Table of Contents

Table of contents ........................................................................................................... 2
Disclaimer .......................................................................................................................... 3
Introduction ....................................................................................................................... 4
 Management Console ..................................................................................................... 4
 Proxy DLL ......................................................................................................................... 4
 Authorization & Security ................................................................................................. 4
 Limitations ....................................................................................................................... 4

How to use the Package Management WebServices - Examples .................................. 5
 Package Management Endpoint Tasks .......................................................................... 5
 Connecting to the PackageManagement Endpoint ..................................................... 5
 Retrieving Package Details ......................................................................................... 6
 Uploading/Downloading Packages ................................................................................. 6

DataRows ......................................................................................................................... 8
 Alerts ................................................................................................................................. 8
 AlertsRow ......................................................................................................................... 8

DataRows - Machines ..................................................................................................... 9
 Machines ......................................................................................................................... 9
 MachinesRow .................................................................................................................. 9
 MachinePackagesRow ..................................................................................................... 9
 MatchResultsRow ........................................................................................................... 10
 MachineDetailsRow ....................................................................................................... 11
 MachineDiagnosticsRow ............................................................................................... 11

DataRows - Packages ................................................................................................... 12
 Packages ........................................................................................................................ 12
 PackagesRow ................................................................................................................. 12
 PackageVersionsRow .................................................................................................... 12
 PatchesRow .................................................................................................................... 13
 PrerequisitesRow ........................................................................................................... 14
 PackageVersionPrerequisitesRow ............................................................................... 15
 PrerequisiteResourceRow ............................................................................................ 15
 PrerequisiteCommandRow ........................................................................................... 15
 PrerequisiteCheckRow .................................................................................................. 16
 PrerequisiteExitCodeRow .............................................................................................. 16

Namespace Index ........................................................................................................... 17
 Hierarchical Index ......................................................................................................... 18
 Class Index ..................................................................................................................... 19

DataAccessServices .................................................................................................... 20
 DataAccessServices.PackageManagement ..................................................................... 21

Class Documentation ................................................................................................... 22

 DataAccessServices.PackageManagement.TableCopier ......................................... 48

Index .............................................................................................................................. 49
# Table of contents

1.  [Disclaimer](#)
2.  [Introduction](#)
3.  [Examples - How To](#)
4.  [DataRows - Machines](#)
5.  [DataRows - Packages](#)
Disclaimer
Copyright © 2020, Ivanti. All rights reserved.
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

To logon as the current user:

```powershell
# Load proxy DLL
Add-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
```

To specify a user to logon as:

```powershell
# Load proxy DLL
Add-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
Username = "MyUsername";    Password = "MyPassword";    Domain = "MyDomain";

# Create connection to the Management Server
```
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

# 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:

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

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

Note: Ensure that PackageGuid and PackageVersionGuid are replaced with the appropriate UpgradeCode and ProductCode from the specified MSI.

```

# Environment Manager product
$EnvironmentManagerKey = "EA13F465-B2B6-4716-ADA4-53D7D84D042F"
```
Both endpoints support block by block upload operations for large files by repeatedly calling `ContinuePackageVersionUpload`. This is recommended for large configurations.
Datarows

Alerts

AlertsRow
## 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.

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineKey</td>
<td>Guid</td>
<td>Unique identifier relating to machine</td>
</tr>
<tr>
<td>GroupKey</td>
<td>Guid</td>
<td>Unique identifier relating to group</td>
</tr>
<tr>
<td>GroupName</td>
<td>String</td>
<td>Name of group</td>
</tr>
<tr>
<td>Platform</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>NetBIOSName</td>
<td>String</td>
<td>Name of net bios</td>
</tr>
<tr>
<td>DistinguishedName</td>
<td>String</td>
<td>Name of distinguished</td>
</tr>
<tr>
<td>OldDistinguishedName</td>
<td>String</td>
<td>Name of old distinguished</td>
</tr>
<tr>
<td>ObjectGuid</td>
<td>Guid</td>
<td></td>
</tr>
<tr>
<td>LastPollTime</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastPollStatus</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>LastUploadTime</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastUploadStatus</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>IsPendingDeletion</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>AlertCount</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>ModifiedGroupTime</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>DiagnosticsError</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>DiagnosticsState</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>DiagnosticsTime</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>Deployed</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>DeployError</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>Offline</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>LastResponseSeconds</td>
<td>Int32</td>
<td></td>
</tr>
</tbody>
</table>

#### 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.

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineKey</td>
<td>Guid</td>
<td>Unique identifier relating to machine</td>
</tr>
<tr>
<td>PackageKey</td>
<td>Guid</td>
<td>Unique identifier relating to package</td>
</tr>
<tr>
<td>Major</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Minor</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Build</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Revision</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of machine package</td>
</tr>
<tr>
<td>Company</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ProductName</td>
<td>String</td>
<td>Name of product</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>Status</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>StatusMessage</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>ChildStatus</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ChildMajor</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ChildMinor</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ChildBuild</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ChildRevision</td>
<td>Int32</td>
<td></td>
</tr>
</tbody>
</table>

**MatchResultsRow**

Represents a match result on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupKey</td>
<td>Guid</td>
<td>Unique identifier relating to group</td>
</tr>
<tr>
<td>GroupName</td>
<td>String</td>
<td>Name of group</td>
</tr>
<tr>
<td>MatchName</td>
<td>String</td>
<td>Name of match</td>
</tr>
<tr>
<td>Difference</td>
<td>Int32</td>
<td></td>
</tr>
</tbody>
</table>
**MachineDetailsRow**
Stores a collection of name \ value pairs containing the machines details such as OS, memory, cpu platform 32 or 64 bit etc.

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineKey</td>
<td>Guid</td>
<td>Unique identifier relating to machine</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of machine detail</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineKey</td>
<td>Guid</td>
<td>Unique identifier relating to machine</td>
</tr>
<tr>
<td>ServerUrl</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of machine diagnostic</td>
</tr>
<tr>
<td>Error</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LatestName</td>
<td>String</td>
<td>Name of latest</td>
</tr>
<tr>
<td>PackageKey</td>
<td>Guid</td>
<td>UpgradeCode property defined in associated MSI files</td>
</tr>
<tr>
<td>Company</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ProductKey</td>
<td>Guid</td>
<td>Unique identifier relating to product</td>
</tr>
<tr>
<td>ProductName</td>
<td>String</td>
<td>Name of product</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>OwnerSid</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>PolicyKey</td>
<td>Guid</td>
<td>Unique identifier relating to policy</td>
</tr>
<tr>
<td>Locked</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>LockedUserName</td>
<td>String</td>
<td>Name of locked user</td>
</tr>
<tr>
<td>LatestMajor</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>LatestMinor</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>LatestBuild</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>LatestRevision</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>SecurityDescriptor</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Dirty</td>
<td>Byte</td>
<td></td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageVersionKey</td>
<td>Guid</td>
<td>ProductCode property defined in associated MSI files</td>
</tr>
<tr>
<td>PackageKey</td>
<td>Guid</td>
<td>UpgradeCode property defined in associated MSI files</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of package version</td>
</tr>
<tr>
<td>Major</td>
<td>Int32</td>
<td>Major version of package</td>
</tr>
<tr>
<td>Minor</td>
<td>Int32</td>
<td>Minor version of package</td>
</tr>
<tr>
<td>Build</td>
<td>Int32</td>
<td>Build version of package</td>
</tr>
<tr>
<td>Revision</td>
<td>Int32</td>
<td>Revision version of package</td>
</tr>
<tr>
<td>DataLength</td>
<td>Int32</td>
<td>Size of package</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>InProgress</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>UserName</td>
<td>String</td>
<td>Name of user</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>Description of package version</td>
</tr>
<tr>
<td>CreatorMajor</td>
<td>Int32</td>
<td>Major version of package creator (console)</td>
</tr>
<tr>
<td>CreatorMinor</td>
<td>Int32</td>
<td>Minor version of package creator (console)</td>
</tr>
<tr>
<td>CreatorBuild</td>
<td>Int32</td>
<td>Build version of package creator (console)</td>
</tr>
<tr>
<td>CreatorRevision</td>
<td>Int32</td>
<td>Revision version of package creator (console)</td>
</tr>
<tr>
<td>DependentMinimumMajor</td>
<td>Int32</td>
<td>Major version of minimum associated agent</td>
</tr>
<tr>
<td>DependentMinimumMinor</td>
<td>Int32</td>
<td>Minor version of minimum associated agent</td>
</tr>
<tr>
<td>DependentMinimumBuild</td>
<td>Int32</td>
<td>Build version of minimum associated agent</td>
</tr>
<tr>
<td>DependentMinimumRevision</td>
<td>Int32</td>
<td>Revision version of minimum associated agent</td>
</tr>
<tr>
<td>DependentMaximumMajor</td>
<td>Int32</td>
<td>Major version of maximum associated agent</td>
</tr>
<tr>
<td>DependentMaximumMinor</td>
<td>Int32</td>
<td>Minor version of maximum associated agent</td>
</tr>
<tr>
<td>DependentMaximumBuild</td>
<td>Int32</td>
<td>Build version of maximum associated agent</td>
</tr>
<tr>
<td>DependentMaximumRevision</td>
<td>Int32</td>
<td>Revision version of maximum associated agent</td>
</tr>
<tr>
<td>PackagesRow</td>
<td>PackagesRow</td>
<td></td>
</tr>
</tbody>
</table>

**PatchesRow**

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

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
</table>

### PatchKey
The unique identifier of the patch

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatchKey</td>
<td>Guid</td>
<td>The unique identifier of the patch</td>
</tr>
<tr>
<td>PackageVersionKey</td>
<td>Guid</td>
<td>The package version that this patch applies to</td>
</tr>
<tr>
<td>PatchCode</td>
<td>Guid</td>
<td>The Patch Code property of the Windows Installer MSP file</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of the patch</td>
</tr>
<tr>
<td>Major</td>
<td>Int32</td>
<td>Major version of package once this patch is applied</td>
</tr>
<tr>
<td>Minor</td>
<td>Int32</td>
<td>Minor version of package once this patch is applied</td>
</tr>
<tr>
<td>Build</td>
<td>Int32</td>
<td>Build version of package once this patch is applied</td>
</tr>
<tr>
<td>Revision</td>
<td>Int32</td>
<td>Revision version of package once this patch is applied</td>
</tr>
<tr>
<td>DataLength</td>
<td>Int32</td>
<td>Size of patch</td>
</tr>
<tr>
<td>InProgress</td>
<td>Boolean</td>
<td>True whenever this patch is being updated</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>TargetMajor</td>
<td>Int32</td>
<td>The version of the patch or package that this patch applies to</td>
</tr>
<tr>
<td>TargetMinor</td>
<td>Int32</td>
<td>The version of the patch or package that this patch applies to</td>
</tr>
<tr>
<td>TargetBuild</td>
<td>Int32</td>
<td>The version of the patch or package that this patch applies to</td>
</tr>
<tr>
<td>TargetRevision</td>
<td>Int32</td>
<td>The version of the patch or package that this patch applies to</td>
</tr>
<tr>
<td>ValidationFlags</td>
<td>Int32</td>
<td>The Validation Flags property of the Windows Installer MSP file</td>
</tr>
</tbody>
</table>

### PrerequisitesRow
Represents a prerequisite on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrerequisitesKey</td>
<td>Guid</td>
<td>Unique identifier relating to prerequisites</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of prerequisite</td>
</tr>
<tr>
<td>Version</td>
<td>Int32</td>
<td>Name of prerequisite</td>
</tr>
<tr>
<td>PlatformInfo</td>
<td>String</td>
<td>Name of prerequisite</td>
</tr>
<tr>
<td>VersionInfo</td>
<td>String</td>
<td>Name of prerequisite</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>Column</td>
<td>DataType</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
</tbody>
</table>

**PackageVersionPrerequisitesRow**

Represents a package version prerequisite on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageVersionKey</td>
<td>Guid</td>
<td>Unique identifier relating to package version</td>
</tr>
<tr>
<td>PrerequisitesKey</td>
<td>Guid</td>
<td>Unique identifier relating to prerequisites</td>
</tr>
<tr>
<td>PrerequisitesRow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PrerequisiteResourceRow**

Represents a prerequisite resource on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceKey</td>
<td>Guid</td>
<td>Unique identifier relating to resource</td>
</tr>
<tr>
<td>PrerequisiteKey</td>
<td>Guid</td>
<td>Unique identifier relating to prerequisite</td>
</tr>
<tr>
<td>Destination</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>DataLength</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>HashCode</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>PrerequisitesRow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PrerequisiteCommandRow**

Represents a prerequisite command on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommandKey</td>
<td>Guid</td>
<td>Unique identifier relating to command</td>
</tr>
<tr>
<td>PrerequisiteKey</td>
<td>Guid</td>
<td>Unique identifier relating to prerequisite</td>
</tr>
<tr>
<td>Action</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Arguments</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>DefaultResult</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>PrerequisitesRow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PrerequisiteCheckRow

Represents a prerequisite check on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrerequisiteCheckKey</td>
<td>Guid</td>
<td>Unique identifier relating to prerequisite check</td>
</tr>
<tr>
<td>PrerequisiteKey</td>
<td>Guid</td>
<td>Unique identifier relating to prerequisite</td>
</tr>
<tr>
<td>CheckType</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>OperatorValue</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Product_ProductCode</td>
<td>Guid</td>
<td></td>
</tr>
<tr>
<td>Product_UpgradeCode</td>
<td>Guid</td>
<td></td>
</tr>
<tr>
<td>File_Path</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Registry_Root</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Registry_Key</td>
<td>String</td>
<td>Unique identifier relating to registry</td>
</tr>
<tr>
<td>Registry_Value</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>OperatingSystem_Message</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>PrerequisitesRow</td>
<td>PrerequisitesRow</td>
<td></td>
</tr>
</tbody>
</table>

### PrerequisiteExitCodeRow

Represents a prerequisite exit code on the server

<table>
<thead>
<tr>
<th>Column</th>
<th>DataType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExitCodeKey</td>
<td>Guid</td>
<td>Unique identifier relating to exit code</td>
</tr>
<tr>
<td>Value</td>
<td>Int32</td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>CommandKey</td>
<td>Guid</td>
<td>Unique identifier relating to command</td>
</tr>
<tr>
<td>ModifiedTime</td>
<td>DateTime</td>
<td>Time modified</td>
</tr>
<tr>
<td>CreationTime</td>
<td>DateTime</td>
<td>Time created</td>
</tr>
<tr>
<td>PrerequisiteCommandRow</td>
<td>PrerequisiteCommandRow</td>
<td></td>
</tr>
</tbody>
</table>
Namespace Index

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

- **DataAccessServices** .................................................................................................................. 20
- **DataAccessServices.PackageManagement** .................................................................................. 21
Hierarchical Index

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

ApplicationException

DataAccessServices.PackageManagement.TableCopier..................................................48

WebService
Class Index

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

DataAccessServices.PackageManagement.PackageManagement_v1 (This class contains legacy functionality to manage packages in the Management Server (version 8.1 or less) database.) ...22
DataAccessServices.PackageManagement.PackageManagement_v2 (This class contains functionality to manage packages in the Management Server (version 8.2 or greater) database.) ..................31
DataAccessServices.PackageManagement.RowFixerNeeded .........................................................47
DataAccessServices.PackageManagement.TableCopier .................................................................48
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

DataAccessServies.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 DataAccessServies.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.

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 **GetServer>Error** ()
  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

Begins a package download.

Parameters:

<table>
<thead>
<tr>
<th>packageKey</th>
<th>The key which represents the package.</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>The major version of the package.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version of the package.</td>
</tr>
<tr>
<td>build</td>
<td>The build version of the package.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision version of the package.</td>
</tr>
</tbody>
</table>

Returns:

A guid which represents a download key.

Guid

Begins an upload of a package version.

Parameters:

<table>
<thead>
<tr>
<th>packageKey</th>
<th>The key which represents the package.</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>Description of the package.</td>
</tr>
<tr>
<td>packageVersionKey</td>
<td>The key which uniquely identifies this package.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified</td>
</tr>
<tr>
<td>dataLength</td>
<td>The length of data that will be uploaded.</td>
</tr>
</tbody>
</table>

Returns:

An upload key used to add data to the upload.


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[]

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

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.


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.

dependenceMinimumBuild, Int32  dependenceMinimumRevision, Int32  dependenceMaximumMajor, Int32  dependenceMaximumMinor, Int32  dependenceMaximumBuild, Int32  dependenceMaximumRevision, string  description, Boolean  supportsMidSessionUpdate, out DateTime  modifiedTime) [inline]

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

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>A guid of the package where the package version is to be created.</td>
</tr>
<tr>
<td>packageVersionKey</td>
<td>A guid for the new package version to be created.</td>
</tr>
<tr>
<td>major</td>
<td>The major version of the package.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version of the package.</td>
</tr>
<tr>
<td>build</td>
<td>The build version of the package.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision version of the package.</td>
</tr>
<tr>
<td>creatorMajor</td>
<td>The major version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorMinor</td>
<td>The minor version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorBuild</td>
<td>The build version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorRevision</td>
<td>The revision version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>dependentMinimumMajor</td>
<td>The major (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumMinor</td>
<td>The minor (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumBuild</td>
<td>The build (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumRevision</td>
<td>The revision (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumMajor</td>
<td>The major (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumMinor</td>
<td>The minor (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumBuild</td>
<td>The build (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumRevision</td>
<td>The revision (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>description</td>
<td>A string describing the package version.</td>
</tr>
<tr>
<td>supportsMidSessionUpdate</td>
<td>A boolean which indicates if this package version supports mid-session installs (if package is agent MSI or MSP).</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time that the package version was last modified.</td>
</tr>
</tbody>
</table>

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

Deletes an existing package from the database.
**Parameters:**

<table>
<thead>
<tr>
<th>key</th>
<th>The key identifying the package that should be deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>

```csharp
```

Deletes a package version from a package.

**Parameters:**

<table>
<thead>
<tr>
<th>packageKey</th>
<th>The key which identifies the package which contains the package version.</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>The major version number of the package to remove.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version number of the package to remove.</td>
</tr>
<tr>
<td>build</td>
<td>The build version number of the package to remove.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision number of the package to remove.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>

```csharp
```

Check that a package version has been correctly uploaded.

**Parameters:**

| packageVersionKey | A guid identifying the package version to check. |

```csharp
```

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.

```csharp
```

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

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

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

Return packages that are currently in progress.

Returns:

A PackagesDataSet_v1 containing the in progress packages.

Int32

Determines the length of a version of a package.

Parameters:

<table>
<thead>
<tr>
<th>packageKey</th>
<th>The key which represents the package.</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>The major version of the package.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version of the package.</td>
</tr>
</tbody>
</table>
### build
The build version of the package.

### revision
The revision version of the package.

**Returns:**
The length of data in bytes.


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

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
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.

  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
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.

```csharp
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.

```csharp
```

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. |

```csharp
Byte []
```

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.

```csharp
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.

```csharp
```

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.

```csharp
```

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>A guid of the package where the package version is to be created.</td>
</tr>
<tr>
<td>packageVersionKey</td>
<td>A guid for the new package version to be created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the package.</td>
</tr>
<tr>
<td>major</td>
<td>The major version of the package.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version of the package.</td>
</tr>
<tr>
<td>build</td>
<td>The build version of the package.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision version of the package.</td>
</tr>
<tr>
<td>creatorMajor</td>
<td>The major version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorMinor</td>
<td>The minor version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorBuild</td>
<td>The build version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorRevision</td>
<td>The revision version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>dependentMinimumMajor</td>
<td>The major (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumMinor</td>
<td>The minor (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumBuild</td>
<td>The build (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumRevision</td>
<td>The revision (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumMajor</td>
<td>The major (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumMinor</td>
<td>The minor (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumBuild</td>
<td>The build (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumRevision</td>
<td>The revision (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>description</td>
<td>A string describing the package version.</td>
</tr>
<tr>
<td>supportsMidSessionUpdate</td>
<td>A boolean which indicates if this package version supports mid-session installs (if package is agent MSI or MSP).</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time that the package version was last modified.</td>
</tr>
</tbody>
</table>

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

Deletes an existing package from the database.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The key identifying the package that should be deleted.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>


Deletes a package version from a package.
Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>The key which identifies the package which contains the package version.</td>
</tr>
<tr>
<td>major</td>
<td>The major version number of the package to remove.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version number of the package to remove.</td>
</tr>
<tr>
<td>build</td>
<td>The build version number of the package to remove.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision number of the package to remove.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>

```java
(Guid packageVersionKey)[inline]
```

Check that a package version has been correctly uploaded.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageVersionKey</td>
<td>A guid which is the package version which should be checked.</td>
</tr>
</tbody>
</table>

```java
PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1
(Guid packageKey)[inline]
```

Returns the machines that have a particular package installed.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>A package key that the machines in the MachinesDataSet_v1 should have installed.</td>
</tr>
</tbody>
</table>

Returns:

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

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

Returns an individual package within the database.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The key that identifies the package to return.</td>
</tr>
</tbody>
</table>

Returns:

A data set describing the package and its versions.
Requires deployment or administrative access.

```java
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

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

Return packages that are currently in progress.

**Returns:**

A PackagesDataSet_v1 containing the in progress packages

---

Int32
(Guid *packageKey*, Int32 *major*, Int32 *minor*, Int32 *build*, Int32 *revision*)

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

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

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

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.
  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**

Begins a package download.

**Parameters:**

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>The key which represents the package.</td>
</tr>
<tr>
<td>major</td>
<td>The major version of the package.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version of the package.</td>
</tr>
<tr>
<td>build</td>
<td>The build version of the package.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision version of the package.</td>
</tr>
</tbody>
</table>

**Returns:**

A guid which represents a download key.

Guid

```csharp
(Guid packageKey, String description, Guid packageVersionKey, out DateTime modifiedTime, int dataLength)[inline]
```

Begins an upload of a package version.

**Parameters:**

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>The key which represents the package.</td>
</tr>
<tr>
<td>description</td>
<td>Description of the package.</td>
</tr>
<tr>
<td>packageVersionKey</td>
<td>The key which uniquely identifies this package.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision version of the package.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified</td>
</tr>
<tr>
<td>dataLength</td>
<td>The length of data that will be uploaded.</td>
</tr>
</tbody>
</table>

**Returns:**

An upload key used to add data to the upload.

```csharp
(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:**

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageVersionKey</td>
<td>A package key guid which should be committed to the server.</td>
</tr>
</tbody>
</table>

Byte []

```csharp
(Guid downloadKey, Int32 offset, Int32 length)[inline]
```

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>downloadKey</td>
<td>The key which represents the download.</td>
</tr>
<tr>
<td>offset</td>
<td>The offset of the first byte to return.</td>
</tr>
<tr>
<td>length</td>
<td>The number of bytes to download.</td>
</tr>
</tbody>
</table>

Returns:

Bytes from the package.

```csharp
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageVersionKey</td>
<td>The key which uniquely identifies this package.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified</td>
</tr>
<tr>
<td>uploadKey</td>
<td>The key which represents the upload.</td>
</tr>
<tr>
<td>offset</td>
<td>The offset of the bytes currently uploaded.</td>
</tr>
<tr>
<td>data</td>
<td>The data to be uploaded.</td>
</tr>
</tbody>
</table>

```csharp
```

Creates a configuration package within the database.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The Guid which identifies the package.</td>
</tr>
<tr>
<td>company</td>
<td>The company name.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the package, i.e. Software, Configuration.</td>
</tr>
<tr>
<td>platform</td>
<td>The platform of the package.</td>
</tr>
<tr>
<td>productKey</td>
<td>The name of the product associated with this package.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>

```csharp
```

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>A guid of the package where the package version is to be created.</td>
</tr>
<tr>
<td>packageVersionKey</td>
<td>A guid for the new package version to be created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the package.</td>
</tr>
<tr>
<td>major</td>
<td>The major version of the package.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version of the package.</td>
</tr>
<tr>
<td>build</td>
<td>The build version of the package.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision version of the package.</td>
</tr>
<tr>
<td>creatorMajor</td>
<td>The major version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorMinor</td>
<td>The minor version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorBuild</td>
<td>The build version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>creatorRevision</td>
<td>The revision version of the console that created this package version (if the package is a configuration).</td>
</tr>
<tr>
<td>dependentMinimumMajor</td>
<td>The major (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumMinor</td>
<td>The minor (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumBuild</td>
<td>The build (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMinimumRevision</td>
<td>The revision (Minimum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumMajor</td>
<td>The major (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumMinor</td>
<td>The minor (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumBuild</td>
<td>The build (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>dependentMaximumRevision</td>
<td>The revision (Maximum version) of an agent that this configuration supports.</td>
</tr>
<tr>
<td>description</td>
<td>A string describing the package version.</td>
</tr>
<tr>
<td>supportsMidSessionUpdate</td>
<td>A boolean which indicates if this package version supports mid-session installs (if package is agent MSI or MSP).</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time that the package version was last modified.</td>
</tr>
</tbody>
</table>

```csharp
```

Deletes an existing package from the database.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The key identifying the package that should be deleted.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>

```csharp
```

Deletes a package version from a package.
### Parameters:

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>The key which identifies the package which contains the package version.</td>
</tr>
<tr>
<td>major</td>
<td>The major version number of the package to remove.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor version number of the package to remove.</td>
</tr>
<tr>
<td>build</td>
<td>The build version number of the package to remove.</td>
</tr>
<tr>
<td>revision</td>
<td>The revision number of the package to remove.</td>
</tr>
<tr>
<td>modifiedTime</td>
<td>The time the package was last modified.</td>
</tr>
</tbody>
</table>

```csharp
```

Check that a package version has been correctly uploaded.

### Parameters:

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageVersionKey</td>
<td>A guid which is the package version which should be checked.</td>
</tr>
</tbody>
</table>

```csharp
PackageManagementWebServiceCode.Schemas.MachinesDataSet_v1
```

Returns the machines that have a particular package installed.

### Parameters:

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packageKey</td>
<td>A package key that the machines in the MachinesDataSet_v1 should have</td>
</tr>
<tr>
<td></td>
<td>installed.</td>
</tr>
</tbody>
</table>

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

```csharp
PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2
```

Returns an individual package within the database.

### Parameters:

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The key that identifies the package to return.</td>
</tr>
</tbody>
</table>

Returns:

A data set describing the package and its versions.

Requires deployment or administrative access.

```csharp
PackageManagementWebServiceCode.Schemas.PackagesDataSet_v2
```
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
```

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
```

Return packages that are currently in progress.

**Returns:**

A PackagesDataSet_v1 containing the in progress packages

```
Int32
(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.

```
```

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

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. |


Unlock a package to allow the package to be modified.

**Parameters:**

| packageKey | A guid package key of the package to unlocked. |

InterfaceStatus_v1

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.PackageManager.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
Index

BeginPackageVersionDownload
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 24
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 32
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 40

BeginPackageVersionUpload
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 24
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 33
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 41

CommitPackageVersion
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 24
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 33
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 41

ContinuePackageVersionDownload
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 25
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 33
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 41

ContinuePackageVersionUpload
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 25
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 34
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 42

CreatePackage
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 25
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 34
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 42

CreatePackageVersion
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 25
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 34
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 42
  DataAccessServices, 20
  DataAccessServices::PackageManagement, 21
  DataAccessServices::PackageManagement::Packa
gemanagement_v1, 22
  DataAccessServices::PackageManagement::Packa
gemanagement_v2, 31
  DataAccessServices::PackageManagement::Packa
gemanagement_v3, 39
  DataAccessServices::PackageManagement::RowFixer
  Needed, 47
  DataAccessServices::PackageManagement::TableCopi
er, 48
  DataAccessServices::PackageManagement::Package
Management_v1
  BeginPackageVersionDownload, 24
  BeginPackageVersionUpload, 24
  CommitPackageVersion, 24
  ContinuePackageVersionDownload, 25
  ContinuePackageVersionUpload, 25
  CreatePackage, 25
  CreatePackageVersion, 25
  DeletePackage, 26
  DeletePackageVersion, 27
  FinalisePackageVersion, 27
  GetMachinesWithPackage, 27
  GetPackageFromKey, 27
  GetPackageFromKeyWithInProgress, 28
  GetPackages, 28
  GetPackagesWithInProgress, 28
  GetPackageVersionLength, 28
  GetServerError, 29
  LockPackage, 29
  UnlockPackage, 29
  VerifyInterface, 29
  WIPSaved, 29
  DataAccessServices::PackageManagement::Package
Management_v2
  BeginPackageVersionDownload, 32
  BeginPackageVersionUpload, 33
  CommitPackageVersion, 33
  ContinuePackageVersionDownload, 33
  ContinuePackageVersionUpload, 34
  CreatePackage, 34
  CreatePackageVersion, 34
  DeletePackage, 35
  DeletePackageVersion, 35
  FinalisePackageVersion, 36
  GetMachinesWithPackage, 36
  GetPackageFromKey, 36
  GetPackageFromKeyWithInProgress, 36
  GetPackages, 37
  GetPackagesWithInProgress, 37
  GetPackageVersionLength, 37
  GetServerError, 37

49
LockPackage, 38
UnlockPackage, 38
VerifyInterface, 38
WIPSaved, 38
DataAccessServices::PackageManagement::Package
Management_v3
BeginPackageVersionDownload, 40
BeginPackageVersionUpload, 41
CommitPackageVersion, 41
ContinuePackageVersionDownload, 41
ContinuePackageVersionUpload, 42
CreatePackage, 42
CreatePackageVersion, 42
DeletePackage, 43
DeletePackageVersion, 43
FinalisePackageVersion, 44
GetMachinesWithPackage, 44
GetPackageFromKey, 44
GetPackageFromKeyWithInProgress, 44
GetPackages, 45
GetPackagesWithInProgress, 45
GetPackageVersionLength, 45
GetServerError, 45
LockPackage, 46
UnlockPackage, 46
VerifyInterface, 46
WIPSaved, 46
DeletePackage
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 26
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 35
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 43
DeletePackageVersion
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 27
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 35
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 43
FinalisePackageVersion
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 27
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 36
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 44
GetMachinesWithPackage
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 27
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 36
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 44
GetPackageFromKey
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 27
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 36
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 44
GetPackageFromKeyWithInProgress
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 28
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 37
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 45
GetPackages
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 28
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 37
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 45
GetPackagesWithInProgress
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 28
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 37
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 45
GetPackageVersionLength
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 28
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 37
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 45
GetServerError
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 29
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 37
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 45
LockPackage
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 29
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 38
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 46
UnlockPackage
DataAccessServices::PackageManagement::Packa
gemanagement_v1, 29
DataAccessServices::PackageManagement::Packa
gemanagement_v2, 38
DataAccessServices::PackageManagement::Packa
gemanagement_v3, 46
VerifyInterface
VerifyInterface
DataAccessServices::PackageManagement::PackageManagement_v1, 29
DataAccessServices::PackageManagement::PackageManagement_v2, 38
DataAccessServices::PackageManagement::PackageManagement_v3, 46
WIPSaved

DataAccessServices::PackageManagement::PackageManagement_v1, 29
DataAccessServices::PackageManagement::PackageManagement_v2, 38
DataAccessServices::PackageManagement::PackageManagement_v3, 46