; }
```

Property value:

Type: System.String

Path to a Windows directory.

Enumerations

Enumerations as used by classes provided in Lib-V core, encoding and decoding.

Namespace:

GridMetric.LibV.Structures

NtSuiteMasks

Specifies enumerated constants used as Windows suite identifier for [HistoryEntry](#).

Syntax:

```
[Flags() ]  
public enum NtSuiteMasks : ushort
```

Note:

Further description of suite mask can be found from Microsoft Platform SDK.

BackOffice

Blade

Datacenter

Enterprise

Embedded

Personal

SingleUserTs

SmallBusiness

SmallBusinessRestricted

Terminal

SecurityAppliance

StorageServer

ComputeServer

HomeServer

NtSystemType

Specifies enumerated constants used as Windows system type identifier for [HistoryEntry](#).

Syntax:

```
public enum NtSystemType : ushort
```

Note:

Further description of system types can be found from Microsoft Platform SDK.

DomainController

Server

Workstation

PackageSaveMode

Specifies enumerated constants used to indicate SFT package saving mode for [HistoryEntry](#).

Syntax:

```
public enum PackageSaveMode : uint
```

Save

Package was saved normally as subsequent version.

SaveAs

Package was saved as an initial version or as branched version from the existing package.

SaveAsNew

Package was saved as branched version from the existing package.

Note: This value is used starting from 4.5 version of SFT and it does not appear in older packages.

ProcessorFamily

Specifies enumerated constants used to indicate processor family for [HistoryEntry](#).

Syntax:

```
public enum ProcessorFamily : uint
```

Note:

Currently all SoftGrid/App-V packages store this value as [Pentium](#). Further description of processor families can be found from Microsoft Platform SDK.

Pentium

ServiceErrorControl

Specifies enumerated constants used to indicate NT service's error control action as used by [VirtualService](#) class.

Syntax:

```
public enum ServiceErrorControl : uint
```

IgnoreProblemAndDisplayNoError

ContinueBootupAndProduceWarning

PanicAndUseLastKnownGood

FailureAndRunDiagnostic

ServiceStartupType

Specifies enumerated constants used to indicate NT service's start-up type as used by [VirtualService](#) class.

Syntax:

```
public enum ServiceStartupType : uint
```

Note:

Only Automatic, Manual and Disabled are valid values for virtual services.

Boot

System

Automatic

Manual

Disabled

ServiceType

Specifies enumerated constants used to indicate NT service type as used by [VirtualService](#) class.

Syntax:

```
public enum ServiceType : uint
```

Note:

Only Service is valid value for virtual services.

[DeviceDriver](#)

[FileSystemDriver](#)

[NetworkAdapter](#)

[Service](#)

[SharedService](#)

SftCompressionType

Specifies enumerated constants used to indicate compression type used for SFT package.

Syntax:

```
public enum SftCompressionType : uint
```

None

Specifies that no compression is applied to a SFT file.

Bzip2

Specifies that BZIP2 compression algorithm is applied to a SFT file.

Note: This compression method has been deprecated starting from version 4.5.

Zlib

Specifies that ZLIB compression algorithm is applied to a SFT file.

SftDecodingOptions

Specifies enumerated constants used as flags to [Decode\(\)](#) method in [SftDecoder](#) class.

Syntax:

```
[Flags() ]  
public enum SftDecodingOptions
```

DEFAULTS

Decodes SFT file using default options.

SKIP_BLOCKSCAN

Do not scan DATA blocks during decoding phase. This makes package decoding faster as only metadata section of the package will be read in.

If DATA blocks are not scanned during decoding phase, accessing individual [VirtualFileBlock](#) -objects in [VirtualFile](#) will trigger Just-In-Time scanning and reading of requested block.

IGNORE_VE_ERRORS

Do not throw exceptions if Virtual Environment cannot be parsed successfully.

This makes it possible to decode SFT files with corrupted virtual environment, although decoded package might be missing of Virtual Registry or VFS mapping information.

DECODE_FILESTRUCTURE

Instructs decoding engine to decode all the files and directories from the package to a physical directory (specified using [InitializeDecoder\(\)](#) method). Otherwise and as default, no files are extracted from package when it is decoded.

Package's virtual environment configuration file, `osguard.cp`, is never extracted when this flag is used. This is to ensure that it can be properly overwritten regardless of the security descriptor, if package is then re-encoded.

Please note that it is not currently supported to make direct modifications to these decoded files for the purpose of re-encoding them to new SFT package. You will have to use [GetStream\(\)](#) method from the appropriate [VirtualFile](#) to make any persistent modification to it.

SKIP_UNSETTABLE_ACLS

When used in conjunction with [DECODE_FILESTRUCTURE](#), instructs decoding engine to ignore errors in setting security descriptors to file or directory. Otherwise exception will be thrown if file or directory has security descriptor that cannot be assigned to extracted copy in the filesystem under current account's security rights.

SftDecodingPhase

Specifies enumerated constants used by [SftDecodingEventArgs](#) -class.

Syntax:

```
public enum SftDecodingPhase
```

HeaderScanAndParse

Decoding engine is processing SFT file metadata headers.

DataBlockScan

Decoding engine is scanning DATA blocks.

VirtualEnvironmentRead

Decoding engine is reading osguard.cp file's contents from the package.

VirtualEnvironmentScan

Decoding engine is parsing virtual environment configuration.

VirtualRegistryParse

Decoding engine is parsing Virtual Registry's contents.

Note that since SFT file contains three separate instances of virtual registry data, this phase will be repeated three times.

VirtualFilesystemParse

Decoding engine is parsing Virtual Filesystem (VFS) entries.

FilesystemDecoding

Decoding engine is extracting package's directories and files to temporary directory if [DECODE_FILESTRUCTURE](#) -flag has been passed to it.

SftEncodingOptions

Specifies enumerated constants used as flags to [Encode\(\)](#) method in [SftEncoder](#) class.

Syntax:

```
[Flags() ]  
public enum SftEncodingOptions
```

DEFAULTS

Encodes SFT file using default options.

SKIP_VE_ENCODING

Do not include Virtual Environment configuration in the package (osguard.cp file).

Note that leaving VE out of the package is generally not recommended.

ENCODE_AS_DIFFERENTIAL

Encode package as DSFT file.

This option is currently not supported and is reserved for future use.

NO_VERSION_INCREMENT

Package version is not automatically increment during encoding.

NO_HISTORY_GENERATION

No new history entry / version entry is generated for encoded package.

NO_TIMESTAMP_INCREMENT

Package timestamps are not automatically incremented during encoding.

REGENERATE_IDENTIFIERS

Regenerates all unique identifiers in the package, making it a completely new package from the SoftGrid/App-V's standpoint.

REMOVE_PACKAGE_HISTORY

Removes existing package version history.

This removal will not affect on automatic generation of new history/version entry for present encoding, which can be suppressed with [NO_HISTORY_GENERATION](#) flag.

STRICT_SEQUENCER_HISTORY

For updated packages, Lib-V will re-use sequencer version information from the previous save entry instead of generating minimum compatible version number. This will ensure more logical version continuation for the package history.

This option has no effect if SFT file-format version is changed during encoding.

SftEncodingPhase

Specifies enumerated constants used by [SftEncodingEventArgs](#) -class.

Syntax:

```
public enum SftEncodingPhase
```

ContentSerialization

Encoding engine is serializing package's content.

Note that this event is emitted throughout the whole encoding process.

HeaderEncoding

Encoding engine is encoding SFT file metadata headers.

VirtualEnvironmentEncoding

Encoding engine is creating virtual environment for the package.

VirtualRegistryEncoding

Encoding engine is encoding Virtual Registry.

This event phase is not currently used and is reserved for future use.

VirtualFilesystemEncoding

Encoding engine is encoding Virtual Filesystem (VFS) entries.

This event phase is not currently used and is reserved for future use.

DatablockOrdering

Encoding engine is creating DATA block entries for the package.

DatablockEncoding

Encoding engine is encoding DATA blocks to SFT file.

SftFileformatType

Specifies enumerated constants used to indicate SFT file format type.

Syntax:

```
public enum SftFileformatType
```

Unknown

File is in unknown file format. This value is never used by Lib-V.

SftFile

File is a normal SFT file. This is currently the only supported format.

DsftFile

File is a differential SFT file. This format is not currently supported by Lib-V.

SftFileformatVersion

Specifies enumerated constants used to indicate SFT file format version.

Syntax:

```
public enum SftFileformatVersion
```

Unknown

SFT file is of unknown version.

Version2xTo3x

SFT file is in format supported by SoftGrid 2.x and 3.x series.

Version4x

SFT file is in format supported by SoftGrid versions 4.0, 4.1 and 4.2.

Version45

SFT file is in format supported by Microsoft App-V 4.5.

Version46

SFT file is in format supported by Microsoft App-V 4.6.

SoftricityFileAttributes

Specifies enumerated constants used in [SoftricityAttributes](#) property in [VirtualFile](#) and [VirtualDirectory](#) classes. These flags tell SoftGrid/App-V specific virtualization characteristics of a directory entry and they are referred to as "Sequencer Attributes" in Microsoft App-V 4.5 Sequencer.

Syntax:

[Flags()]

```
public enum SoftricityFileAttributes : ushort
```

None

No Softricity attributes are applied to a directory entry.

AppData

File/directory is of Application Data type.

AppConfig

File/directory is of Application Configuration type.

Note: This attribute has been deprecated starting from version 4.5.

UserData

File/directory is of User Data type.

UserConfig

File/directory is of User Configuration type.

Note: This attribute has been deprecated starting from version 4.5.

Permanent

File/directory is marked as permanent.

Note: This attribute has been deprecated starting from version 4.5.

Override

File/directory is marked as being overridden.

VirtualFilesystemAttributes

Specifies enumerated constants used in [VfsMappingTarget](#) class. These flags contain Virtual Filesystem (VFS) -specific virtualization information.

Syntax:

[**Flags()**]

```
public enum VirtualFilesystemAttributes : uint
```

None

No attributes specified.

Entry

VFS mapping entry is an existing directory or file.

Font

VFS mapping entry is virtualized as font.

Directory

VFS mapping entry is a pre-existing directory.

Deleted

VFS mapping entry is virtualized as deleted.

All entries that are virtualized as deleted, will fully mask files and directories beneath it.

VirtualFilesystemMappingAttributes

Specifies enumerated constants used in [VfsMappingTarget](#) class. These flags contain Virtual Filesystem (VFS) mapping -specific virtualization information.

Syntax:

[Flags()]

```
public enum VirtualFilesystemMappingAttributes : uint
```

None

No attributes specified.

TargetHasIntermediateValue

Path into which directory entry is mapped to (on top of the real file system) contains Intermediate Values in its name.

This flag specifies that client has to resolve intermediate values to get real folder path (such as %CSIDL_WINDOWS% → C:\Windows).

Active

This mapping information is in active state.

As one file or directory in the SFT's VFS section can map to multiple target paths, this flag will indicate if the target is an active one.

VirtualFilesystemMode

Specifies enumerated constants used in [VfsMode](#) property in [VirtualFile](#) and [VirtualDirectory](#) classes.

Syntax:

```
public enum VirtualFilesystemMode
```

None

Entry is not virtualized within Virtual Filesystem functionality.

This value is used for all the directories and files not placed in the VFS subdirectory of the package's root.

Merge

Entry is merged with path in the real file system.

This value is used only for the directories and setting directory as merged will require identical directory to exist on the client.

Override

Entry overrides possibly conflicting path in the real file system. In pre-4.5 SoftGrid versions, this type of virtualization was referred to as "Virtualized" in the Sequencer.

VirtualRegistryAttributes

Specifies enumerated constants used by [RegistryAttributes](#) property in [VirtualRegistryKey](#) and [VirtualRegistryValue](#) classes. These flags contain virtualization information common for both registry keys and values.

Syntax:

[Flags()]

```
public enum VirtualRegistryAttributes : uint
```

Note:

Not all of these flags can be set to both virtual keys and values. Some of them are only used with the virtual registry keys.

None

No attributes specified.

ValueIsDeleted

This flag is set for virtual registry values that need to be virtualized as deleted into target registry. Deleted virtual values cannot hold any value data.

ValueIsVirtualized

This flag is always set for all virtual registry values and for those keys that hold default value in it.

KeyWasRenamed

This flag is set for virtual registry keys that had been renamed and subsequently set to deleted state.

ValueHasData

This flag is set for virtual registry values that store value data and for those keys that hold default value storing data in it.

KeyOverridesLocal

This flag is set for virtual registry keys which override local equivalent key.

In pre-4.5 SoftGrid versions, this type of virtualization was referred to as "Virtualized" in the Sequencer.

If this flag is not set, virtual registry key will merge its contents with the locally present key.

KeyIsDeleted

This flag is set for virtual registry key that need to be virtualized as deleted into target registry.

All virtual registry keys that are virtualized as deleted fully masks keys and values beneath it.

CapturedWhileMonitoring

This flag is set for virtual registry keys and values that were captured during monitoring operation.

Sequencer sets this attribute for those registry entries which were detected as new or changed during monitoring operation.

KeyCreatedWhileMonitoring

This flag is set for virtual registry keys that were created as new.

VirtualRegistryIMStatusAttributes

Specifies enumerated constants used by [IntermediateValueEncodingStatus](#) property in [VirtualRegistryKey](#) and [VirtualRegistryValue](#) classes. These flags contain virtualization information related to Intermediate Value mapping requirement for key/value names and data.

Syntax:

```
[Flags()]  
public enum VirtualRegistryIMStatusAttributes
```

None

Key's or value's name or data does not contain values encoded with Intermediate Values.

NameOrPathHasEncodedValues

Key's name (or full path) or value's name contain mapped (encoded) Intermediate Values.

Encoded version of the name or path can be read using [RawName](#) and [RawPath](#) properties, whereas automatically decoded name and path can be accessed through [Name](#) and [Path](#) properties.

DataHasEncodedValues

Key's default data or value data contain mapped Intermediate Values.

Encoded version of the data can be read using [RawData](#) property, whereas automatically decoded data can be accessed through [Data](#) property.

Lib-V Metadata and serialization assemblies API reference

Lib-V's App-V metadata files (OSD, SPRJ, Manifest XML and MSI) support is implemented through additional assemblies from the core SFT-handling assemblies.

GridMetric.LibV.Metadata.dll assembly contains core object classes for accessing properties of those metadata files, while GridMetric.LibV.Metadata.Serialization.dll assembly will handle (de)serialization of those metadata files.

If creating a new package using Lib-V, all metadata files can be instantiated from scratch using constructors in main metadata assembly and then serialized to the disk with serialization assembly.

When upgrading an existing package, however - as all App-V package files contain some references to other files in the package - for some metadata files like Manifest XML it is also better to instantiate it anew each time before serialization since Manifest's contents can be re-constructed fully from package's OSD files. Other files containing some unique data in them, like for instance SPRJ, should be brought up-to-date (in case of SPRJ, using [OsdFileNames](#) and [SftFilename](#) properties) using existing deserialized object.

Note:

Although being stored in the separate assembly, serialization support for metadata file objects is implemented through extension methods to make it more convenient for caller. While extension methods are strictly speaking a .NET 3.0 -feature, Lib-V re-declares `ExtensionAttribute` -attribute inside the serialization -assembly which makes extension methods to work still with .NET 2.0 as used by the Lib-V. Only thing required to make this work in the calling code is to add reference to *GridMetric.LibV.Metadata.Serialization* - namespace with using -statement.

Deserialization support is handled through normal static methods inside serialization classes.

ManifestAppElement class

Stores application information in the Manifest XML file. Each <APP> element maps to one published application in the App-V package (i.e. one OSD file).

Namespace:

GridMetric.LibV.Metadata.Manifest

Syntax:

```
public class ManifestAppElement
```

Constructors

Overload List:

Initializes empty application reference.

```
public ManifestAppElement()
```

Initializes application reference based on information from existing OSD file and specified icon subdirectory name. Assumes that application icon file is named based on OSD filename (same base name but with .ico extension).

```
public ManifestAppElement(  
    OsdFile osdFile,  
    string osdFilename,  
    string iconSubdirectoryName  
)
```

Initializes application reference based on information from existing OSD file, specified icon subdirectory name and name of the application icon file.

```
public ManifestAppElement(  
    OsdFile osdFile,  
    string osdFilename,  
    string iconSubdirectoryName,
```

string *osdApplicationIconFilename*
)

Parameters:

osdFile

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

OSD file object reference from which content for manifest application element is automatically read from.

osdFilename

Type: System.String

(Physical) name of the OSD file for the OSD represented by the application element.

iconSubdirectoryName

Type: System.String

Name of subdirectory under App-V package's directory wherein icons are stored. This name is by default [package] name of the App-V package + Icons –suffix (e.g. “Adobe Reader 8 Icons”) for packages created with Microsoft Sequencer.

osdApplicationIconFilename

Type: System.String

(Physical) name of the ICO file for application. If this parameter is not given, name is automatically formed based on OSD's filename by switching .OSD extension to .ICO.

Properties

FileTypes

Gets or sets list of file types associated with the application.

Syntax:

```
public IList<ManifestFileTypeElement> FileTypes { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of file type associations or **null** if none specified.

IconPath

Gets or sets path to application's icon file in package-relative format (e.g. "%SFT_MIME_SOURCE%/TextPad Icons/TextPad.ico").

Syntax:

```
public System.String IconPath { get; set; }
```

Property value:

Type: System.String

Path to an icon file for this application.

Note:

App-V Client recognizes special variable *%SFT_MIME_SOURCE%* as a substitute for directory containing package files.

Name

Gets or sets application name.

Syntax:

```
public System.String Name { get; set; }
```

Property value:

Type: System.String

Application name.

OsdPath

Gets or sets path to application's OSD file in package-relative format (e.g. "%SFT_MIME_SOURCE%/TextPad.osd").

Syntax:

```
public System.String OsdPath { get; set; }
```

Property value:

Type: System.String

Path to an OSD file this application represents.

Note:

App-V Client recognizes special variable *%SFT_MIME_SOURCE%* as a substitute for directory containing package files.

Shortcuts

Gets or sets list of shortcuts associated with the application.

Syntax:

```
public IList<OsdShortcutElement> Shortcuts { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of shortcuts or **null** if none specified.

Version

Gets or sets application version.

Syntax:

```
public System.String Version { get; set; }
```

Property value:

Type: System.String

Application version.

ManifestFile class

Represents Manifest XML file in an object format. Manifest contains publishing -related information from all OSD files in the App-V package.

Namespace:

GridMetric.LibV.Metadata.Manifest

Syntax:

```
public class ManifestFile : MetadataFileBase
```

Constructors

Overload List:

Initializes Manifest file from specified OSD references and SFT file reference, using empty icon directory name. Supplied list of OSD files is used to populate list of [ManifestAppElements](#) in the Manifest object.

```
public ManifestFile(  
    SftPackage sftPackage,  
    NameOsdfFileCollection osdCollection  
)
```

Initializes Manifest file from specified OSD references, SFT file reference and icon subdirectory name. Supplied list of OSD files is used to populate list of [ManifestAppElements](#) in the Manifest object.

```
public ManifestFile(  
    SftPackage sftPackage,  
    NameOsdfFileCollection osdCollection,  
    string iconSubdirectoryName  
)
```

Parameters:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

sftPackage

Type: [GridMetric.LibV.SftPackage](#)

SFT file object for which this Manifest file will have relation to. SFT reference is used to populate ID and version information stored in the Manifest.

osdCollection

Type: [GridMetric.LibV.Metadata.Osd.NameOsdFileCollection](#)

Collection of OSD file objects and associated filenames for those OSD files. OSD collection is used to populate actual Manifest content of application publishing information.

iconSubdirectoryName

Type: System.String

Name of subdirectory under App-V package's directory wherein icons are stored. This name is by default [package] name of the App-V package + Icons –suffix (e.g. “Adobe Reader 8 Icons”) for packages created with Microsoft Sequencer.

Methods

AddOsdFile

Adds new OSD (i.e. application) reference to the Manifest file.

Syntax:

```
public void AddOsdFile(  
    OsdFile newOsdFile,  
    string osdName  
)
```

Parameters:

newOsdFile

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

Specifies OSD file to add.

OsdName

Type: System.String

Physical filename for the OSD file.

ManifestFileTypeElement class

Stores file-type association information for application in the Manifest XML file.

Further information on file type associations in Windows can be found in Microsoft MSDN at <http://msdn.microsoft.com/en-us/library/cc144148.aspx>.

Namespace:

GridMetric.LibV.Metadata.Manifest

Syntax:

```
public class ManifestFileTypeElement
```

Constructors

Initializes new file type association for an application.

```
public ManifestFileTypeElement(  
    string extension,  
    OsdProgIdElement progId,  
    string perceivedType,  
    string contentType,  
    bool? overridden  
)
```

Parameters:

extension

Type: System.String

File extension (without leading dot) this file-type is associated with.

progId

Type: [GridMetric.LibV.Metadata.Osd.OsdProgIdElement](#)

[A Programmatic Identifier](#) for file association. ProgID definitions links file extensions to actual applications handling those extensions and multiple extensions can share the same ProgID.

perceivedType

Type: System.String

[A Classification](#) for the file-extension. Can be **null** if no perceived type is specified for an association.

contentType

Type: System.String

[A MIME type](#) for file extension. Can be **null** if no MIME type is specified for an association.

overridden

Type: System.Nullable<T>

Should be set to **null**.

Properties

ContentType

Gets or sets content type for the file association.

Syntax:

```
public string ContentType { get; set; }
```

Property value:

Type: System.String

Content type.

Extension

Gets or sets file extension (without leading dot) for the file association.

Syntax:

```
public string Extension { get; set; }
```

Property value:

Type: System.String

File extension.

Overridden

Gets or sets override information for the file association. This property should be set to null.

Syntax:

```
public bool? Overridden { get; set; }
```

Property value:

Type: System.Nullable<T>

Is overridden.

PerceivedType

Gets or sets perceived type for the file association.

Syntax:

```
public string PerceivedType { get; set; }
```

Property value:

Type: System.String

Perceived type information.

ProgId

Gets or sets ProgID reference for the file association.

Syntax:

```
public OsdProgIdElement ProgId { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdProgIdElement](#)

Programmatic ID this association links to.

ShellNewEnabled

Gets or sets information if new shell is enabled for association.

Syntax:

```
public bool ShellNewEnabled { get; set; }
```

Property value:

Type: System.Boolean

Is new shell enabled.

ManifestSerializer class

Serializes and deserializes App-V manifest XML file between file and object format.

Namespace:

GridMetric.LibV.Metadata.Serialization

Syntax:

```
public static class ManifestSerializer
```

Static methods

LoadManifestFromFile

Loads manifest from file and deserializes it to object format.

Syntax:

```
public static ManifestFile LoadManifestFromFile (  
    string pathToManifestFile  
)
```

Parameters:

pathToManifestFile

Type: System.String

Full path to manifest file.

Return value:

Type: [GridMetric.LibV.Metadata.Manifest.ManifestFile](#)

Deserialized manifest object.

ParseManifestFromStream

Loads manifest file's contents from specified stream and deserializes it to object format.

Overload list:

Loads manifest file from supplied stream.

```
public static ManifestFile ParseManifestFromStream (  
    Stream manifestFileStream  
)
```

Loads manifest file from supplied stream using forced character encoding instead of auto-detection.

```
public static ManifestFile ParseManifestFromStream (  
    Stream manifestFileStream,  
    Encoding forcedEncoding  
)
```

Parameters:

manifestFileStream

Type: System.IO.Stream

Stream to read manifest file's contents from.

forcedEncoding

Type: System.Text.Encoding

Encoding to use when interpreting the byte sequence from stream.

Return value:

Type: [GridMetric.LibV.Metadata.Manifest.ManifestFile](#)

Deserialized manifest object.

ParseManifestFromXml

Loads manifest from XML document and deserializes it to object format.

Syntax:

```
public static ManifestFile ParseManifestFromXml (  
    XmlDocument manifestXmlRepresentation  
)
```

Parameters:

manifestXmlRepresentation

Type: System.Xml.XmlDocument

XML document representing the manifest.

Return value:

Type: [GridMetric.LibV.Metadata.Manifest.ManifestFile](#)

Deserialized manifest object.

Save

Saves supplied [ManifestFile](#) object into file.

Overload list:

Saves manifest to file in XML format using character encoding specified in the object (by default ANSI).

```
public static void Save (  
    this ManifestFile manifest,  
    string fileName  
)
```

Saves manifest to file in XML format using forced character encoding.

```
public static void Save (  
    this ManifestFile manifest,  
    string fileName,
```

Encoding *requiredEncoding*
)

Parameters:

manifest

Type: [GridMetric.LibV.Metadata.Manifest.ManifestFile](#)

Manifest file to serialize.

fileName

Type: System.String

Path to a file to which save Manifest to. Manifest file is usually named based on package's name and adding "_manifest.xml" to it.

requiredEncoding

Type: System.Text.Encoding

Encoding to use.

Note:

This class extends [ManifestFile](#) –class so that Save(...) -method can be used directly from the manifest object itself. Only requirement is that codefile adds "using GridMetric.LibV.Metadata.Serialization" –clause in its using –block.

SerializeToStream

Serializes supplied [ManifestFile](#) object into stream.

Overload list:

Serializes manifest to supplied stream in XML format using character encoding specified in the object (by default ANSI).

```
public static void SerializeToStream (  
    this ManifestFile manifest,  
    Stream outputStream
```

)

Serializes manifest to supplied stream in XML format using forced character encoding.

```
public static void SerializeToStream (  
    this ManifestFile manifest,  
    Stream outputStream,  
    Encoding requiredEncoding  
)
```

Parameters:

manifest

Type: [GridMetric.LibV.Metadata.Manifest.ManifestFile](#)

Manifest file to serialize.

outputStream

Type: System.IO.Stream

Stream to write serialized manifest to.

requiredEncoding

Type: System.Text.Encoding

Encoding to use.

Note:

This class extends [ManifestFile](#) –class so that `SerializeToStream(...)` -method can be used directly from the manifest object itself. Only requirement is that codefile adds “using `GridMetric.LibV.Metadata.Serialization`” –clause in its using –block.

SerializeToXml

Serializes supplied [ManifestFile](#) object into XML document.

Overload list:

Serializes manifest to XML document using character encoding specified in the object (by default ANSI).

```
public static XmlDocument SerializeToXml (  
    this ManifestFile manifest  
)
```

Serializes manifest to XML document using forced character encoding.

```
public static XmlDocument SerializeToXml (  
    this ManifestFile manifest,  
    Encoding requiredEncoding  
)
```

Parameters:

manifest

Type: [GridMetric.LibV.Metadata.Manifest.ManifestFile](#)

Manifest file to serialize.

requiredEncoding

Type: System.Text.Encoding

Encoding to use.

Return value:

Type: System.Xml.XmlDocument

App-V Manifest file as XML document.

Note:

This class extends [ManifestFile](#) –class so that `SerializeToXml(...)` -method can be used directly from the manifest object itself. Only requirement is that codefile adds “using `GridMetric.LibV.Metadata.Serialization`” –clause in its using –block.

MetadataFileBase

Represents a base class for all metadata files ([OsdFile](#), [SprjFile](#), [ManifestFile](#) and [MsiWrapperPackage](#)) objects.

Namespace:

GridMetric.LibV.Metadata

Syntax:

```
public abstract class MetadataFileBase
```

Properties

Tag

Gets or sets the object that contains data about the metadata file.

Syntax:

```
public object Tag { get; set; }
```

Property value:

Type: System.Object

Object to associate with metadata file.

Note: Lib-V does not use this property, it is for caller to associate any custom information with the metadata files.

.

MetadataSerializer class

Serializes and deserializes App-V package's metadata files.

Namespace:

GridMetric.LibV.Metadata.Serialization

Syntax:

```
public static class MetadataSerializer
```

MsiSerializer class

Serializes and deserializes App-V package MSI wrapper file's information from and to object format.

Namespace:

GridMetric.LibV.Metadata.Serialization

Syntax:

```
public static class MsiSerializer
```

Static methods

LoadMsiFromFile

Loads information from existing App-V MSI file and deserializes it to object format.

Syntax:

```
public static MsiWrapperPackage LoadMsiFromFile (  
    string pathToMsiFile  
)
```

Parameters:

pathToMsiFile

Type: System.String

Full path to MSI file.

Return value:

Type: [GridMetric.LibV.Metadata.Msi.MsiWrapperPackage](#)

Deserialized MSI object.

Save

Saves supplied [MsiWrapperPackage](#) object into file.

Overload list:

Saves customizations to MSI file and updates cabinet inside MSI file with supplied OSD, ICO and Manifest files. Re-generates MSI product and package codes but not upgrade code.

```
public static void Save (  
    this MsiWrapperPackage msiPackage,  
    string msiFileName,  
    IList<string> osdPaths,  
    IList<string> icoPaths,  
    string manifestPath  
)
```

Saves customizations to MSI file using called-supplied MSI template file and updates cabinet inside MSI file with supplied OSD, ICO and Manifest files. Re-generates MSI product and package codes but not upgrade code.

```
public static void Save (  
    this MsiWrapperPackage msiPackage,  
    string msiFileName,  
    IList<string> osdPaths,  
    IList<string> icoPaths,  
    string manifestPath,  
    string msiTemplateFilePath  
)
```

Saves customizations to MSI file and updates cabinet inside MSI file with supplied OSD, ICO and Manifest files.

```
public static void Save (  
    this MsiWrapperPackage msiPackage,  
    string msiFileName,  
    IList<string> osdPaths,  
    IList<string> icoPaths,
```

```

        string manifestPath,
        bool preserveCustomCodes
    )

```

Saves customizations to MSI file using called-supplied MSI template file and updates cabinet inside MSI file with supplied OSD, ICO and Manifest files.

```

public static void Save (
    this MsiWrapperPackage msiPackage,
    string msiFileName,
    IList<string> osdPaths,
    IList<string> icoPaths,
    string manifestPath,
    bool preserveCustomCodes,
    string msiTemplateFilePath
)

```

Parameters:

msiPackage

Type: [GridMetric.LibV.Metadata.Msi.MsiWrapperPackage](#)

MSI wrapper file to save.

msiFileName

Type: System.String

Path to a file to which save MSI to.

osdPaths

Type: System.Collections.Generic.IList<T>

List of full paths to package's up-to-date OSD files which will be put into MSI's internal cabinet file.

icoPaths

Type: System.Collections.Generic.IList<T>

List of full paths to package's up-to-date ICO files which will be put into MSI's internal cabinet file.

manifestPath

Type: System.String

Full path to a package's up-to-date manifest file which will be put into MSI's internal cabinet file.

msiTemplateFilePath

Type: System.String

Full path to a template MSI file to be used as a basis for generated (updated) MSI wrapper package. This MSI needs to derive from original template MSI used by Lib-V (available from customer portal as a separate download) as Lib-V's MSI output functionality relies on having specific MSI properties present for modification/updating.

preserveCustomCodes

Type: System.Boolean

Set to **true** if MSI's [product](#) and [package](#) codes needs to be maintained and not re-generated, **false** if Lib-V can re-generate both codes to conform way App-V Sequencer updates MSI files upon package save (default).

Note:

This class extends [MsiWrapperPackage](#) –class so that Save(...) -method can be used directly from the MSI wrapper object itself. Only requirement is that codefile adds “using GridMetric.LibV.Metadata.Serialization” –clause in its using –block.

MsiWrapperPackage class

Represents App-V package MSI file in an object format. MSI wrapper package can be used to deploy App-V applications into App-V Clients 4.5 and newer configured to accept full stand-alone or streaming usage.

If re-using existing MSI file instead of generating a whole new one, use [MsiSerializer](#) class to deserialize information from source MSI into MsiWrapperPackage object.

Namespace:

GridMetric.LibV.Metadata.Msi

Syntax:

```
public class MsiWrapperPackage : MetadataFileBase
```

Constructors

Overload List:

Initializes a new MSI wrapper package using specified file- and directory name references and [SFT package's unique ID](#).

```
public MsiWrapperPackage (  
    string sftFilename,  
    string manifestFilename,  
    Guid packageId,  
    string iconSubdirectoryName  
)
```

Initializes a new MSI wrapper package using specified file- and directory name references and SFT package object from which necessary attributes are read from.

```
public MsiWrapperPackage (  
    string sftFilename,  
    string manifestFilename,  
    SftPackage sftPackage,  
    string iconSubdirectoryName
```

)

Parameters:

sftFilename

Type: System.String

Physical filename of the SFT file this MSI file has relation to.

manifestFilename

Type: System.String

Physical filename of the Manifest XML file this MSI file has relation to.

sftPackage

Type: [GridMetric.LibV.SftPackage](#)

SFT package object this MSI file has relation to. The reference is used to populate SFT package id information in the MSI file.

iconSubdirectoryName

Type: System.String

Name of subdirectory under App-V package's directory wherein icons are stored. This name is by default [package] name of the App-V package + Icons –suffix (e.g. "Adobe Reader 8 Icons") for packages created with Microsoft Sequencer.

Properties

ArpComments

Sets or gets comment field for package's Add/Remove programs listing.

Syntax:

```
public string ArpComments { get; set; }
```

Property value:

Type: System.String

Comment text.

ArpContact

Sets or gets contact field information for package's Add/Remove programs listing.

Syntax:

```
public string ArpContact { get; set; }
```

Property value:

Type: System.String

Contact information text.

ArpInfoUrl

Sets or gets more information URL / Support link field for package's Add/Remove programs listing.

Syntax:

```
public string ArpInfoUrl { get; set; }
```

Property value:

Type: System.String

Support link text.

IconSubdirectoryName

Sets or gets directory name of the icon subdirectory for package. This is the subdirectory location wherein App-V package's .ICO files are stored in instead of main package directory.

Syntax:

```
public string IconSubdirectoryName { get; set; }
```

Property value:

Type: System.String

Name of the icon subdirectory.

InstallMode

Sets or gets App-V package MSI's installation mode. Controls whether or not the (externally located) SFT file is loaded into App-V Client cache during MSI installation process or left to be streamed on-demand later on.

Syntax:

```
public MsiInstallMode InstallMode { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.MsiInstallMode](#)

Installation mode. By default App-V package MSIs are set to [MsiInstallMode.Standalone](#).

LoadSft

Controls if the SFT is automatically loaded in when MSI installation is run, depending on [InstallMode](#).

Syntax:

```
public bool LoadSft { get; set; }
```

Property value:

Type: System.Boolean

True (default) if SFT is loaded in stand-alone mode or **false** if no loading is attempted.

ManifestFilename

Sets or gets physical filename of the Manifest XML file.

Syntax:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

```
public string ManifestFilename { get; set; }
```

Property value:

Type: System.String

Name of the Manifest XML file.

Important note:

[SftFilename](#) and [ManifestFilename](#) properties **must** always be updated to reflect actual names of the respective files, otherwise MSI installation will end in error.

ManufacturerName

Sets or gets application manufacturer information field for package's Add/Remove programs listing.

Syntax:

```
public string ManufacturerName { get; set; }
```

Property value:

Type: System.String

Application's manufacturer name.

Note:

In App-V Sequencer generated MSI packages this property is always set to "Publisher". It is recommended to change the manufacturer name to actual manufacturer of application inside the App-V package as this information is used by many software inventory tools to report on installed programs.

MsiPackageCode

Sets or gets MSI package code for the wrapper MSI.

Syntax:

```
public Guid MsiPackageCode { get; set; }
```

Property value:

Type: System.Guid

MSI package identification code.

Note:

If *preserveCustomCodes* –parameter is set to true in [Save\(\)](#) extension method for MSI packages, Lib-V will use code defined in this property instead of generating an automatic new one for the resulting MSI file. By default App-V package's MSI file will have new package code for each updated version.

MsiProductCode

Sets or gets MSI product code for the wrapper MSI.

Syntax:

```
public Guid MsiProductCode { get; set; }
```

Property value:

Type: System.Guid

MSI product identification code.

Note:

If *preserveCustomCodes* –parameter is set to true in [Save\(\)](#) extension method for MSI packages, Lib-V will use code defined in this property instead of generating an automatic new one for the resulting MSI file. By default App-V package's MSI file will have new product code for each updated version.

MsiUpgradeCode

Sets or gets MSI upgrade code for the wrapper MSI.

Syntax:

```
public Guid MsiUpgradeCode { get; set; }
```

Property value:

Type: System.Guid

MSI upgrade identification code.

Note:

If *preserveCustomCodes* –parameter is set to true in [Save\(\)](#) extension method for MSI packages, Lib-V will use code defined in this property instead of setting upgrade code to match SFT file's package ID.

OverrideUrl

Sets or gets overridden SFT path to be used by App-V Client instead of streaming information read from OSD file, if MSI is deployed in streaming mode.

Syntax:

```
public string OverrideUrl { get; set; }
```

Property value:

Type: System.String

Overridden streaming path. By default this property is set to **null**, as override URL is usually set by the calling party for MSI. If this property is set to **null** on existing MSI having OVERRIDEURL present, removes the property from MSI file upon serialization.

PackageId

Sets or gets unique identifier for the SFT file used by App-V package.

Syntax:

```
public Guid PackageId { get; set; }
```

Property value:

Type: System.Guid

SFT file's unique identifier.

Important note:

Packageld property **must** always be updated to reflect actual unique identifier of SFT file used by App-V package represented by the wrapper MSI.

ProductName

Sets or gets the name of the App-V package.

Syntax:

```
public string ProductName { get; set; }
```

Property value:

Type: System.String

Product name.

Note:

App-V Sequencer sets MSI's product name information the same as package name defined in OSD file.

ProductVersion

Sets or gets the version of the App-V package.

Syntax:

```
public string ProductVersion { get; set; }
```

Property value:

Type: System.String

Product version information.

Note:

App-V Sequencer sets MSI's product version information the same as package's SFT's internal version number.

SftFilename

Sets or gets physical filename of the SFT file.

Syntax:

```
public string SftFilename { get; set; }
```

Property value:

Type: System.String

Name of the SFT package file.

Important note:

[SftFilename](#) and [ManifestFilename](#) properties **must** always be updated to reflect actual names of the respective files, otherwise MSI installation will end in error.

SftPath

Sets or gets overridden SFT path to be used by App-V Client, if MSI is deployed in standalone mode.

Syntax:

```
public string SftPath { get; set; }
```

Property value:

Type: System.String

Path to a SFT file which will be imported to the App-V Client's cache. By default this property is set to **null**, as SFT path is usually set by the calling party for MSI. If this property is set to **null** on existing MSI having SFTPATH present, removes the property from MSI file upon serialization.

NameOsdFileCollection class

Represents collection of [OsdFile](#) objects and associated file system names for those OSD files when on disk. This class is used by the [ManifestFile](#) and [SprjFile](#) classes to pass group of OSD files to them along with the naming information.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class NameOsdFileCollection : NameObjectCollectionBase
```

Constructors

Overload List:

Initializes an empty collection.

```
public NameOsdFileCollection()
```

Initializes a collection from existing dictionary of [OsdFile](#) and name pairs with [OsdFile](#) as a key.

```
public NameOsdFileCollection (  
    IEnumerable<KeyValuePair<OsdFile,string>> valuePairs  
)
```

Initializes a collection from existing dictionary of [OsdFile](#) and name pairs with [OsdFile](#) as a key, and with optional read-only status flag.

```
public NameOsdFileCollection (  
    IEnumerable<KeyValuePair<OsdFile,string>> valuePairs,  
    bool readOnly  
)
```

Initializes a collection from existing dictionary of name and [OsdFile](#) pairs with name as a key.

```
public NameOsdFileCollection (  
    IEnumerable<KeyValuePair<string, OsdFile>> valuePairs  
)
```

Initializes a collection from existing dictionary of name and [OsdFile](#) pairs with name as a key, and with optional read-only status flag.

```
public NameOsdFileCollection (  
    IEnumerable<KeyValuePair<string, OsdFile>> valuePairs,  
    bool readOnly  
)
```

Parameters:

dictionary

Type: System.Collections.Generic.IEnumerable<TKey,TValue>

Key/value pairs of OSD filename and [OsdFile](#) objects.

readOnly

Type: System.Boolean

Flag indicating if collection is initialized as read-only.

Methods

Add

Adds new filename and associated OSD object pair to the collection.

Syntax:

```
public void Add (  
    string key,  
    OsdFile value  
)
```

Parameters:

key

Type: System.String

Physical filename for the OSD file.

value

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

OSD object to add.

Clear

Empties the collection.

Syntax:

```
public void Clear()
```

Remove

Removes specified filename and associated OSD object from the collection.

Syntax:

```
public void Remove (  
    string key  
)
```

Parameters:

key

Type: System.String

Name of the OSD file to remove.

RemoveAt

Removes OSD filename and object pair item from specified location in the collection.

Syntax:

```
public void RemoveAt (  
    int index  
)
```

Parameters:

key

Type: System.Int32

Position of the item to remove.

Properties

AllKeys

Gets all OSD filenames (keys) in the collection.

Syntax:

```
public string[] AllKeys { get; set; }
```

Property value:

Type: System.String[]

Array of OSD filenames.

AllValues

Gets all OSD objects (values) in the collection.

Syntax:

```
public OsdFile[] AllValues { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdFile\[\]](#)

Array of OSD objects.

OsdCodebaseElement class

Represents CODEBASE element in the OSD file.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdCodebaseElement : OsdElementBase
```

Constructor

Initializes a new CODEBASE element using specified attributes.

```
public OsdCodebaseElement (  
    string streamingUrl,  
    Guid packageGuid,  
    string processFilename,  
    string processParameters,  
    string sysguardFile,  
    Int64 size  
)
```

Parameters:

streamingUrl

Type: System.String

Streaming URI to a SFT file. OsdCodebaseElement will parse and store this information into [OsdCodebaseHref](#) –object format, allowing easier access to different parts of full URI.

packageGuid

Type: System.Guid

Unique identifier for the SFT file referenced by this OSD file.

processFilename

Type: System.String

Executable or file to be launched by App-V Client when OSD file is executed. If referencing to files inside the package's internal directory, drive letter can be omitted (e.g. **q:\foo.001\app.exe** -> **foo.001\app.exe**) from the path.

processParameters

Type: System.String

Parameters passed to an executable specified in the *processFilename* -parameter.

sysguardFile

Type: System.String

Path to a package's virtual environment configuration file, osguard.cp file. This file is always stored under the root directory of the package and can be referenced without any drive letters (e.g. **foo.001\osguard.cp**).

size

Type: System.Int64

Estimated size of the SFT package, in bytes.

Note:

It is responsibility of the caller to maintain each CODEBASE's attributes in accordance with respect to paths inside SFT's internal directory structure and other properties, such as package unique identifier and path an osguard.cp -file.

Properties

Filename

Gets or sets path to an executable or file to be launched by App-V Client when OSD file is executed.

Syntax:

```
public string Filename { get; set; }
```

Property value:

Type: System.String

Program to execute.

Href

Gets or sets streaming URI object representing the source from which the SFT file is loaded by the App-V Client in the streaming scenario.

Syntax:

```
public OsdCodebaseHref Href { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdCodebaseHref](#)

Streaming URI.

PackageGuid

Gets or sets unique identifier into which the SFT referenced has to match.

Syntax:

```
public Guid PackageGuid { get; set; }
```

Property value:

Type: System.Guid

Package's unique identifier.

Parameters

Gets or sets application start up parameters passed to a program launched by App-V Client when OSD file is executed.

Syntax:

```
public string Parameters { get; set; }
```

Property value:

Type: System.String

Parameters to pass.

Size

Gets or sets estimated size of the package's SFT file, in bytes.

Syntax:

```
public Int64 Size { get; set; }
```

Property value:

Type: System.Int64

Size of the package.

SysguardFile

Gets or sets path to a package's virtual environment configuration file.

Syntax:

```
public string SysguardFile { get; set; }
```

Property value:

Type: System.String

Path to osguard.cp file inside the SFT file's internal directory structure.

OsdCodebaseHref class

Represents HREF attribute in CODEBASE element in the OSD file.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdCodebaseHref
```

Constructor

Initializes a new HREF attribute using specified attributes.

```
public OsdCodebaseHref (  
    string hrefString  
)
```

Parameters:

hrefString

Type: System.String

Streaming URI to a SFT file. Constructor will parse textual representation of the URI and store this information in parts, allowing easier access to different parts of full URI through properties.

Methods

ToString

Returns streaming URI in textual representation (as in found inside HREF attribute in OSD file).

Syntax:

```
public override string ToString()
```

Return value:

Type: System.String

Streaming URI in textual format.

Properties

Port

Gets or sets port number for the specified streaming [protocol](#) (e.g. 554 for RTSP). This property is not used for FILE –protocol.

Syntax:

```
public string Port { get; set; }
```

Property value:

Type: System.String

Port number.

Protocol

Gets or sets protocol identifier. Protocols supported by App-V (as of 4.6) are RTSP, RTSPS, HTTP, HTTPS and FILE.

Syntax:

```
public string Protocol { get; set; }
```

Property value:

Type: System.String

Protocol identifier.

Server

Gets or sets server address from where the SFT is streamed from.

Syntax:

```
public string Server { get; set; }
```

Property value:

Type: System.String

Server name.

Note:

Server –property is not used for FILE protocol URIs, which uses [Subdirectory](#) –property for the actual – full - file path for consistency reasons since local file-system paths do not contain server –part while UNC paths would.

SftName

Gets or sets SFT filename App-V Client will stream in.

Syntax:

```
public string SftName { get; set; }
```

Property value:

Type: System.String

SFT's filename.

Note:

It is caller's responsibility to maintain SFT's name up-to-date with regards to what the SFT name will be when encoded by Lib-V. In RTSP and RTSPS scenarios and 4.0+ App-V Servers filename is not reflective of what will be returned to the App-V Client but rather the selection is done based on GUIDs.

Subdirectory

Gets or sets subdirectory on a server from where the SFT file will be streamed in by App-V Client.

Syntax:

```
public string Subdirectory { get; set; }
```

Property value:

Type: System.String

Subdirectory path.

Note:

In RTSP and RTSPS scenarios this path is subdirectory in the App-V server's content directory. In HTTP or HTTPS scenario this will be relative to the virtual directory published from the HTTP server. In FILE scenario, this is the full path to directory where SFT file is stored, either in UNC or local file-system format.

OsdDscCodebaseElement class

Represents extended CODEBASE element in the OSD file as used by Dynamic Suiting feature. This CODEBASE element is always listed under [DEPENDENCIES](#) element.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdDscCodebaseElement : OsdCodebaseElement
```

Constructors

Overload List:

Initializes a new CODEBASE element using specified attributes, omitting attributes not required or meaningful in Dynamic Suiting version of the CODEBASE -element.

```
public OsdDscCodebaseElement (  
    string streamingUrl,  
    Guid packageGuid,  
    string sysguardFile,  
    Int64 size,  
    bool mandatoryForDynamicSuiting  
)
```

Initializes a new CODEBASE element using specified attributes, omitting attributes not required or meaningful in Dynamic Suiting version of the CODEBASE -element.

```
public OsdDscCodebaseElement (  
    string streamingUrl,  
    Guid packageGuid,  
    string sysguardFile,  
    bool mandatoryForDynamicSuiting  
)
```

Parameters:

streamingUrl

Type: System.String

Streaming URI to a linked SFT file. OsdCodebaseElement will parse and store this information into [OsdCodebaseHref](#) –object format, allowing easier access to different parts of full URI.

packageGuid

Type: System.Guid

Unique identifier for the linked SFT file referenced by this OSD file.

sysguardFile

Type: System.String

Path to a linked package's virtual environment configuration file, osguard.cp file. This file is always stored under the root directory of the linked package and can be referenced without any drive letters (e.g. **aux.001\osguard.cp**).

size

Type: System.Int64

Estimated size of the linked SFT package, in bytes.

mandatoryForDynamicSuiting

Type: System.Boolean

Flag to specify whether or not linked package has to be able to be loaded by App-V Client or not. If set to **false**, App-V Client will skip unloadable linked packages. If set to **true**, whole launch process of primary application will be aborted.

Properties

IsMandatory

Gets or sets flag controlling requirement of being able to link to a package.

Syntax:

```
public bool IsMandatory { get; set; }
```

Property value:

Type: System.Boolean

Set to **true** to require App-V Client to load linked package and abort launch if it cannot, **false** if linked package is optional.

OsdDdeExecElement class

Represents file-type association DDE information in the OSD file.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdDdeExecElement : OsdElementBase
```

Constructor

Initializes new DDE information using specified attributes.

```
public OsdDdeExecElement (  
    string application,  
    string topic,  
    string ddeCommand,  
    string ifExec  
)
```

Parameters:

application

Type: System.String

Application to invoke.

topic

Type: System.String

Topic information.

ddeCommand

Type: System.String

DDE command.

ifExec

Type: System.String

IFEXEC information.

Properties

Application

Gets or sets application name to invoke in DDE call.

Syntax:

```
public string Application { get; set; }
```

Property value:

Type: System.String

Name of the application.

Command

Gets or sets command to send in DDE call.

Syntax:

```
public string Command { get; set; }
```

Property value:

Type: System.String

Command string.

IfExec

Gets or sets IFEXEC information.

Syntax:

```
public string IfExec { get; set; }
```

Property value:

Type: System.String

IFEXEC string.

Topic

Gets or sets DDE conversation topic.

Syntax:

```
public string Topic { get; set; }
```

Property value:

Type: System.String

Topic text.

OsdDependenciesElement class

Represents DEPENDENCIES element in the OSD file.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdDependenciesElement : OsdElementBase
```

Constructor

Initializes new DEPENDENCIES element with empty list of linked CODEBASE elements.

```
public OsdDependenciesElement ()
```

Properties

DynamicSuiteCompositionCodebases

Gets list of linked packages.

Syntax:

```
public IList<OsdDscCodebaseElement> DynamicSuiteCompositionCodebases {  
    get; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of linked packages.

OsdDependencyElement class

Represents DEPENDENCY element in the OSD file.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdDependencyElement : OsdElementBase
```

Constructor

Initializes new DEPENDENCY element with empty list of script elements.

```
public OsdDependencyElement ()
```

Properties

ClientVersion

Gets or sets minimum supported App-V Client version for this OSD file.

Syntax:

```
public string ClientVersion { get; set; }
```

Property value:

Type: System.String

Version number of the App-V Client that has to be met at minimum so that the application represented by the OSD is shown on it.

Scripts

Gets or sets list of OSD scripts associated with the package.

Syntax:

```
public IList<OsdScriptElement> Scripts { get; set; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of OSD scripts.

OsdElementBase class

Represents an abstract base class from which all other elements in the Lib-V OSD namespace inherit from.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public abstract class OsdElementBase
```

Properties

UnrecognizedAttributes

Gets a list of XML attributes that were detected during inherited element's parsing (deserialization) but could not be mapped to known OSD attributes. This list is maintained so that custom attributes or attributes unknown to Lib-V can be maintained in the OSD file upon serialization back to disk.

Syntax:

```
public IList<XmlAttribute> UnrecognizedAttributes { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of unknown XML attributes.

UnrecognizedNodes

Gets a list of XML child nodes that were detected during inherited element's parsing (deserialization) but could not be mapped to known OSD elements. This list is maintained so that custom elements or elements unknown to Lib-V can be maintained in the OSD file upon serialization back to disk.

Syntax:

```
public IList<XmlNode> UnrecognizedNodes { get; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of unknown XML nodes.

XmlComments

Gets or sets a list of XML comment nodes that were detected during inherited element's parsing (deserialization). This list is maintained so that custom XML comments otherwise unknown to Lib-V can be maintained and added to in the OSD file upon serialization back to disk.

Syntax:

```
public IList<string> XmlComments { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of XML comments.

Note:

Please be aware, that App-V Sequencer does **not** keep XML comments stored in the OSD files if package is resaved with it!

OsdEnvironmentElement class

Represents virtualized environment variable element in the OSD file.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdEnvironmentElement : OsdElementBase
```

Constructor

Initializes new virtualized environment variable.

```
public OsdEnvironmentElement (  
    string variableName,  
    string variableValue  
)
```

Parameters:

variableName

Type: System.String

Name of the environment variable to virtualize.

variableValue

Type: System.String

Contents of the virtualized environment variable.

Properties

Name

Gets or sets the name of the virtualized environment variable.

Syntax:

```
public string Name { get; set; }
```

Property value:

Type: System.String

Variable's name.

Value

Gets or sets contents of the virtualized environment variable.

Syntax:

```
public string Value { get; set; }
```

Property value:

Type: System.String

Variable's value.

OsdFile class

Represents App-V OSD file in an object format. OSD contains information about one individual application published from the App-V package or application launched into package's context.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdFile : MetadataFileBase
```

Constructors

Overload List:

Initializes new OSD file using specified application name and version information, with no default implementation block.

```
public OsdFile(  
    string applicationName,  
    string applicationVersion  
)
```

Initializes new OSD file using specified application name and version information, and with default implementation block having information derived from supplied SFT object.

```
public OsdFile(  
    string applicationName,  
    string applicationVersion,  
    SftPackage sftPackage  
)
```

Initializes new OSD file using specified application name and version information, package name and with default implementation block having information derived from supplied SFT object.

```

public OsdFile(
    string applicationName,
    string applicationVersion,
    SftPackage sftPackage,
    string packageName
)

```

Initializes new OSD file using specified application name and version information and package name, with no default implementation block.

```

public OsdFile(
    string applicationName,
    string applicationVersion,
    string packageName
)

```

Parameters:

applicationName

Type: System.String

Name of the published application.

applicationVersion

Type: System.String

Version of the published application.

sftPackage

Type: [GridMetric.LibV.SftPackage](#)

SFT file object for which this OSD file will has relation to. SFT reference is used to populate default implementation block for the OSD.

packageName

Type: System.String

Name of the App-V package (PACKAGE NAME).

Methods

AddImplementation

Adds new implementation block to the OSD file.

Overload List:

Adds implementation block with specified manual GUID and root directory name for CODEBASE element initialization.

```
public void AddImplementation(  
    Guid sftPackageId,  
    string packageRootDirectoryName  
)
```

Adds implementation block with SFT file reference from which initialization information is read for CODEBASE element.

```
public void AddImplementation(  
    SftPackage sftPackage  
)
```

Parameters:

sftPackageId

Type: System.Guid

Unique package identifier of the SFT package this OSD file links to.

packageRootDirectoryName

Type: System.String

Name of the package's internal filesystem's root directory. Lib-V uses this information to set osguard.cp file reference in implementation block's CODEBASE element automatically.

sftPackageId

Type: [GridMetric.LibV.SftPackage](#)

Reference to SFT package object.

Note:

CODEBASE element in created implementation block has default streaming URI of "RTSP://%SFT_SOFTGRIDSERVER%:554/softapp.sft" which must be adjusted by the caller to match actual streaming source (if used).

GetCustomOsdElement

Returns custom (i.e. one not in standard OSD schema) XML element associated with the SOFTPKG node.

Syntax:

```
public XmlElement GetCustomOsdElement (  
    string elementName  
)
```

Parameters:

elementName

Type: System.String

Local name for custom XML element to return.

Return value:

Type: System.Xml.XmlElement

XML element.

Note:

This method returns custom element stored in the root of the OSD file (i.e. as child for SOFTPKG element), set either previously through [SetCustomOsdElement](#) -method or originating from unrecognized elements found during parsing.

This method can be used for retrieving custom data stored in the OSD in form of caller-specific XML element.

RemoveCustomOsdElement

Removes custom (i.e. one not in standard OSD schema) XML element associated with the SOFTPKG node.

Syntax:

```
public void RemoveCustomOsdElement (  
    string elementName  
)
```

Parameters:

elementName

Type: System.String

Local name for custom XML element to remove. If element by the specified name is not found, method will do nothing.

SetCustomOsdElement

Adds or updates custom (i.e. one not in standard OSD schema) XML element associated with the SOFTPKG node.

Overload List:

Adds custom element with name, inner text and attributes as string values.

```
public void SetCustomOsdElement (  
    string elementName,  
    string innerText,  
    IDictionary<string, string> attributes  
)
```

Adds custom element directly.

```
public void SetCustomOsdElement (  
    XmlElement customElement  
)
```

Parameters:

elementName

Type: System.String

Local name for the element. If element with the specified name is already in the custom element collection, method will update it with specified inner text and attributes.

innerText

Type: System.String

Inner text to set for element.

attributes

Type: System.Collections.Generic.IDictionary<TKey,TValue>

Collection of attribute names and associated inner texts. If passed as **null**, no attributes are added.

customElement

Type: System.Xml.XmlElement

Element to add as custom element. Will be imported into local XML namespace of OSD file.

Properties

ApplicationName

Gets or sets the name of the application represented by the OSD file.

Syntax:

```
public string ApplicationName { get; set; }
```

Property value:

Type: System.String

Application name. This field has 64 character limit in the OSD schema.

Important note:

Information in [ApplicationName](#) and [ApplicationVersion](#) form the full name for published application which App-V Client uses to identify individual applications added to it. Each application has to have unique name + version combination if they are to co-exist on the same client.

ApplicationVersion

Gets or sets the version of the application represented by the OSD file.

Syntax:

```
public string ApplicationVersion { get; set; }
```

Property value:

Type: System.String

Application version. This field has 16 character limit in the OSD schema.

Important note:

Information in [ApplicationName](#) and [ApplicationVersion](#) form the full name for published application which App-V Client uses to identify individual applications added to it. Each application has to have unique name + version combination if they are to co-exist on the same client.

Comments

Gets or sets the comments for the application (ABSTRACT element).

Syntax:

```
public string Comments { get; set; }
```

Property value:

Type: System.String

Free-form comment text.

DefaultImplementation

Gets the default (i.e. first if multiple) implementation block for the application.

Syntax:

```
public OsdImplementationElement DefaultImplementation { get; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdImplementationElement](#)

Default implementation block.

Note:

While OSD technically allows multiple IMPLEMENTATION blocks for it, this capability is hardly ever used by anybody and thus such situation should be avoided even if Lib-V supports adding several blocks per one application. DefaultImplementation –property will return the first implementation element in the [Implementations](#) –list.

Dependency

Gets or sets dependency element's information for this OSD.

Syntax:

```
public OsdDependencyElement Dependency { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdDependencyElement](#)

Dependency element.

Encoding

Gets or sets character encoding used for the (de)serialized OSD file.

Syntax:

```
public Encoding Encoding { get; set; }
```

Property value:

Type: System.Text.Encoding

Encoding used for the OSD file.

FileAssociations

Gets or sets file associations linked to application represented by this OSD.

Syntax:

```
public OsdFileAssociationsElement FileAssociations { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdFileAssociationsElement](#)

Application's file associations.

Implementations

Gets list of implementation blocks for this OSD file.

Syntax:

```
public IList<OsdImplementationElement> Implementations { get; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of implementation blocks associated with this application.

Note:

To add new implementation, use [AddImplementation](#) –method. All OSD files has have at least one implementation, and preferably only one.

OsdGuid

Gets or sets unique identifier for this OSD file. App-V itself does not track OSD's own GUID in any part of the system, but it is recommended to keep them to allow strong identification of each published application in 3rd party applications.

Syntax:

```
public Guid? OsdGuid { get; set; }
```

Property value:

Type: System.Nullable<T>

OSD file's unique identifier. Does not have relationship to SFT package's unique identifier.

PackageName

Gets or sets the name for the overall package which the OSD is part of. This name must be identical on each OSD file belonging to the same package.

Syntax:

```
public string PackageName { get; set; }
```

Property value:

Type: System.String

Package's name.

Shortcuts

Gets or sets list of (Windows) application shortcuts defined for the application represented by this OSD.

Syntax:

```
public IList<OsdShortcutElement> Shortcuts { get; set; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of application shortcuts associated with this application. App-V Client will use this information to create shortcuts on a client system.

SuiteName

Gets or sets the suite name for the overall package which the OSD is part of. Suite naming has been obsolete since SoftGrid 4.0 and [PackageName](#) -property should be used instead in all 4.0+ packages.

Syntax:

```
public string SuiteName { get; set; }
```

Property value:

Type: System.String

Suite name.

Title

Gets or sets the title for the application. This text is informational and not used directly by App-V.

Syntax:

```
public string Title { get; set; }
```

Property value:

Type: System.String

Application title.

XmlComments

Gets or sets a list of XML comment nodes in OSD's SOFTPKG element. These comments are not used or processed by App-V itself. For element –specific XML comments, please refer to [OsdElementBase](#)'s [XmlComment](#) –property.

Syntax:

```
public IList<string> XmlComments { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of XML comments.

Note:

Please be aware, that App-V Sequencer does **not** keep XML comments stored in the OSD files if package is resaved with it!

OsdFileAssociationsElement class

Represents MGMT_FILEASSOCIATIONS element in the OSD file, recording information about file associations related to a published application.

Each individual file association has one programmatic identifier (ProgIDs) that Windows uses to link association to particular application (virtual in this case by having App-V Client register the association) and one and more file extensions that link to that ProgIDs. Commonly different extensions all have their own programmatic identifiers, but it is not necessarily always so.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdFileAssociationsElement : OsdElementBase
```

Constructor

Initializes new file association element with empty list of file extensions and ProgIDs.

```
public OsdFileAssociationsElement()
```

Properties

FileExtensions

Gets or sets list of file extensions for file associations.

Syntax:

```
public IList<OsdFileExtensionElement> FileExtensions { get; set; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of file extensions associated with the application represented by the OSD.

ProgIdList

Gets or sets list of programmatic identifiers for file associations.

Syntax:

```
public IList<OsdProgIdElement> ProgIdList { get; set; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of ProgID's associated with the application represented by the OSD.

OsdFileExtensionElement class

Represents FILEEXTENSION element in the OSD file, recording information about file extension related to file association.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdFileExtensionElement : OsdElementBase
```

Constructor

Initializes new file extensions using specified attributes.

```
public OsdFileExtensionsElement(  
    string extension,  
    OsdProgIdElement progId,  
    string perceivedType,  
    string contentType,  
    bool? overridden  
)
```

Parameters:

extension

Type: System.String

File's extension without leading dot.

progId

Type: [GridMetric.LibV.Metadata.Osd.OsdProgIdElement](#)

Programmatic identifier [object] this file extensions links to. More information about programmatic identifiers can be found from MSDN documentation at

<http://msdn.microsoft.com/en-us/library/cc144152.aspx>.

perceivedType

Type: System.String

Perceived type for the extension. More information about perceived type can be found from MSDN documentation at <http://msdn.microsoft.com/en-us/library/cc144150.aspx>.

contentType

Type: System.String

Content (MIME) type for the extension. More information about perceived type can be found from MSDN documentation at <http://msdn.microsoft.com/en-us/library/cc144148.aspx>.

overridden

Type: System.Nullable<T>

Is content type overridden. This attribute is usually set to **false** for file extension registration in OSD files and is **null** for file extension elements in the [ManifestFileTypeElement](#)'s [Extension](#) -property.

Properties

ContentType

Gets or sets content type for file extension (e.g. **application/msword** or **text/richtext**).

Syntax:

```
public string ContentType { get; set; }
```

Property value:

Type: System.String

Content type for the extension.

Extension

Gets or sets file's extension by which the association is recognized in the system, without the leading dot (e.g. **docx** or **gif**).

Syntax:

```
public string Extension { get; set; }
```

Property value:

Type: System.String

File extension.

InShellNewMenu

Gets or sets flag indicating if document creation link is created on Windows Explorer's New –context menu.

Syntax:

```
public bool InShellNewMenu { get; set; }
```

Property value:

Type: System.Boolean

true if document type is shown in New menu, **false** otherwise.

Overridden

Gets or sets flag indicating if file association is overridden. This flag is not documented by Microsoft but is usually set to **false** for OSD files generated by Microsoft Sequencer.

Syntax:

```
public bool? Overridden { get; set; }
```

Property value:

Type: System.Nullable<T>

true if file extension is overridden, **false** otherwise. **null** if not set.

PerceivedType

Gets or sets perceived type for file extension, which is a Windows –specific categorization of what kind of document is in question (e.g. **Image** or **Document**).

Syntax:

```
public string PerceivedType { get; set; }
```

Property value:

Type: System.String

Perceived type for the file extension. Is usually set to empty.

ProgId

Gets programmatic identifier this extension links to. If linkage needs to be switched to completely new ProgID, extension has to be instantiated anew with reference to a new [OsdProgIdElement](#) object.

Syntax:

```
public OsdProgIdElement ProgId { get; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdProgIdElement](#)

Programmatic identifier for file extension.

OsdImplementationElement class

Represents IMPLEMENTATION element in the OSD file, recording information about actual implementation of the virtual package.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdImplementationElement : OsdElementBase
```

Constructors

Overload List:

Initializes new empty implementation block.

```
public OsdImplementationElement()
```

Initializes new implementation block with empty codebase and instantiates new process type element with specified VM attributes.

```
public OsdImplementationElement (  
    OsdVmType vmType,  
    OsdSubsystemType subsystemType  
)
```

Initializes new implementation block with codebase and instantiates new process type element with specified VM attributes.

```
public OsdImplementationElement (  
    OsdVmType vmType,  
    OsdSubsystemType subsystemType,  
    OsdCodebaseElement codebase  
)
```

Parameters:

vmType

Type: [GridMetric.LibV.Metadata.Structures.OsdVmType](#)

Process type (bitness).

subsystemType

Type: [GridMetric.LibV.Metadata.Structures.OsdSubsystemType](#)

Process' subsystem type.

codebase

Type: [GridMetric.LibV.Metadata.Osd.OsdCodebaseElement](#)

Codebase element to use for implementation.

Properties

Codebase

Gets or sets codebase for implementation block.

Syntax:

```
public OsdCodebaseElement Codebase { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdCodebaseElement](#)

Codebase to use.

SupportedOperatingSystems

Gets or sets bitmask for defining what operating system versions are supported for the OSD. This controls the visibility of the application on the client (if operating system is not listed, it will not be visible).

Syntax:

```
public OsdOperatingSystemTypes SupportedOperatingSystems { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdOperatingSystemTypes](#)

Flags enumeration of supported operating system. If value **All** is reserved when operating system visibility is not limited.

VirtualEnvironment

Gets or sets virtual environment definition element for the implementation.

Syntax:

```
public OsdVirtualenvElement VirtualEnvironment { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdVirtualenvElement](#)

Virtual environment.

VM

Gets or sets process type definition element for the implementation.

Syntax:

```
public OsdVmElement VM { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdVmElement](#)

Process type information.

WorkingDirectory

Gets or sets working (i.e. current) directory for the process being launched by executing the OSD.

Syntax:

```
public string WorkingDirectory { get; set; }
```

Property value:

Type: System.String

Working directory to set for a launched process.

OsdPoliciesElement class

Represents POLICIES element in the OSD file, recording information about special virtual environment policies in effect.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdPoliciesElement : OsdElementBase
```

Constructor

Initializes new policies element without any policies set.

```
public OsdFileAssociationsElement()
```

Properties

LocalInteractionAllowed

Gets or sets flag indicating if virtual application has out-of-normal COM instantiation ordering rules. Enabling this flag can occasionally fix interaction issues with virtual applications calling on functionality that is provided from locally present applications.

Syntax:

```
public bool? LocalInteractionAllowed { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

EnforceVirtualRegistryAclsOnModify

Gets or sets flag indicating if ACLs are enforced when VREG is modified. This feature is only available in App-V 4.6 SP1 HF8 or SP2 releases, or newer.

Syntax:

```
public bool? EnforceVirtualRegistryAclsOnModify { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

Sxs32Enabled

Gets or sets flag indicating if Windows presents file-system paths of loaded (virtual) DLLs in VFS form instead of their actual Q: -drive based path to a virtual application. Enabling this flag can in rare cases fix issue with application seeing files coming from “incorrect” path.

Syntax:

```
public bool? Sxs32Enabled { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

VirtualFileSystemDisabled

Gets or sets flag indicating if all VFS mappings in the package are disabled at App-V Client for the virtual application. Only useful as debugging or troubleshooting aid.

Syntax:

```
public bool? VirtualFileSystemDisabled { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

VirtualFileSystemPassthrough

Gets or sets flag indicating if VFS pass-through is enabled. This feature is only available in App-V 4.6 SP1 HF8 or SP2 releases, or newer.

Syntax:

```
public bool? VirtualFileSystemPassthrough { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

VirtualRegistryDisabled

Gets or sets flag indicating if all contents of virtual registry in the package are disabled at App-V Client for the virtual application. Only useful as debugging or troubleshooting aid.

Syntax:

```
public bool? VirtualRegistryDisabled { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

VirtualRegistryPassthrough

Gets or sets flag indicating if VREG pass-through is enabled. This feature is only available in App-V 4.6 SP1 HF8 or SP2 releases, or newer.

Syntax:

```
public bool? VirtualRegistryPassthrough { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

VirtualServicesDisabled

Gets or sets flag indicating if all virtualized services in the package are disabled at App-V Client for the virtual application. Only useful as debugging or troubleshooting aid.

Syntax:

```
public bool? VirtualServicesDisabled { get; set; }
```

Property value:

Type: System.Nullable<T>

true if policy is enabled, **false** otherwise. **null** if not set.

OsdProgIdElement class

Represents PROGID element in the OSD file, recording information about programmatic identifier for file extension related to file association.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdProgIdElement : OsdElementBase
```

Constructor

Initializes new programmatic identifier (ProgID) using specified attributes.

```
public OsdProgIdElement (  
    string id,  
    string friendlyName,  
    UInt32? editFlags,  
    bool? overridden  
)
```

Parameters:

id

Type: System.String

Identifier for the ProgID element. Windows names registry subkey for ProgID based on the id and it should be reflective of the application and/or the extension (e.g.

App.jpeg).

friendlyName

Type: System.String

User-friendly name of the association. This is the description text that users would see in Windows Explorer listing of files.

editFlags

Type: System.Nullable<T>

Edit flags for ProgID. More information about structure represented by EditFlags can be found from MSDN documentation at <http://msdn.microsoft.com/en-us/library/bb762506.aspx>. For convenience, App-V supported aspects of Edit flags are provided in `OsdProgIdElement` class through [AlwaysShowExtension](#) and [ConfirmOpenAfterDownload](#) –properties.

overridden

Type: `System.Nullable<T>`

Is ProgId overridden. This attribute is usually set to **false** in OSD files and is **null** for progid elements in the [ManifestFileTypeElement](#).

Note:

More information about programmatic identifiers and attributes related to it can be found from MSDN documentation at <http://msdn.microsoft.com/en-us/library/cc144152.aspx>.

Properties

AlwaysShowExtension

Gets or sets flag indicating if Windows Explorer will always show file's extension regardless of Explorer options set by the user.

Syntax:

```
public bool AlwaysShowExtension { get; set; }
```

Property value:

Type: `System.Boolean`

true if extension is shown, **false** otherwise.

ConfirmOpenAfterDownload

Gets or sets flag indicating if Windows Internet Explorer will always confirm open of the document after it has been downloaded.

Syntax:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

```
public bool ConfirmOpenAfterDownload { get; set; }
```

Property value:

Type: System.Boolean

true if confirmation cannot be disabled, **false** otherwise.

EditFlags

Gets or sets edit flags for programmatic identifier. For convenience, App-V supported aspects of edit flags are provided directly through [AlwaysShowExtension](#) and [ConfirmOpenAfterDownload](#) –properties.

Syntax:

```
public UInt32? EditFlags { get; set; }
```

Property value:

Type: System.Nullable<T>

Bitmask for edit flags. **null** if not set.

FriendlyName

Gets or sets friendly name for the association shown in the Windows Explorer file listing.

Syntax:

```
public string FriendlyName { get; set; }
```

Property value:

Type: System.String

Friendly name.

Icon

Gets or sets path to a document type icon for association.

Syntax:

```
public string Icon { get; set; }
```

Property value:

Type: System.String

Path to icon (.ICO) file used for document type.

Note:

In OSD files, icon files in the package directory are referenced using special variable, %SFT_MIME_SOURCE%, marking the root of the directory holding package files (e.g. %SFT_MIME_SOURCE%/AbiWord Icons/AbiWord Document.ico).

Id

Gets or sets unique identifier string for the ProgID.

Syntax:

```
public string Id { get; set; }
```

Property value:

Type: System.String

Identifier string (e.g. **SoftGrid.Sft**).

Overridden

Gets or sets flag indicating if programmatic id is overridden. This flag is not documented by Microsoft but is usually set to **false** for OSD files generated by Microsoft Sequencer.

Syntax:

```
public bool? Overridden { get; set; }
```

Property value:

Type: System.Nullable<T>

true if ProgID is overridden, **false** otherwise. **null** if not set.

ShellCommandList

Gets or sets list of shell commands (i.e. action verbs shown in Explorer's context menu for the file) associated with the file association.

Syntax:

```
public OsdShellCommandListElement ShellCommandList { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdShellCommandListElement](#)

List holding verbs associated with the file association.

OsdRegistryElement class

Represents REGISTRY element in the OSD file, recording information about OSD level virtual registry overrides. These virtual registry entries always take precedence over ones inside the SFT file and ones that might be cached at the App-V Client into PKG files.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdRegistryElement : OsdElementBase
```

Constructor

Initializes new virtual registry override with empty list of virtual registry keys.

```
public OsdRegistryElement()
```

Properties

RegistryKeys

Gets list of virtual registry keys.

Syntax:

```
public IList<OsdRegkeyElement> RegistryKeys { get; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of virtual registry keys that override entries in the package's main virtual registry.

OsdRegkeyElement class

Represents REGKEY element in the OSD file, recording information about OSD level virtual registry key override.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdRegkeyElement : OsdElementBase
```

Constructor

Initializes new virtual registry key.

```
public OsdRegkeylement (  
    string registryHive,  
    string registryKey,  
    bool? noRedir  
)
```

Parameters:

registryHive

Type: System.String

Registry hive for the key (either **HKLM** for system registry or **HKCU** for user's registry).

registryKey

Type: System.String

Key path inside the specified hive (e.g. **SOFTWARE\Microsoft\Office\Office11**).

noRedir

Type: System.Nullable<T>

Is WOW64 redirection disabled for the specified registry key. Setting to **true** prevents Windows from redirecting 32-bit keys in 64-bit system.

Properties

Hive

Gets or sets registry hive for virtual registry key.

Syntax:

```
public string Hive { get; set; }
```

Property value:

Type: System.String

HKCU for current user's registry hive, **HKLM** for system registry hive.

Key

Gets or sets registry key path.

Syntax:

```
public string Key { get; set; }
```

Property value:

Type: System.String

Full path to key being defined.

NoRedirection

Gets or sets flag indicating if WOW64 redirection is disabled on 64-bit systems.

Syntax:

```
public bool? NoRedirection { get; set; }
```

Property value:

Type: System.Nullable<T>

true if redirection is disabled, **false** if not. **null** if left undefined.

RegistryValues

Gets list of virtual registry values defined for this key.

Syntax:

```
public IList<OsdRegvalueElement> RegistryValues { get; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of virtual registry values defined for the containing key.

OsdRegvalueElement class

Represents REGVALUE element in the OSD file, recording information about OSD level virtual registry value override.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdRegvalueElement : OsdElementBase
```

Constructor

Initializes new virtual registry value.

```
public OsdRegkeylement (  
    OsdRegistryValueType valueType,  
    string valueName,  
    string valueData  
)
```

Parameters:

valueType

Type: [GridMetric.LibV.Metadata.Structures.OsdRegistryValueType](#)

Data type for the registry value.

valueName

Type: System.String

Name of the value.

valueData

Type: System.String

Data (contents) of the value. For binary values, series of numbers in hexadecimal format are given, separated with comma (e.g. **1F,44,78,DF**).

Properties

Data

Gets or sets registry value's data.

Syntax:

```
public string Data { get; set; }
```

Property value:

Type: System.String

Data for the value.

Name

Gets or sets registry value's name.

Syntax:

```
public string Name { get; set; }
```

Property value:

Type: System.String

Name for the value.

ValueType

Gets or sets data type of the value.

Syntax:

```
public OsdRegistryValueType ValueType { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdRegistryValueType](#)

Data type for the value.

OsdScriptElement class

Represents SCRIPT element in the OSD file, recording information about OSD application script.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdScriptElement : OsdElementBase
```

Constructor

Initializes new application script.

```
public OsdRegkeylement (
    OsdScriptType typeOfScript,
    OsdScriptTiming scriptTiming,
    OsdScriptEvent scriptEvent,
    bool runProtected,
    bool? runExtern
)
```

Parameters:

typeOfScript

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptType](#)

Type of the script defined.

scriptTiming

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptTiming](#)

Timing of the script execution.

scriptEvent

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptEvent](#)

Event of the script execution.

runProtected

Type: System.Boolean

Flag indicating if script is run inside virtual environment (**true**) or outside it (**false**).

runExtern

Type: System.Nullable<T>

Flag indicating if script is run with 64-bit command-interpreter in 64-bit system.

Properties

Condition

Gets or sets script execution result conditionals. When App-V Client finishes running OSD script, its return results can be evaluated again ERRORLEVEL value so that further processing is either continued or stopped.

Syntax:

```
public OsdScriptConditional Condition { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptConditional](#)

Script execution result conditional evaluation or **null** if script results are not conditionally evaluated.

Note:

Use [ResultValue](#) -property to set ERRORLEVEL value that is evaluated against.

Content

Gets or sets script's content.

Syntax:

```
public string Content { get; set; }
```

Property value:

Type: System.String

Content. For [Href](#) –type script, this refers to path to an application or other external file to launch, for [Scriptbody](#) –script this holds the complete command-interpreter script.

Event

Gets or sets script launch event.

Syntax:

```
public OsdScriptEvent Event { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptEvent](#)

Event with which the script is started.

Extern

Gets or sets flag indicating if embedded script is launched using 64-bit command interpreter in 64-bit systems or with 32-bit interpreter (default if not set).

Syntax:

```
public bool? Extern { get; set; }
```

Property value:

Type: System.Nullable<T>

true if script is executed with 64-bit command interpreter, **false** if with 32-bit version. **null** if not defined which defaults to 32-bit execution.

Note:

Only applies to 64-bit systems.

Protect

Gets or sets flag indicating if script or external application is run inside the package's virtual environment or outside it.

Syntax:

```
public bool Protect { get; set; }
```

Property value:

Type: System.Boolean

true if script is launched inside the package's VE, **false** if script is launched outside (with access to physical registry and other parts of the system).

ResultValue

Gets or sets ERRORLEVEL result value that gets compared if [Condition](#) –property is set and is not set to [None](#).

Syntax:

```
public Int32 ResultValue { get; set; }
```

Property value:

Type: System.Int32

Error-level value.

Timeout

Gets or sets value (in seconds) that script will be executed. Value of 0 means indefinite execution and script will not time-out. If [Wait](#) –property is set **false**, timeout value is ignored.

Syntax:

```
public Int32? Timeout { get; set; }
```

Property value:

Type: System.Nullable<T>

Timeout in seconds or **null** if timeout is not defined.

Timing

Gets or sets script execution timing.

Syntax:

```
public OsdScriptTiming Timing { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptTiming](#)

Timing of the event with which the script is started.

Type

Gets or sets script type.

Syntax:

```
public OsdScriptType Type { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdScriptType](#)

Type of the script.

Wait

Gets or sets flag indicating if script is run synchronously or asynchronously. If script is executed synchronously, App-V Client will wait for scripts completion before continuing. Specifying timeout value with [Timeout](#) –property implicitly runs scripts synchronously and in that case Wait property does not need to be set.

Syntax:

```
public bool? Wait { get; set; }
```

Property value:

Type: System.Nullable<T>

true if App-V Client waits for script completion, **false** if script is only started and then left running independently of chain of operations. **null** if left undefined.

Note:

Scripts cannot be run asynchronously when [Event](#) is set to [Shutdown](#).

OsdSerializer class

Serializes and deserializes App-V OSD XML file between file and object format.

Namespace:

GridMetric.LibV.Metadata.Serialization

Syntax:

```
public static class OsdSerializer
```

Static methods

LoadOsdFromFile

Loads OSD from file and deserializes it to object format.

Syntax:

```
public static OsdFile LoadOsdFromFile (  
    string pathToOsdFile  
)
```

Parameters:

pathToOsdFile

Type: System.String

Full path to OSD file to load.

Return value:

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

Deserialized OSD file object.

ParseOsdFromStream

Loads OSD file's contents from specified stream and deserializes it to object format.

Overload list:

Loads OSD file from supplied stream.

```
public static OsdFile ParseOsdFromStream (  
    Stream osdFileStream  
)
```

Loads OSD file from supplied stream using forced character encoding instead of auto-detection.

```
public static OsdFile ParseOsdFromStream (  
    Stream osdFileStream,  
    Encoding forcedEncoding  
)
```

Parameters:

osdFileStream

Type: System.IO.Stream

Stream to read OSD file's contents from.

forcedEncoding

Type: System.Text.Encoding

Encoding to use when interpreting the byte sequence from stream.

Return value:

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

Deserialized OSD file object.

ParseOsdFromXml

Loads OSD file from XML document and deserializes it to object format.

Syntax:

```
public static OsdFile ParseOsdFromXml (  
    XmlDocument osdXmlRepresentation  
)
```

Parameters:

osdXmlRepresentation

Type: System.Xml.XmlDocument

XML document representing the OSD file.

Return value:

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

Deserialized OSD file object.

Save

Saves supplied OSD object into file.

Overload list:

Saves OSD to file in XML format using character encoding specified in the object (by default ANSI).

```
public static void Save (  
    this OsdFile osd,  
    string fileName  
)
```

Saves OSD to file in XML format using forced character encoding.

```
public static void Save (  
    this OsdFile osd,  
    string fileName,
```

```
        Encoding encodingToUse
    )
```

Parameters:

osd

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

OSD file to serialize.

fileName

Type: System.String

Path to a file to which save OSD to.

requiredEncoding

Type: System.Text.Encoding

Encoding to use for serialization.

Note:

This class extends [OsdFile](#) –class so that Save(...) -method can be used directly from the OSD object itself. Only requirement is that codefile adds “using GridMetric.LibV.Metadata.Serialization” –clause in its using –block.

SerializeToStream

Serializes supplied OSD object into stream.

Overload list:

Serializes OSD to supplied stream in XML format using character encoding specified in the object (by default ANSI) and with standard OSD XML declaration header (<?xml version="1.0" standalone="no"?>).

```
public static void SerializeToStream (
    this OsdFile osd,
    Stream outputStream
)
```

Serializes OSD to supplied stream in XML format using character encoding specified in the object (by default ANSI), with optional omitting of declaration header.

```
public static void SerializeToStream (  
    this OsdFile osd,  
    Stream outputStream,  
    bool includeDeclarationHeader  
)
```

Serializes OSD to supplied stream in XML format using forced character encoding and with optional omitting of declaration header.

```
public static void SerializeToStream (  
    this OsdFile osd,  
    Stream outputStream,  
    bool includeDeclarationHeader,  
    Encoding encodingToUse  
)
```

Serializes OSD to supplied stream in XML format using forced character encoding and with standard OSD XML declaration header (<?xml version="1.0" standalone="no"?>).

```
public static void SerializeToStream (  
    this OsdFile osd,  
    Stream outputStream,  
    Encoding encodingToUse  
)
```

Parameters:

osd

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

OSD file to serialize.

outputStream

Type: System.IO.Stream

Stream to write serialized OSD to.

includeDeclarationHeader

Type: System.Boolean

Set **true** to write out standard OSD XML declaration header, **false** if resulting XML will not contain any declaration headers.

encodingToUse

Type: System.Text.Encoding

Encoding to use for serialization.

Note:

This class extends [OsdFile](#) -class so that `SerializeToStream(...)` -method can be used directly from the OSD object itself. Only requirement is that codefile adds "using GridMetric.LibV.Metadata.Serialization" -clause in its using -block.

SerializeToXml

Serializes supplied OSD object into XML document.

Overload list:

Serializes OSD to XML document using character encoding specified in the object (by default ANSI) and with standard OSD XML declaration header (<?xml version="1.0" standalone="no"?>).

```
public static XmlDocument SerializeToXml (  
    this OsdFile osd  
)
```

Serializes OSD to XML document using character encoding specified in the object (by default ANSI) with optional omitting of declaration header.

```
public static XmlDocument SerializeToXml (  
    this OsdFile osd,
```

```

        bool includeDeclarationHeader
    )

```

Serializes OSD to XML document using forced character encoding and with standard OSD XML declaration header (<?xml version="1.0" standalone="no"?>).

```

public static XmlDocument SerializeToXml (
    this OsdFile osd,
    Encoding encodingToUse
)

```

Parameters:

osd

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

OSD file to serialize.

includeDeclarationHeader

Type: System.Boolean

Set **true** to write out standard OSD XML declaration header, **false** if resulting XML will not contain any declaration headers.

requiredEncoding

Type: System.Text.Encoding

Encoding to use for serialization.

Note:

This class extends [OsdFile](#) –class so that SerializeToXml (...) -method can be used directly from the OSD object itself. Only requirement is that codefile adds “using GridMetric.LibV.Metadata.Serialization” –clause in its using –block.

OsdShellCmdElement class

Represents SHELLCOMMAND element in the OSD file, recording information about file association's action verb.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdShellCmdElement : OsdElementBase
```

Constructor

Initializes shell command.

```
public OsdShellCmdElement (  
    string name,  
    string friendlyName,  
    string commandParams,  
    bool? overridden  
)
```

Parameters:

name

Type: System.String

Name of the command verb. More information about command verbs can be found from MSDN documentation at <http://msdn.microsoft.com/en-us/library/cc144175.aspx>.

friendlyName

Type: System.String

Friendly name of the command verb. This is the text that will be shown to the user in Explorer's context menu entry.

commandParams

Type: System.String

Parameters to pass to the published application when associated file is opened.

overridden

Type: System.Nullable<T>

Is shell command overridden. This attribute is usually set to **false** for shell command registration in OSD files and is **null** for file shell command elements in the [ManifestFileTypeElement](#).

Properties

DdeExecInfo

Gets or sets DDE information related to command.

Syntax:

```
public OsdDdeExecElement DdeExecInfo { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdDdeExecElement](#)

DDE information for file association invokation.

FriendlyName

Gets or sets friendly name for command.

Syntax:

```
public string FriendlyName { get; set; }
```

Property value:

Type: System.String

Human-readable description text for the verb (action).

Name

Gets or sets name for command.

Syntax:

```
public string Name { get; set; }
```

Property value:

Type: System.String

Command name (e.g. **open**, **print** or **edit**).

Overridden

Gets or sets flag indicating if command is overridden. This flag is not documented by Microsoft but is usually set to **false** for OSD files generated by Microsoft Sequencer.

Syntax:

```
public bool? Overridden { get; set; }
```

Property value:

Type: System.Nullable<T>

true if command is overridden, **false** otherwise. **null** if not set.

Parameters

Gets or sets command parameters to pass to the application when association is invoked.

Syntax:

```
public string Parameters { get; set; }
```

Property value:

Type: System.String

Application –specific parameter string.

OsdShellCommandListElement class

Represents SHELLCOMMANDLIST element in the OSD file, recording information about commands (verbs) related to file association.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdShellCommandListElement : OsdElementBase
```

Constructor

Initializes shell command list.

```
public OsdShellCommandListElement (  
    string defaultCommand  
)
```

Parameters:

defaultCommand

Type: System.String

Name of the default command verb. The name is one of the commands represented by [OsdShellCmdElement](#) –object and listed in [ShellCommands](#) –property.

Properties

Default

Gets or sets default command verb for the association.

Syntax:

```
public string Default { get; set; }
```

Property value:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Type: System.String

Command verb name.

ShellCommands

Gets or sets list of commands associated with this command list (and containing file association).

Syntax:

```
public IList<OsdShellCmdElement> ShellCommands { get; set; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of shell commands.

OsdShortcutElement class

Represents SHORTCUT element in the OSD file, recording information about application shortcut which will be shown on App-V Client.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdShortcutElement : OsdElementBase
```

Constructors

Overload List:

Initializes new application shortcut.

```
public OsdShortcutElement (  
    string location,  
    string linkFilename,  
    bool overridden,  
    string displayName,  
    string iconFilename  
)
```

Initializes new application shortcut.

```
public OsdShortcutElement (  
    string location,  
    string linkFilename,  
    bool overridden,  
    string displayName,  
    string iconFilename,  
    string workingDir,  
    string parameters
```

)

Parameters:

location

Type: System.String

Location on the system to place shortcut to. This location can contain encoded Intermediate Values to specify typical Windows shortcut locations,

%CSDIL_PROGRAMS% for Start Menu's Programs –folder and

%CSIDL_DESKTOPDIRECTORY% for the desktop.

linkFilename

Type: System.String

Physical filename of the link file App-V Client creates on the place specified by *location* –parameter, e.g. **Skype.lnk**. In Manifest –file, physical filename is not used and is set to **null**.

overridden

Type: System.Nullable<T>

Is shortcut overridden. For shortcut elements in OSD files, typically contains **false** and in Manifest –file is set to **null**.

displayName

Type: System.String

Display name for the shortcut. Usually is the same as link's filename but without trailing .lnk –part.

iconFilename

Type: System.String

Physical filename of the icon file used for the shortcut. In OSD files, icon files in the package directory are referenced using special variable, **%SFT_MIME_SOURCE%**, marking the root of the directory holding package files (e.g.

%SFT_MIME_SOURCE%/AbiWord Icons/AbiWord.ico).

workingDir

Type: System.String

Working directory for application referenced by the shortcut. This property is typically set to **null** in App-V created shortcuts but values can exist in some older packages.

parameters

Type: System.String

Application parameters. This property is typically set to **null** in App-V created shortcuts but values can exist in some older packages.

Properties

Display

Gets or sets display name for the shortcut.

Syntax:

```
public string Display { get; set; }
```

Property value:

Type: System.String

Display name.

Filename

Gets or sets physical file system name for shortcut's .LNK file.

Syntax:

```
public string Filename { get; set; }
```

Property value:

Type: System.String

Physical filename of LNK file.

Note:

Not used on shortcut –elements in Manifest file.

Icon

Gets or sets physical file system name for shortcut's .ICO file.

Syntax:

```
public string Icon { get; set; }
```

Property value:

Type: System.String

Physical filename of icon file.

Note:

In OSD files, icon files in the package directory are referenced using special variable, %SFT_MIME_SOURCE%, marking the root of the directory holding package files (e.g. %SFT_MIME_SOURCE%/AbiWord Icons/AbiWord.ico).

Location

Gets or sets location where shortcut will be published to.

Syntax:

```
public string Location { get; set; }
```

Property value:

Type: System.String

Directory location.

Note:

Location can contain encoded Intermediate Values to specify typical Windows shortcut locations, %CSDIL_PROGRAMS% for Start Menu's Programs –folder and %CSIDL_DESKTOPDIRECTORY% for the desktop.

Overridden

Gets or sets flag indicating if shortcut is overridden. This flag is not documented by Microsoft but is usually set to **false** for OSD files generated by Microsoft Sequencer and is set to **null** for Manifest files.

Syntax:

```
public bool? Overridden { get; set; }
```

Property value:

Type: System.Nullable<T>

true if shortcut is overridden, **false** otherwise. **null** if not set.

Parameters

Gets or sets application command-line parameters.

Syntax:

```
public string Parameters { get; set; }
```

Property value:

Type: System.String

Parameters or **null** if not set.

Note:

Usually not found in App-V OSD files.

WorkingDirectory

Gets or sets application working directory path.

Syntax:

```
public string WorkingDirectory { get; set; }
```

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Property value:

Type: System.String

Working directory path or **null** if not set.

Note:

Usually not found in App-V OSD files.

OsdVirtualenvElement class

Represents virtual environment configuration information for application.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdVirtualenvElement : OsdElementBase
```

Constructors

Overload List:

Initializes new empty virtual environment configuration with child-process termination set to false.

```
public OsdVirtualenvElement ()
```

Initializes new empty virtual environment configuration.

```
public OsdVirtualenvElement (  
    bool? terminateChildren  
)
```

Parameters:

terminateChildren

Type: System.Nullable<T>

Are possible child processes started by main process defined in OSD file automatically terminated by App-V Client upon main process termination.

Properties

Dependencies

Gets or sets list of package dependencies.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Syntax:

```
public OsdDependenciesElement Dependencies { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdDependenciesElement](#)

Package dependencies.

EnvironmentVariables

Gets or sets list of virtualized environment variables defined for the virtual application.

Syntax:

```
public IList<OsdEnvironmentElement> EnvironmentVariables { get; set; }
```

Property value:

Type: System.Collection.Generic.IList<T>

List of virtualized environment variables.

Policies

Gets or sets virtual environment policies for the virtual application.

Syntax:

```
public OsdPoliciesElement Policies { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdPoliciesElement](#)

Virtual environment policies.

Registry

Gets or sets OSD virtual registry overrides.

Syntax:

```
public OsdRegistryElement Registry { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdRegistryElement](#)

Virtual registry overrides.

TerminateChildren

Gets or sets flag indicating if child processes will be terminated automatically upon main process (i.e. virtual application launched by OSD) exiting.

Syntax:

```
public bool? TerminateChildren { get; set; }
```

Property value:

Type: System.Nullable<T>

true if child processes are terminated automatically, **false** otherwise. **null** if not set.

OsdVmElement class

Represents VM element in the OSD file, recording information about application process type being started.

Namespace:

GridMetric.LibV.Metadata.Osd

Syntax:

```
public class OsdVmElement : OsdElementBase
```

Constructor

Initializes new process type information.

```
public OsdVmElement (  
    OsdVmType vmType,  
    OsdSubsystemType subsystemType  
)
```

Parameters:

vmType

Type: [GridMetric.LibV.Metadata.Structures.OsdVmType](#)

Process type (bitness).

subsystemType

Type: [GridMetric.LibV.Metadata.Structures.OsdSubsystemType](#)

Process' subsystem type.

Properties

Subsystem

Gets or sets process' subsystem type.

Syntax:

```
public OsdSubsystemType Subsystem { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdSubsystemType](#)

Subsystem type. Usually set to [Windows](#) except for command-line based application having no GUI.

Type

Gets or sets process' bitness.

Syntax:

```
public OsdVmType Subsystem { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.OsdVmType](#)

Process' type i.e. bitness.

SprjDefaultElement class

Represents DEFAULT element in the SPRJ file, recording information about Sequencer option.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjDefaultElement
```

Constructor

Initializes new Sequencer option default.

```
public SprjDefaultElement (  
    string name,  
    string value  
)
```

Parameters:

name

Type: System.String

Name of the option.

value

Type: System.String

Option's value.

Methods

ToString

Returns Sequencer option setting in textual representation.

Syntax:

```
public override string ToString()
```

Return value:

Type: System.String

Option value in textual form (i.e. *Name = Value*).

Properties

Name

Gets or sets name of the option.

Syntax:

```
public string Name { get; set; }
```

Property value:

Type: System.String

Option name.

Value

Gets or sets value of the option.

Syntax:

```
public string Value { get; set; }
```

Property value:

Type: System.String

Option's value.

SprjDeletedFileExtension

Represents a DELETEDOBJECT element in the SPRJ file, recording information about file extension that has been deleted from published application record.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjDeletedFileExtension : SprjDeletedObjectElement
```

Constructor

Initializes new deleted file extension.

```
public SprjDeletedFileExtension (  
    string application,  
    string ext  
)
```

Parameters:

application

Type: System.String

Path to an executable (application) with which the extension was originally associated against.

ext

Type: System.String

File extension.

Properties

Application

Gets or sets application path.

Syntax:

```
public string Application { get; set; }
```

Property value:

Type: System.String

Path to executable deleted extension relates to.

Extension

Gets or sets file extension.

Syntax:

```
public string Extension { get; set; }
```

Property value:

Type: System.String

File extension.

SprjDeletedFileType

Represents a DELETEDOBJECT element in the SPRJ file, recording information about file type association that has been deleted from published application record.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjDeletedFileType : SprjDeletedObjectElement
```

Constructor

Initializes new deleted file type record.

```
public SprjDeletedFileType (  
    string application,  
    string progId  
)
```

Parameters:

application

Type: System.String

Path to an executable (application) with which the file association was originally created against.

progId

Type: System.String

Programmatic identifier of the association.

Properties

Application

Gets or sets application path.

Syntax:

```
public string Application { get; set; }
```

Property value:

Type: System.String

Path to executable deleted file association relates to.

ProgId

Gets or sets programmatic identifier.

Syntax:

```
public string ProgId { get; set; }
```

Property value:

Type: System.String

Programmatic identifier for file association.

SprjDeletedObjectElement

Represents a base class for different types of DELETEDOBJECT elements in the SPRJ file, recording information about object that has been deleted from published application record.

These objects include shortcuts and file type associations which will be recorded in the SPRJ file if user manually removes them from the application.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public abstract class SprjDeletedObjectElement
```

Properties

Type

Gets type of the deleted object.

Syntax:

```
public SprjDeletedObjectType Type { get; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.SprjDeletedObjectType](#)

Type of the deleted object.

SprjDeletedShortcut

Represents a DELETEDOBJECT element in the SPRJ file, recording information about application shortcut that has been deleted from published application record.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjDeletedShortcut : SprjDeletedObjectElement
```

Constructor

Initializes new deleted shortcut record.

```
public SprjDeletedShortcut (  
    string filename  
)
```

Parameters:

filename

Type: System.String

Path to an .LNK file for shortcut.

Properties

Filename

Gets or sets shortcut filename.

Syntax:

```
public string Filename { get; set; }
```

Property value:

Type: System.String

Path to .LNK file for the shortcut.

SprjExclusionElement

Represents an EXCLUSION element in the SPRJ file, recording information about monitoring time exclusion entry for Sequencer.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjExclusionElement : SprjParserItemElement
```

Constructor

Initializes new exclusion item.

```
public SprjExclusionElement (  
    string pattern,  
    SprjParserItemContext context,  
    SprjParserItemType type  
)
```

Parameters:

pattern

Type: System.String

Path to exclude. This path is usually encoded with Intermediate Values.

context

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemContext](#)

Context in which the exclusion is valid in.

type

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemType](#)

Account context in which the exclusion is valid in.

Methods

ToString

Returns Sequencer exclusion item in textual representation.

Syntax:

```
public override string ToString()
```

Return value:

Type: System.String

Exclusion item in textual form.

Properties

Context

Gets or sets exclusion item context.

Syntax:

```
public SprjParserItemContext Context { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemContext](#)

Context of the exclusion item.

SprjFile class

Represents SPRJ (Sequencer Project) file in an object format. SPRJ file contains linkage to other App-V package files as well as Sequencer –related options related to the App-V package.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjFile : MetadataFileBase
```

Constructors

Overload List:

Initializes SPRJ file from specified OSD and SFT filename references, and initializes default exclusion items and Sequencer default settings.

```
public SprjFile(  
    string sftFilename,  
    IList<string> osdFilenames  
)
```

Initializes SPRJ file from specified OSD and SFT filename references and initializes package version information and substitution items from SFT package reference and initializes default exclusion items and Sequencer default settings.

```
public SprjFile(  
    string sftFilename,  
    IList<string> osdFilenames,  
    SftPackage sftPackage  
)
```

Initializes SPRJ file from specified OSD collection and SFT filename references and initializes package version information and substitution items from SFT package reference and initializes default exclusion items and Sequencer default settings.

```
public SprjFile(  
    string sftFilename,  
    NameOsdFileCollection osdCollection,  
    SftPackage sftPackage  
)
```

Parameters:

sftFilename

Type: System.String

Filesystem name of the SFT file associated with the package.

osdFileNames

Type: System.Collections.Generic.IList<T>

List of filesystem names for OSD files associated with the package.

sftPackage

Type: [GridMetric.LibV.SftPackage](#)

Reference to package's SFT object.

osdCollection

Type: [GridMetric.LibV.Metadata.Osd.NameOsdFileCollection](#)

List of (filesystem names for) OSD files associated with the package.

Methods

LoadDefaultExclusions

Loads and replaces exclusion items list in SPRJ file with defaults.

Syntax:

```
public void LoadDefaultExclusions()
```

LoadDefaults

Loads and replaces Sequencer options list in SPRJ file with defaults.

Syntax:

```
public void LoadDefaultExclusions()
```

LoadDefaultSubstitutions

Loads and replaces substitutions items list (i.e. Intermediate Value mappings) in SPRJ file with defaults related to actual package version.

Syntax:

```
public void LoadDefaultSubstitutions(  
    SftPackage sftPackage  
)
```

Parameters:

sftPackage

Type: [GridMetric.LibV.SftPackage](#)

SFT package for which the substitution items will be based on.

Note:

This command will load substitution items specific to bitness of machine on where the command is issued, i.e. on 64-bit machine loaded values will reflect 64-bit Windows installation and on 32-bit machine these will reflect values specific to 32-bit Windows. This command does not take App-V package's "internal bitness" into equation.

Properties

DefaultOsd

Gets or sets default OSD file stored in the SPRJ file.

Syntax:

```
public OsdFile DefaultOsd { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Osd.OsdFile](#)

Template OSD file.

Note:

Default OSD is used in App-V Sequencer as a template for new applications (i.e. OSD files) published from the package and contains all non-application specific OSD elements common to all OSDs in the same package.

Defaults

Gets or sets list of Sequencer options stored in the SPRJ file.

Syntax:

```
public IList<SprjDefaultElement> Defaults { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of Sequencer option defaults.

DeletedObjects

Gets or sets list of deleted objects stored in the SPRJ file.

Syntax:

```
public IList<SprjDeletedObjectElement> DeletedObjects { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of deleted objects.

Encoding

Gets or sets character encoding used for the (de)serialized SPRJ file.

Syntax:

```
public Encoding Encoding { get; set; }
```

Property value:

Type: System.Text.Encoding

Encoding used for the SPRJ file.

Exclusions

Gets or sets list of exclusion items stored in the SPRJ file.

Syntax:

```
public IList<SprjExclusionElement> Exclusions { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of monitoring time exclusion items for package.

OsdFileNames

Gets or sets list of filenames of OSD files associated with the package.

Syntax:

```
public IList<string> OsdFileNames { get; set; }
```

Property value:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

Type: System.Collections.Generic.IList<T>

List of OSD filenames.

Note:

Caller is responsible for maintaining this list up-to-date after package's OSD files has been saved (serialized) out.

ProjectVersion

Gets or sets Sequencer version used to save the package.

Syntax:

```
public string ProjectVersion { get; set; }
```

Property value:

Type: System.String

Full version number of Sequencer from which the package originated. Should be set to closest matching Sequencer version for actual package version or **null** for pre-4.0 packages.

RootFolderNameLong

Gets or sets long file name version of the package's internal root directory.

Syntax:

```
public string RootFolderNameLong { get; set; }
```

Property value:

Type: System.String

Path to a root folder (e.g. Q:\ 7-Zip File Archiver) or null if none set.

Note:

This value exists only for 4.6 SP1 packages and newer.

RootFolderNameLong

Gets or sets short file name version of the package's internal root directory.

Syntax:

```
public string RootFolderNameShort { get; set; }
```

Property value:

Type: System.String

Path to a root folder (e.g. Q 5VLAV1CS.OOU) or null if none set.

Note:

This value exists only for 4.6 SP1 packages and newer.

SftFilename

Gets or sets filename for SFT file.

Syntax:

```
public string ProjectVersion { get; set; }
```

Property value:

Type: System.String

Full version number of Sequencer from which the package originated. Should be set to closest matching Sequencer version for actual package version or **null** for pre-4.0 packages.

Note:

Caller is responsible for maintaining this property up-to-date after package's SFT files has been saved (encoded) out.

Substitutions

Gets or sets list of substitution items (i.e. Intermediate Variable mappings) stored in the SPRJ file.

Syntax:

```
public IList<SprjSubstitutionElement> Substitutions { get; set; }
```

Property value:

Type: System.Collections.Generic.IList<T>

List of substitution items for package.

SprjParserItemElement

Represents a base class for parser item objects in the SPRJ file.

Inherited objects include substitution items and exclusion items.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public abstract class SprjParserItemElement
```

Properties

Context

Gets or sets context for parser item.

Syntax:

```
public SprjParserItemContext Context { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemContext](#)

Context in which the parser item is valid.

Pattern

Gets or sets pattern to match for parser item.

Syntax:

```
public string Pattern { get; set; }
```

Property value:

Type: System.String

Text pattern to match. This is non-encoded version of variable in substitution items (e.g. **C:\Windows\System32**) and [most commonly] encoded variable in exclusion items (e.g. **%CSIDL_HISTORY%**).

Type

Gets or sets type for parser item.

Syntax:

```
public SprjParserItemType Type { get; set; }
```

Property value:

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemType](#)

Type of the parser item in which context the parser item is valid.

SprjSerializer class

Serializes and deserializes App-V SPRJ XML file between file and object format.

Namespace:

GridMetric.LibV.Metadata.Serialization

Syntax:

```
public static class SprjSerializer
```

Static methods

LoadSprjFromFile

Loads SPRJ from file and deserializes it to object format.

Syntax:

```
public static SprjFile LoadSprjFromFile (  
    string pathToSprjFile  
)
```

Parameters:

pathToSprjFile

Type: System.String

Full path to SPRJ file to load.

Return value:

Type: [GridMetric.LibV.Metadata.Sprj.SprjFile](#)

Deserialized SPRJ file object.

ParseSprjFromStream

Loads SPRJ file's contents from specified stream and deserializes it to object format.

Overload list:

Loads SPRJ file from supplied stream.

```
public static SprjFile ParseSprjFromStream (  
    Stream sprjFileStream  
)
```

Loads SPRJ file from supplied stream using forced character encoding instead of auto-detection.

```
public static SprjFile ParseSprjFromStream (  
    Stream sprjFileStream,  
    Encoding forcedEncoding  
)
```

Parameters:

sprjFileStream

Type: System.IO.Stream

Stream to read SPRJ file's contents from.

forcedEncoding

Type: System.Text.Encoding

Encoding to use when interpreting the byte sequence from stream.

Return value:

Type: [GridMetric.LibV.Metadata.Sprj.SprjFile](#)

Deserialized SPRJ file object.

ParseSprjFromXml

Loads SPRJ file from XML document and deserializes it to object format.

Syntax:

```
public static SprjFile ParseSprjFromXml (  
    XmlDocument sprjXmlRepresentation  
)
```

Parameters:

sprjXmlRepresentation

Type: System.Xml.XmlDocument

XML document representing the SPRJ file.

Return value:

Type: [GridMetric.LibV.Metadata.Sprj.SprjFile](#)

Deserialized SPRJ file object.

Save

Saves supplied SPRJ object into file.

Overload list:

Saves SPRJ to file in XML format using character encoding specified in the object (by default ANSI).

```
public static void Save (  
    this SprjFile sprj,  
    string fileName  
)
```

Saves SPRJ to file in XML format using forced character encoding.

```
public static void Save (  
    this SprjFile sprj,
```

```
        string fileName,  
        Encoding encodingToUse  
    )
```

Parameters:

sprj

Type: [GridMetric.LibV.Metadata.Sprj.SprjFile](#)

SPRJ file to serialize.

fileName

Type: System.String

Path to a file to which save SPRJ to.

requiredEncoding

Type: System.Text.Encoding

Encoding to use for serialization.

Note:

This class extends [SprjFile](#) –class so that Save(...) -method can be used directly from the SPRJ object itself. Only requirement is that codefile adds “using GridMetric.LibV.Metadata.Serialization” –clause in its using –block.

SerializeToStream

Serializes supplied SPRJ object into stream.

Overload list:

Serializes SPRJ to supplied stream in XML format using character encoding specified in the object (by default ANSI).

```
public static void SerializeToStream (  
    this SprjFile sprj,  
    Stream outputStream  
)
```

Serializes SPRJ to supplied stream in XML format using forced character encoding.

```
public static void SerializeToStream (  
    this SprjFile sprj,  
    Stream outputStream,  
    Encoding encodingToUse  
)
```

Parameters:

sprj

Type: [GridMetric.LibV.Metadata.Sprj.SprjFile](#)

SPRJ file to serialize.

outputStream

Type: System.IO.Stream

Stream to write serialized SPRJ to.

requiredEncoding

Type: System.Text.Encoding

Encoding to use for serialization.

Note:

This class extends [SprjFile](#) –class so that `SerializeToStream(...)` -method can be used directly from the SPRJ object itself. Only requirement is that codefile adds “using `GridMetric.LibV.Metadata.Serialization`” –clause in its using –block.

SerializeToXml

Serializes supplied SPRJ object into XML document.

Overload list:

Serializes SPRJ to XML document using character encoding specified in the object (by default ANSI).

```
public static XmlDocument SerializeToXml (  
    this SprjFile sprj  
)
```

Serializes SPRJ to XML document using forced character encoding.

```
public static XmlDocument SerializeToXml (  
    this SprjFile sprj,  
    Encoding requiredEncoding  
)
```

Parameters:

sprj

Type: [GridMetric.LibV.Metadata.Sprj.SprjFile](#)

SPRJ file to serialize.

requiredEncoding

Type: System.Text.Encoding

Encoding to use for serialization.

Note:

This class extends [SprjFile](#) –class so that `SerializeToXml(...)` -method can be used directly from the SPRJ object itself. Only requirement is that codefile adds “using `GridMetric.LibV.Metadata.Serialization`” –clause in its using –block.

SprjSubstitutionElement

Represents a SUBSTITUTION element in the SPRJ file, recording information about Intermediate Value mapping pair.

Namespace:

GridMetric.LibV.Metadata.Sprj

Syntax:

```
public class SprjSubstitutionElement : SprjParserItemElement
```

Constructor

Initializes new substitution item.

```
public SprjSubstitutionElement (  
    string pattern,  
    string replacement,  
    SprjParserItemContext context,  
    SprjParserItemType type  
)
```

Parameters:

pattern

Type: System.String

String to match for encoding to variable.

replacement

Type: System.String

Intermediate Value variable *pattern* is substituted to.

context

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemContext](#)

Context in which the substitution is valid in.

type

Type: [GridMetric.LibV.Metadata.Structures.SprjParserItemType](#)

Account context in which the substitution is valid in.

Methods

ToString

Returns substitution item in textual representation.

Syntax:

```
public override string ToString()
```

Return value:

Type: System.String

Substitution item in textual form.

Properties

Replacement

Gets or sets intermediate value (variable) for substitution.

Syntax:

```
public string Replacement { get; set; }
```

Property value:

Type: System.String

Variable name with leading and trailing percentage signs (e.g. **%CSIDL_WINDOWS%**).

Enumerations

Enumerations as used by classes provided in Lib-V metadata assemblies.

Namespace:

GridMetric.LibV.Metadata.Structures

MsiInstallMode

Specifies enumerated constants used to indicate SFT file loading behaviour when running App-V package wrapper MSI.

Syntax:

```
public enum MsiInstallMode : int
```

Standalone

SFT file is loaded into [standalone] client when MSI package is run and [LoadSft](#) is set to true (default). This is default setting for App-V MSIs and requires that SFT file is placed in the same directory as the MSI file or [SftPath](#) is defined.

Streaming

SFT file is not loaded into client when MSI package is run. SFT file is then streamed in on-demand based on information inside OSD files or path set in [OverrideUrl](#).

OsdOperatingSystemTypes

Specifies enumerated constants used to indicate supported operating system values for OSD file.

Syntax:

```
public enum OsdOperatingSystemTypes : long
```

All

OSD supports all operating systems (equals to having no OS –elements in the OSD file).

WinNT

OSD supports Windows NT 4 systems. Deprecated since SoftGrid 3.X versions.

Win2K

OSD supports Windows 2000 workstation systems. Deprecated since SoftGrid 4.0 version.

Win2KTS

OSD supports Windows 2000 Terminal Server systems. Deprecated since SoftGrid 4.0 version.

Win2KSvr

OSD supports Windows 2000 Server systems. Deprecated since SoftGrid 4.0 version.

WinXP

OSD supports 32-bit Windows XP systems.

WinXP64

OSD supports 64-bit Windows XP systems.

Win2003Svr

OSD supports 32-bit Windows Server 2003 systems.

Win2003TS

OSD supports 32-bit Windows Server 2003 Terminal Server systems.

Win2003TS64

OSD supports 64-bit Windows Server 2003 Terminal Server systems.

WinVista

OSD supports 32-bit Windows Vista systems.

WinVista64

OSD supports 64-bit Windows Vista systems.

Win2008Svr

OSD supports 32-bit Windows Server 2008 Server systems.

Win2008TS

OSD supports 32-bit Windows Server 2008 Terminal Server systems.

Win2008TS64

OSD supports 64-bit Windows Server 2008 Terminal Server systems.

Win7

OSD supports 32-bit Windows 7 systems.

Win764

OSD supports 64-bit Windows 7 systems.

Win2008R2TS64

OSD supports 64-bit Windows Server 2008 R2 Remote Desktop Services Session Host systems.

Win8

OSD supports 32-bit Windows 8 systems.

Win864

OSD supports 64-bit Windows 8 systems.

Win2012TS64

OSD supports 64-bit Windows Server 2012 Remote Desktop Services Session Host systems.

OsdRegistryValueType

Specifies enumerated constants used to indicate supported registry value data types for virtual registry overrides in OSD file.

Syntax:

```
public enum OsdRegistryValueType : int
```

Note:

OSD registry value overrides do not support all registry data types normally found in the Windows registry.

REG_SZ

String data-type.

REG_BINARY

Binary data-type.

REG_DWORD

Double-word (32-bit) data-type.

OsdScriptConditional

Specifies enumerated constants used to indicate conditional for OSD script execution's result code.

Syntax:

```
public enum OsdScriptConditional : int
```

None

No specific conditions apply for script execution results.

SuccessResult

Script execution has to return specific errorlevel value so that the script is considered ended successfully (and application is launched, if running script before virtual application startup). Other codes indicate error condition and result failure.

AbortResult

Script execution has to return specific errorlevel value so that the script is considered ended in error (and application launch is stopped, if running script before virtual application startup). Other codes indicate success condition and result continuation.

OsdScriptEvent

Specifies enumerated constants used to indicate specific event along with which the OSD script is executed.

Syntax:

```
public enum OsdScriptEvent : int
```

Stream

Script is executed before or after package streaming (or if package is cached, during initial setup-phase of the launch process).

Launch

Script is executed before or after virtual application launch.

Shutdown

Script is executed after application exists.

OsdScriptTiming

Specifies enumerated constants used to indicate specific timing of the event along with which the OSD script is executed.

Syntax:

```
public enum OsdScriptTiming : int
```

Pre

Script is executed before specified event. Does not apply to [Shutdown](#) -event, which only supports Post –timing.

Post

Script is executed after specified event.

OsdScriptType

Specifies enumerated constants used to indicate type of the OSD script.

Syntax:

```
public enum OsdScriptType : int
```

Href

Script is reference to executable or external script file.

Scriptbody

Script is embedded in the OSD file. Only support Windows command-interpreter scripts.

OsdVmType

Specifies enumerated constants used to indicate process type of the application that OSD file launches when executed.

Syntax:

```
public enum OsdVmType : int
```

Win16

Process in 16-bit executable.

Win32

Process in 32-bit executable.

Win64

Process in 64-bit executable.

OsdSubsystemType

Specifies enumerated constants used to indicate subsystem of the application process that OSD file launches when executed.

Syntax:

```
public enum OsdSubsystemType : int
```

NotSpecified

Subsystem is not defined (i.e. missing from the OSD).

Windows

Subsystem is graphical (GUI) application and App-V Client will try to detect window message loop running in it upon launch.

Console

Subsystem is console application and App-V Client will skip detection of window message loop upon launch.

SprjDeletedObjectType

Specifies enumerated constants used to indicate type of object related to published application and subsequently deleted from the package as recorded in the SPRJ file.

Syntax:

```
public enum SprjDeletedObjectType : int
```

FileExtension

Deleted object is file extension.

FileType

Deleted object is file type association.

Shortcut

Deleted object is application shortcut.

SprjParserItemContext

Specifies enumerated constants used to indicate context to which SPRJ parser item applies to.

Syntax:

```
public enum SprjParserItemContext : int
```

VirtualFilesystem

Parser item applies to virtual filesystem only.

VirtualRegistry

Parser item applies to virtual registry only.

Both

Parser item applies both to virtual registry and virtual filesystem.

SprjParserItemType

Specifies enumerated constants used to indicate type of the SPRJ parser item.

Syntax:

```
public enum SprjParserItemType : int
```

System

Parser item applies to system's context.

User

Parser item applies to user's context.

Appendix A – Lib-V Software License Agreement

Lib-V (hereinafter “SOFTWARE”)

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

1. GRANT OF LICENSE

The Licensee may embed the SOFTWARE within the Licensee’s own software in order to resale or otherwise distribute its own software. The Licensee may not distribute or resale the SOFTWARE as such.

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.

1.2 COMMERCIAL BINARY LICENSE

The Commercial Binary License entitles the Licensee to use the SOFTWARE with a number of developers to which the license has been purchased for and with an unlimited number of Licensee’s own software within the legal entity of the Licensee.

The Commercial Binary License has one version, with Read/Write (RW) privileges.

1.3 SOURCE-CODE ACCESS LICENSE

The Source-Code Access License entitles to use the SOFTWARE with a number of developers to which the license has been purchased for and with an unlimited number of Licensee's own software within the legal entity of the Licensee.

With the Source-Code Access License the Licensee has access to the source code of the SOFTWARE and the right to modify the source code. The Licensee is not allowed to distribute neither the modified nor the unmodified SOFTWARE as such. If the SOFTWARE has been modified, the Licensee is not entitled to product support from GridMetric.

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 Licensee terminates it by destroying the SOFTWARE and its documentation together with all copies. It shall also terminate if Licensee breaches any material term of this Agreement and fails to cure such breach within thirty days of notice thereof. Upon termination Licensee agrees to destroy all copies of the SOFTWARE and its documentation including any SOFTWARE stored on the hard disk of any computer under Licensee's control. Sections 5, 7 and 8 shall survive any termination of this Agreement.

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:

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

- 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.

Copyright © 2008 - 2019 Gridmetric Oy. All rights reserved.

Redistribution, reuse, partial reuse and public display prohibited.

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.