A Yokogawa Company

Development of the Multiflash CAPE-OPEN Interface

Behnam Salimi
Richard Szczepanski

CAPE-OPEN Annual Meeting October 2019

All about EXCELLENCE

Overview

- Some history
- Current release: Multiflash 7.0
- Development for future versions

Multiflash CO Development

- Started in 2000 as part of ***Global CAPE-OPEN*** project
  - MF 3.0/3.1 supported Thermo 1.0 specification
  - VB6
- First public release 2002 with MF 3.2
- 2006 MF 3.5: Added Thermo 1.1 support
- 2008 MF 3.8
  - Completely re-written in C++ by Jasper van Baten
  - Single dll supports Thermo 1.1 and 1.0 and all useful interfaces (Persistence, Edit…)
- 2014 MF 4.4: 64 bit implementation added
- 2015 First threadsafe prototype demonstrated with CO
- 2017 MF 6.2: first release version of threadsafe Multiflash dll

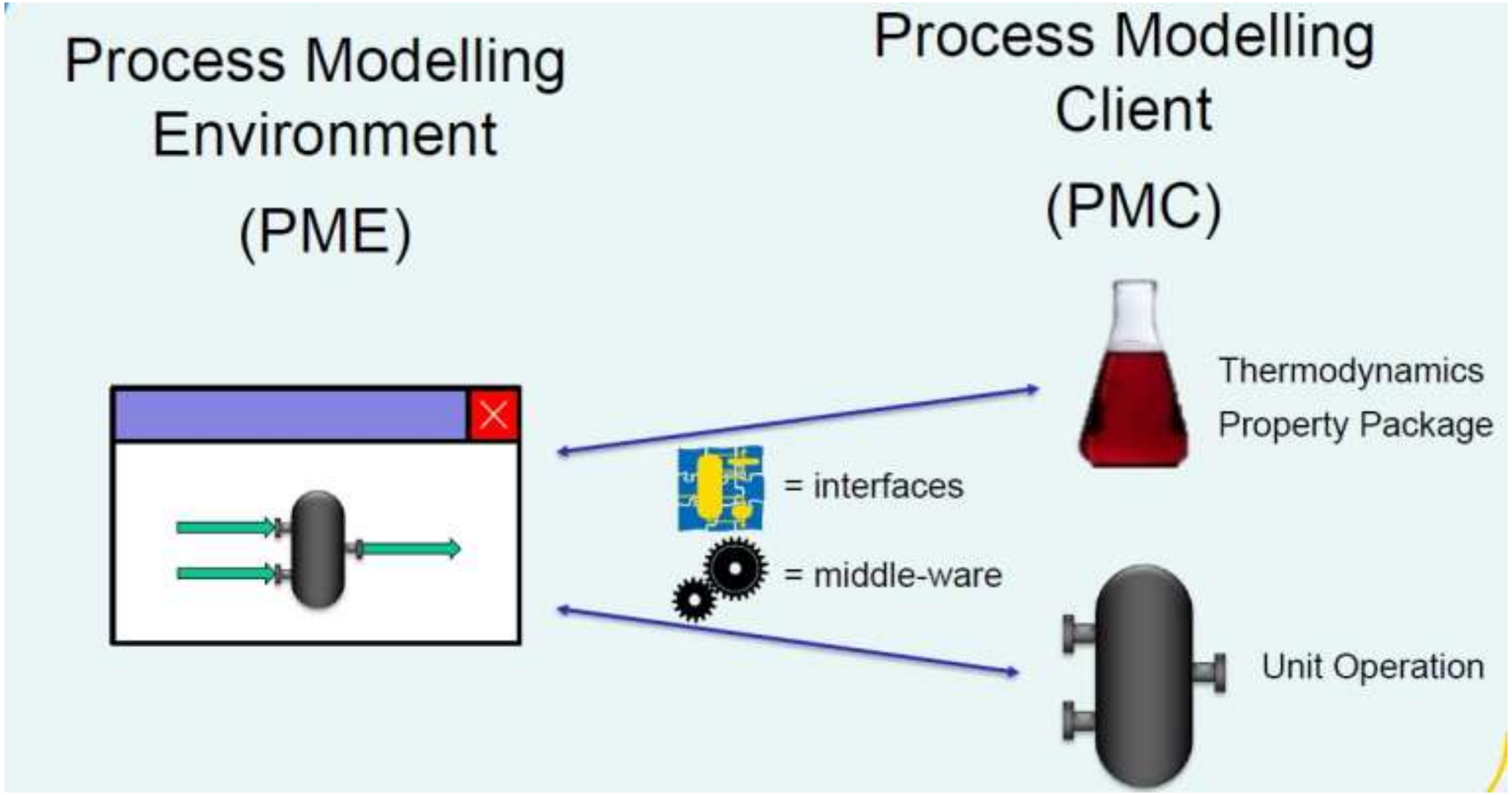
Multiflash 7.1

- Release by end of 2019
- Multiflash dll
  - Almost all calculations and models work in threadsafe mode
  - Updated mercury model, new options for cubic eos and CPA, EOS-CG
  - Better compatibility of models with other simulators
  - Python interface
  - Many other developments mostly concerning GUI
- CAPE-OPEN Interface
  - CO Type Libraries installed with CO-LaN installer
  - Uses threadsafe (MT) API for improved handling of multiple property packages

Future Developments

- Multiflash 7.2
  - COBIA
  - Themo 1.0 support by COM interface
- CO Thermo – my personal view
  - Focus on efficiency of implementation and ease of implementation
  - Aim to make CO Thermo as fast as a native implementation
  - New features
    - Properties and derivatives at specified V, T, n
    - Derivatives of phase equilibrium calculations
    - Support for parameter regression
    - Critical points

Why COBIA


Process Modelling Environment
(PME)
Process Modelling Client
(PMC)
Thermodynamics Property Package
= interfaces
= middle-ware
Unit Operation

Why COBIA

- Multiplatform / No dependence on operating system
- No dependence on commercial products
- Easier on programmers:
  - Data handling
  - Strong data typing
  - Less error prone
- More efficient
- Better error handling
- Support for legacy COM-based CAPE-OPEN

COBIA Interfaces Implementation Requirements

- COBIA Software Development Kit (SDK):
  - Stand-alone installation package
  - Set of tools to create and test software that utilises COBIA
  - To compile the source code of interfaces developed using COBIA IDL
  - To register COBIA components
  - To test developed software
  - It also includes examples, code generators, portions of the COBIA source code, etc.

- COBIA_CodeGen.exe (Command line app)

or

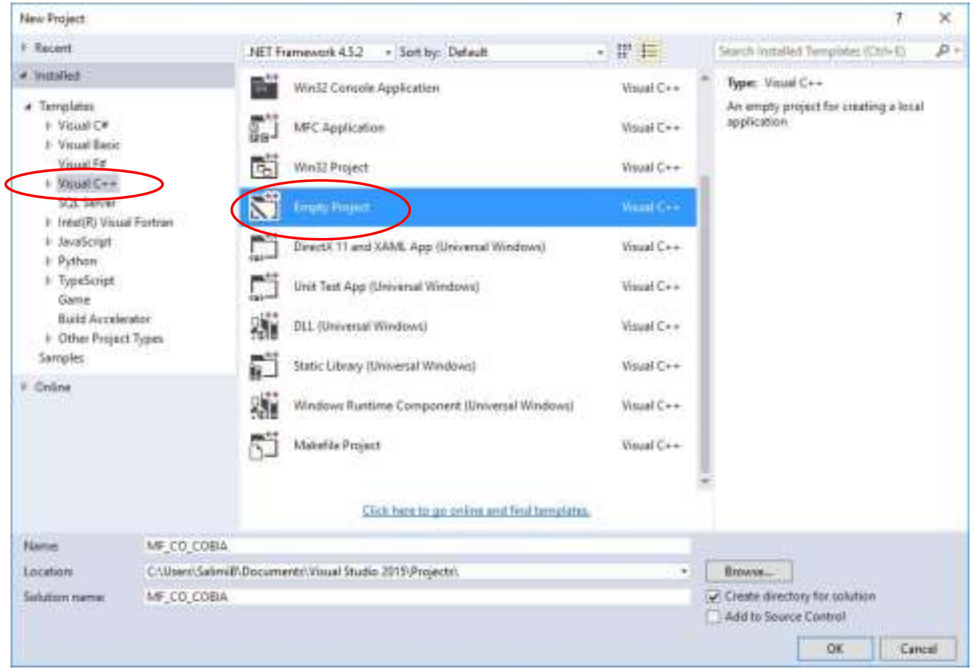
- AmsterCHEM COBIA Class Wizard Add-in for Visual Studio

AmsterCHEM COBIA Class Wizard

- Add-in for Visual Studio to help develop COBIA PMC
- Based on COBIA Code Generation Interface
- Generates classes and the definitions for all the functions in the classes.
- The COBIA Wizard does NOT generate ready to run PMCs!
- It provides a skeleton with Interfaces and Methods.
- The actions in the methods still have to be provided by the developer.
- Help from example document for creation of Unit Operation using the Class Wizard.

Visual Studio Configuration

In Visual Studio start an empty C++ project



New Project

Templates: Visual C++

.NET Framework 4.5.2 · Sort by: Default

Template	Type
Win32 Console Application	Visual C++
MFC Application	Visual C++
Win32 Project	Visual C++
Empty Project	Visual C++
DirectX 11 and XAML App (Universal Windows)	Visual C++
Unit Test App (Universal Windows)	Visual C++
DLL (Universal Windows)	Visual C++
Static Library (Universal Windows)	Visual C++
Windows Runtime Component (Universal Windows)	Visual C++
Makefile Project	Visual C++

Type: Visual C++

An empty project for creating a local application

Name: MF_CO_COBIA

Location: C:\Users\SalimiF\Documents\Visual Studio 2015\Projects\

Solution name: MF_CO_COBIA

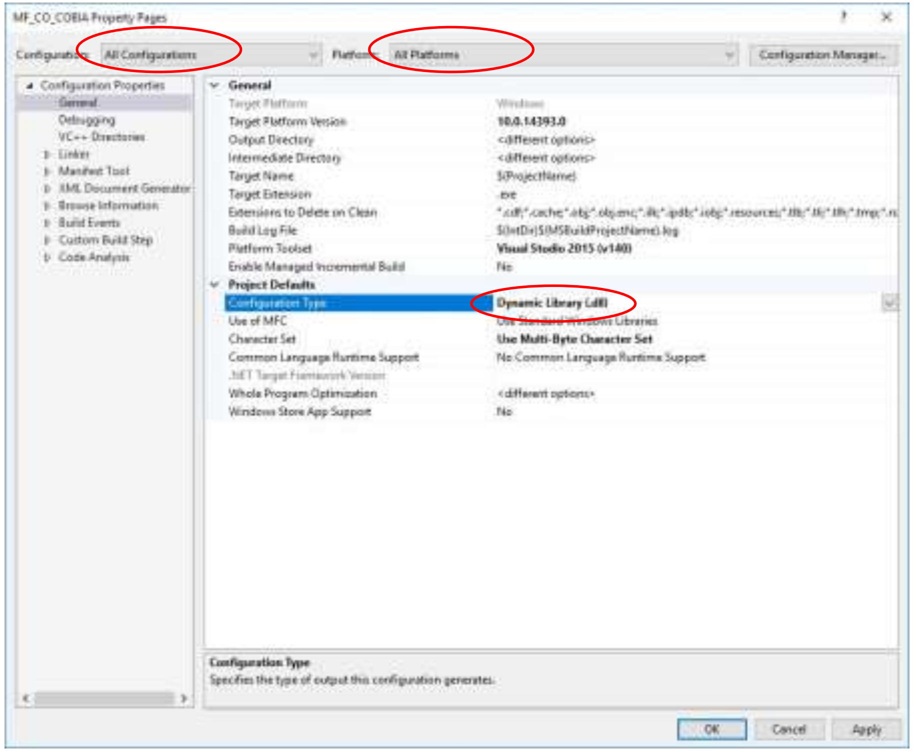
☑ Create directory for solution

☐ Add to Source Control

Visual Studio >= 2015

Visual Studio Configuration

In the project properties set configuration type to DLL Configuration



MF_CO_COBIA Property Pages

Configuration: All Configurations Platform: All Platforms

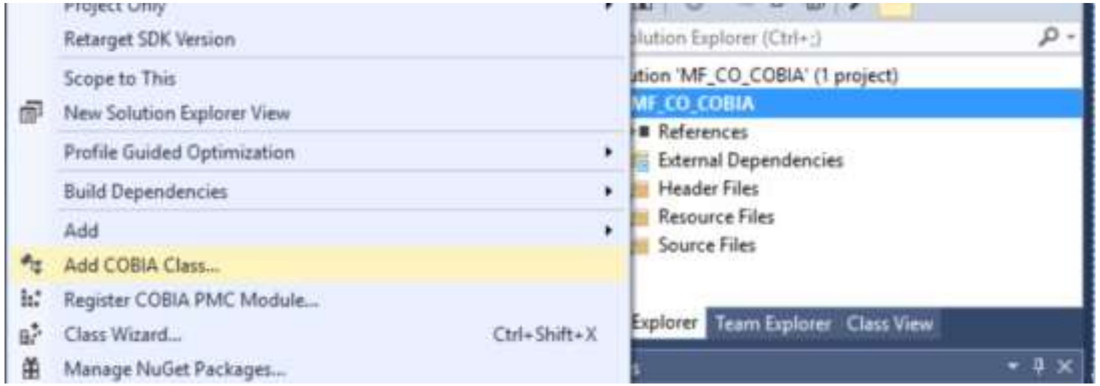
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - Linker
  - Manifest Tool
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Code Analysis

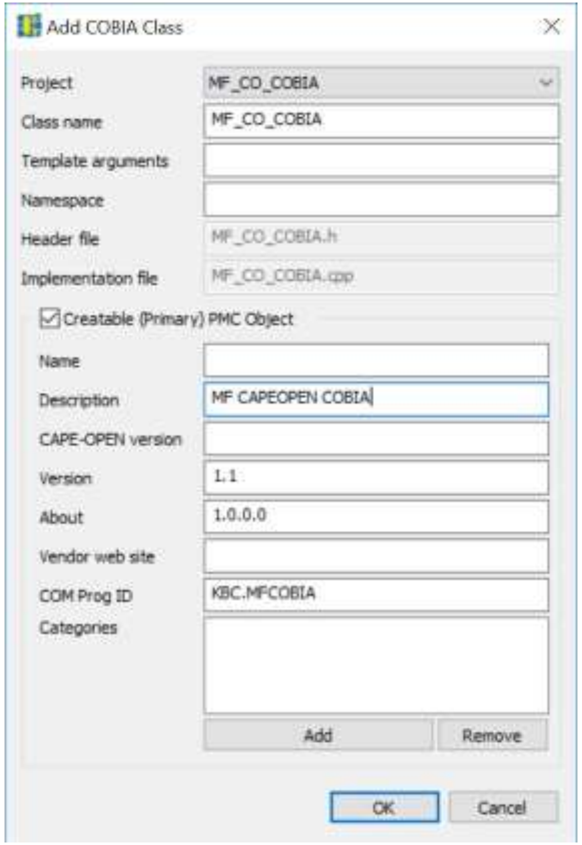
General	
Target Platform	Windows
Target Platform Version	**10.0.14393.0**
Output Directory	<different options>
Intermediate Directory	<different options>
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;*.tli;*.tlh;*.tmp;*.r
Build Log File	$(IntDir)$(MSBuildProjectName).log
Platform Toolset	**Visual Studio 2015 (v140)**
Enable Managed Incremental Build	No
Project Defaults	
Configuration Type	**Dynamic Library (.dll)**
Use of MFC	Use Standard Windows Libraries
Character Set	**Use Multi-Byte Character Set**
Common Language Runtime Support	No Common Language Runtime Support
.NET Target Framework Version	
Whole Program Optimization	<different options>
Windows Store App Support	No

Configuration Type
Specifies the type of output this configuration generates.

Configuration DLL is required to give the PME access to the library

Creation of primary PMC object





Add COBIA Class

Project	MF_CO_COBIA
Class name	MF_CO_COBIA
Template arguments	
Namespace	
Header file	MF_CO_COBIA.h
Implementation file	MF_CO_COBIA.cpp

☑ Creatable (Primary) PMC Object

Name	
Description	MF CAPEOPEN COBIA
CAPE-OPEN version	
Version	1.1
About	1.0.0.0
Vendor web site	
COM Prog ID	KBC.MFCOBIA
Categories	

Visual Studio >= 2015

Creation of primary PMC object

This will generate the following files:

- COBIAEntryPoints.cpp

  Holds the interface to the entry points and registration setting

- MF_CO_COBIA.h

  Includes function that returns a description of the current object for error handling

  Registration info

- MF_CO_COBIA.cpp

  Source file for the MF_CO_COBIA

PMC registration

```
#include <COBIA.h>
#define COBIA_PMC_ENTRY_POINTS
#define COBIA_PMC_DEFAULT_DLLMAIN
#include <COBIA_PMC.h>

//! Define registration scope
/*!
PMC module must implement this function to indicate whether object
registration must be for all users or for the current user.

Return true if registration is for all users, false if registration is for current user only
*/
bool isPMCRegistrationForAllUsers() {
		return false;
}
```

PMC registration

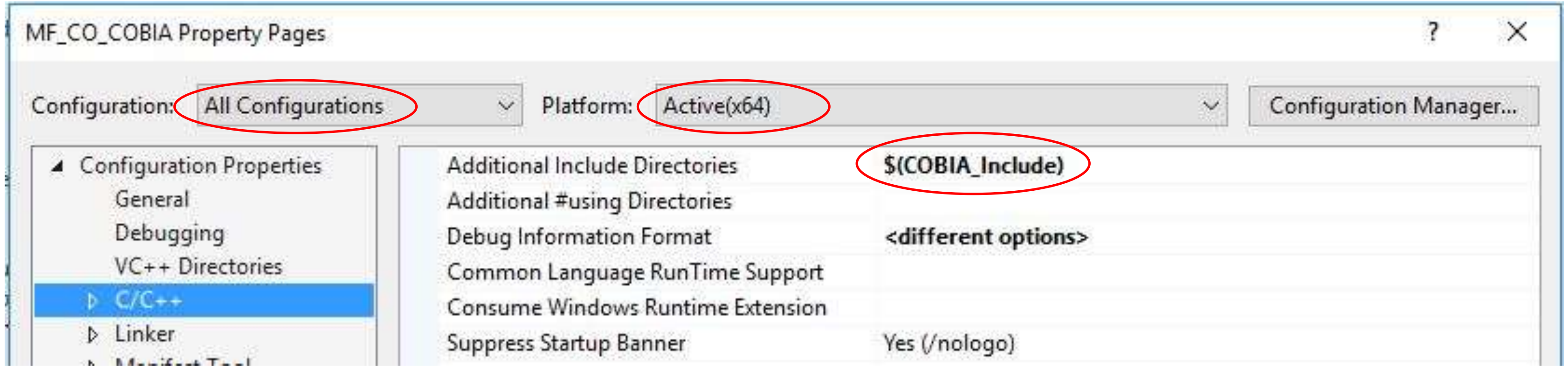
PMC Registration:

- COBIA has its own registry
- COBIA API for direct access to registry
- COBIA API provides PMC registrar component
  - Just fill out the details
  - Takes care of PMC registration
- COBIA PMC registration also registers a COM object (on Windows)

Visual Studio Configuration

The COBIA SDK installer creates an environment variable COBIA_Include pointing to the Include folders.

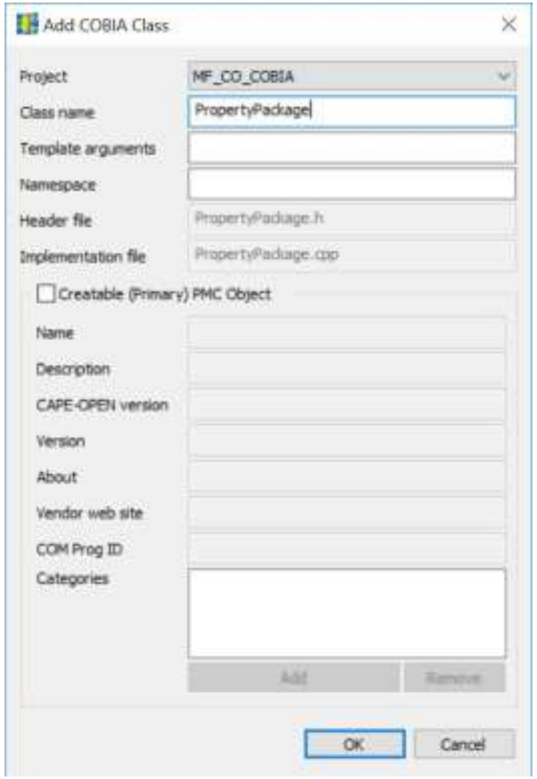
In the project properties set C/C++ Additional Include Directories to: $(COBIA_Include)


MF_CO_COBIA Property Pages

Configuration: All Configurations
Platform: Active(x64)

Additional Include Directories	$(COBIA_Include)
Additional #using Directories	
Debug Information Format	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)

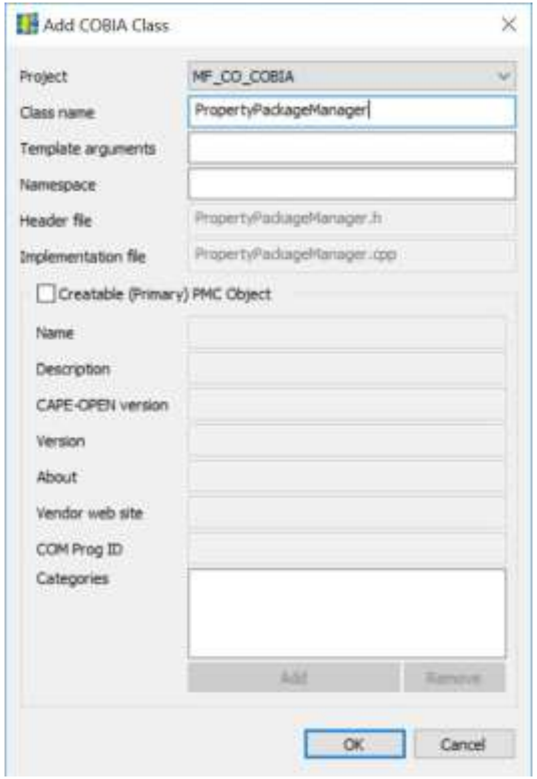
Creation of CO PropertyPackage(PP) and PPManager objects



Add COBIA Class

Project	MF_CO_COBIA
Class name	PropertyPackage
Template arguments	
Namespace	
Header file	PropertyPackage.h
Implementation file	PropertyPackage.cpp

☐ Creatable (Primary) PMC Object



Add COBIA Class

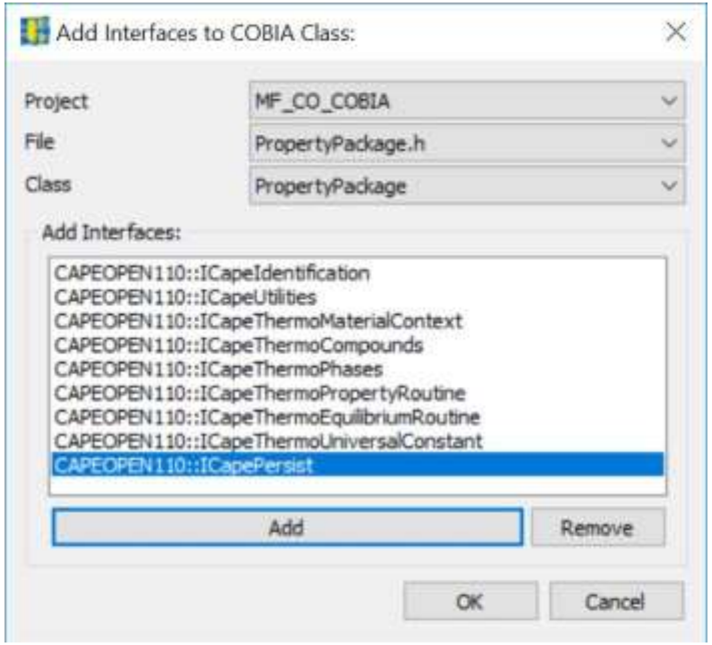
Project	MF_CO_COBIA
Class name	PropertyPackageManager
Template arguments	
Namespace	
Header file	PropertyPackageManager.h
Implementation file	PropertyPackageManager.cpp

☐ Creatable (Primary) PMC Object

Visual Studio >= 2015

PropertyPackage Interfaces

- Right-click PropertyPackage.h
- Select Implement CAPE-OPEN Interface on COBIA Class



Add Interfaces to COBIA Class:

Project: MF_CO_COBIA
File: PropertyPackage.h
Class: PropertyPackage

Add Interfaces:

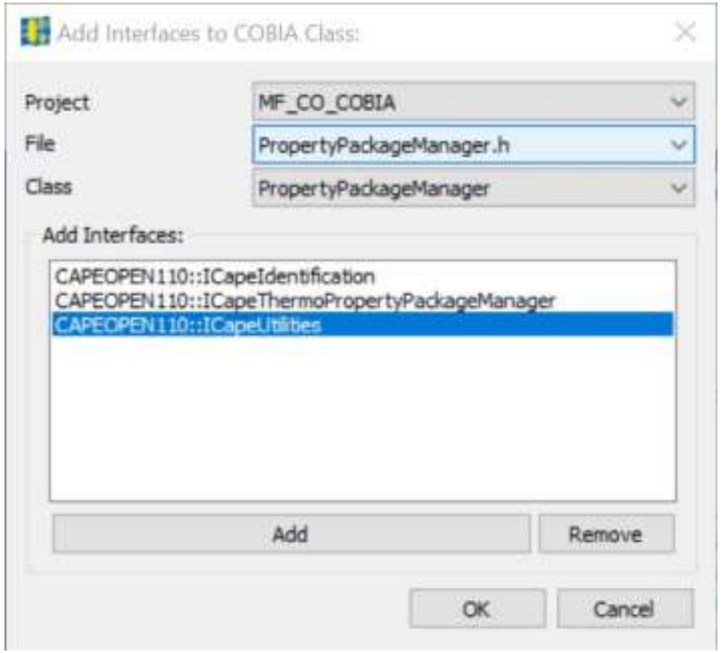
CAPEOPEN110::ICapeIdentification
CAPEOPEN110::ICapeUtilities
CAPEOPEN110::ICapeThermoMaterialContext
CAPEOPEN110::ICapeThermoCompounds
CAPEOPEN110::ICapeThermoPhases
CAPEOPEN110::ICapeThermoPropertyRoutine
CAPEOPEN110::ICapeThermoEquilibriumRoutine
CAPEOPEN110::ICapeThermoUniversalConstant
CAPEOPEN110::ICapePersist

Click add and select the following interfaces:

CAPEOPEN110::ICapeIdentification
CAPEOPEN110:: ICapeUtilities
CAPEOPEN110:: ICapeThermoMaterialContext
CAPEOPEN110:: ICapeThermoCompounds
CAPEOPEN110:: ICapeThermoPhases
CAPEOPEN110:: ICapePropertyRoutine
CAPEOPEN110:: ICapeThermoEquilibriumRoutine
CAPEOPEN110:: ICapeThermoUniversalConstant
CAPEOPEN110:: ICapePersist

PropertyPackageManager Interfaces

- Right-click PropertyPackageManager.h
- Select Implement CAPE-OPEN Interface on COBIA Class


Add Interfaces to COBIA Class:

Project: MF_CO_COBIA
File: PropertyPackageManager.h
Class: PropertyPackageManager

Add Interfaces:

CAPEOPEN110::ICapeIdentification
CAPEOPEN110::ICapeThermoPropertyPackageManager
CAPEOPEN110::ICapeUtilities

Click add and select the following interfaces:

CAPEOPEN110::ICapeIdentification
CAPEOPEN110:: ICapeThermoPropertyPackageManager
CAPEOPEN110:: ICapeUtilities

Remarks on CO PP and PPM creation using Class Wizard

Easier on Programmers: Interface adapter

```
class PropertyPackage :
        public CapeOpenObject<PropertyPackage>,
        public CAPEOPEN110::CapeIdentificationAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeUtilitiesAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeThermoMaterialContextAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeThermoCompoundsAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeThermoPhasesAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeThermoPropertyRoutineAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeThermoEquilibriumRoutineAdapter<PropertyPackage>,
        public CAPEOPEN110::CapeThermoUniversalConstantAdapter<PropertyPackage>,
        public CAPEOPEN110::CapePersistAdapter<PropertyPackage> {
```

Remarks on CO PP and PPM creation using Class Wizard

Easier on Programmers: Generating Stub Code

```
//CAPEOPEN110::ICapeIdentification

void getComponentName(/*out*/ CapeString name) {
}
void putComponentName(/*in*/ CapeString name) {
}
void getComponentDescription(/*out*/ CapeString desc) {
}
void putComponentDescription(/*in*/ CapeString desc) {
}
```

Remarks on CO PP and PPM creation using Class Wizard

Easier on Programmers: Error handling

```
//CAPEOPEN110::ICapeIdentification

void getComponentName(/*out*/ CapeString name) {
name = this->name;
}
void putComponentName(/*in*/ CapeString name) {
If (name.empty()){
throw cape_open_error(COBIAERR_InvalidArgument)
packageName = name;
}
void getComponentDescription(/*out*/ CapeString desc) {

}

void putComponentDescription(/*in*/ CapeString desc) {

}
```

Remarks on BasePropertyPackage (MF PP)

- Started with existing COM-based code (BasePropertyPackage)
- Getting rid of COM specific code and reuse the rest of it

```
//allocate constant BSTR values
STR_MOLECULARWEIGHT=SysAllocString(L"molecularWeight");
```

- Conversion of data types and use COBIA Unified data types:

    COM: LONG, BOOL, BSTR, OLECHAR, …

    COBIA: CapeInteger, CapeBoolean, CapeCapeStringImpl, ..

- Thread safe coding

    Interface class to Lock/Unlock

Overall Experience

- **The Positive**
  - AmsterCHEM COBIA Class Wizard makes it easy to generate the skeleton and framework for the classes selected
  - The available adapter classes are easy to use
  - Easier error handling
  - Less error prone and more efficient
  - Reusing the existing COM based code for many methods
- **The challenges**
  - Which interfaces should be selected
  - Conversion of COM based code to COBIA (type conversion, data allocation, …)
  - Documentation and examples on COBIA such as the one to develop a Unit Operation
  - Multithreading
  - Test and checking interoperability (future)

KBC

A Yokogawa Company

All about EXCELLENCE