CO-LaN: CAPE-OPEN Laboratories Network

Expanding Process Modelling Capability through Software Interoperability Standards
Association loi de 1901 créée le 8 Février 2001

CAPE-OPEN Type Libraries Installers – Developer Guide

Summary

CO-LaN has developed a set of reusable installers for the CAPE-OPEN Type Libraries and associated .NET Primary Interop Assemblies (PIA). The installations are intended for use by all vendors of Process Simulation Software which implements the CAPE-OPEN interoperability standards and by end-users of CAPE-OPEN-compliant software to ensure consistent installation, registration and removal of the Type Libraries and PIAs of the CAPE-OPEN specifications on an end-user's machine. This document explains the requirements governing what the installers need to do; it describes the solutions technologies used to develop the installers; and, it describes how they should be used by software vendors delivering CAPE-OPEN compliant software.

ARCHIVAL INFORMATION

Title	CAPE-OPEN Type Libraries Installers - Developer Guide
Owner	Interoperability SIG
Location	https://colan.repositoryhosting.com/trac/colan_coidl/downloads/13
Document Unique Identifier	92B9D160-B969-11EE-9EC1-0800200C9A66
Distribution	Public
Status	Approved
Document Version Number	1.3
Created Date	2024-02-13
Revision Date	2024-01-29
Number of pages	21

Version History

Document Version Number	RFC Date	Release Date	Comments
0.1			Created by Michael Halloran
0.2			Edited by Michel Pons
0.3			Responses to v0.2 comments by Michael Halloran
0.4			Edited by Michel Pons
0.5			Response to comments from M Pons
0.6			Responses accepted by M. Pons
0.7			Updates following CAPE-OPEN Developer reviews
0.8			Updates revised
0.9			Further comments incorporated
0.99			Added information on Transforms.
1.0		2024-02-13	Approved for publication
1.1		2024-02-28	NSI script examples updated
1.2		2024-04-25	Added reference to NSIS example in RepositoryHosting
1.3		2024-01-29	Replaced the link on page 12 with sample WiX

IMPORTANT NOTICES

Trademark Usage

Many of the designations used by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in CO-LaN publications, and the authors are aware of a trademark claim, the designations have been printed in caps or initial caps.

Contents

1. Introduction

The CAPE-OPEN interface specifications define an interoperability standard allowing Process Simulation environments, Unit Operation models, Physical Property systems and other simulation components from different providers to work together. The interface standards are defined in IDL files which are then compiled to Type Libraries, and the Type Libraries are referenced by the implementations of CAPE-OPEN compliant software. Initially CAPE-OPEN implementations were implemented using Microsoft COM technology but .NET implementations are also being made which raises the requirement for COM .NET interoperability, and interoperability between different .NET versions. These requirements are satisfied by providing Primary Interop Assemblies (PIA) which are .NET assemblies that provide a .NET definition of the CAPE-OPEN interfaces derived from the CAPE-OPEN Type Libraries and which provide a single, fixed set of .NET types that correspond to the CAPE-OPEN interfaces.

The CAPE-OPEN interface specifications are maintained by CO-LaN with Special Interest Groups being responsible for updates to the IDL statement of the interfaces as part of advancing and clarifying the standards.

The Type Libraries and PIAs implement the shared definition of the standards on which all providers of CAPE-OPEN software depend. To ensure consistent installation and removal of these shared CAPE-OPEN components on a computer on which CAPE-OPEN software from different software vendors is installed, CO-LaN has developed various installation packages for the Type Libraries and PIAs. The packages install the necessary files and create the necessary registry entries to define the CAPE-OPEN interfaces and to provide access to them via Type Libraries and PIAs on the Microsoft Windows Operating Systems in both 32- and 64-bit form. The installation packages are based on Windows Installer technology and are provided in the form of reusable merge modules (.msm) intended for use by software providers and as installation packages (.msi) for manual installation by end-users of CAPE-OPEN software.

2. Glossary

- IDL - Interface Definition Language
- PIA – Primary Interop Assembly
- COM – Component Object Model
- TLB – Type Library
- WiX – Windows Installer XML
- NSIS – Nullsoft Scriptable Install System
- MSM – Windows Installer Merge Module
- MSI – Microsoft Installer (file extension)
- GAC – Global Assembly Cache, location for storing shared ,NET Assemblies.
- SIG – Special Interest Group within CO-LaN

3. Intended Audience

This document is intended for vendors and developers of CAPE-OPEN compliant software who need to deliver the Type Libraries and PIAs that contain the COM and .NET implementations of the CAPE-OPEN specifications as part of delivering their own software. It is expected that readers will understand install technology and will be familiar with the tools and concepts used to install software on the Microsoft Windows platform.

4. Requirements

The purpose of the CAPE-OPEN interface specifications is to provide an interoperability standard to be implemented by the vendors of Process Simulation software to allow end users of Process Simulation to combine different software packages from multiple vendors at will.

The primary requirement for CO-LaN and for software vendors is that interoperability is achieved and then maintained through the life cycle of all CAPE-OPEN compliant software on a user's machine. It is a given that there will be CAPE-OPEN software from more than one vendor on a CAPE-OPEN user's machine, since the objective is interoperability between software from different providers. It is also the case that the CAPE-OPEN software installed on a machine will change over time with software being added and deleted.

Interoperability depends on vendors using a single definition of the CAPE-OPEN interfaces during the development cycle. Interoperability also depends on this same definition of the CAPE-OPEN interfaces being installed on end-user machines.

Furthermore, interoperability depends on the definition of CAPE-OPEN interfaces staying on an end-user's machine as vendor software is installed and uninstalled as part of the normal lifecycle of software on a machine.

The CAPE-OPEN Type Libraries and PIAs are required to support 32- and 64-bit software running on 32- and 64-bit editions of the Windows Operating Systems. All versions of Windows from Windows XP onwards and the corresponding versions of Microsoft COM need to be supported. All versions of the .NET Framework need to be supported – current version is 4.6.1.

The 0.9 and 0.93 versions of the CAPE-OPEN standards are deprecated and do not need to be installed on developers' or end-users' machines.

Software developed by CO-LaN is licensed using the Non-Profit Open Software License (https://opensource.org/licenses/NPOSL-3.0). The interactive installers must require that the terms of this License are accepted by the user installing the software. Where the installers are embedded in other installation packages then the Licence terms and CO-LaN ownership must be acknowledged in an attribution file as part of the product installation. The assumption is that software vendors will already deliver or provide attribution files for other 3rd-party components that also require attribution. Attribution files simply contain a list of copyright acknowledgements. They may be provided online, or included in documentation, or installed as files with the software. Examples of attribution best practice are given here:

https://wiki.creativecommons.org/wiki/Best_practices_for_attribution

5. Use Cases

The Type Library and PIA installation packages developed by CO-LaN provide the single implementation of the CAPE-OPEN standards that vendors and users need to achieve interoperability. They are provided in various forms to serve the following Use Cases:

5.1. Software vendor developing CAPE-OPEN compliant software

As a Software Vendor developing CAPE-OPEN compliant software using COM and/or .NET, I need access to the Type Libraries and PIAs containing the interface definitions of the CAPE-OPEN standard.

I want to use the correct Type Libraries and PIAs when developing CAPE-OPEN software.

I want the CAPE-OPEN definitions to be automatically included and easily identifiable in the lists of COM and .NET components that can be referenced from my software projects in MS Visual Studio.

5.2. Software vendor with CAPE-OPEN compliant software already installed

As a Software Vendor with CAPE-OPEN compliant software already installed using earlier Type Libraries installers from CO-LaN, or using a Type Library installer developed independently, I want to be sure that uninstalling the earlier packages and installing the current one will not break any CAPE-OPEN software from any vendor installed using the new Type Library and PIA installers.

5.3. Software vendor delivering CAPE-OPEN software using Windows Installer

As a software vendor who has developed CAPE-OPEN compliant software using the CO-LaN Type Libraries and PIAs, I want to include the same Type Libraries and PIAs as part of the installation of my software.

Since I'm already using a tool for creating Windows Installer packages, I want to use a mechanism that will easily integrate with that tool. My installer may allow per-user and per-machine installations so the CAPE-OPEN installer must allow the same choice at installation time.

When my installation package is uninstalled, I want the CAPE-OPEN Type Libraries and PIAs and all their registry entries to be uninstalled as well, unless they are being referenced by other software packages.

5.4. Software vendor delivering CAPE-OPEN software using other installation technologies

As a software vendor who has developed CAPE-OPEN compliant software using the CO-LaN Type Libraries and PIAs, I want to include the same Type Libraries and PIAs as part of the installation of my software

I cannot use Windows Installer Merge Modules so I need a standalone installation package which can be invoked by my installer and installed silently. My installer may allow per-User and per-Machine installations so the CAPE-OPEN installer must allow the same choice at install time.

When my installation package is uninstalled, I want the CAPE-OPEN Type Libraries and PIAs and all their registry entries to be uninstalled as well unless they are being referenced by other software packages.

5.5. Manual installation by an end-user

As a user of CAPE-OPEN-compliant software I need to be able to install CAPE-OPEN Type Libraries and PIAs to repair CAPE-OPEN software interoperability. In an enterprise environment, I need to be able to install on a per-User basis.

6. Solution Design

The CAPE-OPEN Type Library installer consists of two parts: a reusable “merge module” and an installable package which uses the merge module to provide a mechanism for manual installation of the CAPE-OPEN Type Libraries and PIAs. Both components are provided for 32-bit installation and for 64-bit installation making four deliverables in total. In this document the differences between 32-bit and 64-bit will be ignored unless they are important to the topic in order to avoid unnecessary duplication.

All the logic and data for the correct installation and removal is contained in the merge modules. The separate installation packages use the merge modules and add a User Interface – including presentation of an End User License Agreement - and the logic to handle per-User and per-Machine installations.

The merge module installer creates all the registry entries required by the CAPE-OPEN definitions and installs the Type Libraries for CAPE-OPEN version 1.0 and 1.1 and the corresponding PIAs. It also creates the registry entries for the CAPE-OPEN Component Categories which are used to classify CAPE-OPEN software components. Where the merge module is installed multiple times on a single machine by different software vendors, Windows Installer will automatically reference count the installed components and correctly manage uninstall operations so that CAPE-OPEN Type Libraries and PIAs are not removed while there are outstanding references to them.

The interactive installation packages deliberately make use of the merge modules so that the correct reference counting behavior will result when install and uninstall operations are done manually or through scripting. The Windows Installer Reference Counting means that it is safe for a vendor

package to uninstall the CAPE-OPEN components as part of their own uninstall, either through direct dependency on the merge modules or through scripted use of the MSI.

The CAPE-OPEN Type Library installers are built using the Windows Installer system because it is supported by Microsoft; the WiX toolset makes Windows Installer system available at no cost; it supports a mechanism for reuse of installer components through merge modules; and the resulting standalone installers can be integrated directly with other installer technologies. It is recommended that CAPE-OPEN Software providers use the provided installation packages for these reasons.

Where a vendor uses the Windows Installer system – for example, through WiX or InstallShield or Visual Studio - it is recommended that the CO-LaN MSM files are used. Where other installer technologies are used, which do not support the use of Windows Installer Merge Modules, it is recommended that new MSI packages are created and then are invoked through scripting. This ensures that the correct reference counting behavior is maintained. The simplest way to create such a package is to make and apply a Transform (.mst file) to the supplied MSI. Steps for doing this are described in section 7.

The alternative to using the supplied MSM or MSI files would be to reverse engineer the installers or to develop an independent installer for the Type Libraries. This is not recommended under any circumstances. Doing this means losing the all benefits that come from the use of Windows Installer system and carries the risk of introducing differences in the way the CAPE-OPEN Type Libraries and PIAs are installed. It will also cause software failures when packages are uninstalled.

The CAPE-OPEN Type Library and PIA Installers support per-User and per-Machine installation using a mechanism supported in Windows Installer 5.0 and later.

In summary, the benefits of this approach to providing reusable installers are:

- All software providers install the same CAPE-OPEN definitions in the same way to the same locations ensuring consistency.
- Windows Installer ensures that multiple installations of the same installation components behave correctly when software is uninstalled, removing the risk of one provider's uninstall removing the definitions and files that another provider is also relying on.
- Only a single Primary Interop Assembly created by CO-LaN gets installed. It is a characteristic of .NET COM interoperability that there can only be one Primary Interop Assembly for a given Type Library.
- Explicit definition of registry entries in the Windows Installer system results in reliable installation and removal of all Registry entries associated with CAPE-OPEN definitions rather than relying on Type Library registration.

Examples of how to use the merge module from different installation systems are provided in the next section

6.1. Merge Module Design

The merge modules have to install TLB and PIA files and then to create the registry entries required

to record the locations of the files and to make their contents accessible to other programs according to MS COM requirements.

File Locations

The installers use the standard folder CommonFilesFolder for installing the Type Libraries and they create a sub folder of CommonFilesFolder called CAPE-OPEN and then additional sub-folders called Reference Assemblies for the PIA files, Type Libraries for the TLBs and Licence for the Licence documentation.

```
<CommonFilesFolder>\CAPE-OPEN\Reference Assemblies
<CommonFilesFolder>\CAPE-OPEN\Type Libraries
<CommonFilesFolder>\CAPE-OPEN\Licence
```

The reason for using sub-folders is to ensure there is no risk of accidental uninstallation when earlier versions of the installers are uninstalled on a machine. The location of CommonFilesFolder depends on the installation context chosen at installation time. For a per-User install it resolves to:

```
Users\<user>\Appdata\Local\Programs\Common
```

For a per-Machine installation it resolves to:

```
\Program Files (x86)\Common Files\
```

when using the 32-bit installer on a 64-bit system, and:

```
\Program Files\Common Files
```

when using the the 64-bit installer or on an x86 system.

The installation folders are fixed and the user is not presented with the option to choose alternatives.

In a per-Machine installation the PIAs are also installed to the Global Assembly Cache.

Registry Entries

The location of the Reference Assemblies folder is recorded in the Windows Registry so that the assemblies can be found when a developer wants to add a reference to the CAPE-OPEN definitions in an MS Visual Studio project. The PIAs are registered for .NET versions 2.0, 4.0 and 4.5.

Entries are created for the CAPE-OPEN Component Categories. Component Category data cannot be expressed in IDL, other than as comments, and therefore this data is not present in the Type Libraries and PIAs.

Registry entries are created for the two Type Libraries defining the location of the files. The installer will overwrite any previous entry in the same installation context. Overwritten entries will not be restored on uninstallation. The Type Library FLAGS and HELPDIR entries are created but not populated.

Entries are created for each interface relating them to the GUIDs for the Type Libraries by version.

Per-User and Per-Machine installations

The merge module is designed following the conventions for Single Package Authoring which are support in Windows Installer 5.0 and beyond. Single Package Authoring allows a single installation to work for both per-User (single user) or per-Machine (all users) installation contexts.

Single Package Authoring is described here:

https://msdn.microsoft.com/en-us/library/windows/desktop/dd408068(v=vs.85).aspx

Single Package Authoring uses two WiX properties, `ALLUSERS` and `MSIINSTALLPERUSER,` to define installer behavior.

Setting `ALLUSERS=2` tells Windows Installer that the installer can install per-User or per-Machine
Setting `MSIINSTALLPERUSER=""` tells Windows Installer to perform a per-Machine install
Setting `MSIINSTALLPERUSER=1` tells Windows Installer to perform a per-User Install.

The use of these Properties is further described here:

https://msdn.microsoft.com/en-us/library/windows/desktop/dd408007(v=vs.85).aspx

The Installation Scope dialog shown when using the installer interactively sets these properties in response to the user's choice of installation scope. They can also be set when using the installer on the command line as shown in the examples for Command Line installation in Section 7.

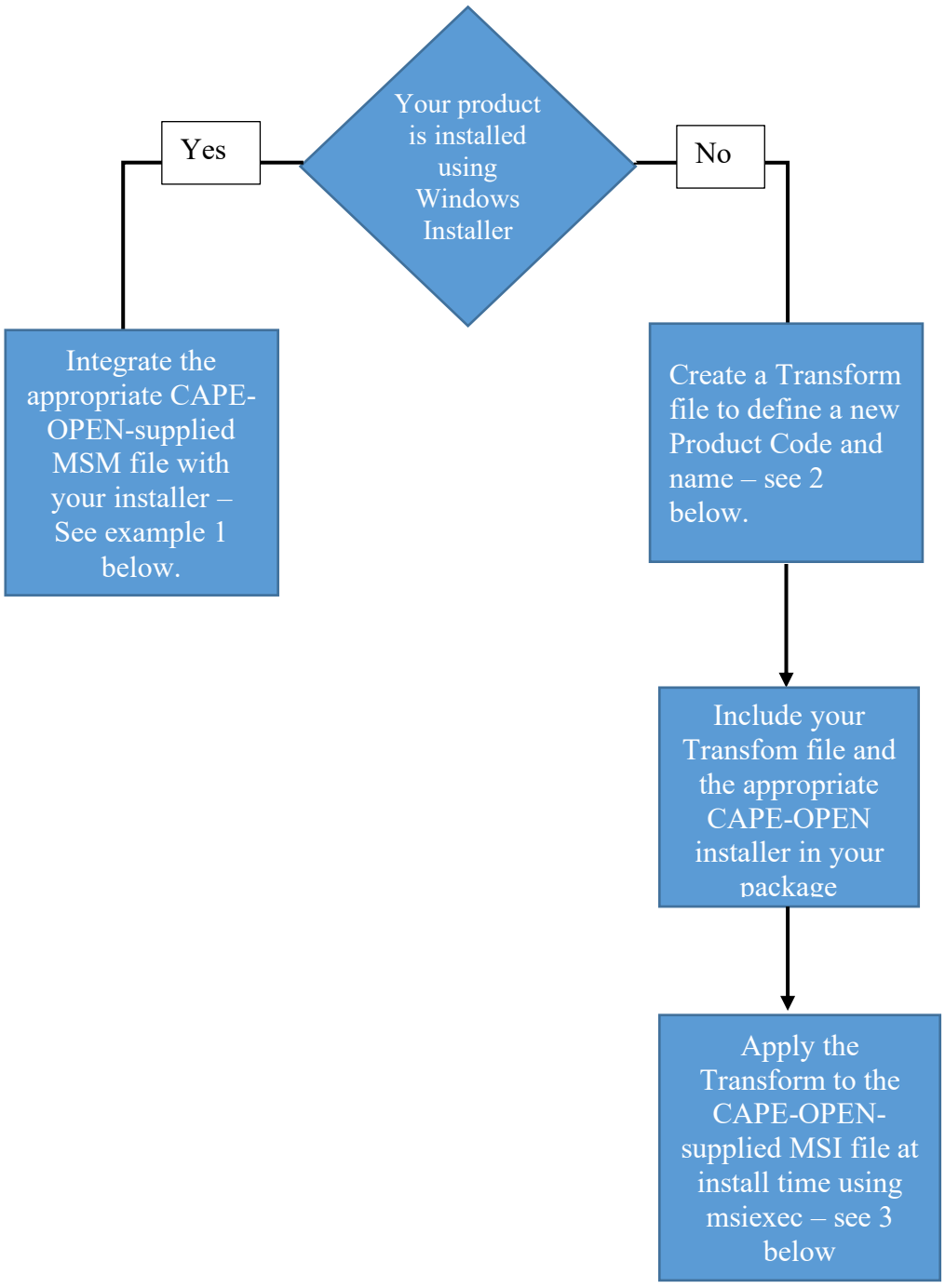
6.2. Interactive Installation Design

The interactive Installation package is built using the merge module and adds a User Interface to prompt for acceptance of the End User License Agreement and to allow a choice between per-User and per-Machine installations.

If the Installation package is invoked after a previous installation has completed, Repair and Remove options are presented. Repair will re-install the CAPE-OPEN files and registry entries, and Remove will uninstall them. The standard dialog used for this function also displays a Change option but since the CAPE-OPEN install does not have a selection of product features that can be changed, this feature is disabled.

7. Examples of how to use the installers

This flowchart summarizes the choices for using the installers:



1. WiX

WiX is documented and downloadable from wixtoolset.org

The interactive installer provides a good example for how to use the merge modules. The following is an example demonstrating the use of the merge module in WiX:

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">

<?define ProductID="QQQQQQQQ-RRRR-SSSS-TTTT-UUUUUUUUUUUU" ?>
<?define ProductUpgradeCode="AAAAAAAA-BBBB-CCCC-DDDD-EEEEEEEEEEEE" ?>
<?define ProductName="CAPE-OPEN for My Product" ?>
<?define Manufacturer="CAPE-OPEN for My Product" ?>

  <!--

  Current user install

  msiexec /i [InstallerName].msi /lv install.log ALLUSERS=2 MSIINSTALLPERUSER=1

  Remove

  msiexec /x {QQQQQQQQ-RRRR-SSSS-TTTT-UUUUUUUUUUUU} /q /lv uninstall.log ALLUSERS=2
MSIINSTALLPERUSER=1

  All users install

  msiexec /i [InstallerName].msi /lv install.log ALLUSERS=2 MSIINSTALLPERUSER=''

  Remove

  msiexec /x {QQQQQQQQ-RRRR-SSSS-TTTT-UUUUUUUUUUUU} /q /lv uninstall.log ALLUSERS=2
MSIINSTALLPERUSER=''


  -->

  <!-- note, change the product code (ProductID) when increasing the Version -->
  <Product Id="$(var.ProductID)" Name="$(var.ProductName)" Language="1033"
Version="1.0.0.0" Manufacturer="$(var.Manufacturer)"
UpgradeCode="$(var.ProductUpgradeCode)">
    <Package InstallerVersion="500" Compressed="yes" />

    <MajorUpgrade DowngradeErrorMessage="A newer version of this product is already
installed."  AllowSameVersionUpgrades="yes"/>

    <!-- default to current user - can be overridden by command line options - see above
-->
    <Property Id="ALLUSERS" Value="2"/>
    <Property Id="MSIINSTALLPERUSER" Value="1"/>

    <Media Id="1" Cabinet="product.cab" EmbedCab="yes"/>

    <Directory Id="TARGETDIR" Name="SourceDir">
```

```
      <Merge Id="CAPEOPENTLBx86msm" Language="1033" SourceFile="CAPE-OPEN Type Libraries
x86.msm" DiskId="1" />
      <Merge Id="CAPEOPENTLBx64msm" Language="1033" SourceFile="CAPE-OPEN Type Libraries
x64.msm" DiskId="1" />
    </Directory>

    <Feature Id="TypeLib" Title="CAPE-OPEN type libraries" Level="1" AllowAdvertise='no'
InstallDefault='local' Absent='disallow'>
      <MergeRef Id="CAPEOPENTLBx86msm"/>
      <MergeRef Id="CAPEOPENTLBx64msm"/>
    </Feature>

    <!-- hide from add/remove programs-->
    <Property Id='ARPSYSTEMCOMPONENT'>1</Property>

  </Product>

</Wix>
```

2. Create a Transform file to support scripted installation of the CAPE-OPEN MSI files for non-Windows Installers.

A Transform file allows modifications to be made to MSI files at the time they are installed. This is a common technique for customizing install packages. When a vendor installs the CAPE-OPEN Type Libraries using the MSI file – for example because the vendor uses NSIS rather than a Windows Install compatible tool – the package needs to be customized to give it a vendor-specific identity. This ensures that Windows Installer reference counting works correctly so that the uninstall of one vendor's product doesn't affect another's.

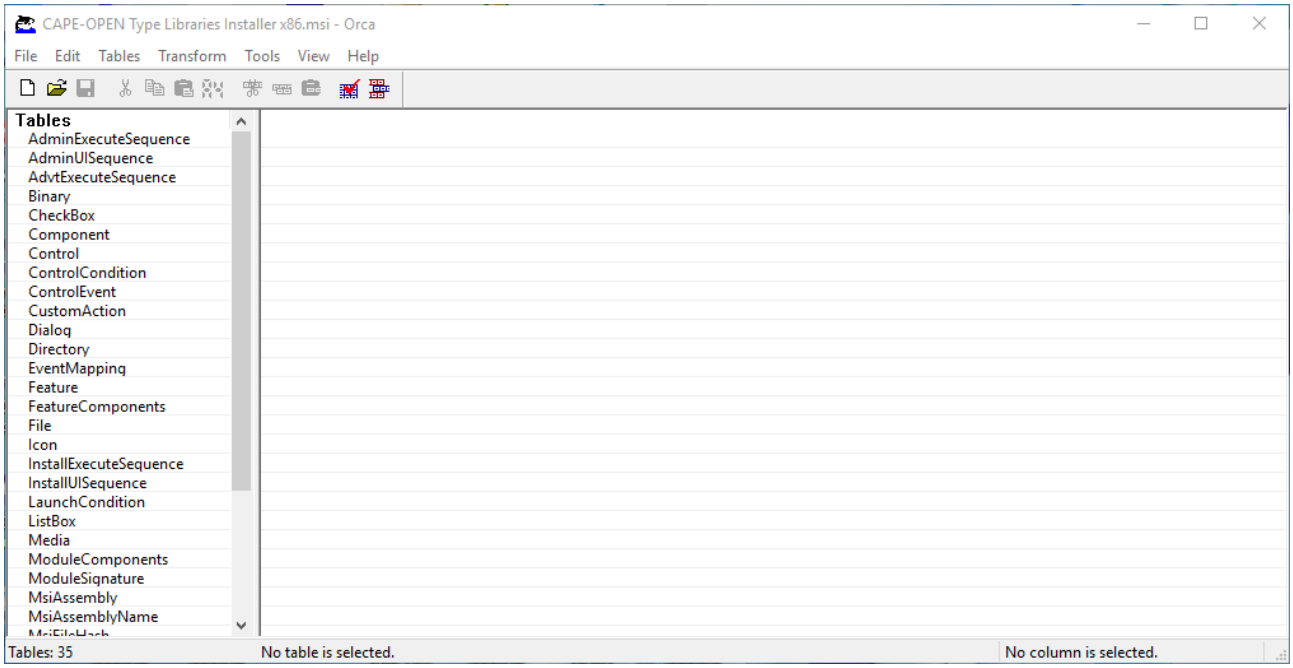
This section explains how to create a Transform file to create a vendor-specific version of the CAPE-OPEN Type Library Installer. The installer will be made vendor-specific by changing its ProductCode and its name.

The simplest way to create a Transform file is using the Orca tool from Microsoft. Orca is part of the Microsoft Windows SDK but is also available as part of a much smaller download for the Windows Installer 4.5 SDK which can be downloaded here:

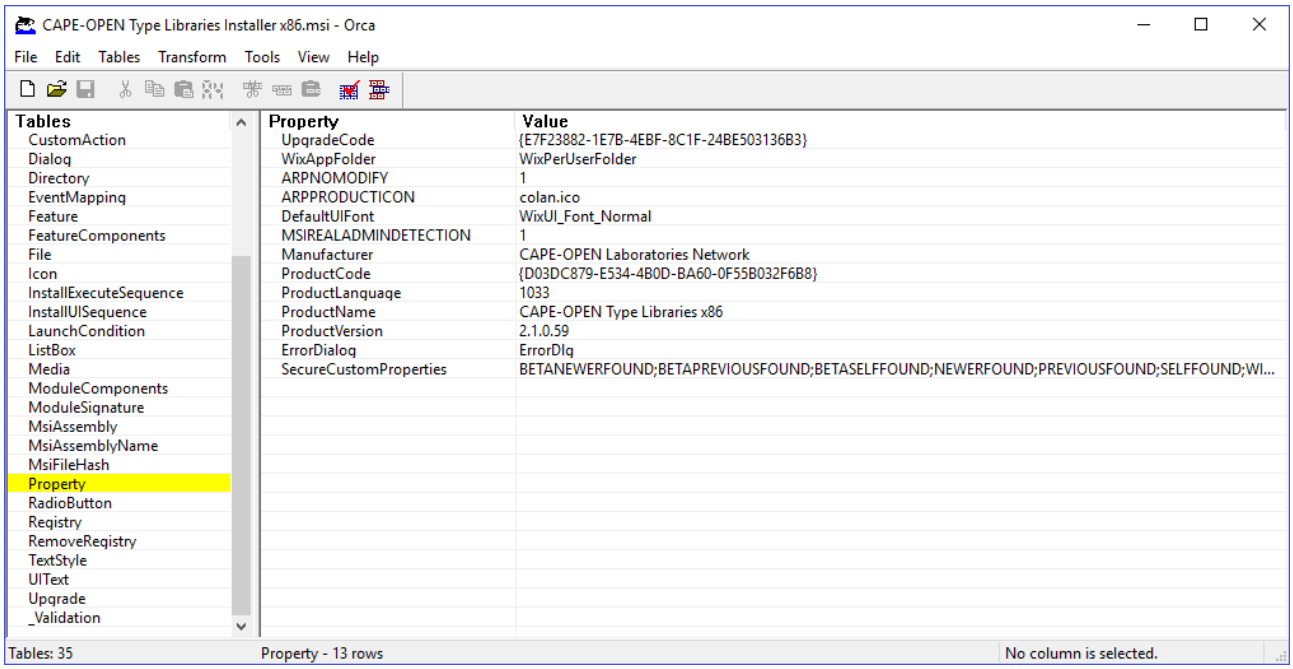
http://download.microsoft.com/download/7/c/4/7c426dfc-46e2-4ded-bab4-3b33600ad7d1/msi45sdk.msi
Open the downloaded file using 7Zip or a similar tool, and extract Orca.msi and install it. If you install msi45sdk.msi you will need to search the folders created to find Orca.msi.

To create a Transform file:

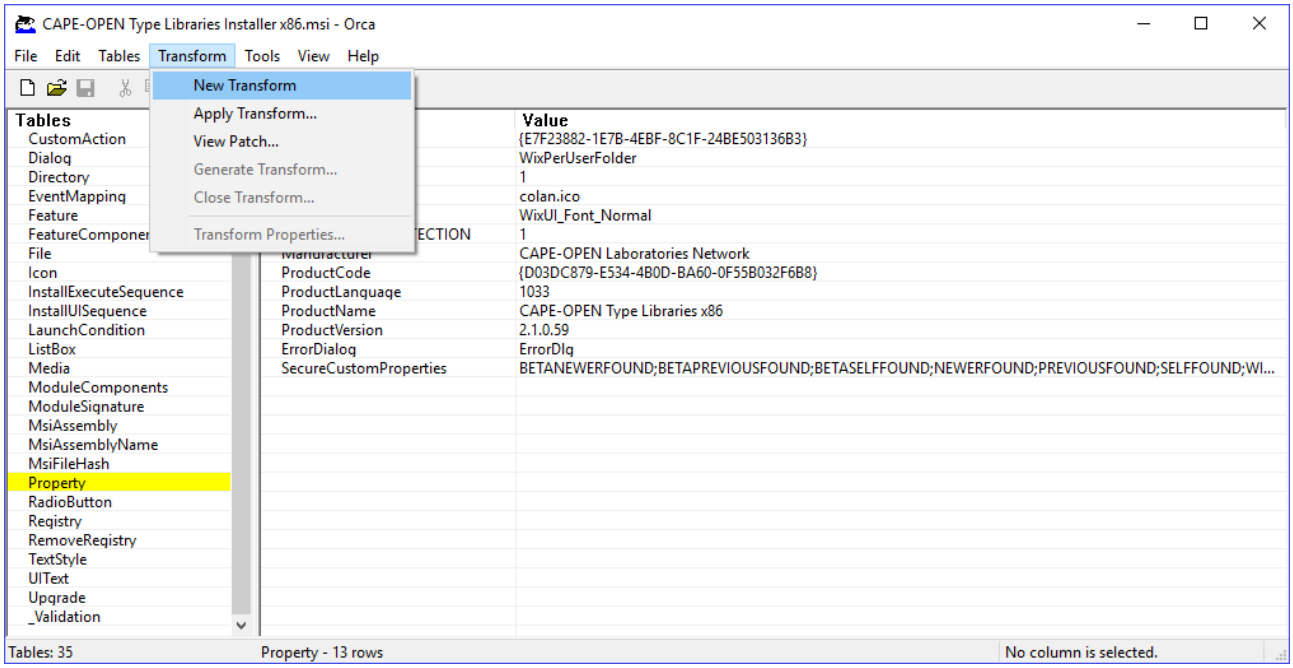
Start Orca and use File Open to open CAPE-OPEN Type Libraries Installer x86.msi (or CAPE-OPEN Type Libraries Installer x64.msi for 64-bit installation):



In the list of Tables scroll down and select Property:

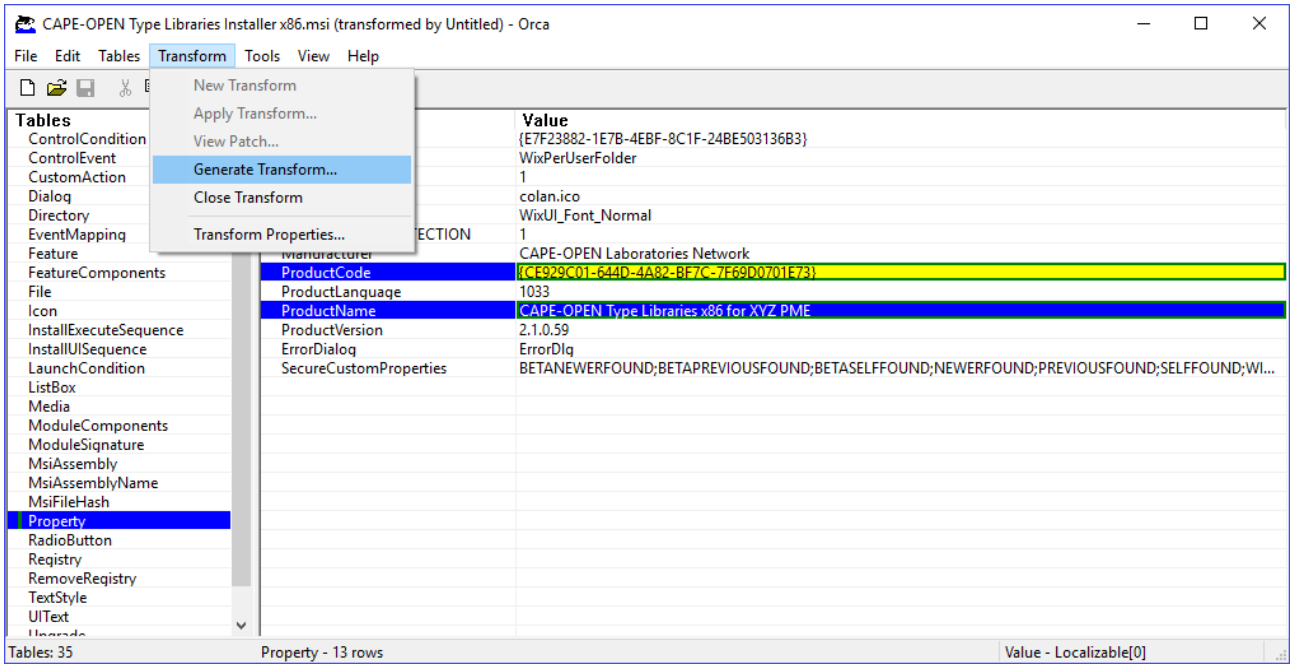


On the Transform menu select New Transform:

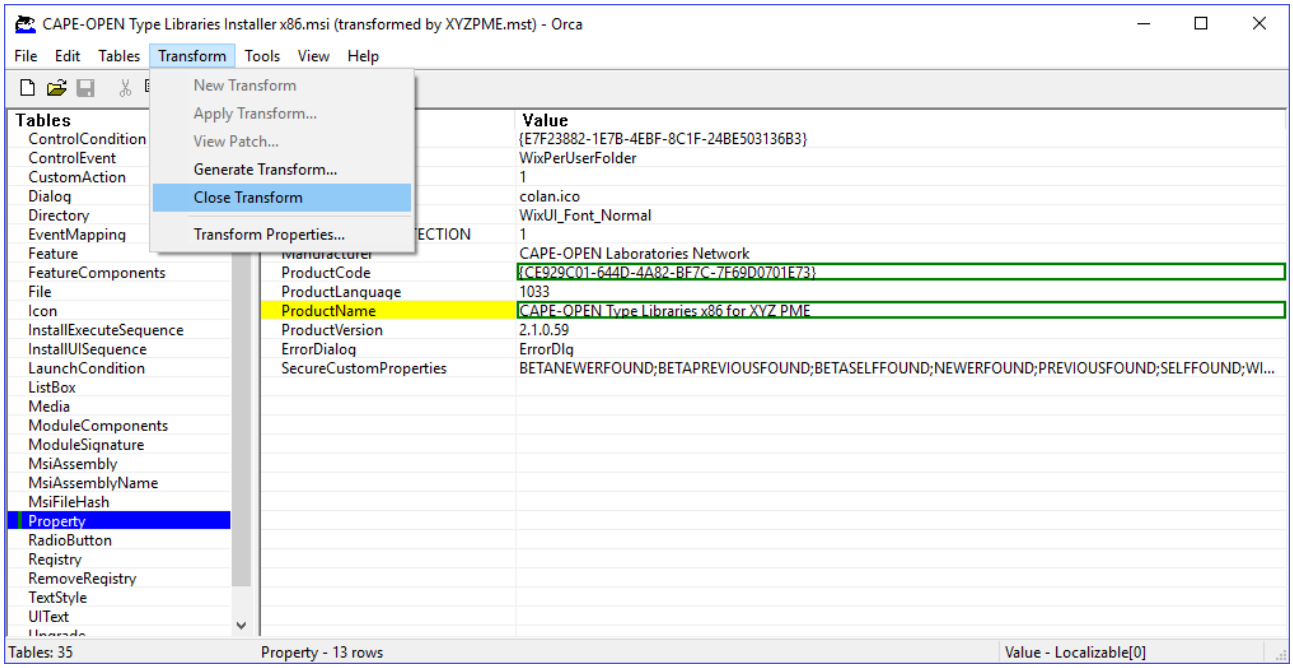


Modify the GUID for ProductCode and the ProductName. (ProductName is modified to make it easy to identify the customized installer in the Control Panel Add/Remove Programs list). To edit a Property value, click on the current value and edit to make changes.

Then use Generate Transform on the Transform menu to create the Transform file.



Use Close Transform on the Transform menu to end the Transform creation process.



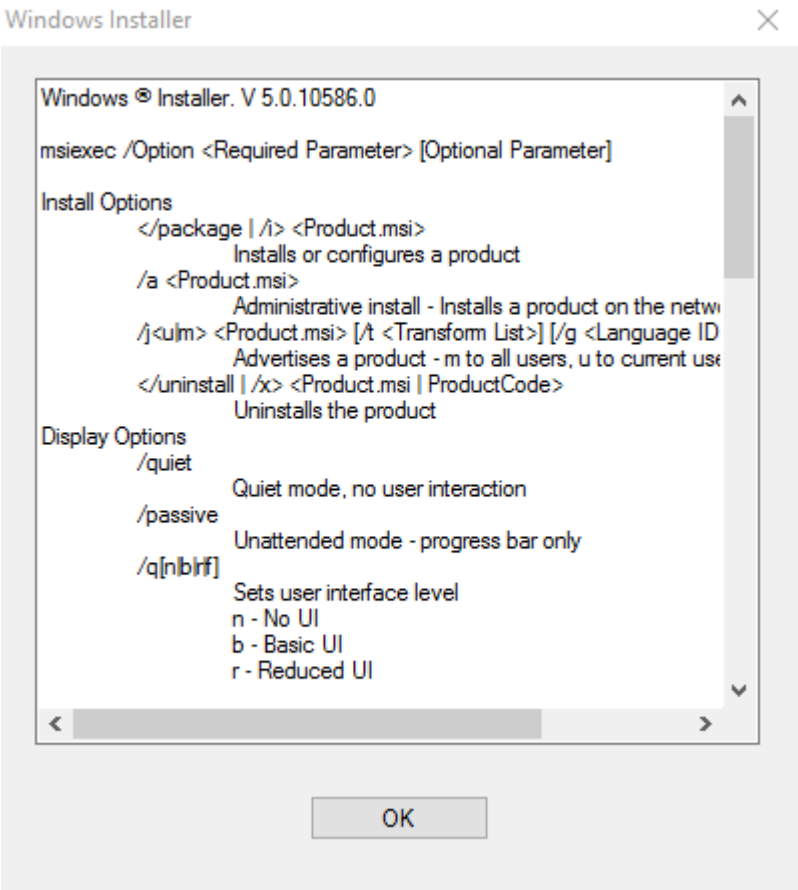
After this step Orca redisplays the original CAPE-OPEN msi file which has not been changed. Orca can now be closed.

3. Manual Installation using the Windows Command line.

Command line installation of an MSI on Windows is possible using the `msiexec` command. To run the command first open a Command Prompt window. The command:

```
msiexec /?
```

displays the following dialog which shows which version of Windows Installer is installed on the machine and the command arguments and options:



More details on the arguments and options can be found here:

https://msdn.microsoft.com/en-gb/library/windows/desktop/aa367988(v=vs.85).aspx

The following examples all assume that the MSI files referred to are in the current directory. If they are not then the full path to the file will need to be included within the “” characters in the value passed for the `/i` command argument.

The examples all use the `/q` option on the assumption that the reason for using the command line is to run the install in a batch mode or from within a non-Windows Installer installation system – see the NSIS examples below. The `/q` option and its variants are used to suppress the installers User Interface dialogs. If the `/q` option is not used in the examples below then the installers will run interactively with all dialogs displayed.

The examples all set the TRANSFORMS and MSINEWINSTANCE properties on the assumption that msiexec is being used as part of a script within another installer such as NSIS.

For a per-User install of the 32-bit installer, customized for use by XYZPME, on a 32- or 64-bit Windows OS with Windows Installer 5.0 installed use the command line:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x86.msi" /q TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=2 MSIINSTALLPERUSER=1
```

or for a 64-bit installation:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x64.msi" /q TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=2 MSIINSTALLPERUSER=1
```

The `/q` option completely suppresses all user interface.

For a per-Machine install of the 32-bit installer on a 32- or 64-bit Windows OS with Windows Installer 5.0 installed use the command line:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x86.msi" /qb TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=2 MSIINSTALLPERUSER=""
```

or for a 64-bit installation:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x64.msi" /qb TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=2 MSIINSTALLPERUSER=""
```

Note the use of the `/qb` option in the per-Machine examples. This option allows a basic UI only and is required so that the Windows User Account Control (UAC) dialog is displayed to allow the user to confirm the installation. Just using /q on its own for a per-Machine installation will cause the installation to fail with error 1603 because the default (silent) response to the UAC dialog is "NO".

Alternatively a silent per-Machine install using `/q` will work if `msiexec` is run with Administrator privileges, for example by using "Run as Administrator" to start the Command Prompt used to run the msiexec command. It will also work if Group Policy on the machine allows an MSI to always run with elevated privileges.

On a machine where the version of Windows Installer is earlier than 5.0 use the following commands. For a per-User 32-bit install :

```
msiexec /i "CAPE-OPEN Type Libraries Installer x86.msi" /q TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=""
```

or for a 64-bit installation:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x64.msi" /q TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=""
```

For a 32-bit per-Machine install use:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x86.msi" /qb TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=1
```

or for a 64-bit installation:

```
msiexec /i "CAPE-OPEN Type Libraries Installer x64.msi" /qb TRANSFORMS=XYZPME.mst
MSINEWINSTANCE=1 ALLUSERS=1
```

With any version of Windows Installer, to uninstall the 32-bit installer use:

```
Msiexec /x "CAPE-OPEN Type Libraries Installer x86.msi" /q TRANSFORMS=XYZPME.mst

Or

Msiexec /x {Your-ProductCode-GUID}
```

Or for x64:

```
Msiexec /x "CAPE-OPEN Type Libraries Installer x64.msi" /q

Or

Msiexec /x {Your-ProductCode-GUID}
```

Note that when uninstalling a per-Machine installation the `/qb` option is required so that the UAC dialog is displayed. As in the installation examples `/q` will work if msiexec is run with Administrator privileges or if Group Policy gives Windows Installer elevated privileges.

4. NSIS

NSIS is available from nsis.sourceforge.net

Note that the same logic applies for other non-Windows Installer systems, although the syntax will be different in each case.

NSIS does not use Windows Installer technology and is therefore not able to use merge modules directly. However, it is possible for NSIS to run external installation packages. As explained above in Section 6 – Solution Design, the MSI packages provided by COLAN use the merge modules to ensure that the correct reference counting behavior will result as packages which rely on the CAPE-OPEN Type Libraries and PIAs are installed and uninstalled on a machine. However, each where the MSI is used int his way a Transform file must be applied to create a new installation package which will ensure the correct reference counting behavior.

In an NSIS script the `ExecWait` command is used to execute Windows commands. To run an external installation package as part of an NSIS installation use `ExecWait` with the Windows Installer msiexec command. All the examples given above in the section describing Manual Installation using the Windows Command line can be used within NSIS using `ExecWait` with the appropriate syntax.

As an example, for a 32-bit per-Machine install on a machine with Windows Installer 5.0 installed use the command line:

```
  ; Install the CAPE-OPEN Type Libraries

  ExecWait '"msiexec" /i "$INSTDIR\CAPE-OPEN Type Libraries Installer x86.msi" /qb
TRANSFORMS="$INSTDIR\\XYZPMC.mst" MSINEWINSTANCE=1 ALLUSERS=$\"$\"'
```

Note that the two files referred to in this command: “CAPE-OPEN Type Libraries Installer x86.msi” and “XYZPME.mst” must be installed by the .nsi script to $INSTDIR so that this command will execute properly on the target machine during installation.

The other examples given in the Manual Installation section above, and the `msiexec` command line in general, can also be used following the same pattern. Note the use of the ‘ and “ characters and $\ as an escape sequence in the command passed to `ExecWait`.

An example NSIS script that uses these techniques can be found here:

http://colan.repositoryhosting.com/svn_public/colan_examples/MixerSplitterExamples/Installer/CPPInstallerWithTypeLibrariesMSI.nsi

To use commands appropriate to the Windows Installer version available on the user's machine use `GetDLLVersion` on MSI.dll. The `GetDLLVersion` function is described in this function reference:

http://nsis.sourceforge.net/Docs/Chapter4.html#ffunctioncommands

And here is an example: http://nsis.sourceforge.net/Detect_MSI_3.1