ACTIAN™

Installation Toolkit Handbook

How to Integrate Zen Program Installations into Your Own

Zen v16

Activate Your Data™

Contents

About This Document

This documentation contains information for developers about how to integrate a Zen product installation with another application installation.

We would appreciate your comments and suggestions about this document. Your feedback can determine what we write about the use of our products and how we deliver information to you. Please post your feedback in the community forum on the Actian Zen website.

Customizing Zen Installations

The following topics cover concepts and procedures for customizing Zen installations on Windows systems. Customizing the installation allows you to bundle all or part of a Zen product with your application.

Customized Installation Overview

This topic covers the technology and customization settings used for Zen product installations. On Windows systems, Zen installation uses Microsoft Installer (MSI). The PTKSetup.ini file contains default settings that you can change for custom installations.

Installer Executable

For most installation scenarios, the installer executable file should be used for installations. The installer is an InstallShield package that performs a variety of checks before installation. It also detects whether you have 32- or 64-bit Windows, launches the appropriate installation, and automatically provides all 32- and 64-bit client components appropriate to your system.

If bundling or embedding the Zen installation with another software package or scripting the installation of Zen, do **not** use the installer executable. For these scenarios, see the topics Embedding Zen in Your Application and Using Silent Installation for detailed instructions to install Zen.

The following table describes Zen installers on Windows operating systems.

Product	Installation Package	Description
Enterprise Server	Install_Zen_EnterpriseServer.exe	• Installs 32-bit engine on 32-bit operating system. • Installs 64-bit engine on 64-bit operating system. • Installs all client components.
Cloud Server	Install_Zen_CloudServer.exe	• Installs 32-bit engine on 32-bit operating system. • Installs 64-bit engine on 64-bit operating system. • Installs all client components.
Workgroup	Install_Zen_Workgroup_Engine.exe	• Installs 32-bit engine on both 32- and 64-bit operating systems. • Installs all client components.
Client	Install_Zen_Client.exe	• Installs 32-bit components on 32-bit operating system. • Installs both 32- and 64-bit components on 64-bit system.

Product	Installation Package	Description
Client Reporting Engine	Install_Zen_Reporting_Engine.exe	• Installs 64-bit engine only. • Installs all client components.

PTKSetup.ini Settings

PTKSetup.ini contains all settings needed for a typical installation. This file is used by every Zen product. On the installation media, it is located in the same folder as the .msi file that uses it.

PTKSetup.ini is divided into two parts:

- Properties Section - Contains settings used during every installation.
- Registry Migration Section - Contains settings used with versions of Zen before PSQL v10.

Each setting has a current value that is the default for a typical installation. Comments in the file explain the settings and give all accepted values.

Caution! You must use the specific PTKSetup.ini file included with the version of the product that you are embedding. Because installer technology and installation settings can change from version to version, the file must match the product that it accompanies.

Properties Section

The following table lists keys in the PTKSetup.ini Properties section. The keys are grouped into categories. The value for each setting is contained in a key.

Categories	Keys
Setup Type	PVSW_PSQL_INSTALL_TYPE
Destination Folder (Custom Setup Only)	PVSW_PSQL_SKIP_INSTALLDIR
Directory Locations (Custom setup only)	PVSW_PSQL_DATADIR PVSW_PSQL_INSTDIR32 PVSW_PSQL_INSTDIR64 PVSW_PSQL_ARCHIVE_DIR
Registration Page	PVSW_PSQL_UI_NO_REGISTER PVSW_OEM_REGISTER_HTML

Categories	Keys
License (Workgroup or server engines only)	PVSW_PSQL_LICENSE_KEY PVSW_PSQL_SKIP_LICENSE
Engine as Service or Application (Workgroup or cache engines only)	PVSW_RUN_CCE_AS_SVC PVSW_RUN_WGE_AS_SVC PVSW_APP_WAIT_TIME
Optional Features - Custom Setup Only	
Data Access Features	PVSW_PSQL_INSTALL_ADONET PVSW_PSQL_INSTALL_BTRBOX PVSW_PSQL_INSTALL_DTO PVSW_PSQL_INSTALL_JCL PVSW_PSQL_INSTALL_JDBC PVSW_PSQL_INSTALL_PDAC
Java Utilities Feature	PVSW_PSQL_INSTALL_DDFB PVSW_PSQL_INSTALL_PCC PVSW_PSQL_INSTALL_DOCUMENTATION PVSW_PSQL_INSTALL_NOTIFYVIEWER PVSW_PSQL_INSTALL_JAVAUTILS
Other Utility Features	PVSW_PSQL_INSTALL_CBOL PVSW_PSQL_INSTALL_COREUTILS PVSW_PSQL_INSTALL_PSA

Registry Migration Section

The Registry Migration section of the PTKSetup.ini file can be used when upgrading from PSQL v9 or earlier. The settings in this section enable migration of selected registry values from the earlier to the later version.

All of the configuration settings are explained in detail in *Advanced Operations Guide*.

Registry Migration Settings Format

The Registry Migration section lists each setting for the previous Zen version first, followed by an equal (=) sign. The setting for Zen is listed to the right of the equal sign.

The format used for registry migration settings is *<Setting for Prior Version>=<Setting for New Version>*.

Using Registry Migration Settings

Comment out any settings that should not be migrated during installation from any version of Zen before v10. If the key to be migrated already exists, the earlier registry setting replaces it.

Note: When upgrading versions of PSQL v10 and later, the Zen installations ignore this section and migrate all registry settings automatically.

To view your current settings, use the **bcfg** utility described in *Advanced Operations Guide*. This utility allows you to capture your current configuration settings in a report that may be useful for considering needed changes.

Configuring Zen After Installation

After installing Zen, you can use the configuration functions available in the Distributed Tuning Interface (DTI) to configure and tune your database engines. The functions available in DTI allow you to perform a variety of tasks to check current settings and make changes to them programmatically.

For information on using the DTI functions to configure a database engine after installation, see the list of configuration function groups described in *Distributed Tuning Interface Guide*.

The Zen DTO object is a COM Interface to perform Zen configuration functions similar to the DTI interface.

Improving System Performance

In an enterprise environment where reimaging or some other technique may be the preferred method for workstation recovery, and System Restore is never used, the additional time and disk space required to create restore points may be recovered and performance improved by disabling System Restore for installer activity. To disable System Restore, see the Microsoft help system for detailed information.

This setting affects only installer-initiated restore activity and is available in Group Policy to aid its deployment to workstations.

Note: System Restore is a very important feature of Windows and in most circumstances it is recommended that you do not disable it. Disabling System Restore is applicable only for corporate scenarios where the feature is not used.

Customizing Installation Content

You can change what is installed for Zen by using one of the following methods:

- Using PTKSetup.ini Settings
- Changing Installation Package Size and Features Using CAB Files
- Embedding Zen in Your Application

Using PTKSetup.ini Settings

Once you have modified PTKSetup.ini by uncommenting any properties and setting them to new values, you can simply run the installer from the command line to use those properties in the .ini file.

Note: If you include an MSI property on the command line, but it is uncommented and set in PTKSetup.ini, the PTKSetup.ini value is used.

The available PTKSetup.ini properties are covered in PTKSetup.ini Settings.

Specifying a Zen Key

Before PSQL v11, you could specify a Zen authorization key as an attribute in the PTKSetup.ini file. The same key could be used for multiple installations. Since PSQL v11, each installation requires a unique key. You can still specify a key in PTKSetup.ini, but it must be unique to the machine where the installation runs.

Various means are used for distribution of keys, such as a key printed on the product packaging or included inside the box or in email to the end user. Whatever method you use to distribute keys, one way to ensure uniqueness of the key in PTKSetup.ini is for your application installation to prompt the end user for the key. Your application installation could then write that key to a local copy of PTKSetup.ini and continue with the installation process that uses PTKSetup.ini. The Zen installation process will automatically authorize a valid key if one is specified in PTKSetup.ini. Note that part of key validity is that it must be unique for the machine on which the installation is being performed.

Using an Alternate Configuration File Location with the Install Executable

The MSI Public Property PVSW_CFG_FILE can be set to use an alternate configuration file by passing a fully qualified file path name on the command line when you run the installer executable. The property and its value are passed to the underlying msiexec Windows process.

Before passing the property, determine whether its value contains spaces. If not, your command should resemble the following:

```
Install_<product>.exe /v"PVSW_CFG_FILE=c:\temp\ConfigFile.ini"
```

Note: All properties after **/v** are passed to msiexec. The **/v** option has no space after it. The string of properties to be passed must be enclosed in double quotation marks. If one of the properties is a path name that may include spaces, the path must also be enclosed in double quotation marks, and they must both be escaped with a backslash (\"), as shown here:

```
Install_<product>.exe /v"PVSW_CFG_FILE=\"c:\temp\My Folder\ConfigFile.ini\""
```

Combinations of the examples shown here can be used to pass multiple property settings:

```
Install_<product>.exe /v"PROPERTY1=0 PROPERTY2=\"c:\My Path\File.txt\""
```

See also Prerequisites for Installing a Zen MSI.

To install Zen with no documentation using a command line property

1. Make sure that MSI prerequisites are met. See Prerequisites for Installing a Zen MSI.
2. At a command prompt, enter the following command:

   ```
   Install_<product>.exe /v"/l*v \"%temp%\Zen_<version>_Install.log\"
   PVSW_PSQL_INSTALL_DOCUMENTATION=0"
   ```

3. Press Enter.

   The installation program runs interactively, without the documentation option.

To install Zen with key authorization using a command line property

1. Make sure that MSI prerequisites are met. See Prerequisites for Installing a Zen MSI.
2. At a command prompt, enter the following command using a unique key for the system where the installation is running:

   ```
   Install_<product>.exe /v"/l*v \"%temp%\Zen_<version>_Install.log\" PVSW_PSQL_LICENSE_KEY=NG2ZE-
   ZKS3C-D2CFK-9IR6K-G2C3X"
   ```

Note: All properties after **/v** are passed to msiexec. The **/v** option has no space after it. The string of properties to be passed must be enclosed in double quotation marks. If one of the properties is a path name that may include spaces, the path must also be enclosed in double quotation marks, and they must both be escaped with a backslash (\"), as shown in the example above.

Changing Installation Package Size and Features Using CAB Files

The Zen installation includes cabinet (CAB) files that can be removed from your installation package to decrease the overall size of the installation package and avoid using unnecessary files. Removing these files automatically removes the corresponding feature from the list of available features in the graphical user interface during a custom installation.

This section covers the following topics:

Working with CAB Files and Future Updates

CAB files are located in the same directory as the MSI. CAB files beginning with an underscore (_) are required for that installation type.

The core set of CAB files does not change. However, Zen updates contain additional CAB files from the original set released for the major version of the product to cover newly added components or features. Be sure and obtain the latest files for the installation type and optional features you need.

Required CAB Files

CAB files that are required are designated by a name that begins with an underscore. The following table lists the cabinet files required for each installation package type.

Cabinet File	Enterprise Server	Cloud Server	Client	Workgroup	Reporting
_BCW.cab			X	X	
_C32_64b.cab	X	X	X	X	
_CE32_64.cab			X		
_CmnEn64.cab	X	X			X
_DbEng32.cab	X	X		X	X

Cabinet File	Enterprise Server	Cloud Server	Client	Workgroup	Reporting
_DbEng.cab	X	X		X	X
_DRM64.cab	X	X		X	X
_PSQL.cab	X	X	X	X	X
_PSQL64.cab	X	X	X	X	X
_RpEng64.cab					X
_Srvr32.cab	X	X			
_Srvr64.cab	X	X			
_WGE.cab				X	

Required Files Other than CAB

In addition to the files listed under Required CAB Files, your custom installation must have the following items in a folder called Data, located in the same folder where the installer executable runs:

- 409 folder and all files within it
- 411 folder and all files within it
- Data folder and all files within it, except optional .cab files listed in the next section
- ISSetupPrerequisites folder and all files within it
- Setup*Product*32_x86.exe
- Setup*Product*64_x64.exe

  Where *Product* is EnterpriseServer, Client, Workgroup, or Reporting, with two special cases:

  - For Workgroup, the 64-bit setup file is called SetupWorkgroup32_x64.exe.
  - For Client Reporting Engine, only the 64-bit file SetupReporting64_x64.exe is used.

Optional CAB Files

The following table lists optional features by the installation package type that applies them. If you want to install them, you must leave them in their location under the Data folder described in the previous topic. For more information on these features, see Zen Optional Features in *Getting Started with Zen*.

Cabinet File	Enterprise Server	Enterprise Server	Client	Workgroup[1]	Reporting
ADONET.cab	X	X	X	X	X
BtreveDos.cab	X	X	X	X	X
CSE32Cmn.cab	X	X	X	X	X
CSE32Eng.cab	X	X		X	
CSE64Eng.cab	X	X			X
CSECmEng.cab	X	X		X	X
DDFB.cab	X	X	X	X	X
Docs.cab	X	X	X	X	X
DTO.cab DTO64.cab	X	X	X	X	X
EclipRCP.cab	X	X	X	X	X
JCL.cab JCL64.cab	X	X	X	X	X
JDBC.cab JDBC64.cab	X	X	X	X	X
JRE.cab	X	X	X	X	X
JreUtils.cab	X	X	X	X	X
NVUtil.cab	X	X	X	X	X
PDAC.cab PDAC64.cab	X	X	X	X	X
PSA.cab	X	X	X	X	X
Utils.cab	X	X	X	X	X
Utils2.cab	X	X	X	X	X
WPMCS32.cab WPMCS64.cab	X	X			
WPMCS64R.cab					X
ZenCC.cab	X	X	X	X	X

Cabinet File	Enterprise Server	Enterprise Server	Client	Workgroup[1]	Reporting

[1] Zen Workgroup is a 32-bit application. However, separate installation packages are provided for 32- and 64-bit operating systems.

Optional Feature Cabinet File Dependencies

Certain optional features require that other components accompany them in order to function. The following table lists the optional features that require other components.

	Cobol Schema Executor	DDF Builder	ZenCC[1]	System Analyzer	User Documentation	Notification Viewer	Other Utilities[2]
CSE32Cmn.cab	X[3]						
CSE32Eng.cab	X[4]						
CSE64Eng.cab	X[5]						
CSECmEng.cab	X[6]						
DDFB.cab		X					
Docs.cab					X		
EclipRCP.cab		X	X		X	X	X
JRE.cab		X	X		X	X	X
JreUtils.cab							X
NVUtil.cab						X	
PSA.cab				X			
Utils2.cab							X
Utils.cab				X			X
ZenCC.cab			X		X		

	Cobol Schema Executor	DDF Builder	ZenCC[1]	System Analyzer	User Documentation	Notification Viewer	Other Utilities[2]

[1] Zen Control Center

[2] Includes all other utilities not listed in this table. The additional utilities that require Java, such as Phone Authorization Wizard, bcfg, and bmon, require EclipRCP.cab, JRE.cab, and JreUtils.cab. The common utilities (non-Java) such as License Administrator and Function Executor, require Utils2.cab and Utils.cab.

[3] Required for Client, Workgroup, Enterprise Server, and Cloud Server

[4] Required for Workgroup, Enterprise Server, and Cloud Server

[5] Required for Enterprise Server

[6] Required for Workgroup, Enterprise Server, and Cloud Server

CAB File Installation Example

The following provides an example of customizing your installation using CAB files. This example installs the Workgroup engine with documentation as the only optional feature.

Note: Documentation requires that you install the Eclipse framework and Zen Control Center.

To package a Workgroup Engine with Documentation

This example is for a system running on a Windows operating system.

1. Select the required CAB files for the Workgroup Engine (see Required CAB Files). These files include the following:
   - _BCW.cab
   - _C32_64b.cab
   - _DbEng.cab
   - _DbEng32.cab
   - _DRM64.cab
   - _PSQL.cab
   - _PSQL64.cab
   - _WGE.cab
2. Select the optional feature CAB files for the documentation (see Optional Feature Cabinet File Dependencies). These files include the following:
   - Docs.cab

- EclipRCP.cab
- ZenCC.cab
- JRE.cab

The files listed in these two steps are the only CAB files that you need in order to install the Workgroup Engine and documentation. All other CAB files may be removed from the installation package. All non-CAB files described under Required Files Other than CAB must also remain in the installation package.

Embedding Zen in Your Application

To embed Zen in your application, we recommend using Windows Installer. Originally called the Microsoft Installer, its acronym is MSI and its file extension is .msi.

This section covers the following topics:

- Prerequisites for Installing a Zen MSI
- Zen Installers and Related MSI Files
- Customizing Zen Embedding with a Microsoft Transform
- Combining MSI Options and PTKSetup.ini Settings

Prerequisites for Installing a Zen MSI

Zen MSI installation requires the following:

- For Windows 8.1 or Server 2012 R2: Apply Windows Update 3118401.
- The target system must contain the minimum version of Windows Installer supported by Zen, which is Windows Installer 5.0. To see what version of the Windows Installer you have on your system, run `msiexec /?` at a command prompt. If you are unsure whether the system has MSI 5.0 or later, install using the installation executable with the **/v** option.
- To install Zen using the msiexec command line without running the installer executable, you must ensure that all prerequisites are installed and include the property and value `PSQL_PREREQS_INSTALLED=Y` on the msiexec command line.
- On Windows Vista and later editions, the process that calls the Zen installer must already be running with elevated user credentials before starting the Zen installation using the MSI file. The installer must run as an elevated administrator user account.
- If you are upgrading from PSQL v11 SP3 and the current version is between 11.30.000 and 11.31.050, then the patch PSQLv11.31.049.msp (located in the installation folder

"\ISSetupPrerequisites{1BCDC56D-D2CC-4125-8B24-4D249AFBFFE7}\") must be applied or the v12 installation displays an error message when PSQL v11 SP3 is being removed. The error message is displayed even if the upgrade installation runs silently. Alternatively, you can apply the latest available PSQL v11 SP3 patch before upgrading.

- If you are upgrading from PSQL v12 Server, Vx Server, or Workgroup and the current version is between 12.00.000 and 12.01.066, then before the upgrade you must apply the patch PSQLv12.01.068.msp, found in the installer folder \ISSetupPrerequisites\{D61586D9-7EFA-4ED8-8A47-6E569A92D4D9}. Without this patch, the upgrade deletes the data folder defaultdb. Alternatively, before upgrading you can apply the latest available PSQL v12.01 patch.
- Before you install the upgrade on any 64-bit version of Windows, you must first install the 64-bit Microsoft Visual C++ 2019 Runtime Libraries or later, found in the installer folder \ISSetupPrerequisites\{2F044C80-608C-4274-AB1D-96D67E047307}.
- Before you install the upgrade on both 32- and 64-bit versions of Windows, you must first install 32-bit Microsoft Visual C++ 2019 Runtime Libraries or later, found in the installer folder \ISSetupPrerequisites{7BB553E0-BAA5-4184-965C-AEEB89B82D46}.

In addition to the syntax and options displayed by running `msiexec /?` at a command prompt, MSI technology is fully documented by Microsoft on its website.

Zen Installers and Related MSI Files

The following table lists Zen products and their MSI files.

Product	MSI Files
Enterprise Server	ActianZenv16EnterpriseServer64_x64.msi ActianZenv16EnterpriseServer32_x86.msi
Cloud Server	ActianZenv16CloudServer64_x64.msi ActianZenv16CloudServer32_x86.msi
Workgroup	ActianZenv16WGE32_x64.msi ActianZenv16WGE32_x86.msi Workgroup is a 32-bit application. However, separate installation packages are provided for 32- and 64-bit operating systems to install the needed client components.
Client	ActianZenv16Client64_x64.msi ActianZenv16Client32_x86.msi
Reporting Engine	ActianZenv16Reporting64_x64.msi

Customizing Zen Embedding with a Microsoft Transform

The Microsoft transform (MST) provides a method of customizing your embedded installation. An .mst file is a set of changes to use during installation. Using a transform file to customize your installation allows for flexibility in selecting features and simplifies product updates.

To generate a MSI transform, you must work with the installer database that stores application information needed during installation. You can use several tools to generate a transform file, including InstallShield, Microsoft Orca, or the MSI utilities in the Windows Platform SDK.

Example: Creating an Embedded Installation Without Shortcuts

A common use case for a transform file is to embed a Zen installation without creating shortcuts in the Start menu or on the Desktop. The following steps provide an example of this customization.

1. Make a copy of the original Zen .msi file.
2. Open the .msi copy using your tool of choice.
3. In the Action column of the Shortcut table, modify the InstallExecuteSequence record to change the CreateShortcuts value.

   Original

   ```
   CreateShortcuts   <null_value>  <sequence_number>
   ```

   Revised

   ```
   CreateShortcuts   NOSHORTCUTS   <sequence_number>
   ```

   The undefined property name NOSHORTCUTS will cause the CreateShortcuts action to be skipped.
4. Use your tool to generate the .mst file by comparing the original .msi file to the modified copy.
5. Name the generated transform file, such as MyTransform.mst.
6. Include the transform file with the installation files and add the TRANSFORMS= option to your command line – in this case, TRANSFORMS=MyTransform.mst.

Here is an example installation based on these steps:

```
msiexec.exe /i {path}\ActianZenv16<product-type><platform>.msi /qn REBOOT=ReallySuppress /l*v
"%temp%\Zen_<version>_Install.log" TRANSFORMS=MyTransform.mst PSQL_PREREQS_INSTALLED=Y
```

Combining MSI Options and PTKSetup.ini Settings

You can combine MSI options and parameters and PTKSetup.ini file settings in a Zen installation. To do this, be sure to disable the property in the .ini file by commenting it out. Then, run the installation from the command line using the options you disabled in the file.

The following examples combine methods used in the previous examples.

To remove Zen documentation from an installation

1. In the PTKSetup.ini file, find the following property:

```
;PVSW_PSQL_INSTALL_DOCUMENTATION=1
```

2. Remove the semicolon and change the value for this property to 0, so that it reflects the following:

```
PVSW_PSQL_INSTALL_DOCUMENTATION=0
```

3. Make sure MSI prerequisites are met. See Prerequisites for Installing a Zen MSI.

4. At a command prompt, run the following command:

```
msiexec.exe /i <ActianZenProductName>.msi /qn PSQL_PREREQS_INSTALLED=Y
```

To include Zen key authorization in an installation

1. In the PTKSetup.ini file, find the following property:

```
;PVSW_PSQL_LICENSE_KEY=
```

2. Remove the semicolon and change the value for this property to a unique key for the machine on which the installation runs, so that it resembles the following:

```
PVSW_PSQL_LICENSE_KEY=NG2ZE-ZKS3C-D2CFK-9IR6K-G2C3X
```

   See also Specifying a Zen Key.

3. Make sure MSI prerequisites are met. See Prerequisites for Installing a Zen MSI.

4. At a command prompt, run the following command:

```
msiexec.exe /i <ActianZenProductName>.msi /qn PSQL_PREREQS_INSTALLED=Y
```

Using an Alternate Configuration File Location with the MSI File

Use the MSI file only if the MSI prerequisites are met. See Prerequisites for Installing a Zen MSI.

The MSI Public Property PVSW_CFG_FILE can be set to use an alternate configuration file by passing a fully qualified file path name on the command line.

Before passing the property, determine whether its value contains spaces. If not, your command should resemble the following:

```
msiexec.exe /i "{path}\MSIPackage.msi" PSQL_PREREQS_INSTALLED=Y PVSW_CFG_FILE=c:\temp\ConfigFile.ini
```

If the property value contains spaces, you must enclose it in double quotation marks as in this example:

```
msiexec.exe /i "{path}\MSIPackage.msi" PSQL_PREREQS_INSTALLED=Y
PVSW_CFG_FILE="c:\My Folder\ConfigFile.ini"
```

Using Silent Installation

A silent installation runs without interaction from a user and by displaying little to no user interface during installer execution. It can be accomplished in two ways:

- Using msiexec with the **/quiet** or **/q** option and its extensions. The most common extension **/qn** displays no UI.
- Using an InstallShield-based installer executable with the **/s** option, adding the **/w** option to keep the executable running during MSI installation, and adding the **/v** option to pass command line options to msiexec. Note that only Zen setup<product><bitness>.exe installer executables can use the **/w** option. You **cannot** use the **/w** option with the Install_Zen_<product>.exe launcher application. For example, the first of these commands can use **/w**, while the second cannot:

```
    SetupEnterpriseServer64_x64.exe /w
    Install_Zen_Client.exe
```

Because silent installations cannot launch other installations automatically, you may need to execute other installations to make sure that the product is completely up to date. See Special Considerations for Silent (or Basic UI) Installation.

Silent Installation Examples Using Only the Installer Executable

The following is an example of installing Zen Client silently with a log file in the user's temp folder:

```
Install_Zen_Client.exe /s /v"/qn /l*v \"%temp%\Zen_<version>_SilentInstall.log\""
```

Note: All properties after **/v** are passed to msiexec. The **/v** option has no space after it. The string of properties to be passed must be enclosed in double quotation marks. If one of the properties is a path name that may include spaces, the path must also be enclosed in double quotation marks, and they must both be escaped with a backslash (\"), as shown in the example above.

The next example creates a log file in the user's temp folder and sets license key and service installation options:

```
Install_Zen_Workgroup_Engine.exe /s /v"/qn REBOOT=ReallySuppress PVSW_PSQL_LICENSE_KEY={key_value}
PVSW_RUN_WGE_AS_SVC=Y TRANSFORMS=MyTransform.mst /l*v \"%temp%\Zen_<version>_Install.log\""
```

Note: If you set no log file name and file location, a default file is written to the %temp% folder. The file name is Zen_<version>_<product>_<architecture>_Install.log. For example, Zen_v16_Server_x64_Install.log.

Note: All properties after **/v** are passed to msiexec. The **/v** option has no space after it. The string of properties to be passed must be enclosed in double quotation marks. If one of the properties is a path name that may include spaces, the path must also be enclosed in double quotation marks, and they must both be escaped with a backslash (\"), as shown in the example above.

Silent Installation Examples Using MSI

The embedding of an .msi file for one product in the installation of another product is often done by silent installation. The following example creates a log file in the user's temp folder and specifies the license key and service installation options. See also Prerequisites for Installing a Zen MSI.

```
msiexec.exe /i {path}\ActianZenv16WGE32_x86.msi /qn REBOOT=ReallySuppress
PVSW_PSQL_LICENSE_KEY={key_value} PSQL_PREREQS_INSTALLED=Y PVSW_RUN_WGE_AS_SVC=N
TRANSFORMS=MyTransform.mst /l*v "%temp%\Zen_v16_Install.log"
```

This example uses a Microsoft transform file (.mst) to customize the installation. For more information, see Customizing Zen Embedding with a Microsoft Transform.

Note: Unless you specify a log file name and file location, no log file is created.

Silent Installation Example Using Scripting

If you want to script a silent installation of Zen, you cannot use the installer executables listed under Customized Installation Overview because they do not remain running for the duration of the installation. Instead, use one of the two following options. Both examples are for Zen Enterprise Server and both use Microsoft PowerShell, which we recommend over Windows batch scripts to better handle scripting complexity. When scripting the Zen installation, make sure you sequence the installation of all Zen prerequisites, including the Microsoft Visual C++ runtime, **before** invoking the Zen installation itself by setup launcher or by MSI.

The following PowerShell script examples are provided each on its own page for more convenient copying and pasting.

PowerShell script to silently install Zen Enterprise Server using setup launcher

```
# Sample script for silently installing Zen Enterprise Server Engine that automatically ensures that
# the required Microsoft Visual C++ 2019 runtimes are installed before invoking the Zen MSI
# installation launcher.
# 1) The PowerShell execution policy allows scripts to be executed.
# 2) The PowerShell session is running with Administrator privileges if UAC is enabled.
# 3) Zen Server Engine installation files are located in a dir named "server" along with this script.
# 4) The bundled Microsoft Visual C++ Runtime installers are invoked BEFORE Zen installation is
#    invoked.
#
$currentFolder= & {Split-Path $myInvocation.ScriptName}
$installVCRT32="$currentFolder\Data\ISSetupPrerequisites\{7BB553E0-BAA5-4184-965C-
AEEB89B82D46}\VC_redist.x86.exe"
$vc_redist32Log=join-path $env:TEMP "\vcredist2015_x86.log"
$vc_redist32Args="/quiet","/norestart","/log $vc_redist32Log"
$is64Bit=$False
if ([System.IntPtr]::Size -eq 4) {
 $installFile="$currentFolder\Data\SetupEnterpriseServer32_x86.exe"
} else {
 $is64Bit=$True
 $installFile="$currentFolder\Data\SetupEnterpriseServer64_x64.exe"
 $installVCRT64="$currentFolder\Data\ISSetupPrerequisites\{2F044C80-608C-4274-AB1D-
96D67E047307}\VC_redist.x64.exe"
 $vc_redist64Log=join-path $env:TEMP "\vcredist2015_x64.log"
 $vc_redist64Args="/quiet","/norestart","/log $vc_redist64Log"
}
$setupArgs='/s','/w','/v/qn'
#
# Always install 32-bit MS VCRT.
if (Test-Path "$installVCRT32") {
 Try {
  Write-Host "Starting Microsoft Visual CRT (x86) installation at"(Get-Date).ToString()"..."
  $proc=(Start-Process -FilePath $installVCRT32 -ArgumentList $vc_redist32Args -Wait -PassThru)
  $handle=$proc.Handle # cache proc.Handle http://stackoverflow.com/a/23797762/1479211
  $proc.WaitForExit();
  If ($proc.ExitCode -ne 0) {
   Write-Error "The MS install returned error code: $($proc.ExitCode)"
  } else {Write-Host "The MS install completed successfully at"(Get-Date).ToString()"."}
 }
 Catch {Write-Error "Unknown error: " + [string]$error[0]; exit -1}
}

if ($is64Bit) {
 # Install 64-bit MS VCRT
 if (Test-Path "$installVCRT64") {
  Try {
   Write-Host "Starting Microsoft Visual CRT (x64) installation at"(Get-Date).ToString()"..."
   $proc=(Start-Process -FilePath $installVCRT64 -ArgumentList $vc_redist64Args -Wait -PassThru)
   $handle=$proc.Handle # cache proc.Handle http://stackoverflow.com/a/23797762/1479211
   $proc.WaitForExit();
   If ($proc.ExitCode -ne 0) {
    Write-Error "The MS install returned error code: $($proc.ExitCode)"
   } else {
    Write-Host "The MS install completed successfully at"(Get-Date).ToString()"."
   }
  }
  Catch {Write-Error "Unknown error: " + [string]$error[0]; exit -1}
 }
} else {Write-Host "Skipping installation of Microsoft Visual C++ (x64) on 32-bit system..."}

# Zen install using setup launcher application.
If (Test-Path "$installFile") {
 Try {
  Write-Host "Starting Zen installation at"(Get-Date).ToString()"..."
```

```
  $proc=(Start-Process -FilePath $installFile -ArgumentList $setupArgs -NoNewWindow -Wait -PassThru)
  $handle=$process.Handle # cache proc.Handle http://stackoverflow.com/a/23797762/1479211
  $proc.WaitForExit();
  If ($proc.ExitCode -ne 0) {
   Write-Warning "$_ exited with status code $($proc.ExitCode)"
  } else {
   Write-Host "The setup completed successfully at"(Get-Date).ToString()"."
  }
 }
 Catch {Write-Error "Installing Zen: " + [string]$error[0]; exit -1}
} else {Write-Warning "Cannot install Zen, unable to locate '$installFile'."; exit -1}
```

PowerShell script to silently install Zen Enterprise Server using Microsoft Installer (MSI)

```
# Sample script for silently installing Zen Enterprise Server Engine that
# automatically ensures that the required Microsoft Visual C++ 2019 runtimes
# are installed before invoking the Zen MSI installation.
#
# Requirements:
# 1) The PowerShell execution policy allows scripts to be executed.
# 2) The PowerShell session is running with Administrator privileges
# if UAC is enabled.
# 3) The Zen Server Engine installation files are located in a folder named
# "server" in the same location as this script.
# 4) The bundled Microsoft Visual C++ Runtime installers are invoked BEFORE the Zen
# installation is invoked.
$currentFolder= & {Split-Path $myInvocation.ScriptName}
$installVCRT32="$currentFolder\Data\ISSetupPrerequisites\{7BB553E0-BAA5-4184-965C-
AEEB89B82D46}\VC_redist.x86.exe"
$vc_redist32Log=join-path $env:TEMP "\vcredist2015_x86.log"
$vc_redist32Args="/quiet","/norestart","/log $vc_redist32Log"
$is64Bit=$False
if ([System.IntPtr]::Size -eq 4) {
 $installFile="$currentFolder\Data\Data\ActianZenv16EnterpriseServer32_x86.msi"
} else {
 $is64Bit=$True
 $installFile="$currentFolder\Data\Data\ActianZenv16EnterpriseServer64_x64.msi"
 $installVCRT64="$currentFolder\Data\ISSetupPrerequisites\{2F044C80-608C-4274-AB1D-
96D67E047307}\VC_redist.x64.exe"
 $vc_redist64Log=join-path $env:TEMP "\vcredist2015_x64.log"
 $vc_redist64Args="/quiet","/norestart","/log $vc_redist64Log"
}
$msiexecArgs="/i '"$installFile'"","/qn","REBOOT=ReallySuppress","PSQL_PREREQS_INSTALLED=Y"

# Always install 32-bit MS VCRT.
if (Test-Path "$installVCRT32") {
 Try {
  Write-Host "Starting Microsoft Visual CRT (x86) installation at"(Get-Date).ToString()"..."
  $proc=(Start-Process -FilePath $installVCRT32 -ArgumentList $vc_redist32Args -Wait -PassThru)
  $handle=$proc.Handle # cache proc.Handle http://stackoverflow.com/a/23797762/1479211
  $proc.WaitForExit();
  If ($proc.ExitCode -ne 0) {
   Write-Error "The MS install returned error code: $($proc.ExitCode)"
  } else {Write-Host "The MS install completed successfully at"(Get-Date).ToString()"."}
 }
 Catch {Write-Error "Unknown error: " + [string]$error[0]; exit -1}
}

if ($is64Bit) {
 # Install 64-bit MS VCRT
 if (Test-Path "$installVCRT64") {
  Try {
   Write-Host "Starting Microsoft Visual CRT (x64) installation at"(Get-Date).ToString()"..."
```

```
   $proc=(Start-Process -FilePath $installVCRT64 -ArgumentList $vc_redist64Args -Wait -PassThru)
   $handle=$proc.Handle # cache proc.Handle http://stackoverflow.com/a/23797762/1479211
   $proc.WaitForExit();
   If ($proc.ExitCode -ne 0) {
    Write-Error "The MS install returned error code: $($proc.ExitCode)"
   } else {
    Write-Host "The MS install completed successfully at"(Get-Date).ToString()"."
   }
  }
  Catch {Write-Error "Unknown error: " + [string]$error[0]; exit -1}
 }
} else {Write-Host "Skipping installation of Microsoft Visual C++ (x64) on 32-bit system..."}

# Zen install using MSI file.
if (Test-Path "$installFile") {
 Try {
   Write-Host "Starting Zen installation at"(Get-Date).ToString()"..."
   $proc=(Start-Process -FilePath "msiexec.exe" -ArgumentList $msiexecArgs -Wait -PassThru)
   $handle=$proc.Handle # cache proc.Handle http://stackoverflow.com/a/23797762/1479211
   $proc.WaitForExit();
   If ($proc.ExitCode -ne 0) {
     Write-Error "The install returned error code: $($proc.ExitCode)"
   } else {Write-Host "The install completed successfully at"(Get-Date).ToString()"."}
 }
 Catch {Write-Error "Unknown error: " + [string]$error[0]; exit -1}
} else {Write-Warning "Cannot install Zen, unable to locate '$installFile'."; exit -1}
```

Handling Product Updates

Product updates for Zen are compressed, self-extracting .zip archives. Files in these archives are used to update, or patch, an existing Zen installation.

This section covers the following topics:

- Installing a Product Update
- Removing a Product Update
- Special Considerations for Silent (or Basic UI) Installation

Installing a Product Update

When you run the .exe file for a product update, it asks you to choose a location for the extracted update files (default is %temp% folder) and whether to launch the update executable (default is yes).

When one of the update executables is launched, it verifies the bitness of the installed Zen and launches the appropriate Microsoft patch package (.msp).

- Update_Zen_Client.exe
- Update_Zen_EnterpriseServer.exe
- Update_Zen_Workgroup_Engine.exe

Example: Installing a Product Update Silently

```
Update_Zen_Client.exe /s /v"/qn /l*v \"%temp%\Zen_Client_silent_update.log\""
```

Note: All properties after **/v** are passed to msiexec. The **/v** option has no space after it. The string of properties to be passed must be enclosed in double quotation marks. If one of the properties is a path name that may include spaces, the path must also be enclosed in double quotation marks, and they must both be escaped with a backslash (\"), as shown in the example above.

Removing a Product Update

On Windows systems, in most cases a Zen product update can be "rolled back" by uninstalling it. This removal restores Zen binaries to their versions before the product update. See the note in this section about unremovable patches.

To remove a product update for Zen from the command line, you need the following:

- Original installation package (.msi) or the Product Code GUID for the installation. Each Zen installation type has a different product code GUID.
- Original Microsoft patch (.msp) file.

Either of the following commands can remove a product update:

- `msiexec /package <{path_to_MSI} or Product_Code_GUID> /uninstall <path_to_MSP> /l*v "%temp%\uninstall_patch.log"`
- `msiexec /i <{path_to_MSI} or Product_Code_GUID> MSIPATCHREMOVE=<path_to_MSP> /l*v "%temp%\uninstall_patch.log"`

Note: In certain cases, a Zen patch cannot be removed. A warning message states "Uninstallation of the patch package is not supported," which also is entered in the installation log file. To remove the patch, uninstall and reinstall Zen, and then apply only the patches you desire.

Windows does not list Zen updates under Programs and Features to ensure that you are prompted to elevate to Administrator when uninstalling or modifying the Zen installation.

Special Considerations for Silent (or Basic UI) Installation

Two types of installations cannot launch other installations automatically:

- A silent installation
- An MSI Basic UI installation, which is a reduced UI installation initiated using the MSI command line option **/qb** or **/passive** and displays only a progress bar while installing

In both these cases, unlike an installation executed from the installer executable UI, the installation does not ensure that all needed patches are applied. When you install silently or in Basic UI mode, you may need to apply these patches independently to make sure the product is complete.

The next two topics give examples of such independent updates:

- Product Patch Before an Upgrade
- Product Patch for a Current Release

Product Patch Before an Upgrade

An *upgrade* moves the product version from a previous release to the current release, such as 11.31 to 12.00. You may need to apply a patch *before* you upgrade the release. Patches are provided as .msp files. See the release notes for the current release, which lists any required patches. Be sure to install these first before upgrading.

For example, an optional patch is available for upgrading from a major release to a service pack. Install this patch if you want to keep personalized settings applied in the original release. The patch may be available only from the Actian website, so you must download the .msp file to apply it.

Use the following command to silently install the optional patch:

```
Product_type_platform.msp /qn /l*v "%temp%\product_beforeupdate.log"
```

where:

Product is the product, and *type* and *platform* are the product type and the platform on which the product runs. For example, ActianZenv16Patch_EnterpriseServer32_x86.msp is for the Enterprise Server on a 32-bit platform.

Product Patch for a Current Release

The current release may require a patch *after* the silent installation. If a patch is required, you can find its .msp file in the installation media in the ~\Redist\Data folder.

Install any .msp file in that folder to complete installation of the current release. For example, the following command silently installs a patch for an Enterprise Server on a 64-bit platform:

```
path_to_installation_media\Redist\Data\ActianZenv16Patch_EnterpriseServer64_x64.msp /qn /l*v
"%temp%\Zenv16_InstalledPatch.log"
```