ZW3D API Introduction

This document only gives basic introduction to ZW3D API. We recommend C/C++ for developing add-ons based on ZW3D. We will keep improving the API to make it more supportive to help our partners develop powerful applications based on ZW3D.
Should you have any suggestions or requirements, please feel free to contact us.
You could email zdn@zwsoft.com for help.
Thanks.

Contents

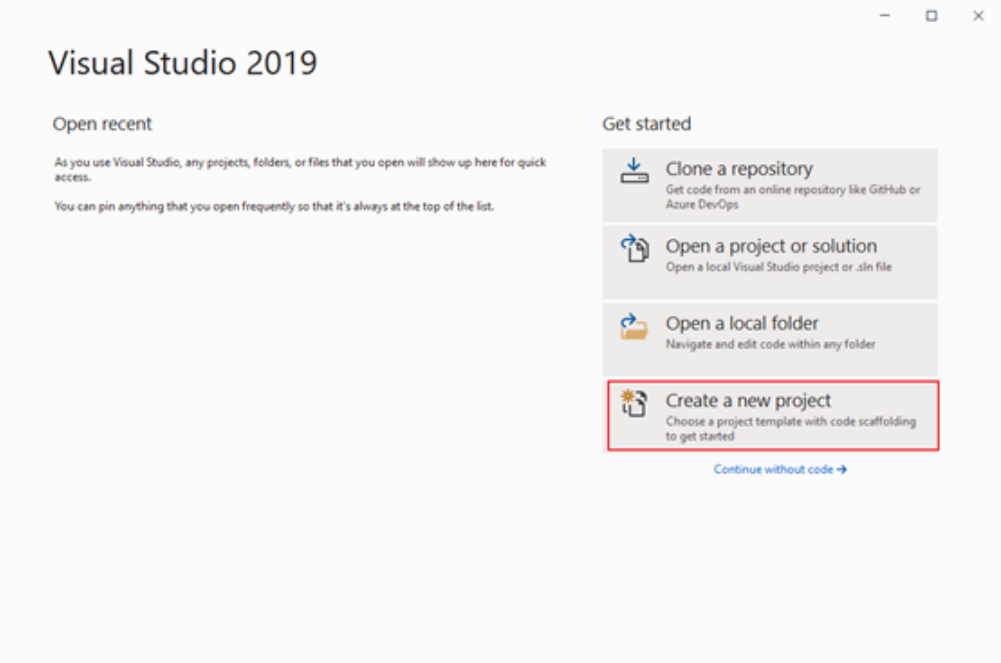
ZW3D API Introduction

Chapter 1: Start with ZW3D API

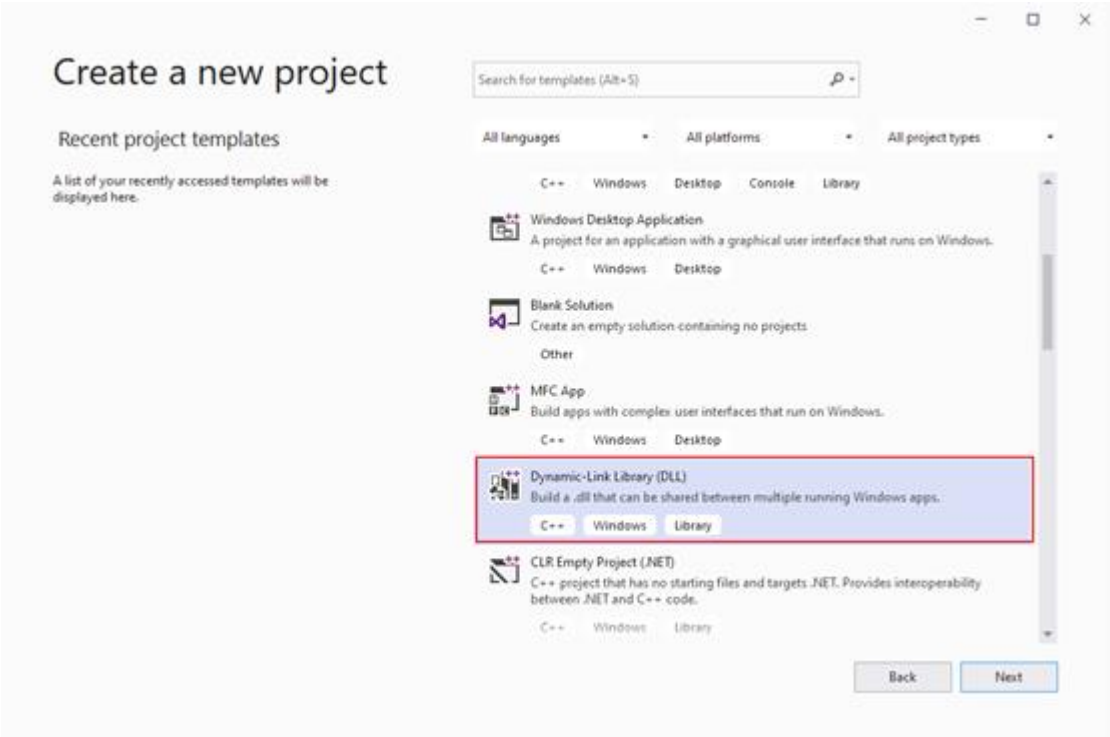
1. System requirement
   a) Windows 7 or above
   b) Visual studio 2019 (or any another IDE for C/C++)
   c) ZW3D 2012 or above
   d) Qt5.9.7(for Windows)

   Note: *After installing Qt, you can double-click the script of "CopyQtDll.bat" in api folder of ZW3D installation directory to complete installing ZW3D custom control. Once the installation is complete, you can open Qt Designer plug-in and use custom control provided by ZW3D.*

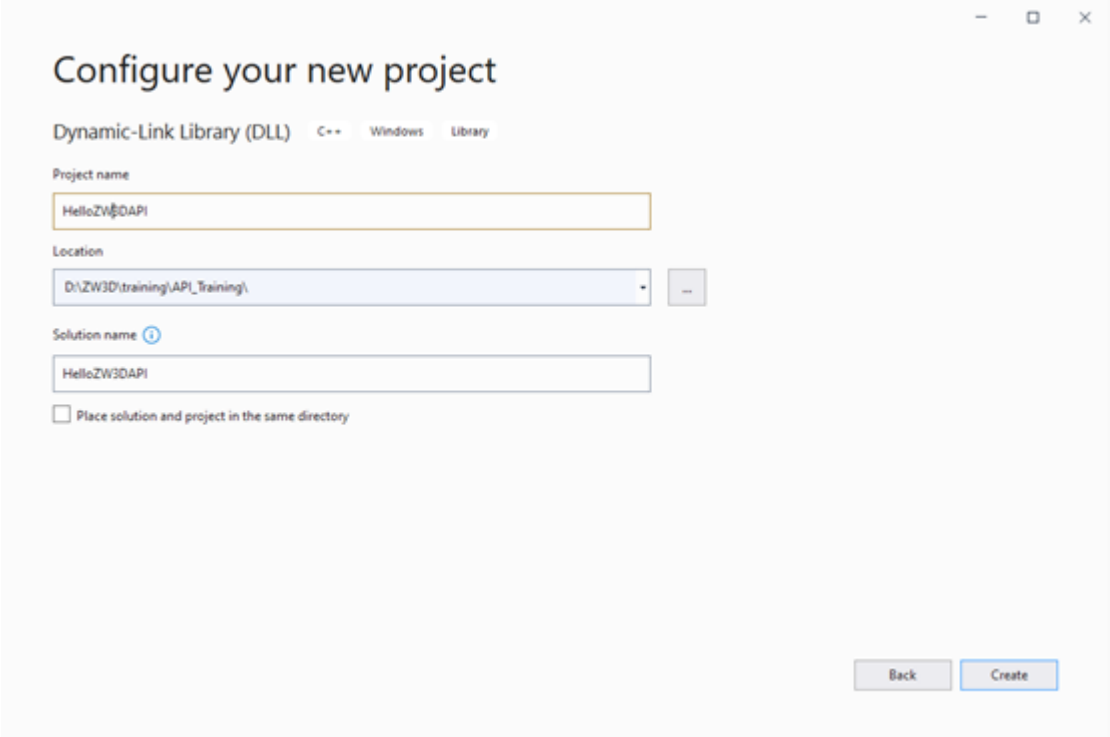
2. Create a project
   a) Open Visual Studio 2019 and choose ***Create a new project.***



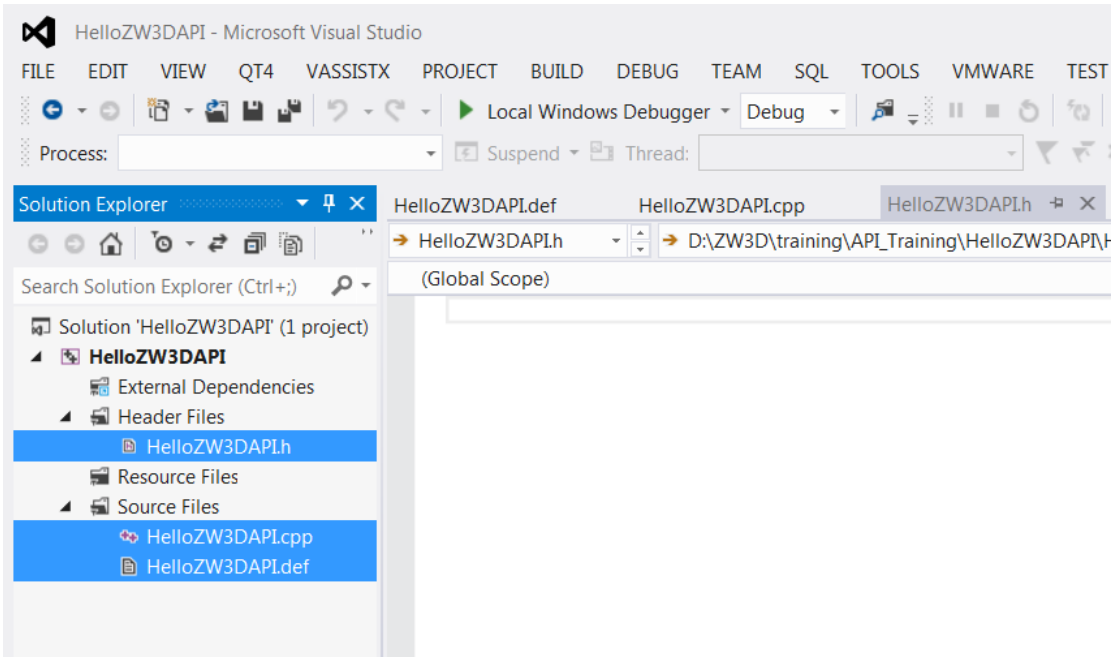
   b) Choose ***Dynamic Link Library (DLL)*** and click ***Next***.



c) Input name as ***HelloZW3DAPI***, and then click ***Create.***

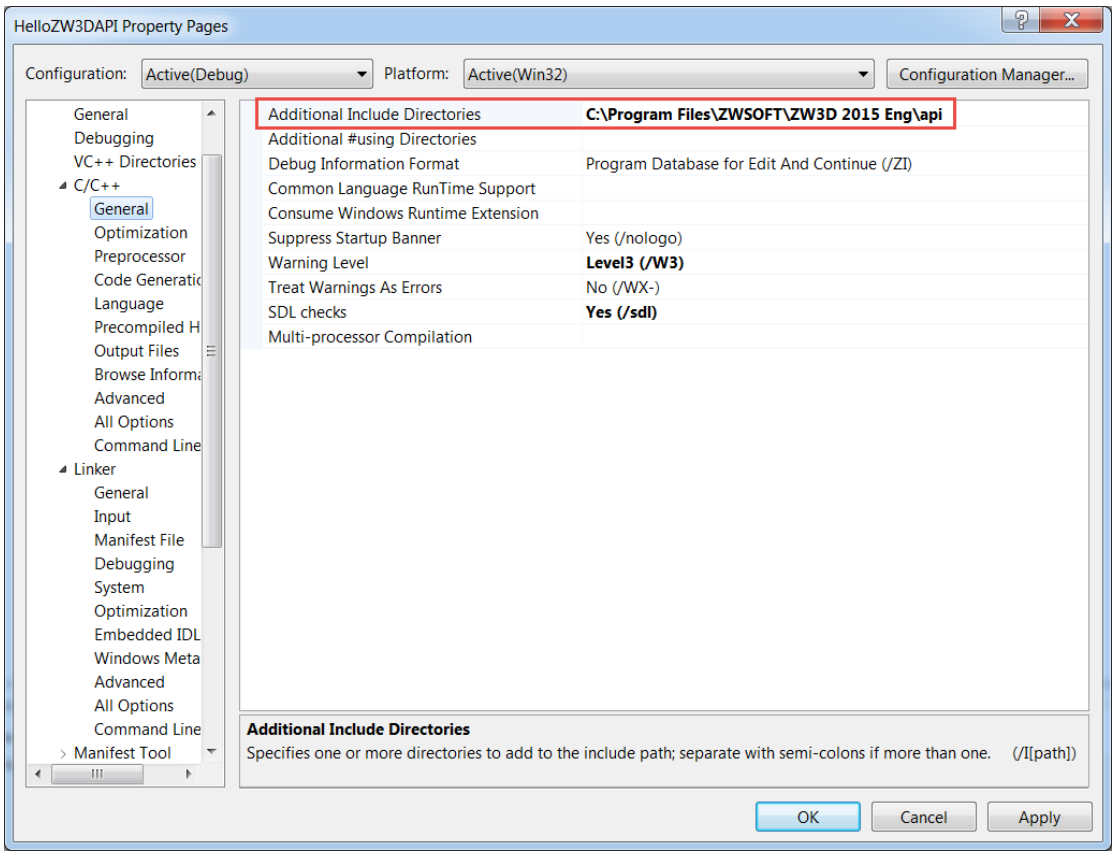


3. Add HelloZW3DAPI.h, HelloZW3DAPI.cpp HelloZW3DAPI.def to this project.

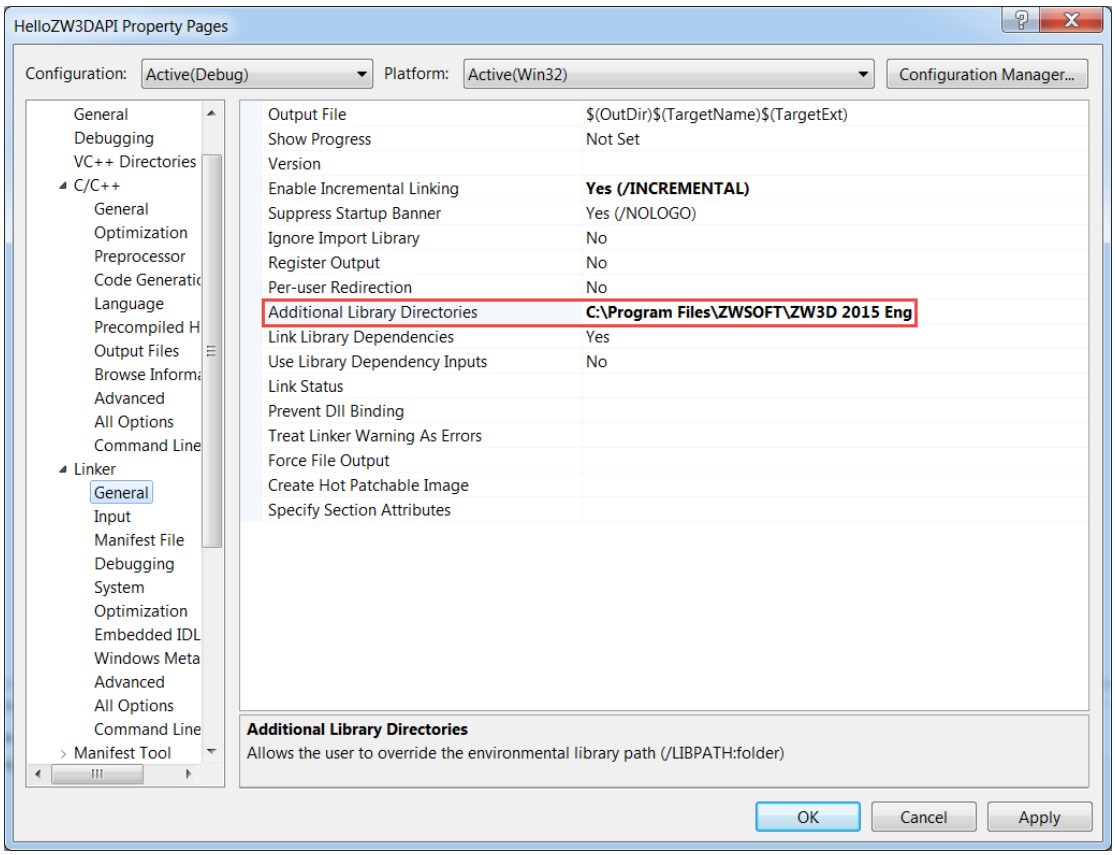


4. Set the project.

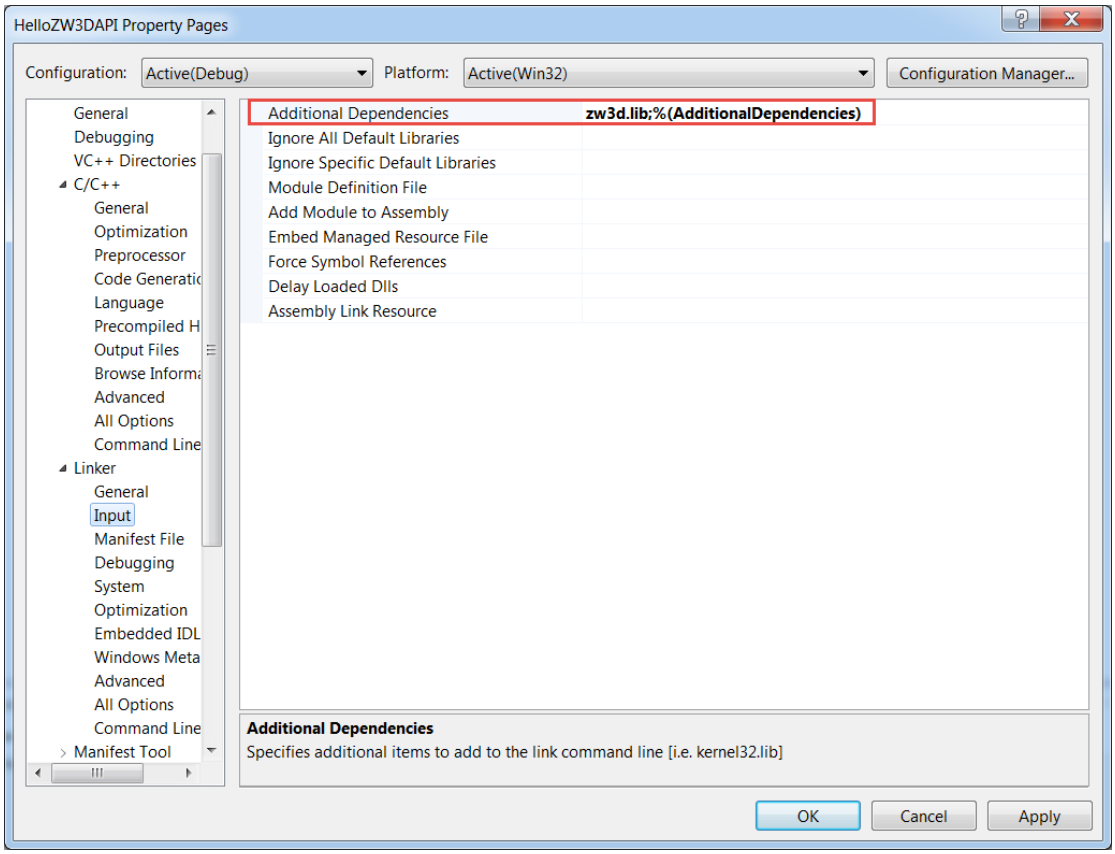
Right click on the project and choose Properties. Then, add the directory of ZW3D API head file to ***C/C++ → General → Additional Include Directories***.



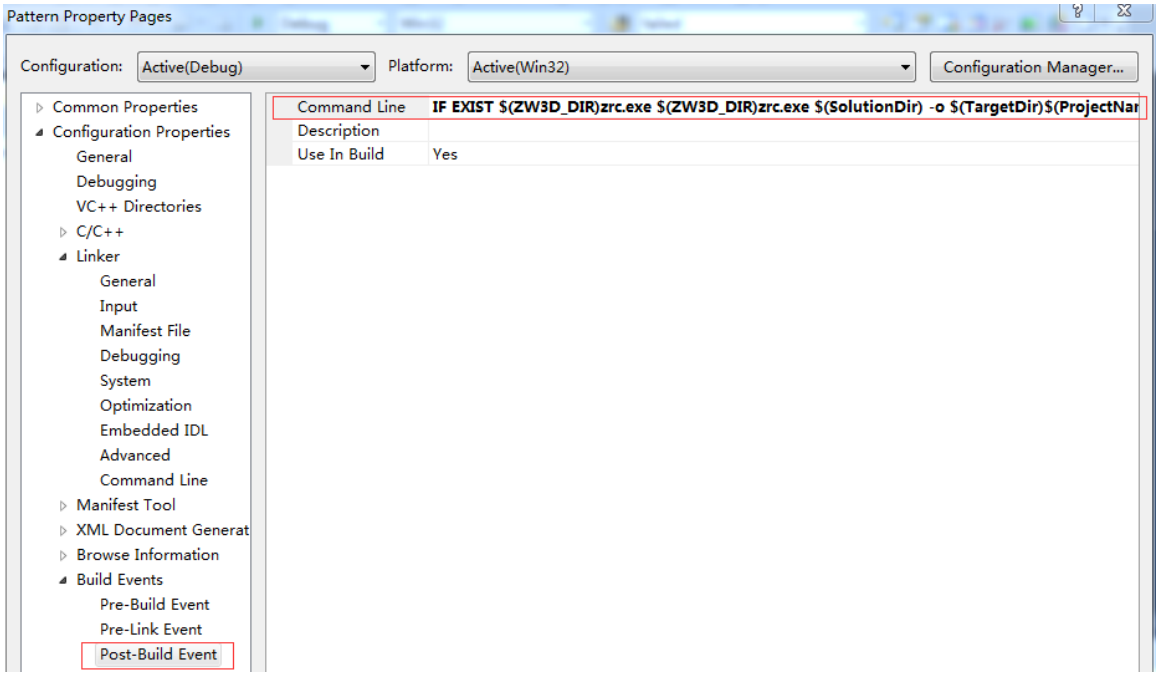
a) Add the directory of ZW3D API library file to ***Linker → General → Additional Library Directories***.



b) Add ZW3D API library to ***Linker → Input → Additional Dependencies***.



c) To compile resource, add command line to ***Build Events→Post-Build Event→Command Line***.



(*Note:* *If there is no resource file in your project. Such as image/UI. This step is optional)*

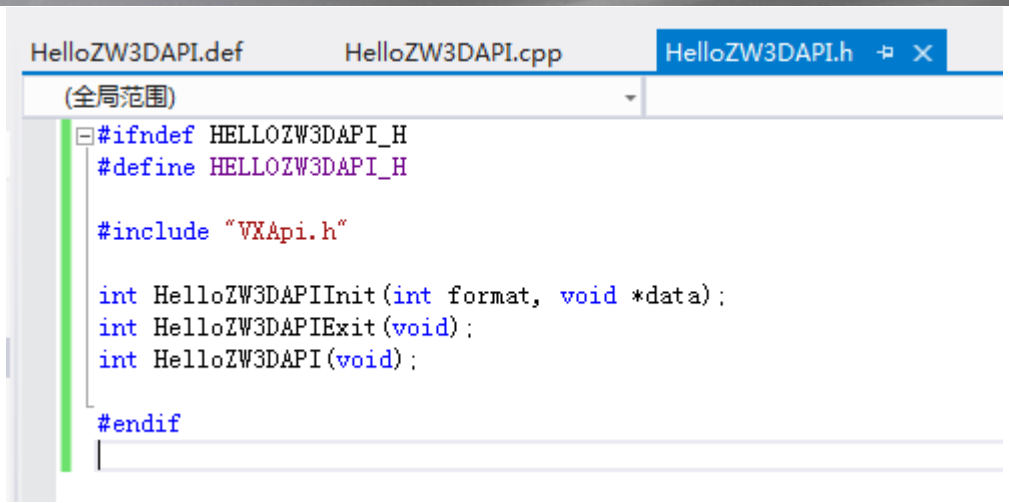
The contents as shown below:

```
IF EXIST "$(ZW3D_DIR)zrc.exe" "$(ZW3D_DIR)zrc.exe" "$(SolutionDir)\." -o "$(TargetDir)$(ProjectName).zrc"
```

(*Note:* *ZW3D_DIR is an environment variable whose value is ZW3D installation path***)**

5. Define the functions in HelloZW3DAPI.h. You can copy the following code directly.

```
#ifndefHELLOZW3DAPI_H
#defineHELLOZW3DAPI_H

#include"VXApi.h"

IntHelloZW3DAPIInit(int format, void *data);
IntHelloZW3DAPIExit(void);
IntHelloZW3DAPI(void);

#endif
```



Note: *The prefix of the functions must be the same as the project, which means @customizedApp@Init()* and *@customizedApp@Exit()* are the entrance functions of the application. When *@customizedApp@.dll* is loaded by ZW3D, *@customizedApp@Init()* and *@customizedApp@Exit()* will be checked to know the customized functions. *(@customizedApp@ means the name of any project you have created.)*

6. Implement the functions in HelloZW3DAPI.cpp. You can copy the following code directly.

```
#include"HelloZW3DAPI.h"

IntHelloZW3DAPIInit(int format, void *data)
{
 cvxCmdFunc("HelloZW3DAPI", (void*)HelloZW3DAPI, VX_CODE_GENERAL);

 return0;
}

IntHelloZW3DAPIExit(void)
{
 cvxCmdFuncUnload("HelloZW3DAPI");

 return0;
}

IntHelloZW3DAPI(void)
{
 cvxMsgDisp("Hello ZW3D API!");

 return0;
}
```

HelloZW3DAPI.def | HelloZW3DAPI.cpp | HelloZW3DAPI.h

(全局范围)

```
#include "HelloZW3DAPI.h"

int HelloZW3DAPIInit(int format, void *data)
{
    cvxCmdFunc("HelloZW3DAPI", (void*)HelloZW3DAPI, VX_CODE_GENERAL);
    return 0;
}

int HelloZW3DAPIExit(void)
{
    cvxCmdFuncUnload("HelloZW3DAPI");
    return 0;
}

int HelloZW3DAPI(void)
{
    cvxMsgDisp("Hello ZW3D API!");
    return 0;
}
```

7. Define the Module Definition file, HelloZW3DAPI.def.
   You can copy the following code directly.

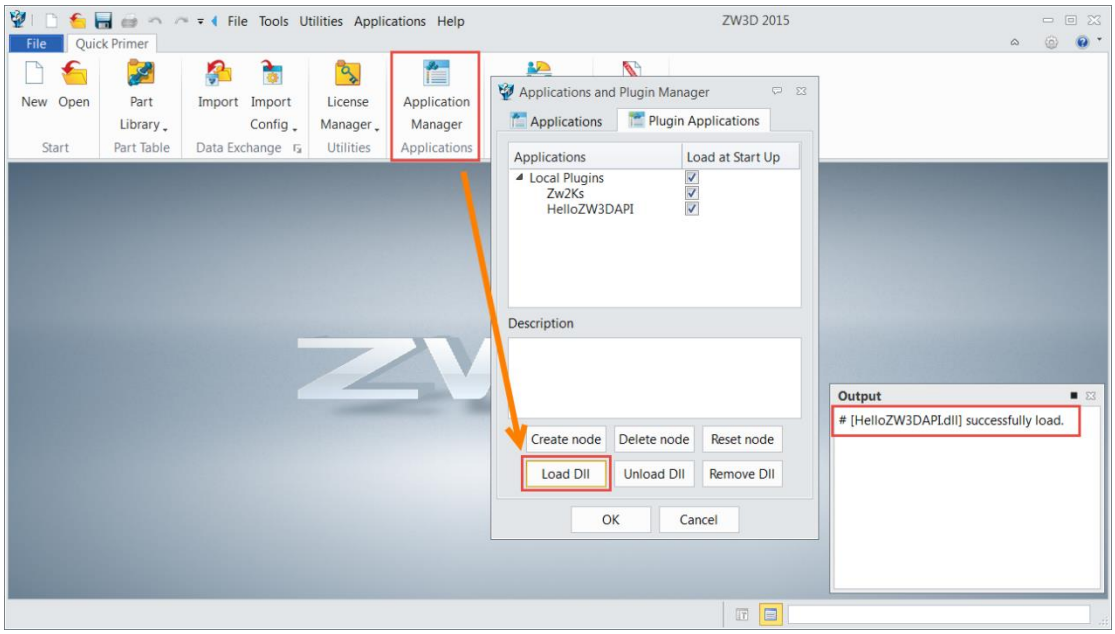
```
LIBRARY HelloZW3DAPI.dll
EXPORTS
    HelloZW3DAPIInit
    HelloZW3DAPIExit
    HelloZW3DAPI
```

HelloZW3DAPI.def | HelloZW3DAPI.cpp | HelloZW3DAPI.h

```
LIBRARY HelloZW3DAPI.dll
EXPORTS
    HelloZW3DAPIInit
    HelloZW3DAPIExit
    HelloZW3DAPI
```

8. Build the project.
   Right click on this project to build the project. Then, you can find HellowZW3DAPI.dll in the directory ".\HelloZW3DAPI\Debug\HelloZW3DAPI.dll".

9. Load HelloZW3DAPI.dll.
   a) Use ZW3D ***Applications Manager*** to load the DLL and a message in the ***Output*** dialog will show whether the command fails or succeeds.
   Note**:** *This path will be remembered in registration when ZW3D is closed, and next time it will automatically follow the path to load the file.*

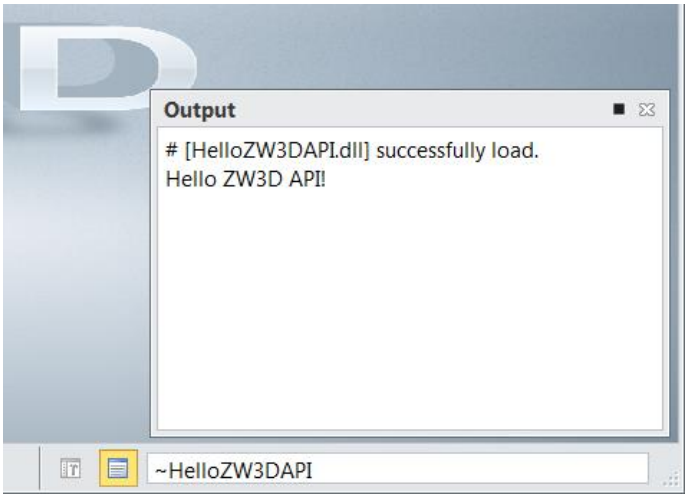


b) Copy HelloZW3DAPI.dll to the installation folder. “C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs”, then start ZW3D.

Note: *ZW3D will search for applications in “.\apilibs”. If any, they will be loaded.*

10. Run this application.

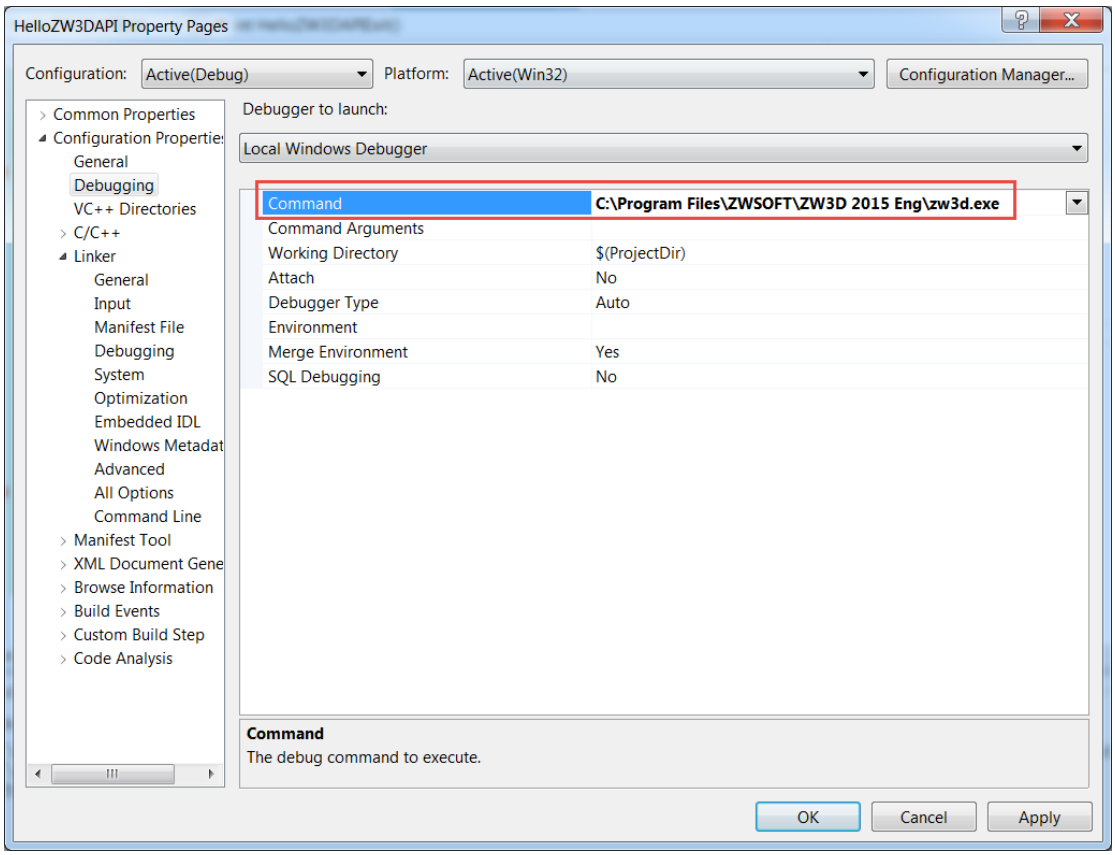
Input “~HelloZW3DAPI” in the command line, then press Enter. You can find “Hello ZW3D API!” was shown in the Output dialog.



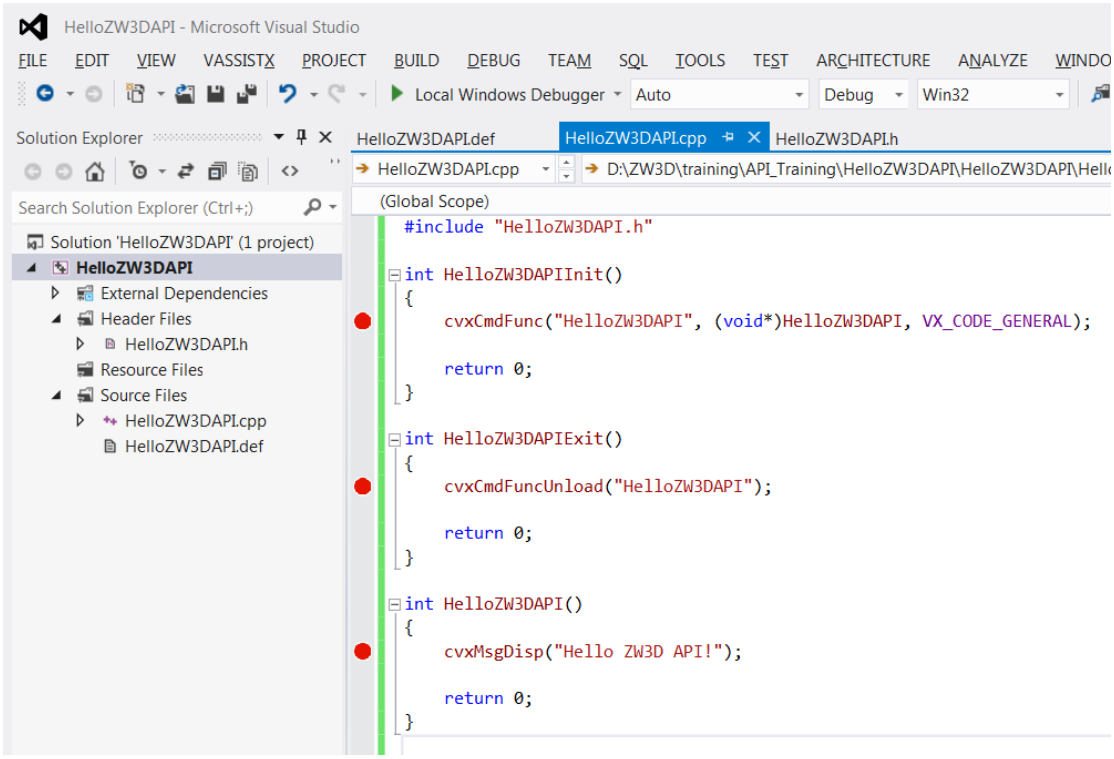
Note: *Function name can be defined to any name, and it is not necessary to be the same as the project. But it is a good habit to define it the same as the name of project.*

11. Debug the application.

a) Right click on the project and set the Debugging Command as below. Then run “***Local Windows Debugger***” or press F5 to start the debugger.



b) Set the break point for each function.

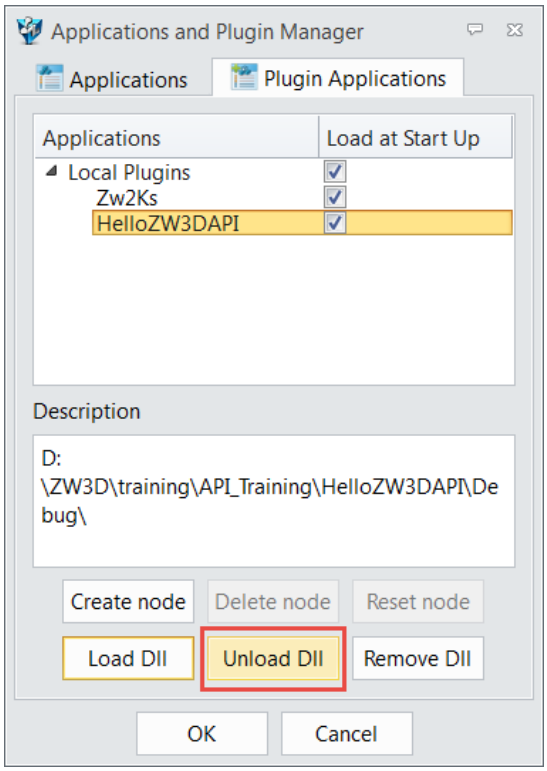


c) Reference step 9 → a) to Load the DLL, you will find the following situations.

i. The debugger will get into HelloZW3DAPIInit() when the DLL is loaded.

ii. The debugger will get into HelloZW3DAPI() when "~HelloZW3DAPI" (step 10) is run.

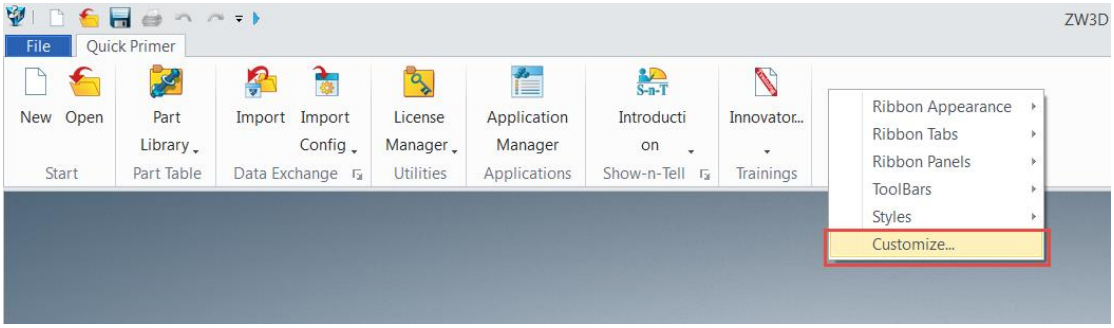
iii. The debugger will get into HelloZW3DAPIExit() when the application is unloaded.



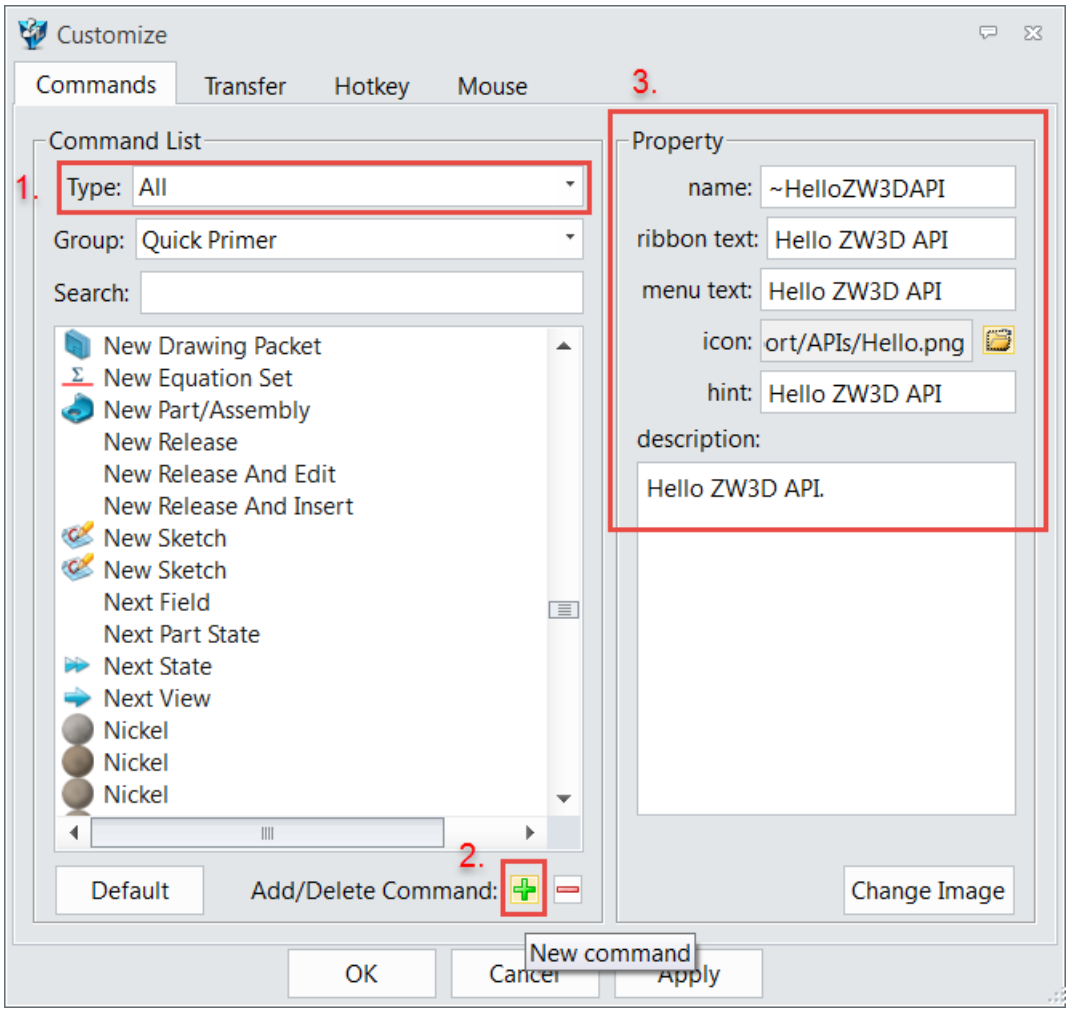
Chapter2: Customized Menu / Ribbon / Toolbar

We create the first customized command in ZW3D. But input the command is complex. I will introduce how to put this command in the Menu / Ribbon / Toolbar.

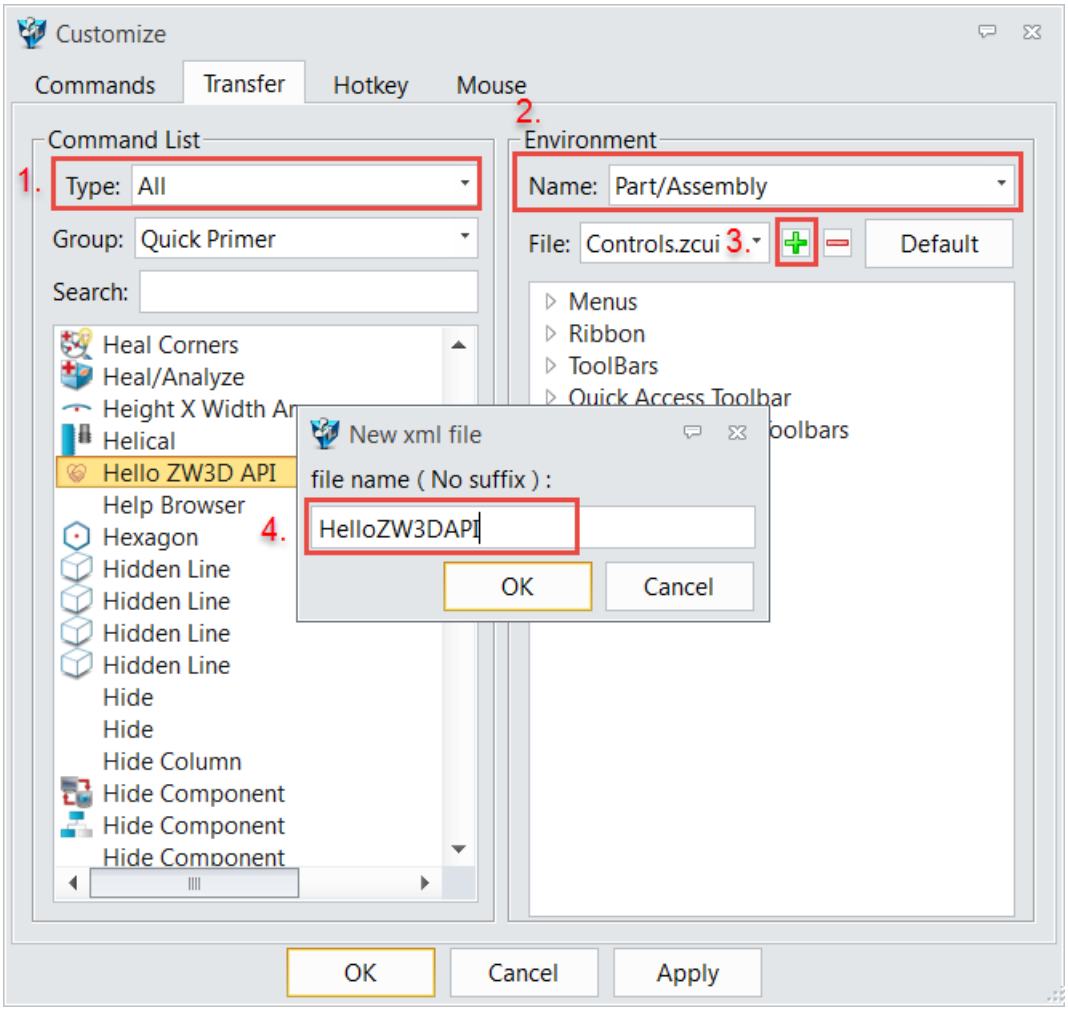
1. Open the ZW3D and right click on the space of the ribbon. Then, click ***Customize…***



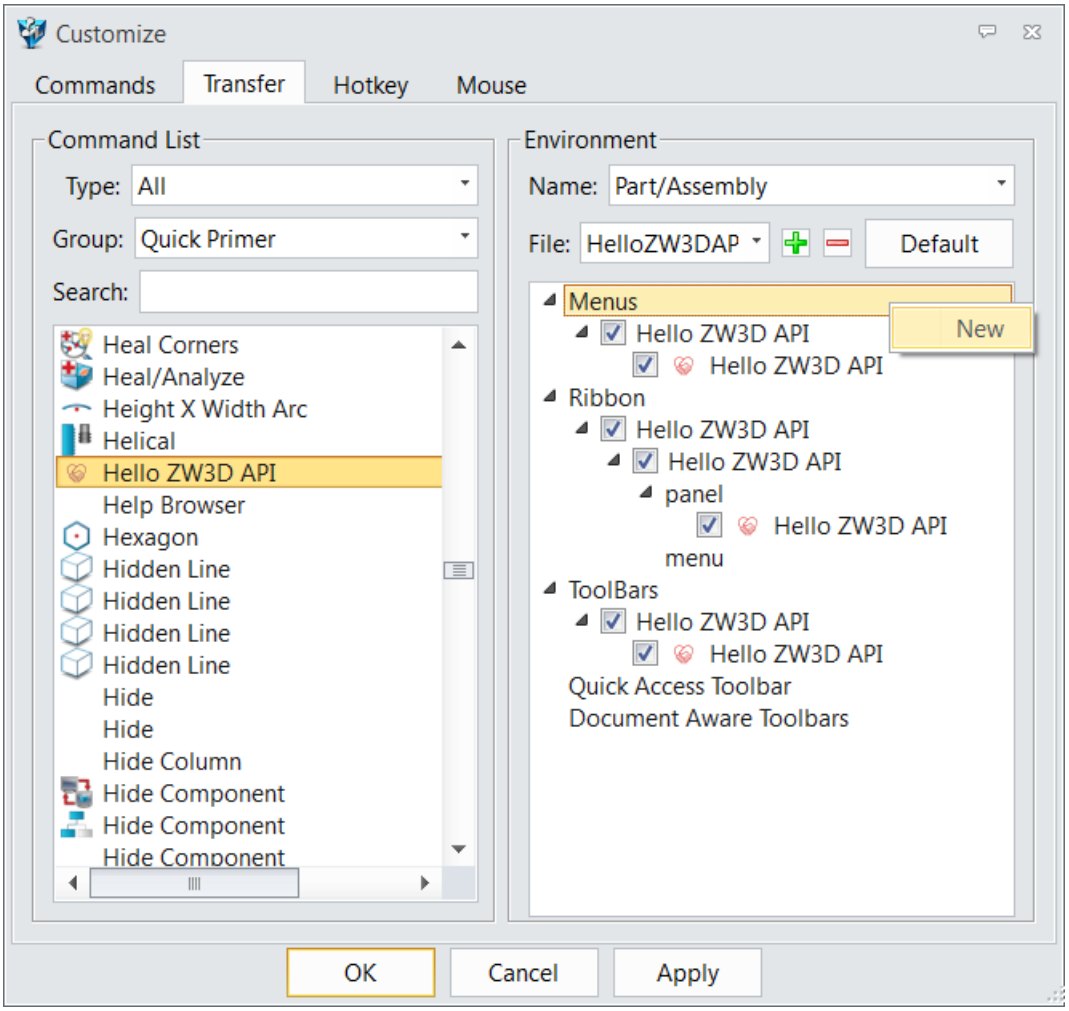
2. Define your command.
   a) Change the type to ***All***.
   b) Add a new command.
   c) Change the Property of the new command.



3. Change the tap to **Transfer** to create your Own Menu / Ribbon / Toolbar.
   a) Change the Type to ***All***.
   b) Change the Environment to ***Part/Assembly***.
   c) Add a new xml file.
   d) Give the name to **HelloZW3DAPI**.

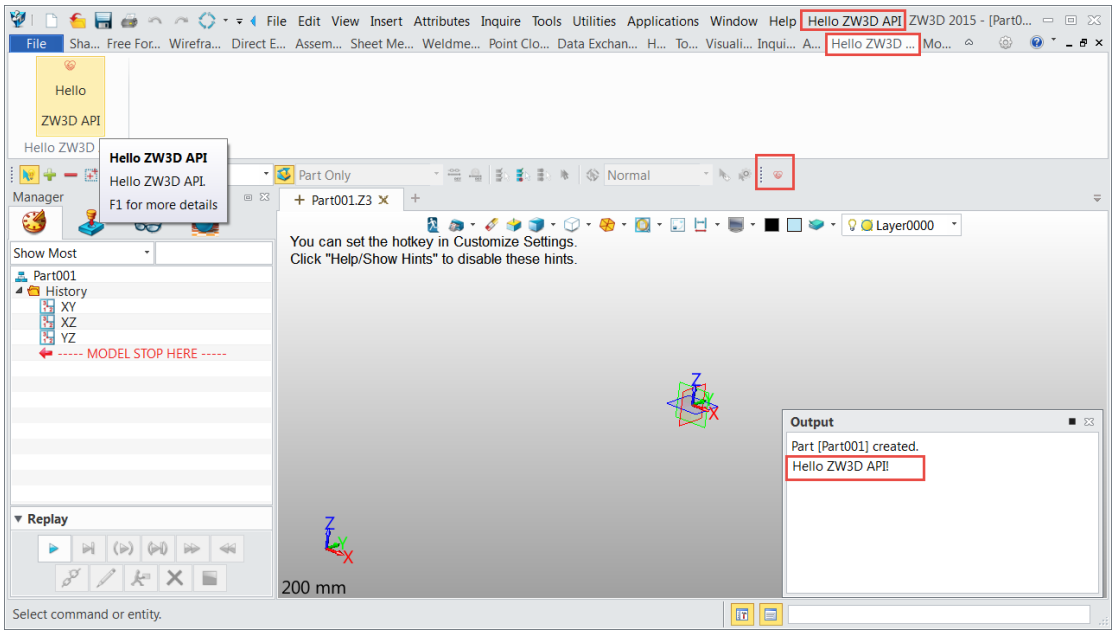


   e) Customize the new xml file.
      i. Right click on the Menus / Ribbon / ToolBars to create new items.
      ii. Drag the “Hello ZW3D API” command from left to the new items you just have created.
      iii. Please refer to the following picture to create the same Menu / Ribbon / Toobar.



f) Click ***OK*** to finish the customization.

g) Then, create a new part, and you can find the Menu / Ribbon / Toolbar there. You can press the button to run the command.



Note: if you get this warning: "ALERT: Unable to find HelloZW3DAPI: No such symbol." It means ZW3D cannot find your command. You need to load the DLL first. Please refer to Chapter 1, point 9 to load the DLL.

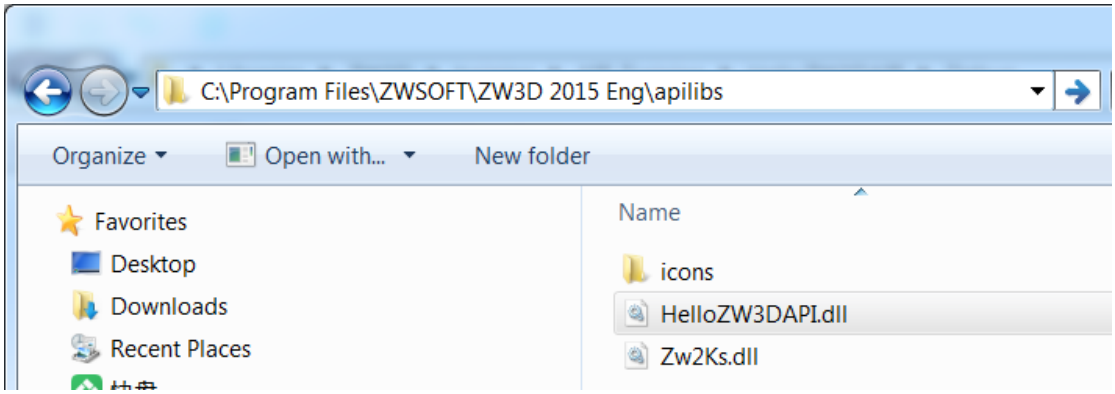
4. Share the customization to other people.
   a) Get the customized UI XML file (HelloZW3DAPI.zcui) from the user's folder:
      If your ZW3D version is prior to 2020, get from:
      %appdata%\ZW3D 2015Eng\profiles\Default\Environment-2\Controls

      If your ZW3D version after 2020(included), get from:
      %appdata%\ZWSOFT\ZW3D\ZW3D 2022Eng\custom\profiles\Default\Environment-2\Controls

   b) Get the customized Command XML file (User.zcui) from the user's folder:
      If your ZW3D version is prior to 2020, get from:
      %appdata%\ZW3D 2015Eng\profiles\Default\Action

      If your ZW3D version is after 2020, get from:
      %appdata% \ZWSOFT\ZW3D \ZW3D 2022Eng\custom\profiles\Default\Action

   c) Copy the files to the same directory in another PC.
   d) Copy the DLL and the image to the installation directory, and the image must be put in the folder which named as icons.
      C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs



Note: *you can rename the XML to any other name. But you must put them in the right directory. ZW3D will load them automatically.*

Chapter 3: ZW3D UI Designer introduction

1. What is ZW3D UI Designer?

   ZW3D UI Designer is a UI designer based on QT 5.9.7 ZWSOFT bought the QT license from Digia. We developed our own UI controls based on QT technology. ZW3D UI Designer is a plugin based on QT Designer. Users need to download and install QT by themselves, and copy the plugin provided by ZW3D to the QT Designer directory, start QT Designer where you can see the UI control defined by ZW3D.

   Note: *You only can use ZW3D UI Designer for UI designer. All the coding logic, you cannot use QT technology. ZW3D UI Designer does not include the QT libraries. So, you need to buy the commercial version if you need to use the QT functions.*

   (1) The specific operations as follow:

   a. Find dll in plugins\\designer in the installation package, copy to QT installation directory of designer

   CommonControlsPlugin.dll

   QtnRibbonDsgn.dll

   b. Find the following dll in the installation package, copy to QT bin directory

   CommonControls.dll

   ResourceSystem.dll

   logging.dll

   QtnRibbon.dll

   (2) This can be done by adding script:

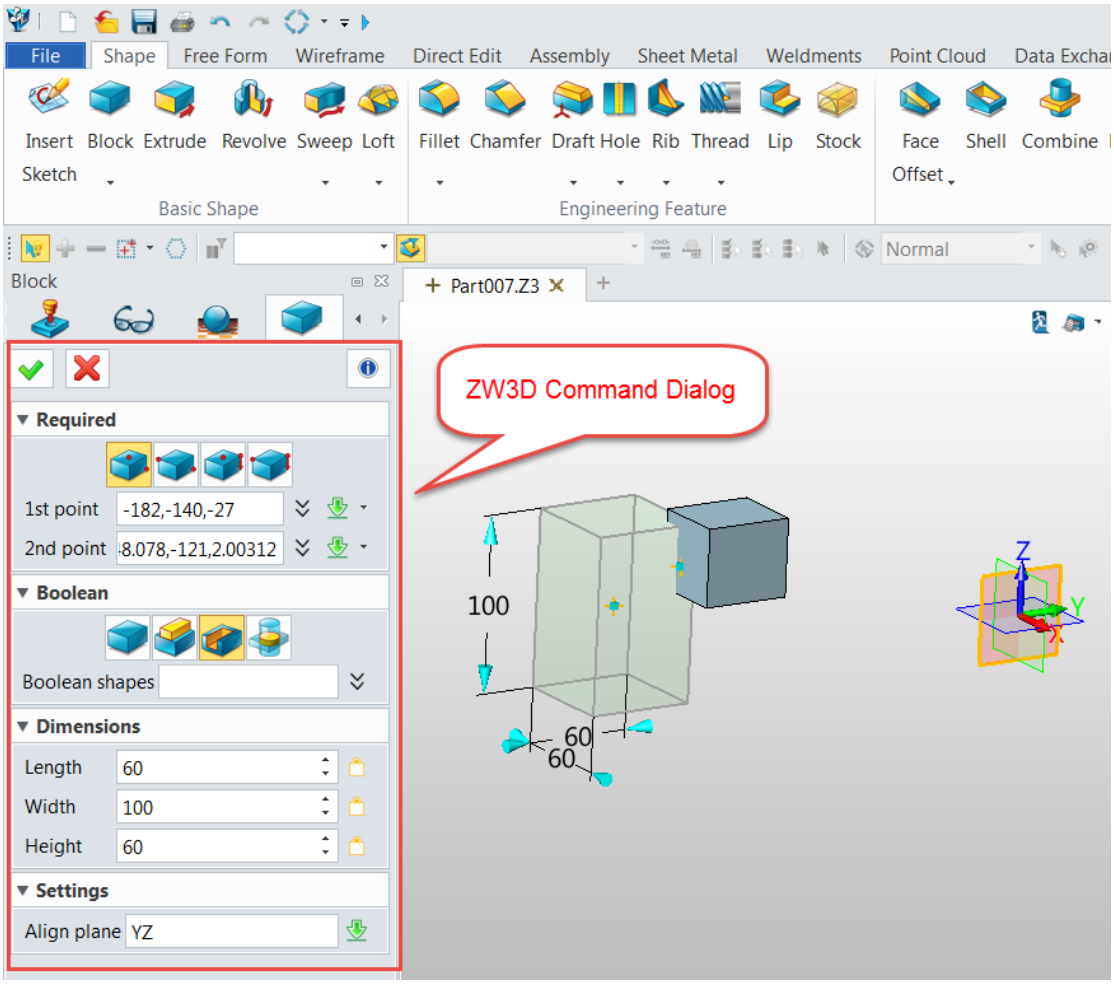
   Create a CopyQTDll.bat file in the “api” directory in the installation path, copy the following contents. **Note:** **Modify QTPath**

```
%@echo off%
set CurrentPath=%~dp0
cd /d %CurrentPath%
cd ..
set ZW3DPath=%cd%
if "%QTDIR%"=="" (
    set QTPath=D:\Qt\Qt5.9.7\5.9.7\msvc2017_64
) else (
    set QTPath=%QTDIR%
)
COPY "%ZW3DPath%\plugins\designer\CommonControlsPlugin.dll" "%QTPath%\plugins\designer"
COPY "%ZW3DPath%\plugins\designer\QtnRibbonDsgn.dll" "%QTPath%\plugins\designer"
COPY "%ZW3DPath%\logging.dll" "%QTPath%\bin"
COPY "%ZW3DPath%\CommonControls.dll" "%QTPath%\bin"
COPY "%ZW3DPath%\QtnRibbon.dll" "%QTPath%\bin"
```

```
COPY "%ZW3DPath%\ResourceSystem.dll" "%QTPath%\bin"
Pause
```

2. What is ZW3D Command Dialog?

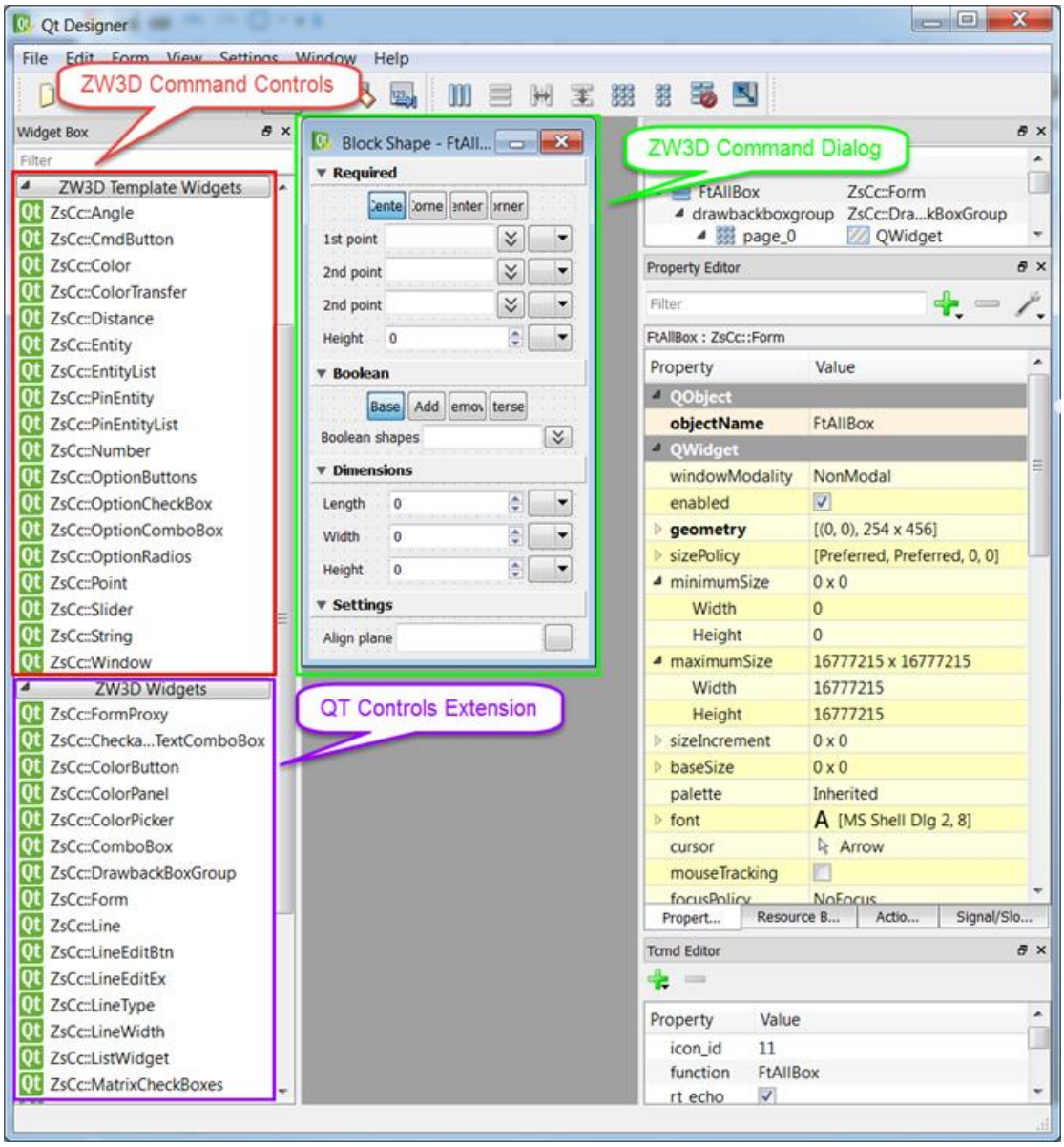
   ZW3D Command Dialog is the special dialog for interaction when the user runs a command in ZW3D. ZW3D UI Designer support the special controls to get the necessary values from ZW3D modeling space or get/set some special values which only can be used in ZW3D.



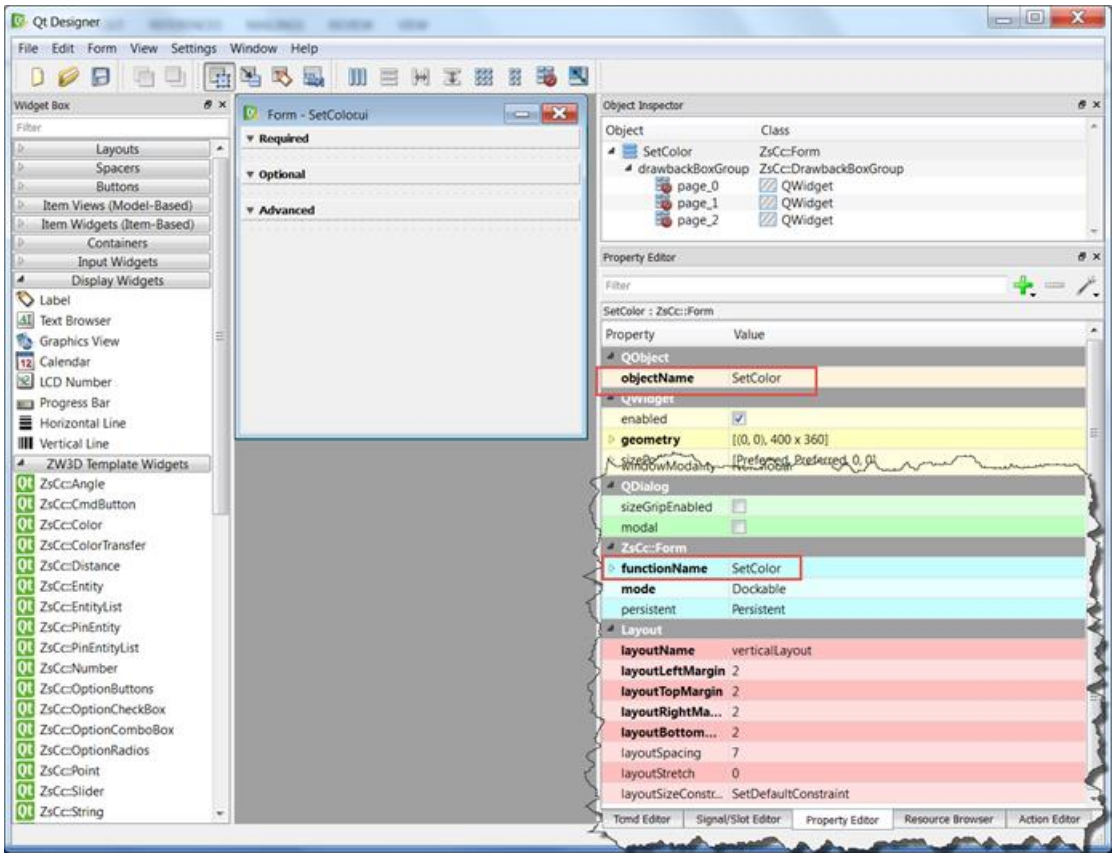
3. ZW3D UI Designer introduction.

   Form the following picture, you can find there are two extension groups on the left side. The group marked in red is the controls for ZW3D Command Controls. The controls are used for ZW3D Command Dialog design.

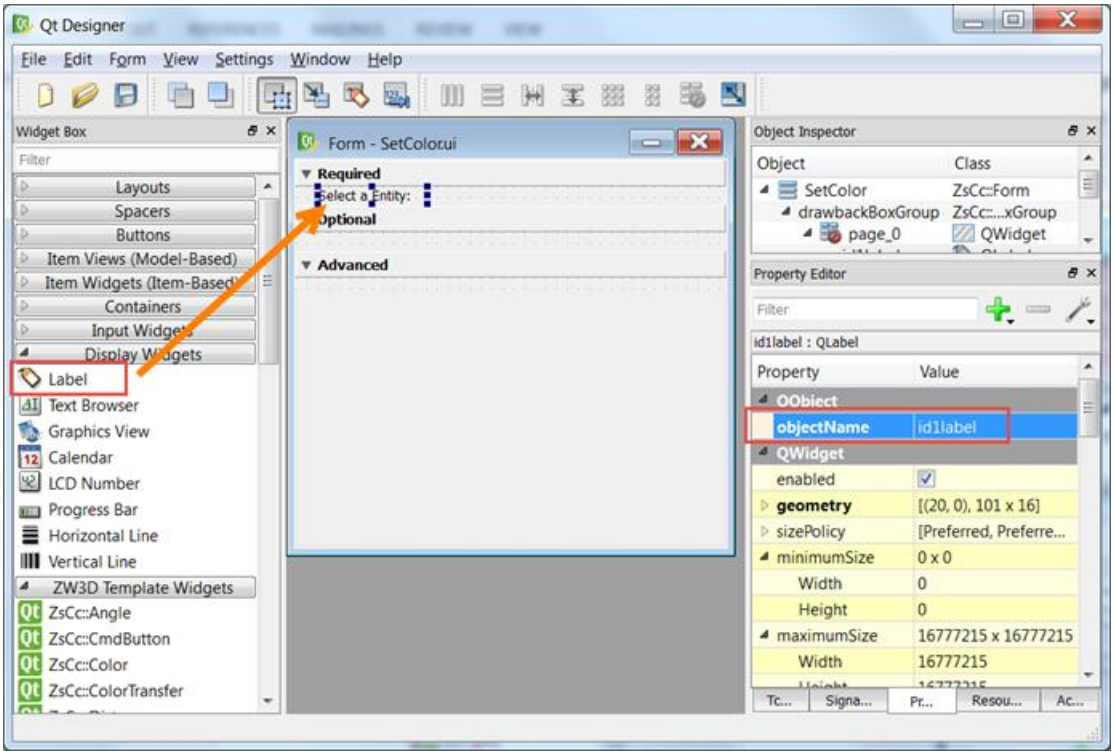
   The group marked in purple is the extension of Qt controls. They are used for Qt dialog design. We don't suggest using them if you don't have some special requirement.



4. How to create ZW3D Command dialog?
   a) Open ZW3D UI Designer. And create a new Form based on ZsCc::Form. You can get the basic form as the picture below. Then, set the objectName and functionName to **SetColor**.

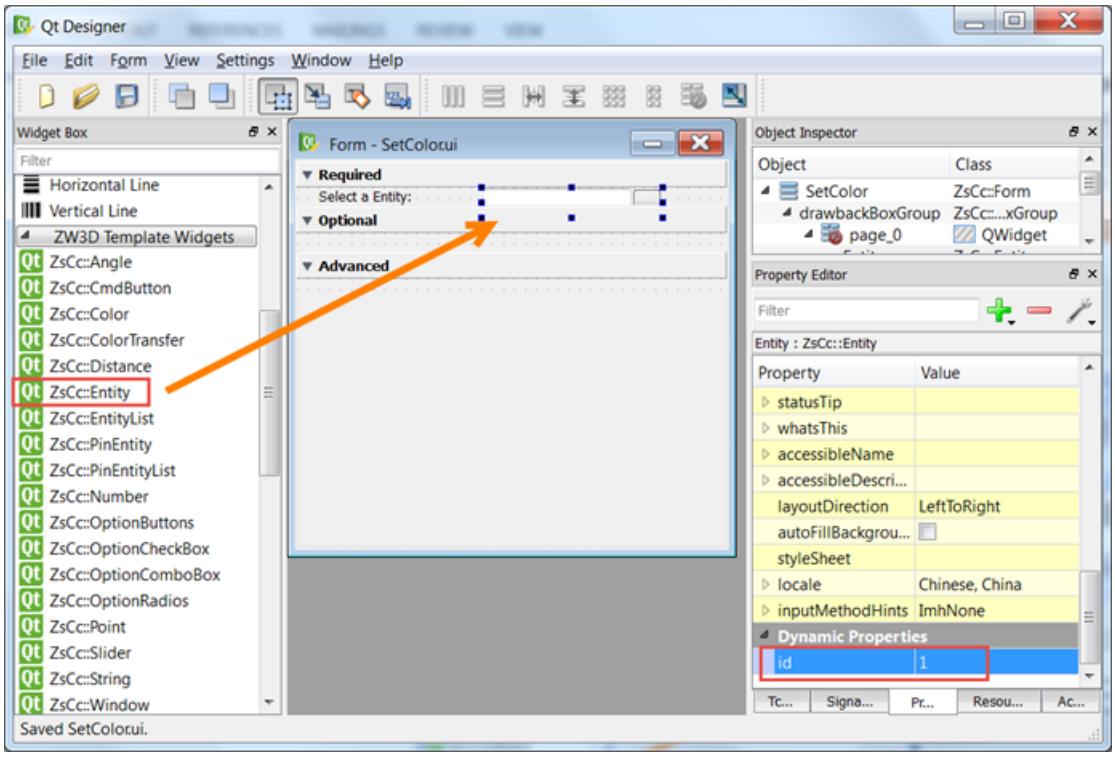


b) Drag the Label to the Required area, change the text to "***Select an Entity:***". Set the **object Name** to **id1label**.

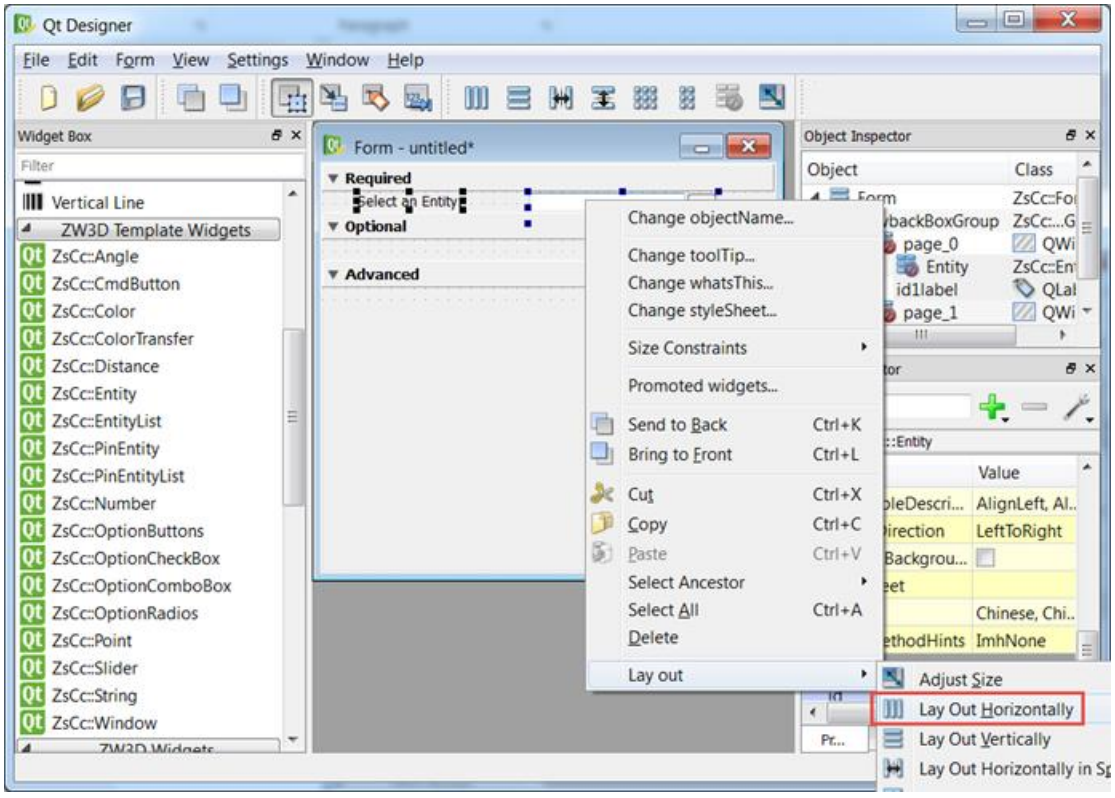


c) Drag the Entity control to the Required area. Set the ID to 1. The number of the ID can be any number, but it must be the same as the number in the Label's object Name

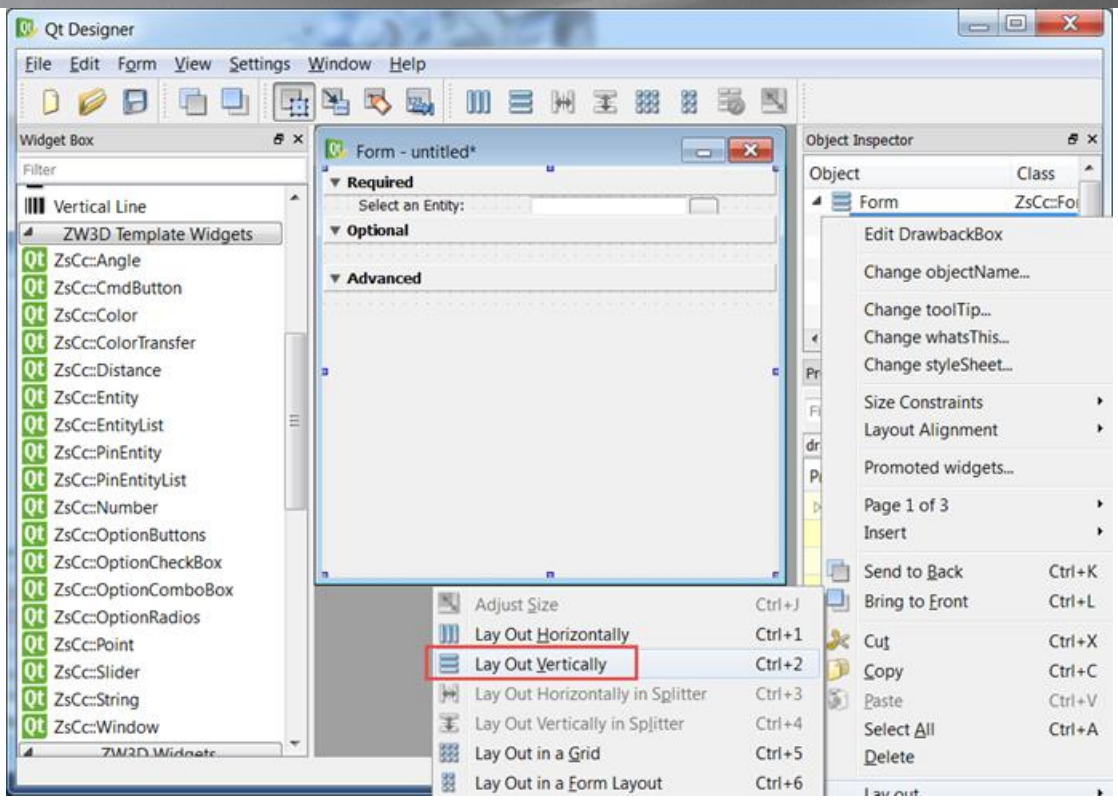
which means these two controls are a group.



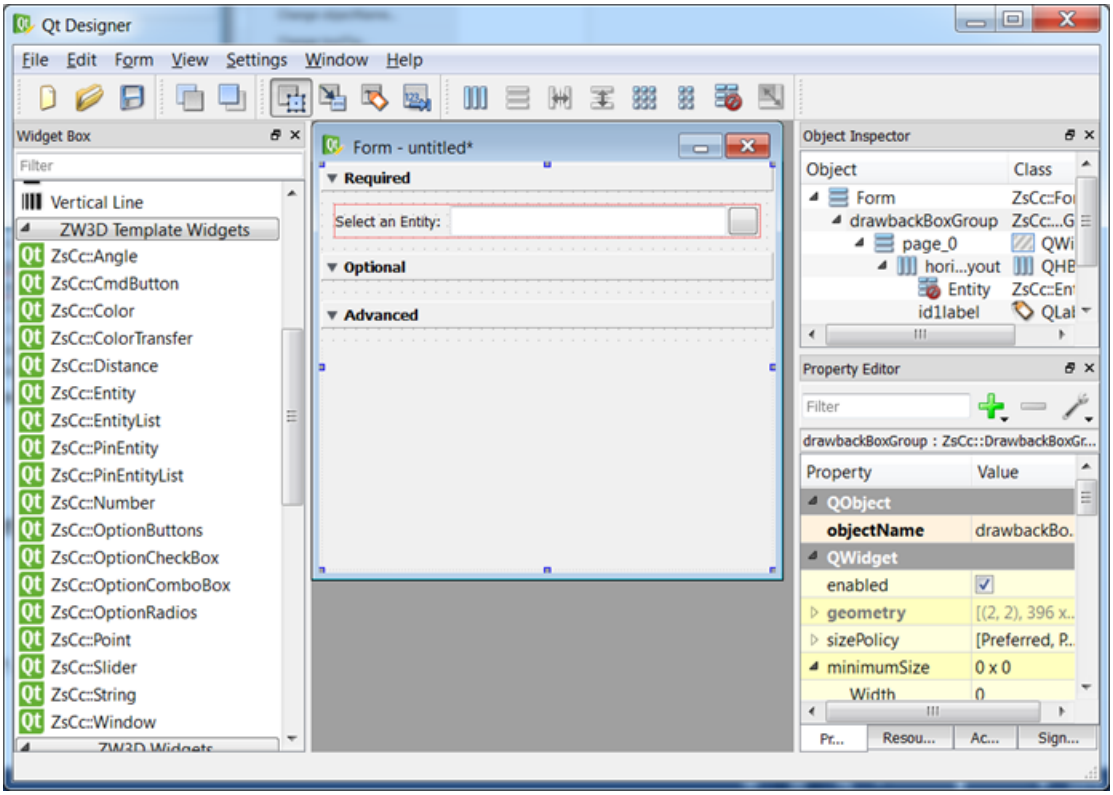
d) Use Qt function to layout these two controls. Choose both controls and right click to choose **Lay Out Horizontally**.



e) Right click on drawbackBoxGroup and choose **Lay Out Vertically**.



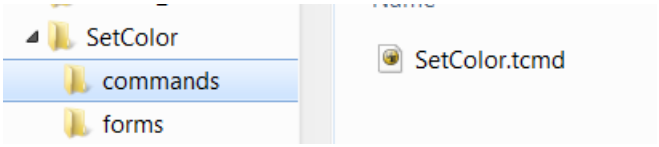
f) You will get the command dialog as follows:



g) Save the dialog to **SetColor** folder, named as **SetColor.ui**. You need to edit **SetColor**.tcmd by manual and refer to the platform template files provided by ZW3D. Please see details about key properties of tcmd in Chapter 5.

h) Put **SetColor.ui** in **forms** folder and put **SetColor.tcmd** in commands folder. You can

refer to the following figure:

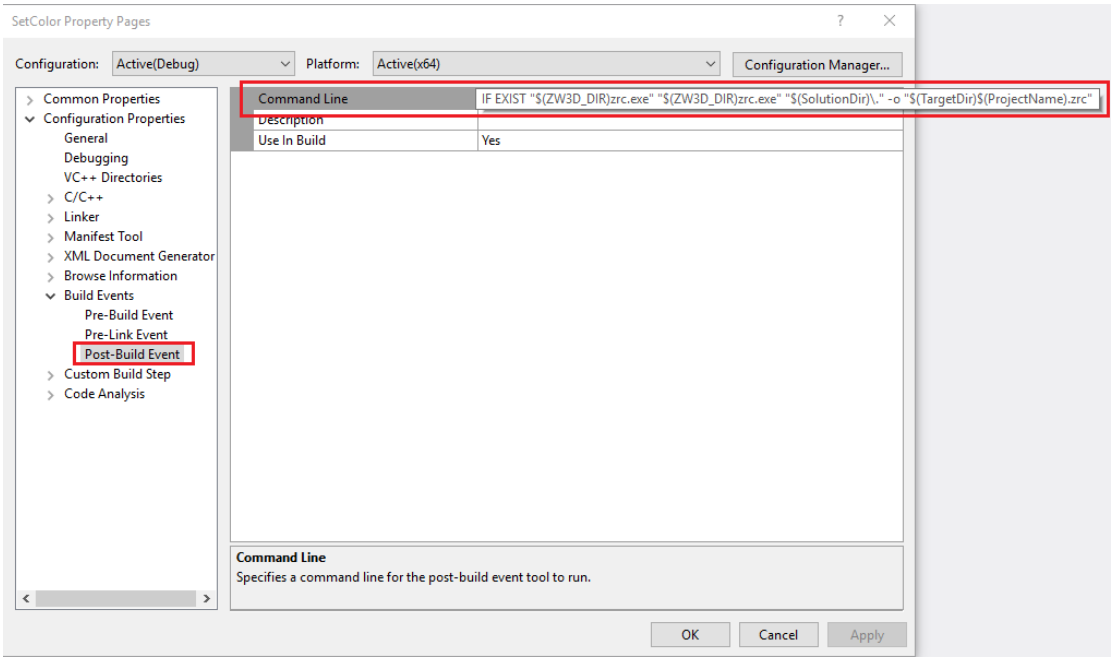


i) To compile resource, your project need add command line to ***Build Events→Post-Build Event→Command Line***.

The contents as shown below:

```
IF EXIST "$(ZW3D_DIR)zrc.exe" "$(ZW3D_DIR)zrc.exe" "$(SolutionDir)\." -o "$(TargetDir)$(ProjectName).zrc"
```

(Note: ZW3D_DIR is an environment variable whose value is ZW3D installation path)



Chapter 4: Use ZW3D Command Dialog

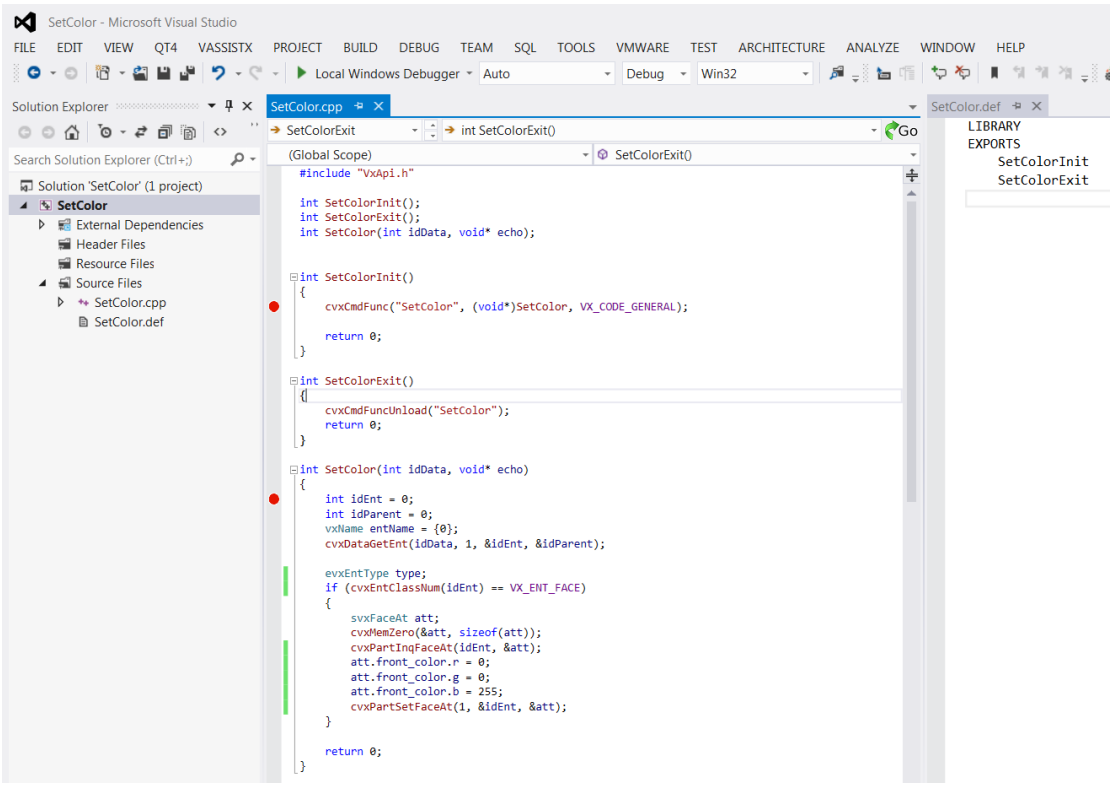
1. Refer to Chapter 1 to create an empty project and name it as **SetColor**.
2. Add a SetColor.cpp in the project, and input the code as follows:

```
#include"VxApi.h"

intSetColor(intidData, void* echo);

intSetColorInit(int format, void *data)
{
    cvxCmdFunc("SetColor", (void*)SetColor, VX_CODE_GENERAL);
    return0;
}
```

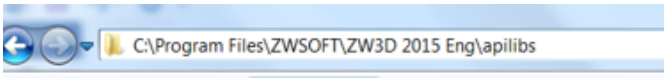
```
intSetColorExit(void)
{
	cvxCmdFuncUnload("SetColor");
	return0;
}

intSetColor(intidData, void* echo)
{
	intidEnt = 0;
	intidParent = 0;
	vxNameentName = {0};
	cvxDataGetEnt(idData, 1, &idEnt, &idParent);
		evxEntTypetype;
		if (cvxEntClassNum(idEnt) == VX_ENT_FACE)
		{
		svxFaceAtatt;
		cvxMemZero(&att, sizeof(att));
		cvxPartInqFaceAt(idEnt, &att);
		att.front_color.r = 0;
		att.front_color.g = 0;
		att.front_color.b = 255;
		cvxPartSetFaceAt(1, &idEnt, &att);
	}
	return0;
}
```

3. Add the Module Definition file, SetColor.def. Copy the code as follows:

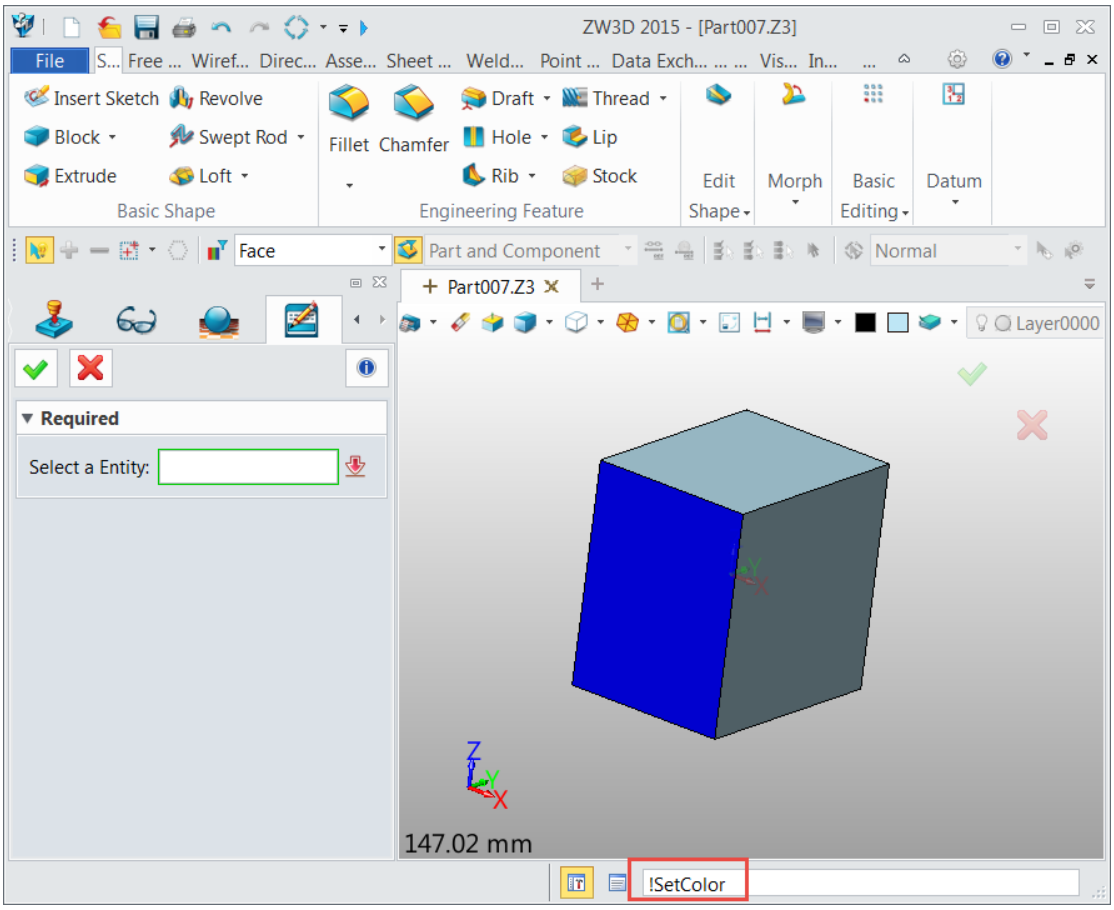
```
LIBRARY SetColor.dll
EXPORTS
	SetColorInit
	SetColorExit
	SetColor
```



4. Refer to Chapter 1 to set the configuration of the project and build to get **SetColor.dll** and **SetColor.zrc**. Now, we have the dll and zrc. Copy all the file to ZW3D installation directory.
   Note: *you must close ZW3D before you copy the files.*

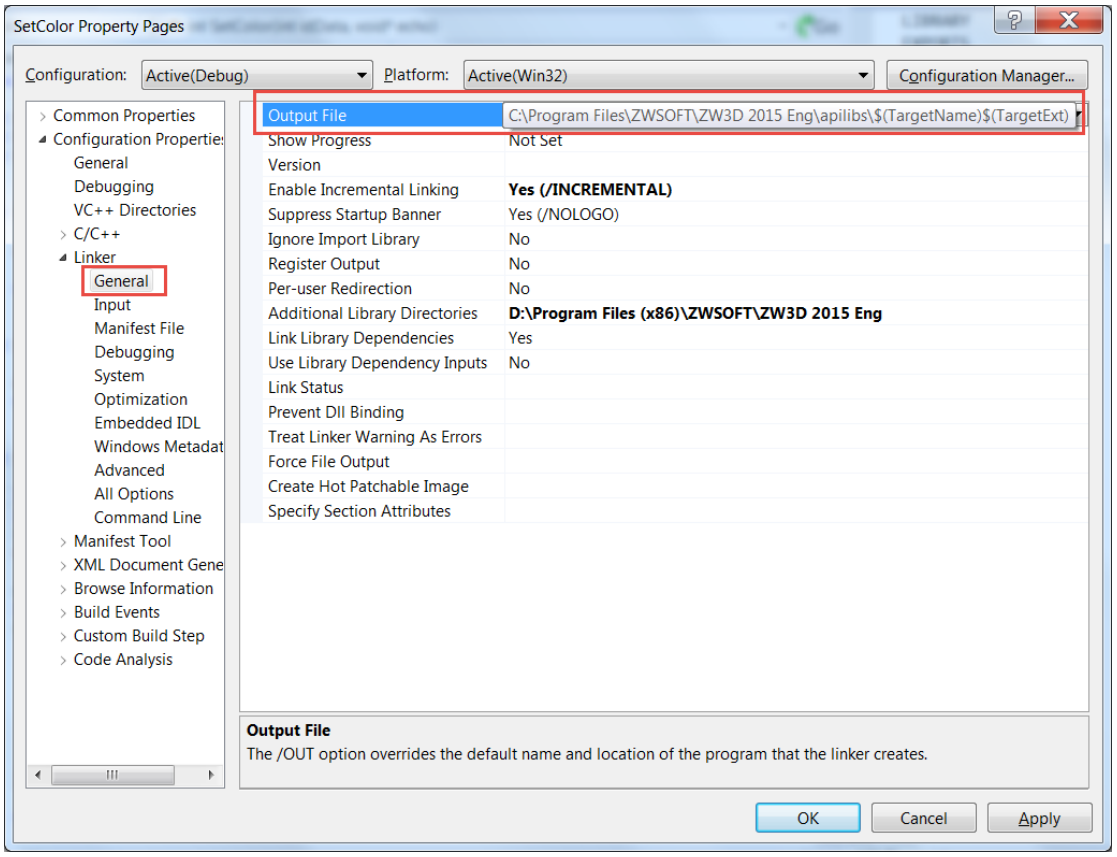


5. Start ZW3D, create a new part and draw a box. Then, input "***!SetColor***" in the command line. You can get the command dialog as follows. Select a surface and press OK, you can find the color of the surface was changed to blue.



6. Debug this project. Set the Output File to "C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs\$(TargetName)$(TargetExt)". You can refer to the picture as follows. Then, rebuild the project. You can set the break point and debug this project.

***Note**: ZW3D will load all the DLL in the "apilibs" directory automatically.*


SetColor Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadat
Advanced
All Options
Command Line
Manifest Tool
XML Document Gene
Browse Information
Build Events
Custom Build Step
Code Analysis
Output File
C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs\$(TargetName)$(TargetExt)
Show Progress
Not Set
Version
Enable Incremental Linking
Yes (/INCREMENTAL)
Suppress Startup Banner
Yes (/NOLOGO)
Ignore Import Library
No
Register Output
No
Per-user Redirection
No
Additional Library Directories
D:\Program Files (x86)\ZWSOFT\ZW3D 2015 Eng
Link Library Dependencies
Yes
Use Library Dependency Inputs
No
Link Status
Prevent Dll Binding
Treat Linker Warning As Errors
Force File Output
Create Hot Patchable Image
Specify Section Attributes
Output File
The /OUT option overrides the default name and location of the program that the linker creates.
OK
Cancel
Apply

Chapter 5: How to call ZW3D functions?

1. Get ZW3D command dialogs.
   Download all ZW3D command dialogs from the following link:
   https://dl.zwsoft.com/zw3d/Products/ZW3D_API/2023/zw3d-command-dialog_2023.rar
2. Tcmd key properties
   (1) Define Template command symbol
   Take Extrude command as an example. Define the relevant symbols of template command as follows:

```
<template name="FtAllExt">
    <property name="icon_id">11</property>
    <property name="function">FtAllExt</property>
    <property name="rt_echo">true</property>
    <property name="mod_entity">true</property>
    <property name="echo_obj">FtAllExtEO</property>
    <property name="fast_echo_obj">FtAllExtEO</property>
    <property name="init">FtBaseExtInit</property>
    <property name="init_after">FtBaseExtInitAf</property>
    <property name="show_tol">true</property>
    <property name="multi_cmd">14,</property>
    <property name="ref_id">32</property>
    <property name="esc_dlg">true</property>
```

   Introduction to common symbols as follows:

   a. **"template name"** is the command name

   b. "function" is the performed function of command, which is often the same name as command. Define function:

   int fun(int idData, int *idOut); or

   int fun(int idData);

   Note: idData is the corresponding container to the command. Use cvxDataSet to store the data of the corresponding field and use cvxDataGet to get the data of the corresponding field.

   c. **"echo_obj"** is the preview function of command(optional) and define function

   Int funEo(int idData);

   d. **"init"** is the initial function of command which can initialize data to the parameter container. As form has not been built when calling init, ui related interface access UI data cannot be called. Define function:

   void fun_init(int idData);

e. **"init_after"** is the initial function of command. At this point, form has been built, you can initialize some status of ui. Define function as follows:

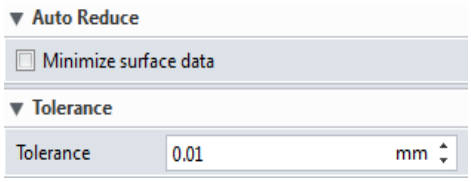
```
void fun_initAft(int idData);
```

f. **"term"** is exit function which will not be called in normal execution. If term involves with resource release, it may need to explicitly call term function. Define function as follows:

```
void term();
```

g. **"multi_cmd"** is mode of multiple commands. In the same command, according to the value of field of some option type, control the display status of field corresponding control. Such as in Block command, switch 8 types of fields, the other control status in the interface will change accordingly.

h. **"show_tol"** controls feature command interface whether to display Tolerance page as follows:



i. **"esc_dlg"** Whether the close command is automatically displayed when the command is executed slowly

j. **"custom"** self-defines feature command in user-defined feature as follows:

```
    <template name="TestCmd">
        <property name="function">CustomOp</property>
        <property name="custom">TestCmd.c</property>
        <parameters>
            <parameter trigger="true" luid="1" description="Distance" type="distance">
                <property name="prompt">Enter distance.</property>
            </parameter>
        </parameters>
    </template>
</templates>
```

standard function defined by ZW3D

command execution function defined by user

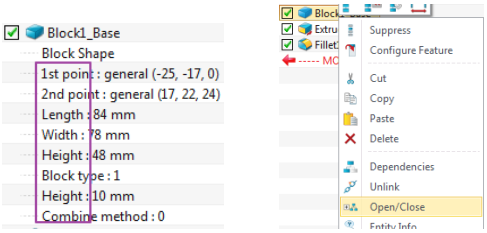
(2) Define field

Define field parameters as follows:

```
<parameter luid="14" description="Combine method" type="option">
    <property name="options">@sym_int=0,|enable=FtBoolHasShape,|auto_log,|reactivate,</property>
    <property name="callback">FtAllExtCb</property>
</parameter>
<parameter luid="1" description="Profile P" type="entity">
    <property name="options">pause_id=70,cmd=CdProfNew,filter=UiFiCrvFac,ent_extend,|rgn=FtPrfRgnCb,no_qpick_ech
    <property name="list">1</property>
    <property name="callback">FtAllPrfCapCb</property>
    <property name="prompt">|CdStrProfile|</property>
    <property name="inv_pick">true</property>
</parameter>
```

a. **Parameters self attributes:**

- **trigger**-command execution symbol which was a required input previously and is now an optional input. Meanwhile, you can end the command by middle mouse button.
- **description**- field characters to display. Currently, this parameter is used to display the parameter name after the open operation is performed on the feature node.



- **checker**-User checks whether the value of this field is empty. If empty, end the command, used for set-list.
- **type**-null, entity, point, option, number/distance/angle, form, command, continue, string.

b. **Option general properties are set as follows:**

- Allow to input empty： empty_ok
- Field condition of availability:|enable=fun. Compared with callback, the number of calls enable function will be more frequent, other control parameter changes will be triggered for calling.

  enable function defines as follows:

  int funEn();

  Note: return 1, the field is not available, otherwise return 0.
- no_qpick_echo： do not execute echo function when choosing entity

c. **callback-callback function, definition as follows:**

- callback function：<property name="callback">funCb</property>

int funCb(char* formName, int idField, int idData);

d. **next-skip to the next field after this control input, and properties defines as follows:**

- automatically skip: <property name="next">10</property>

Automatically skip to the control id=10 after this control input.

(3) entity

Get entity input, generally corresponding to the entity or the entity list control (Entity/EntityList/EntityTable)

General properties of entity define as follows:

a. **options**-control the comprehensive options of control behavior

General behaviors include:

- Filter: /shell/face/curve/edge/, etc., custom_filter=fun

a. /face/curve/edge/ as entity type and separate with '/'. If entity inside component can be selected, it can be changed to /Eface/Edge/…

b. custom_filter is user-defined filter setting which can be set limitation to select.

Function defines as follows:

```
int funFilter(int idx_ent);
```

Note: return 1, idx_ent can be selected; otherwise, return 0.

c. General entity filter goes as the following table, separate with '/':

Entity object	Filter
Shape	shell
Face	face
Edge	edge
Line	curve
Point	point
Curve list	clist
Sketch	sketch
Datum	datum
Axis	axis
CSYS	csys
Feature	ftr
Component	comp
Text	text

Dimension	dim

- Select object to check and limit:chk_line, chk_plane(use with other filters)
  a. chk_line: check whether it is a line
  b. chk_plane: check whether it is a plane

b. **list**-Multiple choice or not, 1 means multiple choice, single choice is not allowed to set, the properties are defined as follows:

```
<property name="list">1</property>
```

c. **comp**-generally, used from setlist associating form, and used as a pair with field form in sub template command, and store data from the template command, the properties are defined as follows:

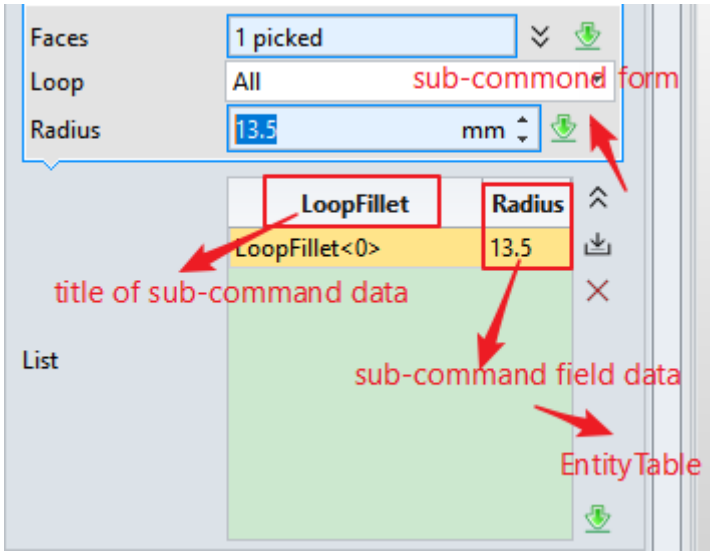
```
<property name="comp">31</property>
```

a. SetList the method to define sub-command is as follows:

```
<parameter luid="39" description="Set" type="form">
    <property name="options">empty_ok,</property>
    <property name="comp">40</property>
    <property name="template">FtFlltLoopSet</property>
</parameter>
<parameter luid="40" description="Loop fillet list" type="entity">
    <property name="options">empty_ok,</property>
    <property name="list">1</property>
    <property name="comp">39</property>
    <property name="info_id">5</property>
    <property name="format">LoopFillet</property>
</parameter>
```

form of the set-list
template name of the sub-commond
save the data of sub-command. what stored in each group of data is data container of the sub-command
after folding the table list, display field data of sub-commond
title of the sub-command display data

b. UI display as follows:



(4) point

Get point input, generally it's corresponding to point control

a. **options**-the comprehensive options to control the control behaviors

- Filter: /face/curve/edge/, etc.,custom_filter=fun, similar with entity control:or/direction/ means direction.
- ddd_drag: dray dynamically, it's available if control as point and not available as direction
- on_ent: the required point on entity
- hi_ent: the selected entity highlights
- get_dir: means direction which will automatically get direction value
- rev_dir: means direction which can reverse direction through mouse clicking the direction arrow

(5) 2.3 option

Get user's options and it's generally corresponding to the controls of (OptionButtons/ OptionCheckBox/ OptionComboBox/ OptionRadios)

a. **options**-control the comprehensive options of control behaviors

- Automatically initialize data: |auto_log(option)
- Set default value: @sym_int=0

(6) number

Get value input, without unit, generally corresponding to number control (Number)

a. **options**-control the comprehensive options of the control behavior

- Set default value: @sym_dbl=0.0(will record the last set value), or val=0.0
- Set the minimum value: min=-10000.0
- Set the maximum value: max=10000.0
- Set the increment: inc=15.0,

(7) distance

Get value input, with unit, generally generate dimension, corresponding to distance control (Distance)

a. **options**-control the comprehensive options of the control behavior

- Set basic value the same as number
- ddd setting: property name="dim", please see *Dynamic Drag Dimension DDD Development Guide* for details

(8) angel

Get value angle input, generally corresponding to angle control (Angle)

a. **options**-control the comprehensive options of the control behavior

- Set base value the same as number
- ddd setting: property name="dim", please see *Dynamic Drag Dimension DDD Development Guide* for details

(9) form

Display a specified GUI form, corresponding to (FormProxy), which is used in setlist case, embedded a sub-template command, or display a GUI form.

Corresponding to setlist setting is as follows:

a. **options**-control comprehensive options of the control behavior

- Allow to input empty: empty_ok

b. comp-generally used in setlist associated entity, please see entity control comp

- <property name="comp">2</property>, 2 serves as entity control id

c. Template-sub-template setting

- <property name="template">FtFlltEdgSet</property>

(10) continue

Waiting, generally used in middle mouse button to end command, set between the required option and optional option.

a. **options**-control the comprehensive options of the control behavior

- End command: ~CdSkipEnd,

b. prompt-prompt

- Prompt: <property name="prompt"><middle-click> to finish.</property>

(11) string

Get character string input, generally corresponding to character string control (String)

(12) Field access and setting

a. Setting

cvxDataSet used in setting the data of specified field

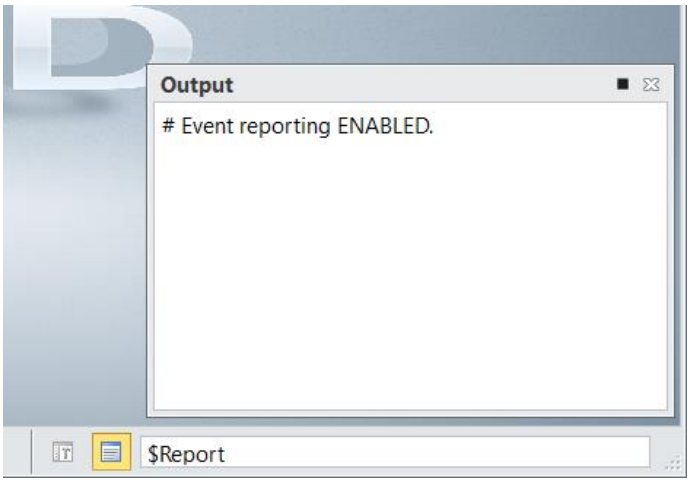
b. Get

(13) cvxDataGet user gets specified data of field

3. Reference to ZW3D command dialog.

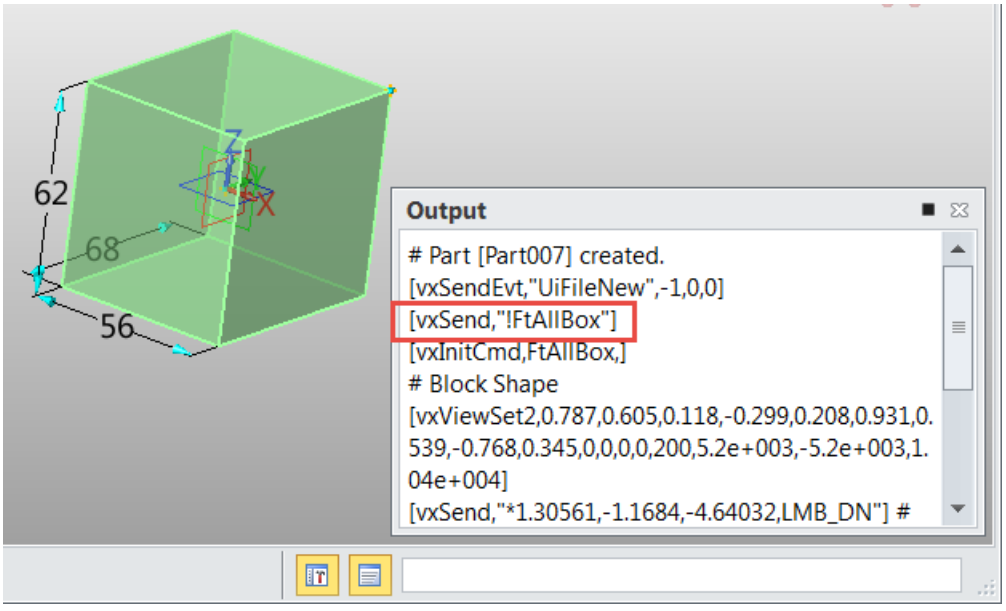
a) Run "$Report" in the command line of ZW3D.

You will get the following message after you run the command successfully.

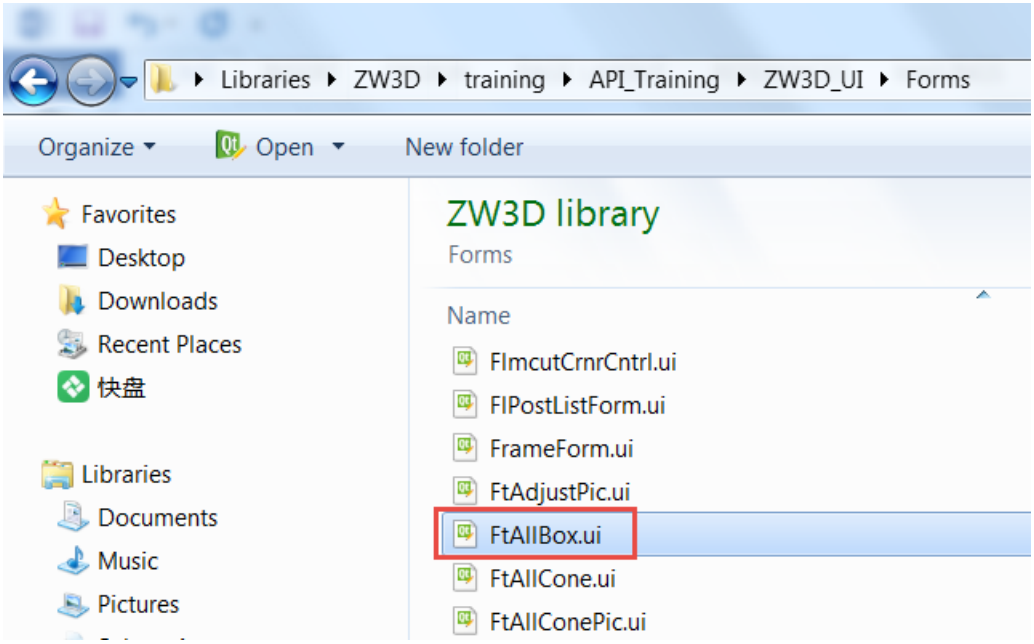


b) Run the command you need to know, for example the function to create Box.

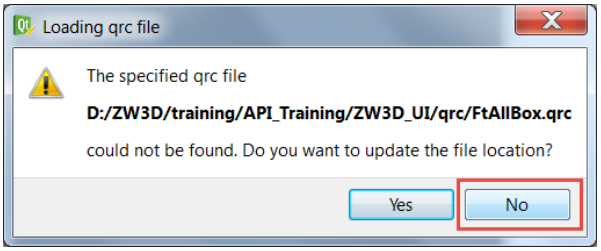
You will get the message as below, "[vxSend,"!FtAllBox]", which means ZW3D run "!FtAllBox" to create the box.



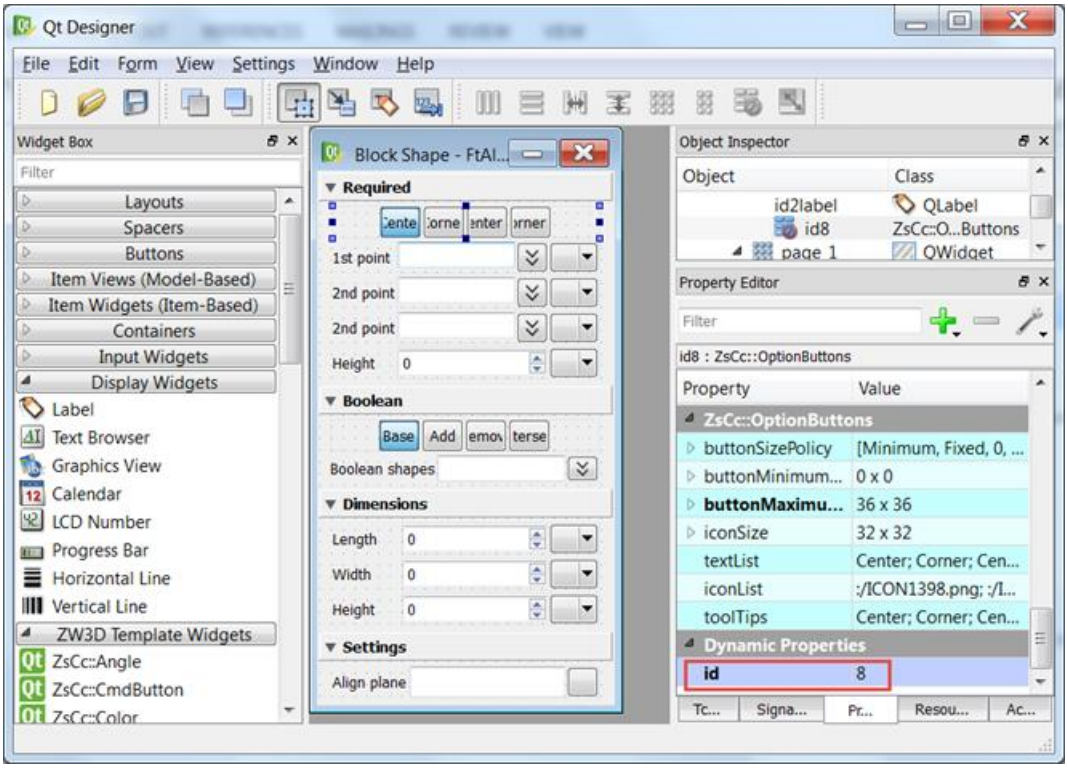
c) Go to ZW3D Command dialog folder, you can find a command dialog named as "FtAllBox.ui".



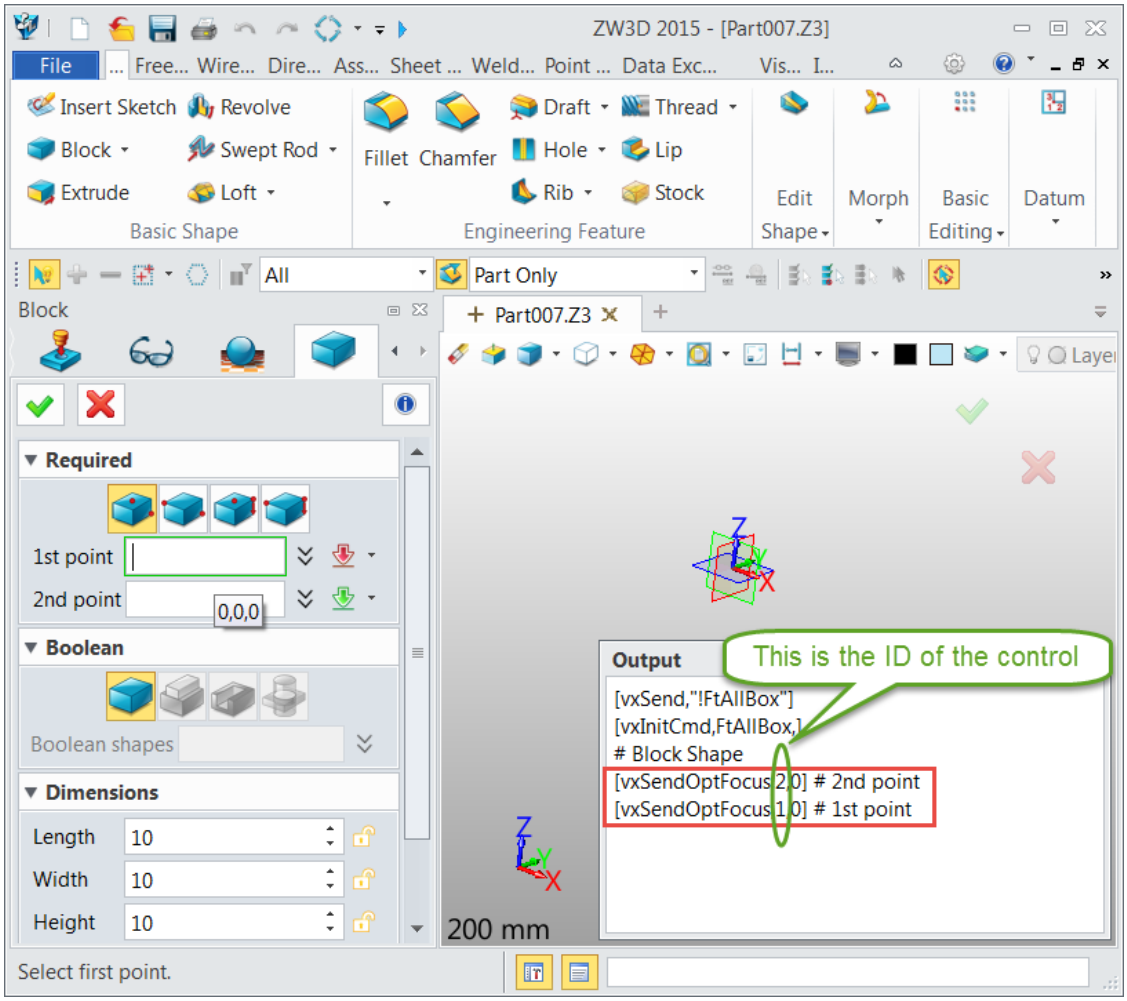
d) Use ZW3D UI designer to open this dialog, you will get a warning. Press **No** to ignore it.



e) You will know all the IDs of the controls.



f) You will find there are two "2nd point" controls. To learn the difference, refer to step b) to know the function. Then, change your mouse on different controls in the command dialog. You can get different message if you change the controls. You can find the ID in the messages.



4. Now, you already know the function name and the control IDs. Let's use ZW3D function to create a box although ZW3D also support the API function to create a box [cvxPartBox()].

a) Refer to chapter 1 to create a new project and named as "myBox".

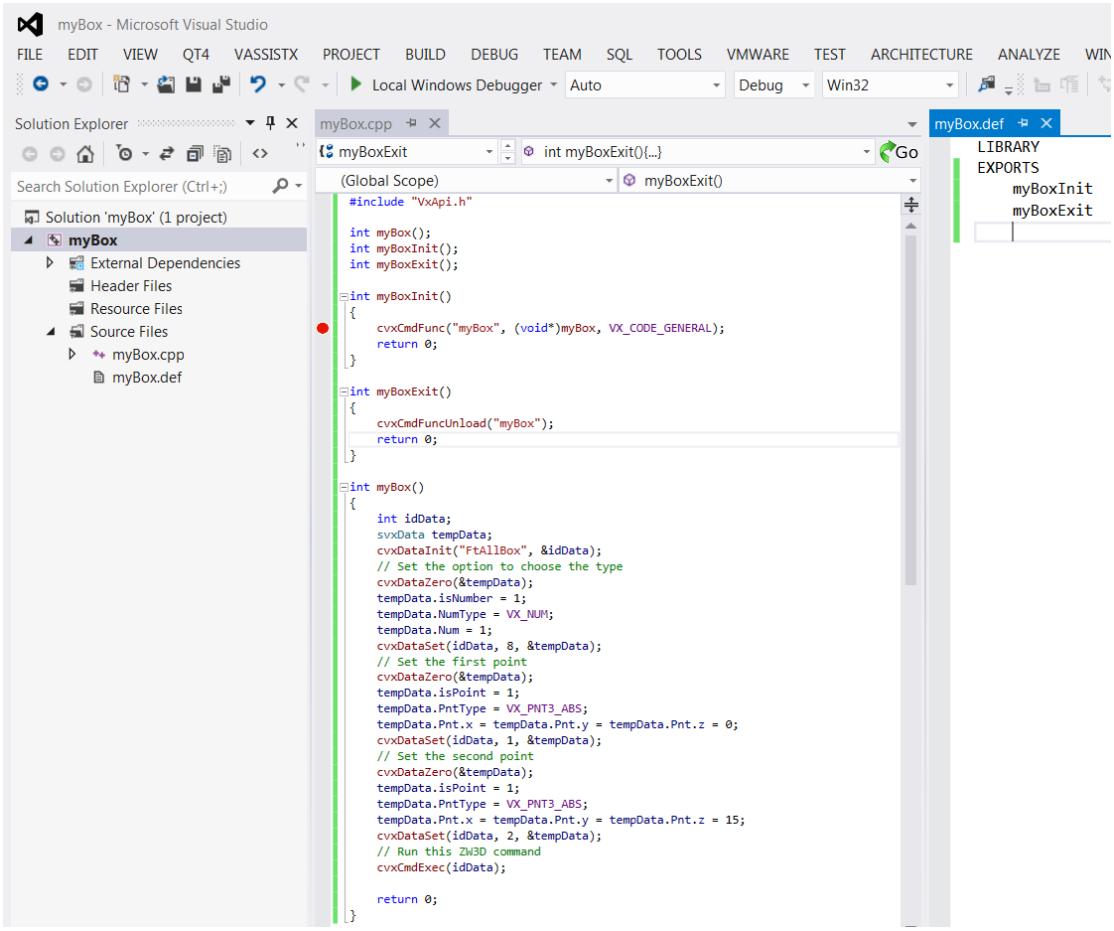
b) Add myBox.cpp, and copy the code as follows:

```
#include"VxApi.h"

intmyBox();

intmyBoxInit(int format, void *data)
{
    cvxCmdFunc("myBox", (void*)myBox, VX_CODE_GENERAL);
    return0;
}
```

```
intmyBoxExit(void)
{
    cvxCmdFuncUnload("myBox");
    return0;
}

intmyBox()
{
    intidData;
    svxDatatempData;
    cvxDataInit("FtAllBox", &idData);
    // Set the option to choose the type
    cvxDataZero(&tempData);
    tempData.isNumber = 1;
    tempData.NumType = VX_NUM;
    tempData.Num = 1;
    cvxDataSet(idData, 8, &tempData);
    // Set the first point
    cvxDataZero(&tempData);
    tempData.isPoint = 1;
    tempData.PntType = VX_PNT3_ABS;
    tempData.Pnt.x = tempData.Pnt.y = tempData.Pnt.z = 0;
    cvxDataSet(idData, 1, &tempData);
    // Set the second point
    cvxDataZero(&tempData);
    tempData.isPoint = 1;
    tempData.PntType = VX_PNT3_ABS;
    tempData.Pnt.x = tempData.Pnt.y = tempData.Pnt.z = 15;
    cvxDataSet(idData, 2, &tempData);
    // Run this ZW3D command
    cvxCmdExec(idData);

    return0;
}
```

c) Add the Module Definition file, myBox.def. Copy the code as follows:

```
LIBRARY myBox.dll
EXPORTS
    myBoxInit
    myBoxExit
    myBox
```

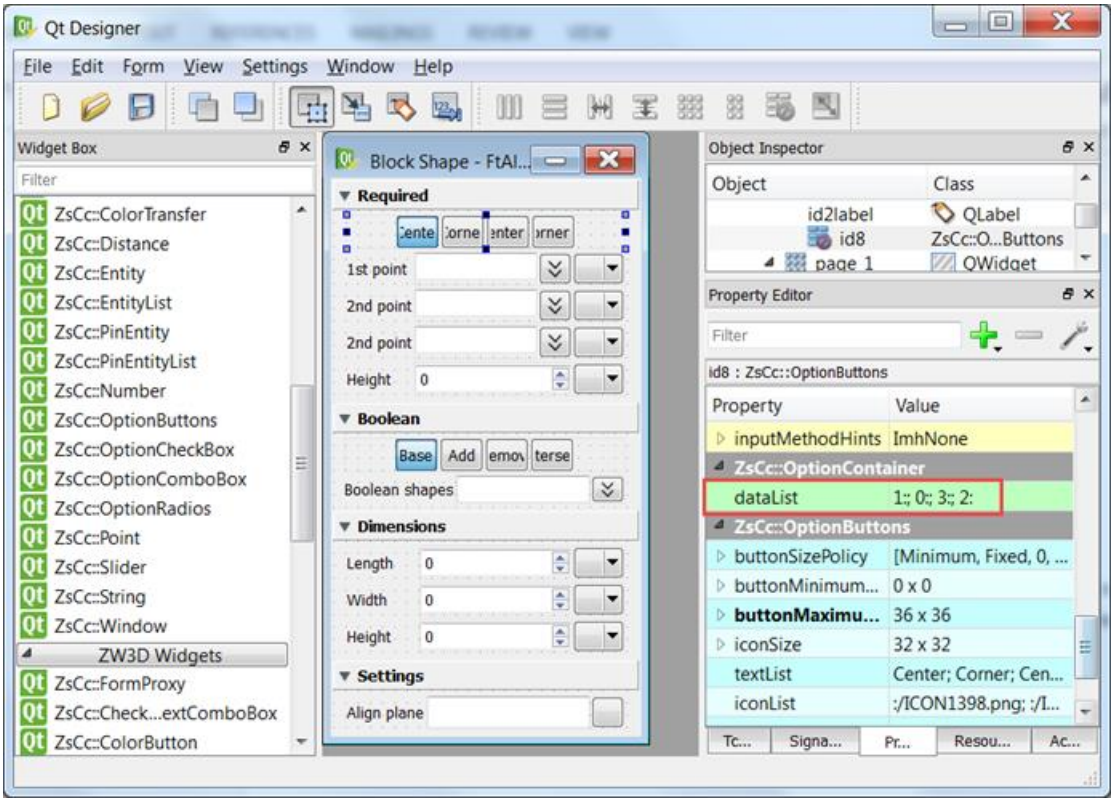


Build this project and load the DLL in ZW3D. Run "~myBox" in the command line after you create a new part. This command will create a box in the modeling space.

5. The value of the ZsCc::optionbuttons.

   After clicking on the control, you can find the **DataList** property, where the value order of the control should be 1, 0, 3, 2. So you need to set the right value in your code. The following code means this project use the first button. If you want to use the second button, you need to set the value to 0.

```
// Set the option to choose the type
    cvxDataZero(&tempData);
    tempData.isNumber = 1;
    tempData.NumType = VX_NUM;
    tempData.Num = 1;
    cvxDataSet(idData, 8, &tempData);
```


Qt Designer
File Edit Form View Settings Window Help
Widget Box
Filter
ZsCc::ColorTransfer
ZsCc::Distance
ZsCc::Entity
ZsCc::EntityList
ZsCc::PinEntity
ZsCc::PinEntityList
ZsCc::Number
ZsCc::OptionButtons
ZsCc::OptionCheckBox
ZsCc::OptionComboBox
ZsCc::OptionRadios
ZsCc::Point
ZsCc::Slider
ZsCc::String
ZsCc::Window
ZW3D Widgets
ZsCc::FormProxy
ZsCc::Check...extComboBox
ZsCc::ColorButton
Block Shape - FtAl...
Required
1st point
2nd point
2nd point
Height 0
Boolean
Boolean shapes
Dimensions
Length 0
Width 0
Height 0
Settings
Align plane
Object Inspector
Object
Class
id2label
QLabel
id8
ZsCc::O...Buttons
page 1
QWidget
Property Editor
Filter
id8 : ZsCc::OptionButtons
Property
Value
inputMethodHints
ImhNone
ZsCc::OptionContainer
dataList
1:; 0:; 3:; 2:
ZsCc::OptionButtons
buttonSizePolicy
[Minimum, Fixed, 0, ...
buttonMinimum...
0 x 0
buttonMaximu...
36 x 36
iconSize
32 x 32
textList
Center; Corner; Cen...
iconList
:/ICON1398.png; :/I...

Chapter 6: ZW3D Register information

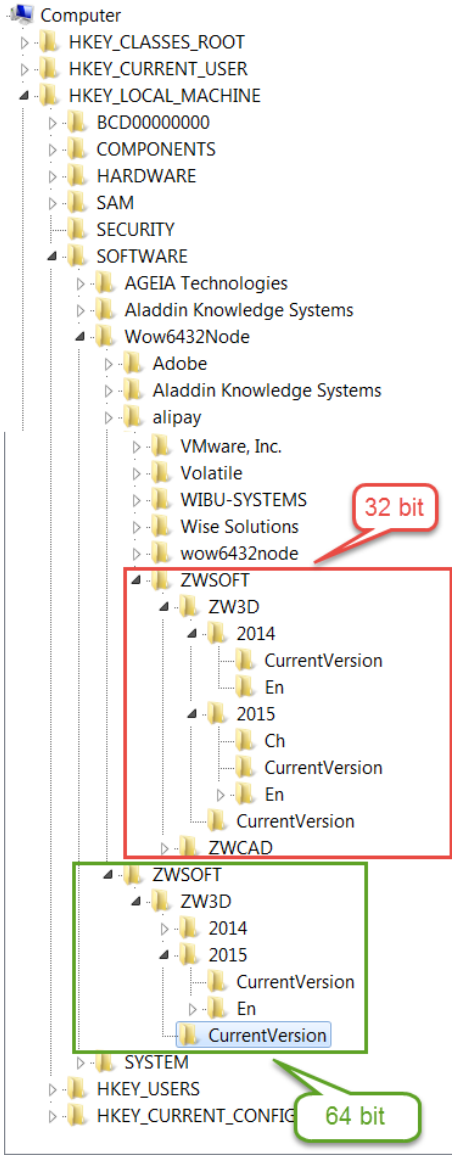
The register of ZW3D is saved in:

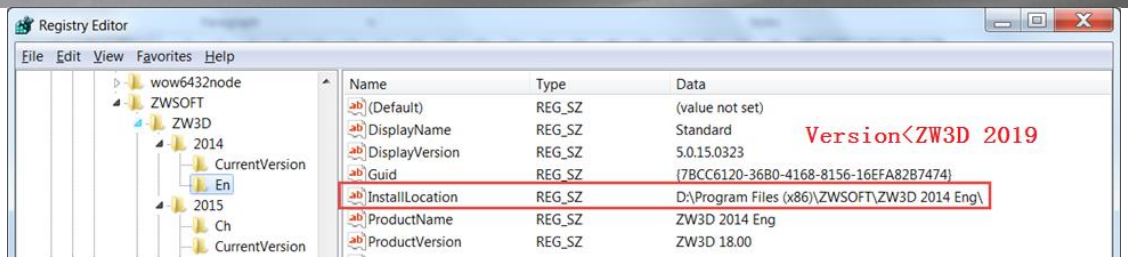
64 bit: [HKEY_LOCAL_MACHINE\SOFTWARE\ZWSOFT\ZW3D]

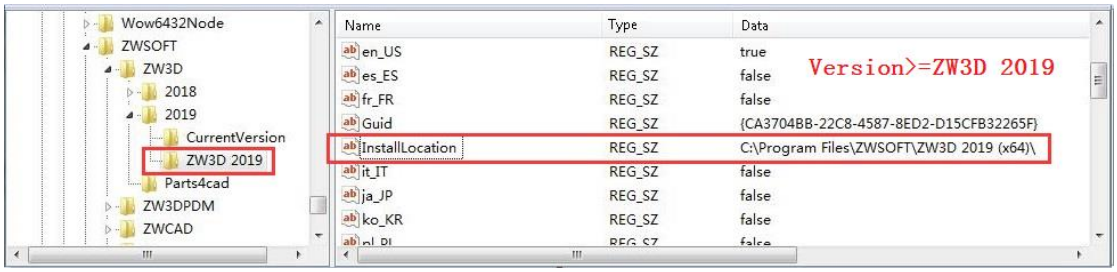
32 bit: [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ZWSOFT\ZW3D]

The **CurrentVersion** in the root directory remembers the version run last time. The **CurrentVersion** in the special version directory remembers the language run last time.

You can also get the detailed information in each language directory, like installation path.




Registry Editor
File Edit View Favorites Help
wow6432node
ZWSOFT
ZW3D
2014
CurrentVersion
En
2015
Ch
CurrentVersion
Name Type Data
(Default) REG_SZ (value not set)
DisplayName REG_SZ Standard
DisplayVersion REG_SZ 5.0.15.0323
Guid REG_SZ {7BCC6120-36B0-4168-8156-16EFA82B7474}
InstallLocation REG_SZ D:\Program Files (x86)\ZWSOFT\ZW3D 2014 Eng\
ProductName REG_SZ ZW3D 2014 Eng
ProductVersion REG_SZ ZW3D 18.00
Version<ZW3D 2019


Wow6432Node
ZWSOFT
ZW3D
2018
2019
CurrentVersion
ZW3D 2019
Parts4cad
ZW3DPDM
ZWCAD
Name Type Data
en_US REG_SZ true
es_ES REG_SZ false
fr_FR REG_SZ false
Guid REG_SZ {CA3704BB-22C8-4587-8ED2-D15CFB32265F}
InstallLocation REG_SZ C:\Program Files\ZWSOFT\ZW3D 2019 (x64)\
it_IT REG_SZ false
ja_JP REG_SZ false
ko_KR REG_SZ false
Version>=ZW3D 2019