ivanti | Management Center

Package Management Web Service API Guide

Version 2023.1

Table of Contents

Disclaimer

Introduction

This document details the web services interface exposed by the AppSense Management Server. While primarily used by the AppSense Management Console this interface is available for others to use.

Caution

AppSense reserves the right to modify any API classes or method signatures without warning.

Users are advised that changes to the API will occur over the span of releases and updates, and that any scripts that use these APIs should be carefully tested with new versions of AppSense products before being deployed in a production environment.

Since the Management Console uses the same API to communicate with the Management Server, anything possible within the Management Console is also possible through the API.

Management Console

Proxy DLL

The Management Console product ships with a web services proxy DLL used by the console for communication with the server. Third party tools may use this interface for convenience, or may access the APIs directly via the web services described in this document. The following DLLs are provided:

- DataAccess Endpoints: [InstallDir]\Management Center\Console\ManagementConsole.WebServices.dll
- PackageManagement Endpoint: [InstallDir]\Management Center\Console\PackageManager.dll (this also requires CommonDialogs.dll)

Caution

The AppSense Management Console must be installed before using the Proxy DLLs directly to ensure that any dependant DLLs are also available. This step is not required if the APIs are accessed directly via the web services

Authorization & Security

A full set of credentials should be specified when using the Proxy DLL by passing an instance of NetworkCredential. An example of this can be found under “Connecting to the PackageManagement Endpoint”

Limitations

Some types defined in the proxy DLL are not described in this document, as they are based on standard data types defined in the .NET framework. Users looking for documentation on DataSet types should refer to MSDN for further details: http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx

How to use the Package Management WebServices - Examples

Package Management Endpoint Tasks

Connecting to the PackageManagement Endpoint

Firstly, ensure that the latest version of the Management Console is installed. Before using the API a connection to the Management Server must be established. This involves:

- Loading the proxy DLL
- Instantiating a NetworkCredential instance
- Call the GetPackageServer method of PackageManagement.PackageServerFactory

To logon as the current user:

```
# Load proxy DLLAdd-Type -Path "${Env:ProgramFiles}\AppSense\Management
Center\Console\PackageManager.dll"
```

```
# Management Server URL$url = "http://localhost:7751/ManagementServer"
```

```
# Get NetworkCredential instance$credentials =
[System.Net.CredentialCache]::DefaultCredentials$credential = $credentials.GetCredential($url,
"Basic")
```

```
# Create connection to the Management
Server[PackageManagement.PackageServerFactory]::GetPackageServer($url, $credential)
```

To specify a user to logon as:

```
# Load proxy DLLAdd-Type -Path "${Env:ProgramFiles}\AppSense\Management
Center\Console\PackageManager.dll"
```

```
# Management Server URL$url = "http://localhost:7751/ManagementServer"
```

```
# Get NetworkCredential instance$credential = New-Object System.Net.NetworkCredential -Property
Username = "MyUsername";    Password = "MyPassword";    Domain = "MyDomain";}
```

```
# Create connection to the Management
Server[PackageManagement.PackageServerFactory]::GetPackageServer($url, $credential)
```

Retrieving Package Details

Packages are managed using the PackagesWebService. For convenience when using the Proxy DLL a reference to this web service is available from ManagementConsole.WebServices.Packages once a connection is established. Packages can be retrieved using the PackagesService.GetPackages() method. This returns a PackageDataSet (based on the standard .NET DataTable class). This in turn contains a Packages property which is an IEnumerable of type PackageRow (based on the standard .NET DataRow class). You can then further filter packages using PackageDataSet.Packages.Select or alternatively in PowerShell use the Where-Object.

```
# Get Packages Web Service reference$PackagesWebService =
[PackageManagement.PackageServerFactory]::GetPackageServer($url, $credential)
```

```
# Retrieve list of packages$PackagesDataSet = $PackagesWebService.GetPackages()$Packages =
$PackagesDataSet.Packages
```

Uploading/Downloading Packages

Management Center packages can be created through the API using the PackagesWebService. Each package stored in the Management Center may contain multiple versions of a package.The process of uploading packages involves:

- Creating the package (if no instances of the package already exist)
- Creating a package version
- Getting a package upload key
- Finalising the package version
- Commit the package
- Unlock the package

When creating a package a product key needs to be specified. The following products are supported:

Name	ProductPK
Environment Manager	EA13F465-B2B6-4716-ADA4-53D7D84D042F
Performance Manager	969D739C-F6BA-4F8E-A210-632FC2B97D70
Application Manager	A1FD1D32-4F66-4958-9A81-9B72053661B2
Management Center	C30C105F-9961-41EA-8DD8-9DF4606E56FB

While packages can be uploaded using the DataAccess Endpoints, it is recommended that developers use the PackageManagment Endpoint.

Note: Ensure that PackageGuid and PackageVersionGuid are replaced with the

appropriate UpgradeCode and ProductCode from the specified MSI.

```
Add-Type -Path "${Env:ProgramFiles}\AppSense\Management Center\Console\PackageManager.dll"$url =
"http://localhost:7751/ManagementServer"$credentials =
[System.Net.CredentialCache]::DefaultCredentials$credential = $credentials.GetCredential($url,
"Basic")$PackagesWebService = [PackageManagement.PackageServerFactory]::GetPackageServer($url,
$credential)
```

```
# Environment Manager product$EnvironmentManagerKey = "EA13F465-B2B6-4716-ADA4-53D7D84D042F"
```

```
# Properties for newly created package$PackageGuid = "<Upgrade code from MSI Property table goes
here>"$PackagePlatform =
[PackageManagement.PackageManagementWebService_v2.PackagePlatform_v1]::PlatformIndependent
```

```
# Create package$PackagesWebService.CreatePackage($PackageGuid, "MyPackage", "MyCompany",
"msi/configuration", $PackagePlatform, $EnvironmentManagerKey)
```

```
# Create package version$PackageVersionGuid = "<Product code from MSI Property table goes
here>"$UploadDateTime = $PackagesWebService.CreatePackageVersion($PackageGuid,
$PackageVersionGuid, "MyPackageVersion", 8, 3, 0, 0, 8, 3, 0, 0, 8, 3, 0, 0, 8, 3, 0, 0,
"MyPackageDescription")
```

```
# Open file for upload$UploadFile = "C:\Configurations\EMConfiguration.msi"$UploadBytes =
[System.IO.File]::ReadAllBytes($UploadFile)
```

```
# Start package upload$UploadGuid = $PackagesWebService.BeginPackageVersionUpload($PackageGuid,
"API uploaded package", $PackageVersionGuid, $UploadBytes.Length, [ref] $UploadDateTime)
```

```
# Upload package$PackagesWebService.ContinuePackageVersionUpload($PackageVersionGuid, [ref]
$UploadDateTime, $UploadGuid, 0, $UploadBytes)
```

```
# Finalize package version$PackagesWebService.FinalisePackageVersion($PackageVersionGuid)
```

```
# Commit package version$PackagesWebService.CommitPackageVersion($PackageVersionGuid)
```

```
# Unlock package$PackagesWebService.UnlockPackage($PackageGuid)
```

Both endpoints support block by block upload operations for large files by repeatedly calling ContinuePackageVersionUpload. This is recommended for large configurations.

DataRows - Machines

Machines

MachinesRow

Provides data on each machine in the Management Center. The Platform column contains either 1 for a 32-bit machine, or 2 for a 64-bit machine. The GroupFK column is a foreign key relating to the group that the machine is a member of. This has the value of null for machines in the unassigned group.

Column	DataType	Description
MachineKey	Guid	Unique identifier relating to machine
GroupKey	Guid	Unique identifier relating to group
GroupName	String	Name of group
Platform	Int32	
NetBiosName	String	Name of net bios
DistinguishedName	String	Name of distinguished
OldDistinguishedName	String	Name of old distinguished
ObjectGuid	Guid	
LastPollTime	DateTime	
LastPollStatus	Int32	
LastUploadTime	DateTime	
LastUploadStatus	Int32	
IsPendingDeletion	Boolean	
AlertCount	Int32	
CreationTime	DateTime	Time created
ModifiedTime	DateTime	Time modified

ModifiedGroupTime	DateTime	
DiagnosticsError	Boolean	
DiagnosticsState	Int32	
DiagnosticsTime	DateTime	
Deployed	Int32	
DeployError	Boolean	
Offline	Boolean	
DNS	String	
LastResponseSeconds	Int32	

MachinePackagesRow

The CCA detects the installation state of all packages which have been added to the Management Center's database. This information is sent to the Management Server when the CCA polls, and is stored in the MachinePackages table. The Status column indicates the progress through the installation of the package:

- Pending Install
- Checking Prerequisites
- Downloading
- Download Failed
- Installing
- Installed
- Install Failed
- Pending Upgrade
- Upgrade Failed
- Pending Uninstall
- Uninstalling
- Uninstall Failed
- Uninstalled
- Install Prerequisite Failed
- Unmanaged

The StatusMessage column will contain an error message if the Status column is a failure.

Column	DataType	Description
MachineKey	Guid	Unique identifier relating to machine

PackageKey	Guid	Unique identifier relating to package
Major	Int32	
Minor	Int32	
Build	Int32	
Revision	Int32	
Name	String	Name of machine package
Company	String	
Type	String	
Platform	Int32	
ProductName	String	Name of product
CreationTime	DateTime	Time created
ModifiedTime	DateTime	Time modified
Status	Int32	
StatusMessage	String	
ChildStatus	Int32	
ChildMajor	Int32	
ChildMinor	Int32	
ChildBuild	Int32	
ChildRevision	Int32	
CertificateKey	Guid	Unique identifier of the associated PFX package (certificate)
CertificateThumbprints	String	Comma separated list of thumbprints for the

		certificates in the PFX package (certificate)
CertificatePasssword	String	Encrypted password to open the PFX package

MatchResultsRow

Represents a match result on the server

Column	DataType	Description
GroupKey	Guid	Unique identifier relating to group
GroupName	String	Name of group
MatchName	String	Name of match
Difference	Int32	

MatchResultsRow

Represents a match result on the server

Column	DataType	Description
GroupKey	Guid	Unique identifier relating to group
GroupName	String	Name of group
MatchName	String	Name of match
Difference	Int32	

MachineDetailsRow

Stores a collection of name \ value pairs containing the machines details such as OS, memory, cpu platform 32 or 64 bit etc.

Column	DataType	Description
MachineKey	Guid	Unique identifier relating to machine
Name	String	Name of machine detail
Value	String	

MachineDiagnosticsRow

Stores the results of any diagnostic tests that have been performed on the machine.

Column	DataType	Description
MachineKey	Guid	Unique identifier relating to machine
ServerUrl	String	
Name	String	Name of machine diagnostic
Error	Boolean	
Message	String	

DataRows - Packages

Packages

PackagesRow

Stores the version independent properties of a package, such as the platform and type. The platform column can be 0 for platform independent, 1 for 32-bit and 2 for 64-bit. The type column can be "msi/configuration" for configurations, and "msi/agent" for agents. If the package has been locked by a user, then the Locked column is set to 1 and the LockedUserName set to the name of the user who owns the lock.

Column	DataType	Description
LatestName	String	Name of latest
PackageKey	Guid	UpgradeCode property defined in associated MSI files
Company	String	
Type	String	
Platform	Int32	
ProductKey	Guid	Unique identifier relating to product
ProductName	String	Name of product
CreationTime	DateTime	Time created
ModifiedTime	DateTime	Time modified
OwnerSid	String	
PolicyKey	Guid	Unique identifier relating to policy
Locked	Boolean	
LockedUserName	String	Name of locked user
LatestMajor	Int32	

LatestMinor	Int32	
LatestBuild	Int32	
LatestRevision	Int32	
SecurityDescriptor	String	
Dirty	Byte	
CertificateKey	Guid	Unique identifier of the associated PFX certificate package

PackageVersionsRow

Stores the actual data for each version of a package. The name is stored on a per package basis to accommodate renames of configs and tags of agents (such as beta). The Major, Minor, Build and Revision fields form the unique version number for the package. The InProgress column identifies a version of a package which is currently being modified, and hence should not be deployed. The creator versions store the version number of the console which created a configuration, and the dependent minimum and maximum columns representing the minimum and maximum versions of agents that the configuration is compatible with.

Column	DataType	Description
PackageVersionKey	Guid	ProductCode property defined in associated MSI files
PackageKey	Guid	UpgradeCode property defined in associated MSI files
Name	String	Name of package version
Major	Int32	Major version of package
Minor	Int32	Minor version of package
Build	Int32	Build version of package
Revision	Int32	Revision version of package

DataLength	Int32	Size of package
CreationTime	DateTime	Time created
ModifiedTime	DateTime	Time modified
InProgress	Boolean	
UserName	String	Name of user
Description	String	Description of package version
CreatorMajor	Int32	Major version of package creator (console)
CreatorMinor	Int32	Minor version of package creator (console)
CreatorBuild	Int32	Build version of package creator (console)
CreatorRevision	Int32	Revision version of package creator (console)
DependentMinimumMajor	Int32	Major version of minimum associated agent
DependentMinimumMinor	Int32	Minor version of minimum associated agent
DependentMinimumBuild	Int32	Build version of minimum associated agent
DependentMinimumRevision	Int32	Revision version of minimum associated agent
DependentMaximumMajor	Int32	Major version of maximum associated agent
DependentMaximumMinor	Int32	Minor version of maximum associated agent
DependentMaximumBuild	Int32	Build version of maximum associated agent

DependentMaximumRevision	Int32	Revision version of maximum associated agent
PackagesRow	PackagesRow	

PatchesRow

Stores meta-data for a patch, including the package version that the patch applies.

Column	DataType	Description
PatchKey	Guid	The unique identifier of the patch
PackageVersionKey	Guid	The package version that this patch applies to
PatchCode	Guid	The Patch Code property of the Windows Installer MSP file
Name	String	Name of the patch
Major	Int32	Major version of package once this patch is applied
Minor	Int32	Minor version of package once this patch is applied
Build	Int32	Build version of package once this patch is applied
Revision	Int32	Revision version of package once this patch is applied
DataLength	Int32	Size of patch
InProgress	Boolean	True whenever this patch is being updated
CreationTime	DateTime	Time created

ModifiedTime	DateTime	Time modified
TargetMajor	Int32	The version of the patch or package that this patch applies to
TargetMinor	Int32	The version of the patch or package that this patch applies to
TargetBuild	Int32	The version of the patch or package that this patch applies to
TargetRevision	Int32	The version of the patch or package that this patch applies to
ValidationFlags	Int32	The Validation Flags property of the Windows Installer MSP file

CertificatesRow

Stores meta-data for a PFX certificate package, including the thumbprints of the contained certificates and the password for the package.

Column	DataType	Description
CertificateKey	Guid	The unique identifier of the PFX certificate package
PackageKey	Guid	The package key that this certificate is associated with
Name	String	The name of the PFX certificate file
DataLength	Int32	The size of the PFX certificate file

CreationTime	DateTime	The date and time when the PFX certificate file was uploaded
ModifiedTime	DateTime	The date and time that the certificate row was last modified
InProgress	Boolean	Set to true when the PFX certificate file is being uploaded
UserName	String	The name of the user that uploaded the PFX certificate file
Description	String	Description of the PFX certificate file
Password	String	The encrypted password for the PFX certificate file
Thumbprints	String	Comma separated list of thumbprints of the certificates contained in the PFX certificate file
EarliestExpiry	DateTime	The date and time of the earliest expiry time of the contained certificates
IsDeployed	Boolean	Set to true when the PFX certificate file is assigned to a group

PrerequisitesRow

Represents a prerequisite on the server

Column	DataType	Description

PrerequisitesKey	Guid	Unique identifier relating to prerequisites
Name	String	Name of prerequisite
Version	Int32	
PlatformInfo	String	
VersionInfo	String	
ModifiedTime	DateTime	Time modified
CreationTime	DateTime	Time created

PackageVersionPrerequisitesRow

Represents a package version prerequisite on the server

Column	DataType	Description
PackageVersionKey	Guid	Unique identifier relating to package version
PrerequisitesKey	Guid	Unique identifier relating to prerequisites
PrerequisitesRow	PrerequisitesRow	

PrerequisiteResourceRow

Represents a prerequisite resource on the server

Column	DataType	Description
ResourceKey	Guid	Unique identifier relating to resource
PrerequisiteKey	Guid	Unique identifier relating to prerequisite

Destination	String	
DataLength	Int32	
HashCode	String	
Valid	Boolean	
ModifiedTime	DateTime	Time modified
CreationTime	DateTime	Time created
PrerequisitesRow	PrerequisitesRow	

PrerequisiteCommandRow

Represents a prerequisite command on the server

Column	DataType	Description
CommandKey	Guid	Unique identifier relating to command
PrerequisiteKey	Guid	Unique identifier relating to prerequisite
Action	Int32	
Type	Int32	
Path	String	
Arguments	String	
DefaultResult	Int32	
ModifiedTime	DateTime	Time modified
CreationTime	DateTime	Time created
PrerequisitesRow	PrerequisitesRow	

PrerequisiteCheckRow

Represents a prerequisite check on the server

Column	DataType	Description
PrerequisiteCheckKey	Guid	Unique identifier relating to prerequisite check
PrerequisiteKey	Guid	Unique identifier relating to prerequisite
CheckType	Int32	
Condition	Int32	
OperatorValue	Int32	
Data	String	
Product_ProductCode	Guid	
Product_UpgradeCode	Guid	
File_Path	String	
Registry_Root	String	
Registry_Key	String	Unique identifier relating to registry_
Registry_Value	String	
OperatingSystem_Message	String	
ModifiedTime	DateTime	Time modified
CreationTime	DateTime	Time created
PrerequisitesRow	PrerequisitesRow	

PrerequisiteExitCodeRow

Represents a prerequisite exit code on the server

Column	DataType	Description
ExitCodeKey	Guid	Unique identifier relating to exit code
Value	Int32	
Result	String	
CommandKey	Guid	Unique identifier relating to command
ModifiedTime	DateTime	Time modified
CreationTime	DateTime	Time created
PrerequisiteCommandRow	PrerequisiteCommandRow	

Namespace Index

Namespace List

Here is a list of all documented namespaces with brief descriptions:

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Namespace Documentation

DataAccessServices Namespace Reference

Namespaces

- namespace PackageManagement

DataAccessServices.PackageManagement Namespace Reference

Classes

- class **Implementation**
- class PackageManagement_v1
- *This class contains legacy functionality to manage packages in the Management Server (version 8.1 or less) database.* class PackageManagement_v2
- *This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.* class PackageManagement_v3
- *This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.* class RowFixerNeeded
- class TableCopier

Class Documentation

DataAccessServices.PackageManagement.PackageManagement_v1 Class Reference

This class contains legacy functionality to manage packages in the Management Server (version 8.1 or less) database.

Inheritance diagram for DataAccessServices.PackageManagement.PackageManagement_v1:

Public Member Functions

- Guid BeginPackageVersionDownload (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision)
  Begins a package download.
- Guid BeginPackageVersionUpload (Guid packageKey, String description, Guid packageVersionKey, out DateTime modifiedTime, int dataLength)
  Begins an upload of a package version.
- void CommitPackageVersion (Guid packageVersionKey)
  Once a package has been fully uploaded it must be committed, this clears the In Progress flag and allows the package to be deployed.
- Byte[] ContinuePackageVersionDownload (Guid downloadKey, Int32 offset, Int32 length)
  Continues a package download. The bytes of the package from 'offset' to 'offset + length' are returned.
- void ContinuePackageVersionUpload (Guid packageVersionKey, ref DateTime modifiedTime, Guid uploadKey, Int32 offset, Byte[] data)
  Continues an upload of a package version.
- void CreatePackage (Guid key, String name, String company, String type, PackageManagementWebServiceCode.Schemas.PackagePlatform_v1 platform, Guid productKey, out DateTime modifiedTime)
  Creates a configuration package within the database.
- void CreatePackageVersion (Guid packageKey, Guid packageVersionKey, Int32 major, Int32 minor, Int32 build, Int32 revision, Int32 creatorMajor, Int32 creatorMinor, Int32 creatorBuild, Int32 creatorRevision, Int32 dependentMinimumMajor, Int32 dependentMinimumMinor, Int32 dependentMinimumBuild, Int32 dependentMinimumRevision, Int32 dependentMaximumMajor, Int32 dependentMaximumMinor, Int32 dependentMaximumBuild, Int32 dependentMaximumRevision, string description, Boolean supportsMidSessionUpdate, out DateTime modifiedTime)
  Creates a package version entry in the database ready to be uploaded.
- void DeletePackage (Guid key, DateTime?modifiedTime)
  Deletes an existing package from the database.
- void DeletePackageVersion (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision, DateTime?modifiedTime)
  Deletes a package version from a package.

- void FinalisePackageVersion (Guid packageVersionKey)
  Check that a package version has been correctly uploaded.
- PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1 GetMachinesWithPackage (Guid packageKey)
  Returns the machines that have a particular package installed.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 GetPackageFromKey (Guid key)
  Returns an individual package within the database.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 GetPackageFromKeyWithInProgress (Guid key)
  Returns a package from the database that is currently in progress.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 GetPackages ()
  Returns all packages and their versions stored within the database.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 GetPackagesWithInProgress ()
  Return packages that are currently in progress.
- Int32 GetPackageVersionLength (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision)
  Determines the length of a version of a package.
- String GetServerError ()
  Returns any errors associated with the server. Verifies that the connection to the database is valid and that the database schema is the correct version number.
- void LockPackage (Guid packageKey)
  Locks a package for editing stopping other users from making changes to the package.
- void UnlockPackage (Guid packageKey)
  Unlock a package to allow the package to be modified,
- InterfaceStatus_v1 VerifyInterface ()
  This method returns an InterfaceStatus_v1 if the current user can login to the current database.
- void WIPSaved (Guid packageKey)
  Sets the package as a Work In Progress package which is not deployable.

Detailed Description

This class contains legacy functionality to manage packages in the Management Server (version 8.1 or less) database.

Member Function Documentation

Guid DataAccessServices.PackageManagement.PackageManagement_v1.BeginPackageVersionDownload (Guid packageKey, *Int32* major, *Int32* minor, *Int32* build, *Int32* revision*)* `[inline]`

Begins a package download.

Parameters:

packageKey	The key which represents the package.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.
revision	The revision version of the package.

Returns:

A guid which represents a download key.

Guid DataAccessServices.PackageManagement.PackageManagement_v1.BeginPackageVersionUpload (Guid packageKey*, String* description*, Guid* packageVersionKey*, out DateTime* modifiedTime*, int* dataLength*) [inline]*

Begins an upload of a package version.

Parameters:

packageKey	The key which represents the package.
description	Description of the package.
packageVersionKey	The key which uniquely identifies this package.
modifiedTime	The time the package was last modified
dataLength	The length of data that will be uploaded.

Returns:

An upload key used to add data to the upload.

void DataAccessServices.PackageManagement.PackageManagement_v1.CommitPackageVersion (Guid packageVersionKey*) [inline]*

Once a package has been fully uploaded it must be committed, this clears the In Progress flag and allows the package to be deployed.

Parameters:

packageVersionKey	A package key guid which should be committed to the server.

Byte []
DataAccessServices.PackageManagement.PackageManagement_v1.ContinuePackageVersionDownload (Guid* downloadKey*, Int32* offset*, Int32* length*) `[inline]`

Continues a package download. The bytes of the package from 'offset' to 'offset + length' are returned.

Parameters:

downloadKey	The key which represents the download.
offset	The offset of the first byte to return.
length	The number of bytes to download.

Returns:

Bytes from the package.

void
DataAccessServices.PackageManagement.PackageManagement_v1.ContinuePackageVersionUpload (Guid* packageVersionKey*, ref DateTime* modifiedTime*, Guid* uploadKey*, Int32* offset*, Byte[]* data*) `[inline]`

Continues an upload of a package version.

Parameters:

packageVersionKey	The key which uniquely identifies this package.
modifiedTime	The time the package was last modified
uploadKey	The key which represents the upload.
offset	The offset of the bytes currently uploaded.
data	The data to be uploaded.

void DataAccessServices.PackageManagement.PackageManagement_v1.CreatePackage (Guid* key*, String* name*, String* company*, String* type*, PackageManagementWebServiceCode.Schemas.PackagePlatform_v1* platform*, Guid* productKey*, out DateTime* modifiedTime*) `[inline]`

Creates a configuration package within the database.

Parameters:

key	The Guid which identifies the package.
name	The display name of the package.
company	The company name.
type	The type of the package, i.e. Software, Configuration.
platform	The platform of the package.
productKey	The name of the product associated with this package.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v1.CreatePackageVersion (Guid* packageKey*, Guid* packageVersionKey*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*, Int32* creatorMajor*, Int32* creatorMinor*, Int32* creatorBuild*, Int32* creatorRevision*, Int32* dependentMinimumMajor*, Int32* dependentMinimumMinor*, Int32* dependentMinimumBuild*, Int32* dependentMinimumRevision*, Int32* dependentMaximumMajor*, Int32* dependentMaximumMinor*, Int32* dependentMaximumBuild*, Int32* dependentMaximumRevision*, string* description*, Boolean* supportsMidSessionUpdate*, out DateTime* modifiedTime*) `[inline]`

Creates a package version entry in the database ready to be uploaded.

Parameters:

packageKey	A guid of the package where the package version is to be created.
packageVersionKey	A guid for the new package version to be created.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.

revision	The revision version of the package.
creatorMajor	The major version of the console that created this package version (if the package is a configuration).
creatorMinor	The minor version of the console that created this package version (if the package is a configuration).
creatorBuild	The build version of the console that created this package version (if the package is a configuration).
creatorRevision	The revision version of the console that created this package version (if the package is a configuration).
dependentMinimumMajor	The major (Minimum version) of an agent that this configuration supports.
dependentMinimumMinor	The minor (Minimum version) of an agent that this configuration supports.
dependentMinimumBuild	The build (Minimum version) of an agent that this configuration supports.
dependentMinimumRevision	The revision (Minimum version) of an agent that this configuration supports.
dependentMaximumMajor	The major (Maximum version) of an agent that this configuration supports.
dependentMaximumMinor	The minor (Maximum version) of an agent that this configuration supports.
dependentMaximumBuild	The build (Maximum version) of an agent that this configuration supports.
dependentMaximumRevision	The revision (Maximum version) of an agent that this configuration supports.
description	A string describing the package version.
supportsMidSessionUpdate	A boolean which indicates if this package version supports mid-session installs (if package is agent MSI or MSP).
modifiedTime	The time that the package version was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v1.DeletePackage (Guid* key, *DateTime?* modifiedTime*) [inline]

Deletes an existing package from the database.

Parameters:

key	The key identifying the package that should be deleted.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v1.DeletePackageVersion (Guid* packageKey*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*, DateTime?* modifiedTime*) `[inline]`

Deletes a package version from a package.

Parameters:

packageKey	The key which identifies the package which contains the package version.
major	The major version number of the package to remove.
minor	The minor version number of the package to remove.
build	The build version number of the package to remove.
revision	The revision number of the package to remove.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v1.FinalisePackageVersion (Guid* packageVersionKey*) `[inline]`

Check that a package version has been correctly uploaded.

Parameters:

packageVersionKey	A guid identifying the package version to check.

PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v1.GetMachinesWithPackage (Guid* packageKey*) `[inline]`

Returns the machines that have a particular package installed.

Parameters:

packageKey	A package key that the machines in the MachinesDataSet_v1 should have installed.

Returns:

A MachinesDataSet_v1 that identifies machines that have a particular package installed.

***PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v1.GetPackageFromKey (Guid* key*)* `[inline]`**

Returns an individual package within the database.

Parameters:

key	The key that identifies the package to return.

Returns:

A data set describing the package and its versions.

Requires deployment or administrative access.

***PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v1.GetPackageFromKeyWithInProgress (Guid* key*)* `[inline]`**

Returns a package from the database that is currently in progress.

Parameters:

key	A guid of the package to be retrieved.

Returns:

A PackagesDataSet_v1 containing the in progress package

***PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v1.GetPackages ()* `[inline]`**

Returns all packages and their versions stored within the database.

Returns:

A data set consisting of all packages and versions.

Requires deployment or administrative access.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v1.GetPackagesWithInProgress () `[inline]`

Return packages that are currently in progress.

Returns:

A PackagesDataSet_v1 containing the in progress packages

Int32 DataAccessServices.PackageManagement.PackageManagement_v1.GetPackageVersionLength (Guid **packageKey,** ***Int32*** **major,** ***Int32*** **minor,** ***Int32*** **build,** ***Int32*** **revision***)*** `[inline]`

Determines the length of a version of a package.

Parameters:

packageKey	The key which represents the package.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.
revision	The revision version of the package.

Returns:

The length of data in bytes.

String DataAccessServices.PackageManagement.PackageManagement_v1.GetServerError () `[inline]`

Returns any errors associated with the server. Verifies that the connection to the database is valid and that the database schema is the correct version number.

Returns:

A String containing the server error

void DataAccessServices.PackageManagement.PackageManagement_v1.LockPackage (Guid **packageKey***) [inline]*

Locks a package for editing stopping other users from making changes to the package.

Parameters:

packageKey	The guid package key of the package to be locked.

void DataAccessServices.PackageManagement.PackageManagement_v1.UnlockPackage (Guid **packageKey***) [inline]*

Unlock a package to allow the package to be modified,

Parameters:

packageKey	A guid package key of the package to unlocked.

InterfaceStatus_v1 DataAccessServices.PackageManagement.PackageManagement_v1.VerifyInterface () [inline]

This method returns an InterfaceStatus_v1 if the current user can login to the current database.

Returns:

A InterfaceStatus_v1 enum describing the interface status

void DataAccessServices.PackageManagement.PackageManagement_v1.WIPSaved (Guid **packageKey***) [inline]*

Sets the package as a Work In Progress package which is not deployable.

Parameters:

packageKey	A guid for the package key to be set as WIP.

The documentation for this class was generated from the following file:

- PackageManagement_v1.cs

DataAccessServices.PackageManagement.PackageManagement_v2 Class Reference

This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.

Inheritance diagram for DataAccessServices.PackageManagement.PackageManagement_v2:

Public Member Functions

- Guid BeginPackageVersionDownload (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision)
  Begins a package download.
- Guid BeginPackageVersionUpload (Guid packageKey, String description, Guid packageVersionKey, out DateTime modifiedTime, int dataLength)
  Begins an upload of a package version.
- void CommitPackageVersion (Guid packageVersionKey)
  Once a package has been fully uploaded it must be committed, this clears the In Progress flag and allows the package to be deployed.
- Byte[] ContinuePackageVersionDownload (Guid downloadKey, Int32 offset, Int32 length)
  Continues a package download. The bytes of the package from 'offset' to 'offset + length' are returned.
- void ContinuePackageVersionUpload (Guid packageVersionKey, ref DateTime modifiedTime, Guid uploadKey, Int32 offset, Byte[] data)
  Continues an upload of a package version.
- void CreatePackage (Guid key, String company, String type, PackageManagementWebServiceCode.Schemas.PackagePlatform_v1 platform, Guid productKey, out DateTime modifiedTime)
  Creates a configuration package within the database.
- void CreatePackageVersion (Guid packageKey, Guid packageVersionKey, String name, Int32 major, Int32 minor, Int32 build, Int32 revision, Int32 creatorMajor, Int32 creatorMinor, Int32 creatorBuild, Int32 creatorRevision, Int32 dependentMinimumMajor, Int32 dependentMinimumMinor, Int32 dependentMinimumBuild, Int32 dependentMinimumRevision, Int32 dependentMaximumMajor, Int32 dependentMaximumMinor, Int32 dependentMaximumBuild, Int32 dependentMaximumRevision, string description, Boolean supportsMidSessionUpdate, out DateTime modifiedTime)
  Creates a package version entry in the database ready to be uploaded.
- void DeletePackage (Guid key, DateTime?modifiedTime)
  Deletes an existing package from the database.
- void DeletePackageVersion (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision, DateTime?modifiedTime)
  Deletes a package version from a package.
- void FinalisePackageVersion (Guid packageVersionKey)
  Check that a package version has been correctly uploaded.

- PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1 GetMachinesWithPackage (Guid packageKey)
  Returns the machines that have a particular package installed.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackageFromKey (Guid key)
  Returns an individual package within the database.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackageFromKeyWithInProgress (Guid key)
  Returns a package from the database that is currently in progress.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackages ()
  Returns all packages and their versions stored within the database.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackagesWithInProgress ()
  Return packages that are currently in progress.
- Int32 GetPackageVersionLength (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision)
  Determines the length of a version of a package.
- String GetServerError ()
  Returns any errors associated with the server. Verifies that the connection to the database is valid and that the database schema is the correct version number.
- void LockPackage (Guid packageKey)
  Locks a package for editing stopping other users from making changes to the package.
- void UnlockPackage (Guid packageKey)
  Unlock a package to allow the package to be modified,
- InterfaceStatus_v1 VerifyInterface ()
  This method returns an InterfaceStatus_v1 if the current user can login to the current database.
- void WIPSaved (Guid packageKey)
  Sets the package as a Work In Progress package which is not deployable.

Detailed Description

This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.

Member Function Documentation

Guid DataAccessServices.PackageManagement.PackageManagement_v2.BeginPackageVersionDownload (Guid packageKey, *Int32* major, *Int32* minor, *Int32* build, *Int32* revision*)* `[inline]`

Begins a package download.

Parameters:

packageKey	The key which represents the package.

major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.
revision	The revision version of the package.

Returns:

A guid which represents a download key.

Guid DataAccessServices.PackageManagement.PackageManagement_v2.BeginPackageVersionUpload (Guid* packageKey*, String* description*, Guid* packageVersionKey*, out DateTime* modifiedTime*, int* dataLength*) [inline]

Begins an upload of a package version.

Parameters:

packageKey	The key which represents the package.
description	Description of the package.
packageVersionKey	The key which uniquely identifies this package.
revision	The revision version of the package.
modifiedTime	The time the package was last modified
dataLength	The length of data that will be uploaded.

Returns:

An upload key used to add data to the upload.

void DataAccessServices.PackageManagement.PackageManagement_v2.CommitPackageVersion (Guid* packageVersionKey*) [inline]

Once a package has been fully uploaded it must be committed, this clears the In Progress flag and allows the package to be deployed.

Parameters:

packageVersionKey	A package key guid which should be committed to the server.

Byte []
DataAccessServices.PackageManagement.PackageManagement_v2.ContinuePackageVersionDownload (Guid* downloadKey*, Int32* offset*, Int32* length*) `[inline]`

Continues a package download. The bytes of the package from 'offset' to 'offset + length' are returned.

Parameters:

downloadKey	The key which represents the download.
offset	The offset of the first byte to return.
length	The number of bytes to download.

Returns:

Bytes from the package.

void
DataAccessServices.PackageManagement.PackageManagement_v2.ContinuePackageVersionUpload (Guid* packageVersionKey*, ref DateTime* modifiedTime*, Guid* uploadKey*, Int32* offset*, Byte[]* data*) `[inline]`

Continues an upload of a package version.

Parameters:

packageVersionKey	The key which uniquely identifies this package.
modifiedTime	The time the package was last modified
uploadKey	The key which represents the upload.
offset	The offset of the bytes currently uploaded.
data	The data to be uploaded.

***void DataAccessServices.PackageManagement.PackageManagement_v2.CreatePackage (Guid* key*, String* company*, String* type*, PackageManagementWebServiceCode.Schemas.PackagePlatform_v1* platform*, Guid* productKey*, out DateTime* modifiedTime*)* `[inline]`**

Creates a configuration package within the database.

Parameters:

key	The Guid which identifies the package.
company	The company name.
type	The type of the package, i.e. Software, Configuration.
platform	The platform of the package.
productKey	The name of the product associated with this package.
modifiedTime	The time the package was last modified.

***void DataAccessServices.PackageManagement.PackageManagement_v2.CreatePackageVersion (Guid* packageKey*, Guid* packageVersionKey*, String* name*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*, Int32* creatorMajor*, Int32* creatorMinor*, Int32* creatorBuild*, Int32* creatorRevision*, Int32* dependentMinimumMajor*, Int32* dependentMinimumMinor*, Int32* dependentMinimumBuild*, Int32* dependentMinimumRevision*, Int32* dependentMaximumMajor*, Int32* dependentMaximumMinor*, Int32* dependentMaximumBuild*, Int32* dependentMaximumRevision*, string* description*, Boolean* supportsMidSessionUpdate*, out DateTime* modifiedTime*)* `[inline]`**

Creates a package version entry in the database ready to be uploaded.

Parameters:

packageKey	A guid of the package where the package version is to be created.
packageVersionKey	A guid for the new package version to be created.
name	The name of the package.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.

revision	The revision version of the package.
creatorMajor	The major version of the console that created this package version (if the package is a configuration).
creatorMinor	The minor version of the console that created this package version (if the package is a configuration).
creatorBuild	The build version of the console that created this package version (if the package is a configuration).
creatorRevision	The revision version of the console that created this package version (if the package is a configuration).
dependentMinimumMajor	The major (Minimum version) of an agent that this configuration supports.
dependentMinimumMinor	The minor (Minimum version) of an agent that this configuration supports.
dependentMinimumBuild	The build (Minimum version) of an agent that this configuration supports.
dependentMinimumRevision	The revision (Minimum version) of an agent that this configuration supports.
dependentMaximumMajor	The major (Maximum version) of an agent that this configuration supports.
dependentMaximumMinor	The minor (Maximum version) of an agent that this configuration supports.
dependentMaximumBuild	The build (Maximum version) of an agent that this configuration supports.
dependentMaximumRevision	The revision (Maximum version) of an agent that this configuration supports.
description	A string describing the package version.
supportsMidSessionUpdate	A boolean which indicates if this package version supports mid-session installs (if package is agent MSI or MSP).
modifiedTime	The time that the package version was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v2.DeletePackage (Guid* key, *DateTime?* modifiedTime*) [inline]

Deletes an existing package from the database.

Parameters:

key	The key identifying the package that should be deleted.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v2.DeletePackageVersion (Guid* packageKey*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*, DateTime?* modifiedTime*) `[inline]`

Deletes a package version from a package.

Parameters:

packageKey	The key which identifies the package which contains the package version.
major	The major version number of the package to remove.
minor	The minor version number of the package to remove.
build	The build version number of the package to remove.
revision	The revision number of the package to remove.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v2.FinalisePackageVersion (Guid* packageVersionKey*) `[inline]`

Check that a package version has been correctly uploaded.

Parameters:

packageVersionKey	A guid which is the package version which should be checked.

PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v2.GetMachinesWithPackage (Guid* packageKey*) `[inline]`

Returns the machines that have a particular package installed.

Parameters:

packageKey	A package key that the machines in the MachinesDataSet_v1 should have installed.

Returns:

A MachinesDataSet_v1 that identifies machines that have a particular package installed.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v2.GetPackageFromKey (Guid* key*) ***`[inline]`***

Returns an individual package within the database.

Parameters:

key	The key that identifies the package to return.

Returns:

A data set describing the package and its versions.

Requires deployment or administrative access.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v2.GetPackageFromKeyWithInProgress (Guid* key*) ***`[inline]`***

Returns a package from the database that is currently in progress.

Parameters:

key	A guid of the package to be retrieved.

Returns:

A PackagesDataSet_v1 containing the in progress package

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v2.GetPackages () ***`[inline]`***

Returns all packages and their versions stored within the database.

Returns:

A data set consisting of all packages and versions.

Requires deployment or administrative access.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v2.GetPackagesWithInProgress () [inline]

Return packages that are currently in progress.

Returns:

A PackagesDataSet_v1 containing the in progress packages

Int32 DataAccessServices.PackageManagement.PackageManagement_v2.GetPackageVersionLength (Guid* packageKey*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*) [inline]

Determines the length of a version of a package.

Parameters:

packageKey	The key which represents the package.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.
revision	The revision version of the package.

Returns:

The length of data in bytes.

String DataAccessServices.PackageManagement.PackageManagement_v2.GetServerError () [inline]

Returns any errors associated with the server. Verifies that the connection to the database is valid and that the database schema is the correct version number.

void DataAccessServices.PackageManagement.PackageManagement_v2.LockPackage (Guid* packageKey*) [inline]

Locks a package for editing stopping other users from making changes to the package.

Parameters:

packageKey	The guid package key of the package to be locked.

void DataAccessServices.PackageManagement.PackageManagement_v2.UnlockPackage (Guid* packageKey*) `[inline]`

Unlock a package to allow the package to be modified,

Parameters:

packageKey	A guid package key of the package to unlocked.

InterfaceStatus_v1 DataAccessServices.PackageManagement.PackageManagement_v2.VerifyInterface () `[inline]`

This method returns an InterfaceStatus_v1 if the current user can login to the current database.

Returns:

A InterfaceStatus_v1 enum describing the interface status

void DataAccessServices.PackageManagement.PackageManagement_v2.WIPSaved (Guid* packageKey*) `[inline]`

Sets the package as a Work In Progress package which is not deployable.

Parameters:

packageKey	A guid for the package key to be set as WIP.

The documentation for this class was generated from the following file:

- PackageManagement_v2.cs

DataAccessServices.PackageManagement.PackageManagement_v3 Class Reference

This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.

Inheritance diagram for DataAccessServices.PackageManagement.PackageManagement_v3:

Public Member Functions

- Guid BeginPackageVersionDownload (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision)
  Begins a package download.
- Guid BeginPackageVersionUpload (Guid packageKey, String description, Guid packageVersionKey, out DateTime modifiedTime, int dataLength)
  Begins an upload of a package version.
- void CommitPackageVersion (Guid packageVersionKey)
  Once a package has been fully uploaded it must be committed, this clears the In Progress flag and allows the package to be deployed.
- Byte[] ContinuePackageVersionDownload (Guid downloadKey, Int32 offset, Int32 length)
  Continues a package download. The bytes of the package from 'offset' to 'offset + length' are returned.
- void ContinuePackageVersionUpload (Guid packageVersionKey, ref DateTime modifiedTime, Guid uploadKey, Int32 offset, Byte[] data)
  Continues an upload of a package version.
- void CreatePackage (Guid key, String company, String type, PackageManagementWebServiceCode.Schemas.PackagePlatform_v1 platform, Guid productKey, out DateTime modifiedTime)
  Creates a configuration package within the database.
- void CreatePackageVersion (Guid packageKey, Guid packageVersionKey, String name, Int32 major, Int32 minor, Int32 build, Int32 revision, string marketingVersion, Int32 creatorMajor, Int32 creatorMinor, Int32 creatorBuild, Int32 creatorRevision, Int32 dependentMinimumMajor, Int32 dependentMinimumMinor, Int32 dependentMinimumBuild, Int32 dependentMinimumRevision, Int32 dependentMaximumMajor, Int32 dependentMaximumMinor, Int32 dependentMaximumBuild, Int32 dependentMaximumRevision, string description, Boolean supportsMidSessionUpdate, out DateTime modifiedTime)
  Creates a package version entry in the database ready to be uploaded.
- void DeletePackage (Guid key, DateTime?modifiedTime)
  Deletes an existing package from the database.
- void DeletePackageVersion (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision, DateTime?modifiedTime)
  Deletes a package version from a package.
- void FinalisePackageVersion (Guid packageVersionKey)
  Check that a package version has been correctly uploaded.

- PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1 GetMachinesWithPackage (Guid packageKey)
  Returns the machines that have a particular package installed.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackageFromKey (Guid key)
  Returns an individual package within the database.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackageFromKeyWithInProgress (Guid key)
  Returns a package from the database that is currently in progress.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackages ()
  Returns all packages and their versions stored within the database.
- PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 GetPackagesWithInProgress ()
  Return packages that are currently in progress.
- Int32 GetPackageVersionLength (Guid packageKey, Int32 major, Int32 minor, Int32 build, Int32 revision)
  Determines the length of a version of a package.
- String GetServerError ()
  Returns any errors associated with the server. Verifies that the connection to the database is valid and that the database schema is the correct version number.
- void LockPackage (Guid packageKey)
  Locks a package for editing stopping other users from making changes to the package.
- void UnlockPackage (Guid packageKey)
  Unlock a package to allow the package to be modified,
- InterfaceStatus_v1 VerifyInterface ()
  This method returns an InterfaceStatus_v1 if the current user can login to the current database.
- void WIPSaved (Guid packageKey)
  Sets the package as a Work In Progress package which is not deployable.

Detailed Description

This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.

Member Function Documentation

Guid DataAccessServices.PackageManagement.PackageManagement_v3.BeginPackageVersionDownload (Guid packageKey, *Int32* major, *Int32* minor, *Int32* build, *Int32* revision*)* `[inline]`

Begins a package download.

Parameters:

packageKey	The key which represents the package.

major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.
revision	The revision version of the package.

Returns:

A guid which represents a download key.

Guid DataAccessServices.PackageManagement.PackageManagement_v3.BeginPackageVersionUpload (Guid* packageKey*, String* description*, Guid* packageVersionKey*, out DateTime* modifiedTime*, int* dataLength*) [inline]

Begins an upload of a package version.

Parameters:

packageKey	The key which represents the package.
description	Description of the package.
packageVersionKey	The key which uniquely identifies this package.
revision	The revision version of the package.
modifiedTime	The time the package was last modified
dataLength	The length of data that will be uploaded.

Returns:

An upload key used to add data to the upload.

void DataAccessServices.PackageManagement.PackageManagement_v3.CommitPackageVersion (Guid* packageVersionKey*) [inline]

Once a package has been fully uploaded it must be committed, this clears the In Progress flag and allows the package to be deployed.

Parameters:

packageVersionKey	A package key guid which should be committed to the server.

Byte []
DataAccessServices.PackageManagement.PackageManagement_v3.ContinuePackageVersionDownload (Guid* downloadKey*, Int32* offset*, Int32* length*) `[inline]`

Continues a package download. The bytes of the package from 'offset' to 'offset + length' are returned.

Parameters:

downloadKey	The key which represents the download.
offset	The offset of the first byte to return.
length	The number of bytes to download.

Returns:

Bytes from the package.

void
DataAccessServices.PackageManagement.PackageManagement_v3.ContinuePackageVersionUpload (Guid* packageVersionKey*, ref DateTime* modifiedTime*, Guid* uploadKey*, Int32* offset*, Byte[]* data*) `[inline]`

Continues an upload of a package version.

Parameters:

packageVersionKey	The key which uniquely identifies this package.
modifiedTime	The time the package was last modified
uploadKey	The key which represents the upload.
offset	The offset of the bytes currently uploaded.
data	The data to be uploaded.

***void DataAccessServices.PackageManagement.PackageManagement_v3.CreatePackage (Guid* key*, String* company*, String* type*, PackageManagementWebServiceCode.Schemas.PackagePlatform_v1* platform*, Guid* productKey*, out DateTime* modifiedTime*)* `[inline]`**

Creates a configuration package within the database.

Parameters:

key	The Guid which identifies the package.
company	The company name.
type	The type of the package, i.e. Software, Configuration.
platform	The platform of the package.
productKey	The name of the product associated with this package.
modifiedTime	The time the package was last modified.

***void DataAccessServices.PackageManagement.PackageManagement_v3.CreatePackageVersion (Guid* packageKey*, Guid* packageVersionKey*, String* name*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*, string* marketingVersion*, Int32* creatorMajor*, Int32* creatorMinor*, Int32* creatorBuild*, Int32* creatorRevision*, Int32* dependentMinimumMajor*, Int32* dependentMinimumMinor*, Int32* dependentMinimumBuild*, Int32* dependentMinimumRevision*, Int32* dependentMaximumMajor*, Int32* dependentMaximumMinor*, Int32* dependentMaximumBuild*, Int32* dependentMaximumRevision*, string* description*, Boolean* supportsMidSessionUpdate*, out DateTime* modifiedTime*)* `[inline]`**

Creates a package version entry in the database ready to be uploaded.

Parameters:

packageKey	A guid of the package where the package version is to be created.
packageVersionKey	A guid for the new package version to be created.
name	The name of the package.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.

revision	The revision version of the package.
creatorMajor	The major version of the console that created this package version (if the package is a configuration).
creatorMinor	The minor version of the console that created this package version (if the package is a configuration).
creatorBuild	The build version of the console that created this package version (if the package is a configuration).
creatorRevision	The revision version of the console that created this package version (if the package is a configuration).
dependentMinimumMajor	The major (Minimum version) of an agent that this configuration supports.
dependentMinimumMinor	The minor (Minimum version) of an agent that this configuration supports.
dependentMinimumBuild	The build (Minimum version) of an agent that this configuration supports.
dependentMinimumRevision	The revision (Minimum version) of an agent that this configuration supports.
dependentMaximumMajor	The major (Maximum version) of an agent that this configuration supports.
dependentMaximumMinor	The minor (Maximum version) of an agent that this configuration supports.
dependentMaximumBuild	The build (Maximum version) of an agent that this configuration supports.
dependentMaximumRevision	The revision (Maximum version) of an agent that this configuration supports.
description	A string describing the package version.
supportsMidSessionUpdate	A boolean which indicates if this package version supports mid-session installs (if package is agent MSI or MSP).
modifiedTime	The time that the package version was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v3.DeletePackage (Guid* key, *DateTime?* modifiedTime*) [inline]

Deletes an existing package from the database.

Parameters:

key	The key identifying the package that should be deleted.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v3.DeletePackageVersion (Guid* packageKey*, Int32* major*, Int32* minor*, Int32* build*, Int32* revision*, DateTime?* modifiedTime*) `[inline]`

Deletes a package version from a package.

Parameters:

packageKey	The key which identifies the package which contains the package version.
major	The major version number of the package to remove.
minor	The minor version number of the package to remove.
build	The build version number of the package to remove.
revision	The revision number of the package to remove.
modifiedTime	The time the package was last modified.

void DataAccessServices.PackageManagement.PackageManagement_v3.FinalisePackageVersion (Guid* packageVersionKey*) `[inline]`

Check that a package version has been correctly uploaded.

Parameters:

packageVersionKey	A guid which is the package version which should be checked.

PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1 DataAccessServices.PackageManagement.PackageManagement_v3.GetMachinesWithPackage (Guid* packageKey*) `[inline]`

Returns the machines that have a particular package installed.

Parameters:

packageKey	A package key that the machines in the MachinesDataSet_v1 should have installed.

Returns:

A MachinesDataSet_v1 that identifies machines that have a particular package installed.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v3.GetPackageFromKey (Guid* key*) ***`[inline]`***

Returns an individual package within the database.

Parameters:

key	The key that identifies the package to return.

Returns:

A data set describing the package and its versions.

Requires deployment or administrative access.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v3.GetPackageFromKeyWithInProgress (Guid* key*) ***`[inline]`***

Returns a package from the database that is currently in progress.

Parameters:

key	A guid of the package to be retrieved.

Returns:

A PackagesDataSet_v1 containing the in progress package

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v3.GetPackages () ***`[inline]`***

Returns all packages and their versions stored within the database.

Returns:

A data set consisting of all packages and versions.

Requires deployment or administrative access.

PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2 DataAccessServices.PackageManagement.PackageManagement_v3.GetPackagesWithInProgress () `[inline]`

Return packages that are currently in progress.

Returns:

A PackagesDataSet_v1 containing the in progress packages

Int32 DataAccessServices.PackageManagement.PackageManagement_v3.GetPackageVersionLength (Guid **packageKey,** ***Int32*** **major,** ***Int32*** **minor,** ***Int32*** **build,** ***Int32*** **revision)** `[inline]`

Determines the length of a version of a package.

Parameters:

packageKey	The key which represents the package.
major	The major version of the package.
minor	The minor version of the package.
build	The build version of the package.
revision	The revision version of the package.

Returns:

The length of data in bytes.

String DataAccessServices.PackageManagement.PackageManagement_v3.GetServerError () `[inline]`

Returns any errors associated with the server. Verifies that the connection to the database is valid and that the database schema is the correct version number.

void DataAccessServices.PackageManagement.PackageManagement_v3.LockPackage (Guid **packageKey)** `[inline]`

Locks a package for editing stopping other users from making changes to the package.

Parameters:

packageKey	The guid package key of the package to be locked.

void DataAccessServices.PackageManagement.PackageManagement_v3.UnlockPackage (Guid* packageKey*) `[inline]`

Unlock a package to allow the package to be modified,

Parameters:

packageKey	A guid package key of the package to unlocked.

InterfaceStatus_v1 DataAccessServices.PackageManagement.PackageManagement_v3.VerifyInterface () `[inline]`

This method returns an InterfaceStatus_v1 if the current user can login to the current database.

Returns:

A InterfaceStatus_v1 enum describing the interface status

void DataAccessServices.PackageManagement.PackageManagement_v3.WIPSaved (Guid* packageKey*) `[inline]`

Sets the package as a Work In Progress package which is not deployable.

Parameters:

packageKey	A guid for the package key to be set as WIP.

The documentation for this class was generated from the following file:

- PackageManagement_v3.cs

DataAccessServices.PackageManagement.RowFixerNeeded Class Reference

Inheritance diagram for DataAccessServices.PackageManagement.RowFixerNeeded:

Public Member Functions

- **RowFixerNeeded** (DataTable targetTable, DataColumn targetColumn, DataTable sourceTable, DataColumn sourceColumn)

The documentation for this class was generated from the following file:

- TableCopier.cs

DataAccessServices.PackageManagement.TableCopier Class Reference

Public Member Functions

- delegate void **FixUpRow** (System.Data.DataRow targetRow, System.Data.DataRow sourceRow)

Static Public Member Functions

- static bool **CanCopy** (DataColumn targetColumn, DataColumn sourceColumn)
- static void **CopyTable** (System.Data.DataTable sourceTable, System.Data.DataTable targetTable, FixUpRow rowFixer)
- static bool **HasMaxLength** (System.Data.DataColumn col)

The documentation for this class was generated from the following file:

- TableCopier.cs