# ZW3D API 简介

本文档仅提供 ZW3D API 的基本介绍。我们推荐使用 C/C++开发基于 ZW3D 的附加组件。我们将不断改进 API，使其更加支持我们合伙人基于 ZW3D 开发强大的应用程序。

如您有任何建议或诉求，请随时联系我们。
您可以邮件到 zdn@zwsoft.com 寻求帮助。
谢谢合作！

中望 3D™，ZW3D™是广州中望龙腾软件股份有限公司正在注册中的商标。
中望 3D™图案，ZW3D™图案是广州中望龙腾软件股份有限公司正在注册中的商标。
中望®、中望软件®、ZWCAD®, ZWSOFT®及其图案均为广州中望龙腾软件股份有限公司已注册成功的商标。中望 CAD™及其图案是广州中望龙腾软件股份有限公司正在注册中的商标。

打印于中华人民共和国。

# 目录

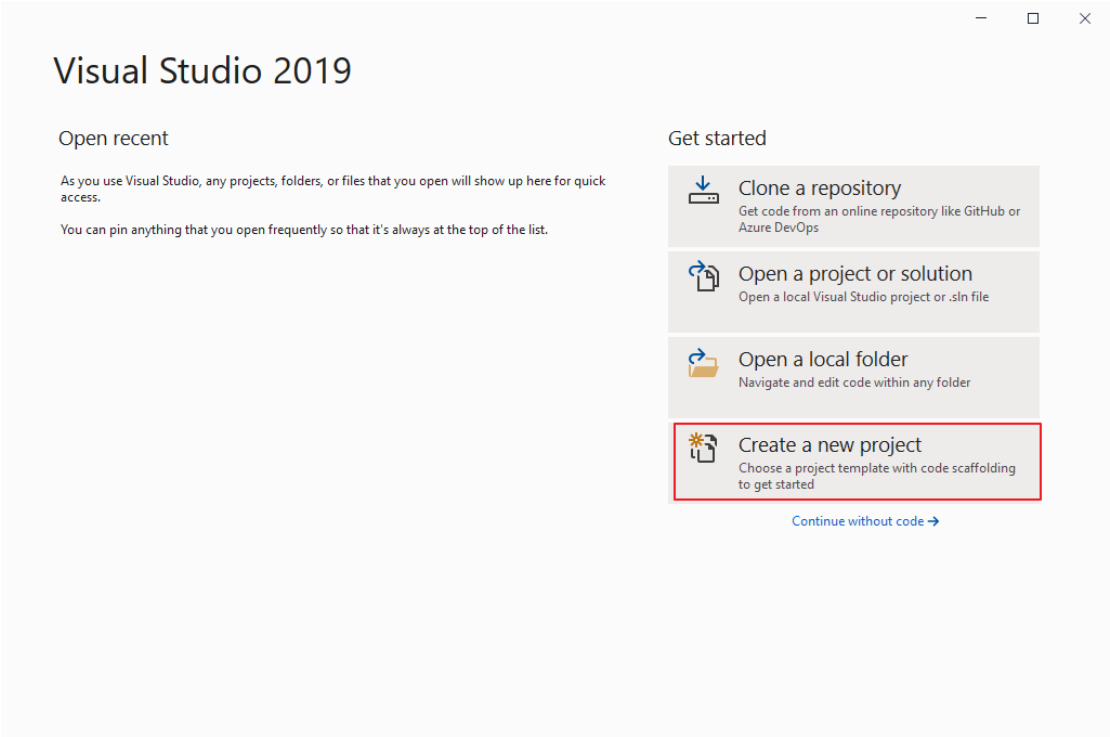
# ZW3D API 简介

## 第一章开始 ZW3D API

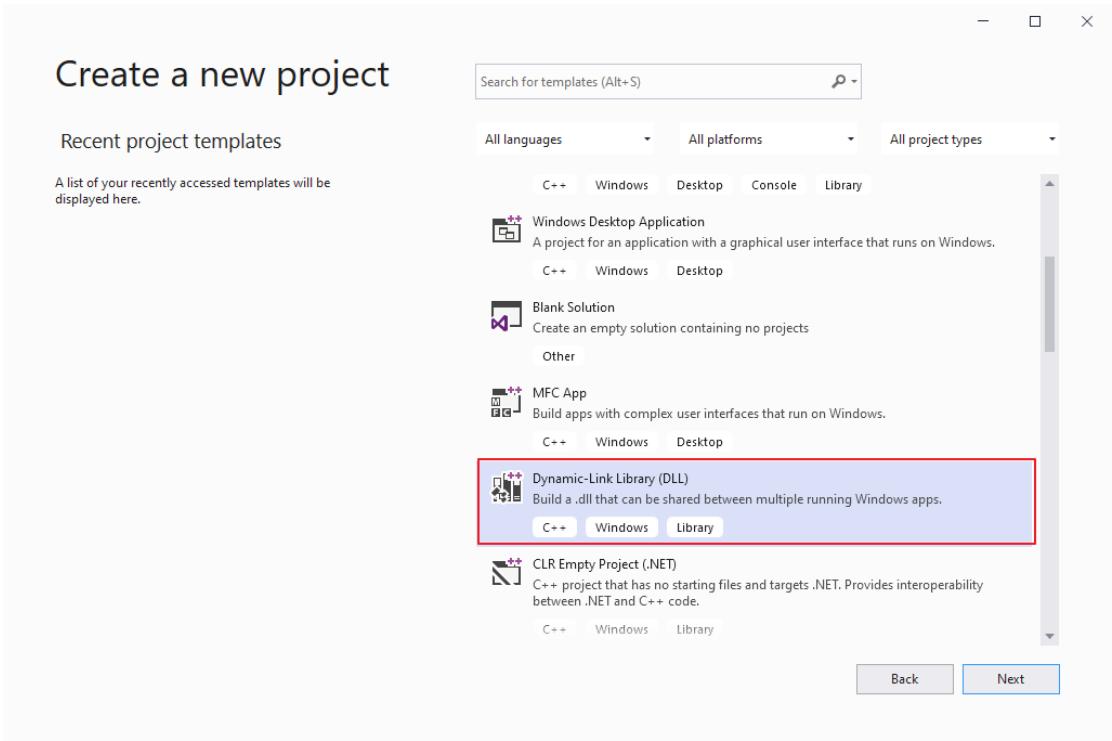
1. 系统要求
   a) Windows 7 或以上
   b) Visual studio 2019 （或 C/C++的任何其他集成开发环境）
   c) ZW3D 2012 或以上版本
   d) Qt5.9.7(for Windows)

   ***备注：****用户在安装 Qt 之后，可以双击 ZW3D 安装目录下 api 文件夹中的“CopyQtDll.bat”脚本完成 ZW3D 自定义控件的安装。安装完成之后，打开 Qt Designer 插件，即可使用 ZW3D 提供的自定义控件。*

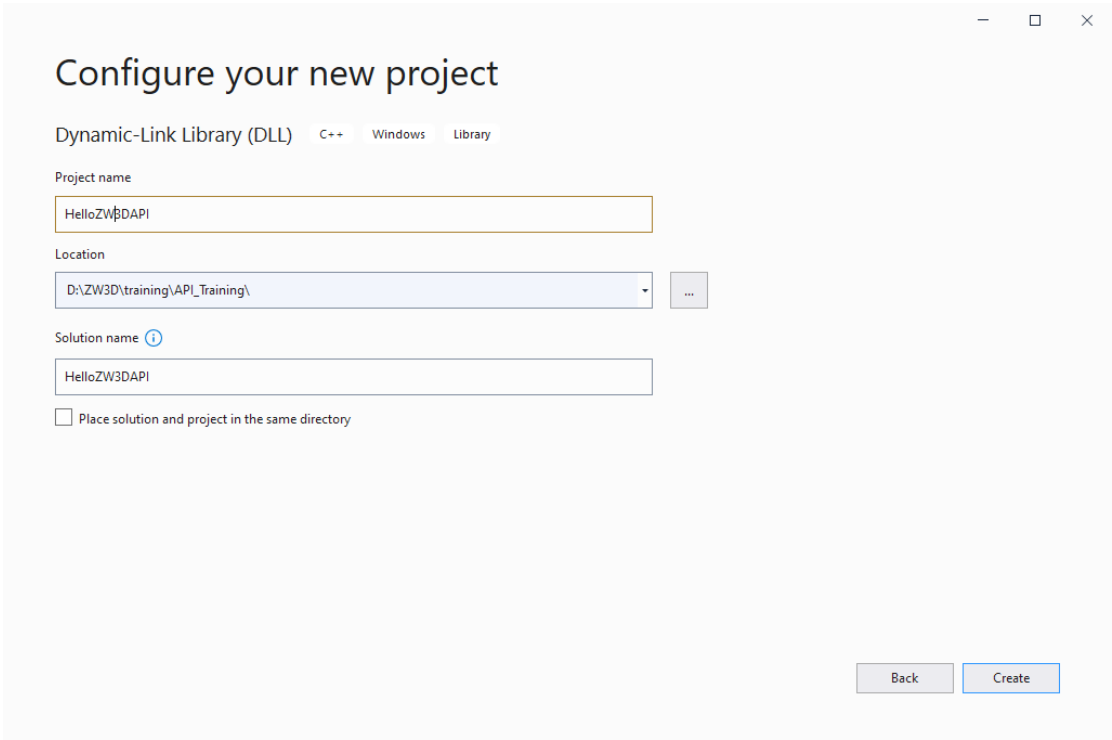
2. 新建项目
   a) 打开 Visual Studio 2019，选择***创建一个新项目。***



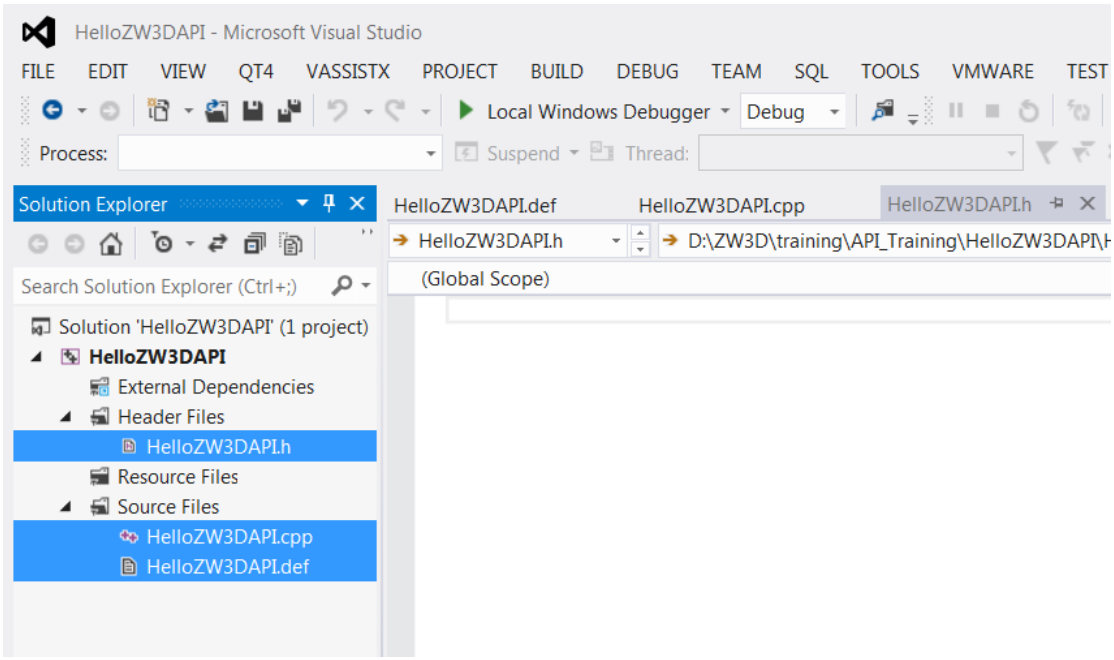

   b) 选择***动态链接库***，然后点击 ***下一步***。




c) 输入名称为 ***HelloZW3DAPI***，然后点击***创建***。

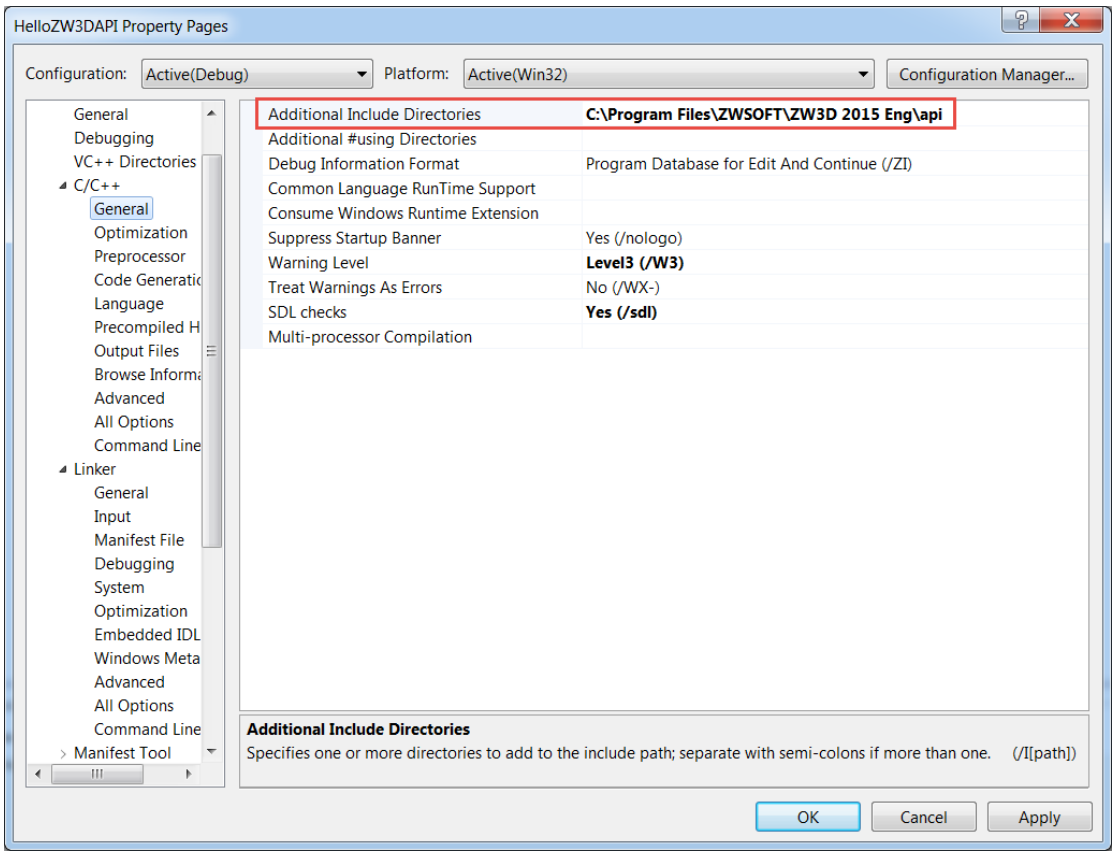



3. 参考下图，在项目下添加以下文件：HelloZW3DAPI.h、HelloZW3DAPI.cpp、HelloZW3DAPI.def。

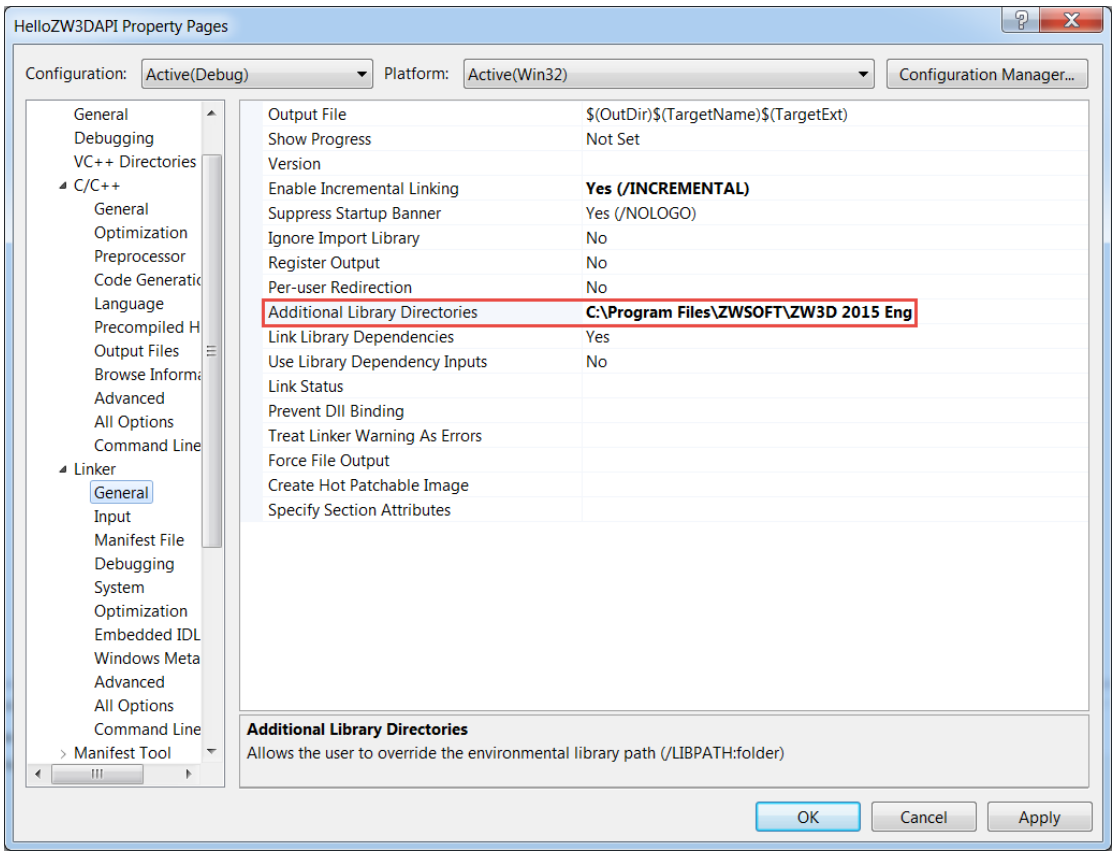



4. 设置项目。

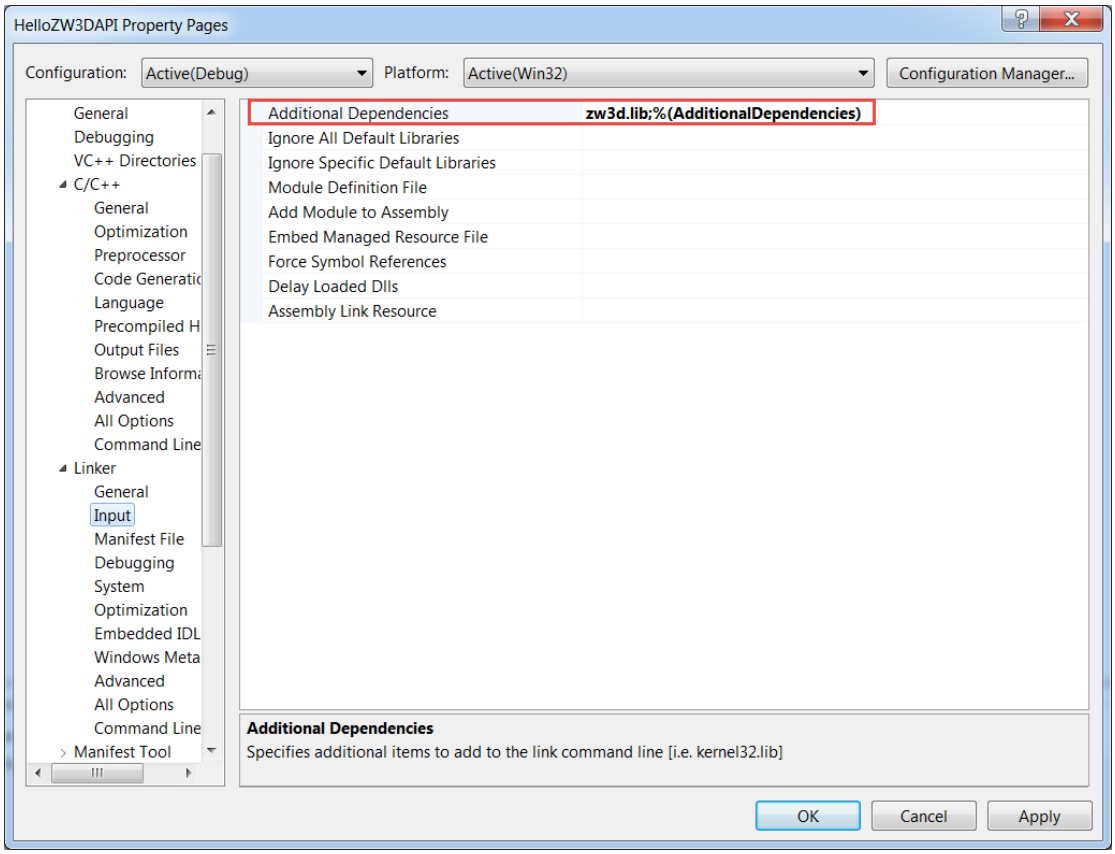
   右键点击项目，选择***属性***，将 ZW3D API 头文件目录添加至：***C/C++ →常规→附加包含目录。***



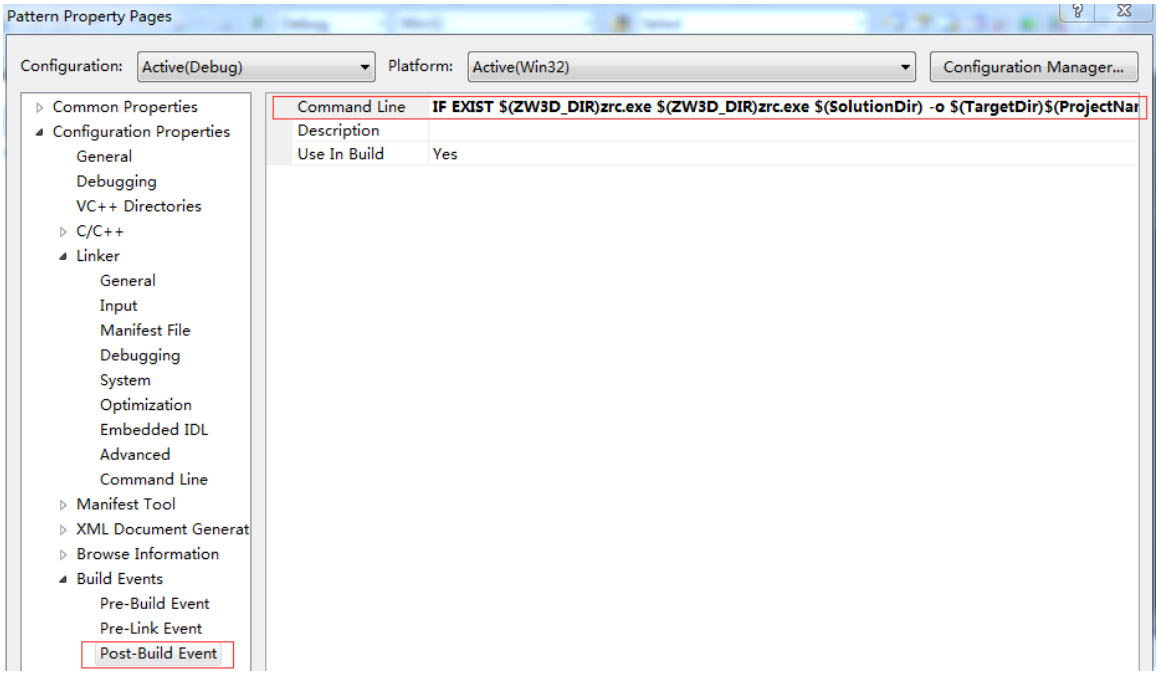

a) 将 ZW3D API 库文件目录添加至：***链接器→常规→附加库目录。***




b) 将 ZW3D API 库添加至：***链接器→输入→附加依赖项***



c) 要编译资源，请将以下的命令行添加至：***生成事件→后期生成事件→命令行。***




（***注意：****如您的项目下没有资源文件，比如图片/UI，可忽略此步*）

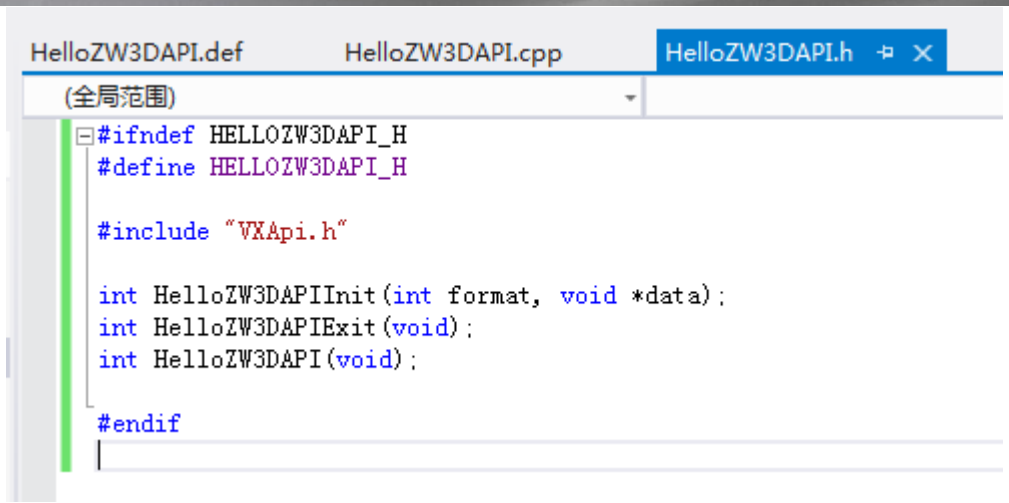
*命令行的内容如下：*

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

（***注意：****ZW3D_DIR 是一个环境变量，其值是 ZW3D 安装路径*）

5. 在 HelloZW3DAPI.h 添加函数定义，您可以直接复制以下代码：

```
#ifndefHELLOZW3DAPI_H
#defineHELLOZW3DAPI_H

#include"VXApi.h"

intHelloZW3DAPIInit(int format, void *data);
intHelloZW3DAPIExit(void);
intHelloZW3DAPI(void);

#endif
```




***注意：****函数的前缀必须与项目名保持一致，这意味着：*
*@customizedApp@Init() 和 @customizedApp@Exit() 是应用程序的入口函数。当 ZW3D 加载@customizedApp@.dll 时，会检查@customizedApp@Init() 和 @customizedApp@Exit() 以了解自定义函数。*
*(@customizedApp@ 表示您所创建的任何项目的名称)*

6. 在 HelloZW3DAPI.cpp 中实现函数。您可以直接复制以下代码

```
#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. 定义模块定义文件 HelloZW3DAPI.def。
   您可以直接复制以下代码：

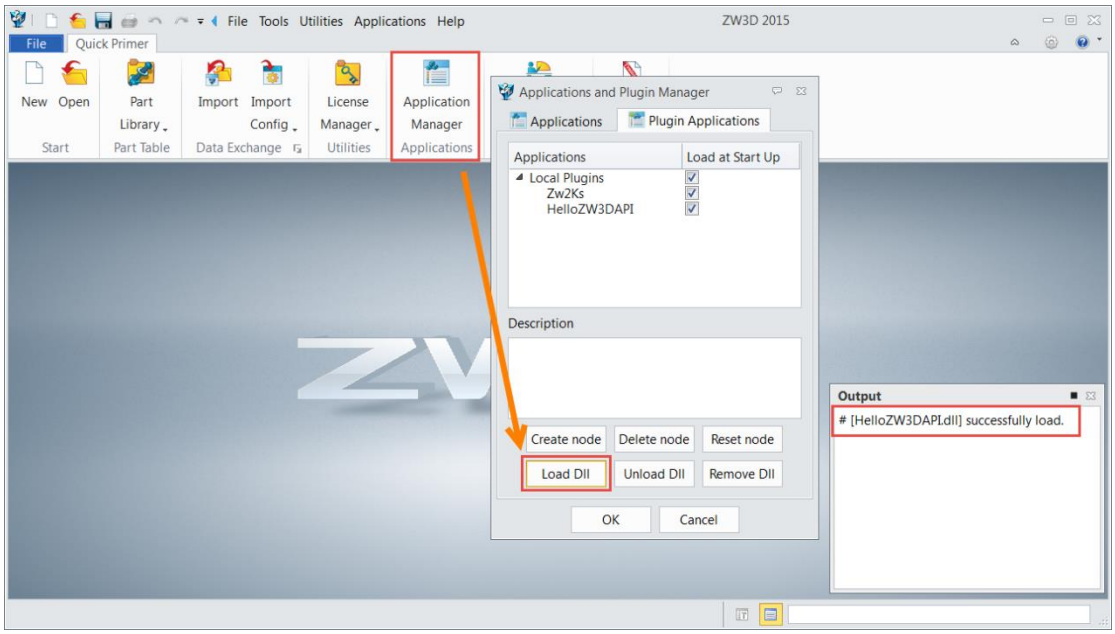
```
LIBRARY HelloZW3DAPI.dll
EXPORTS
    HelloZW3DAPIInit
    HelloZW3DAPIExit
    HelloZW3DAPI
```

HelloZW3DAPI.def | HelloZW3DAPI.cpp | HelloZW3DAPI.h

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

8. 生成项目。
   右键点击项目，生成项目。您可以在目录：".\HelloZW3DAPI\Debug\HelloZW3DAPI.dll"找到 HellowZW3DAPI.dll。

9. 加载 HelloZW3DAPI.dll。
   a) 使用 ZW3D 的***应用程序管理器***来加载这个 DLL 文件，***输出***对话框的信息将显示命令成功还是失败。
   ***注意:****关闭 ZW3D 时，会将 DLL 的路径记录到注册表，再次启动 ZW3D 时将自动加载此 DLL。*

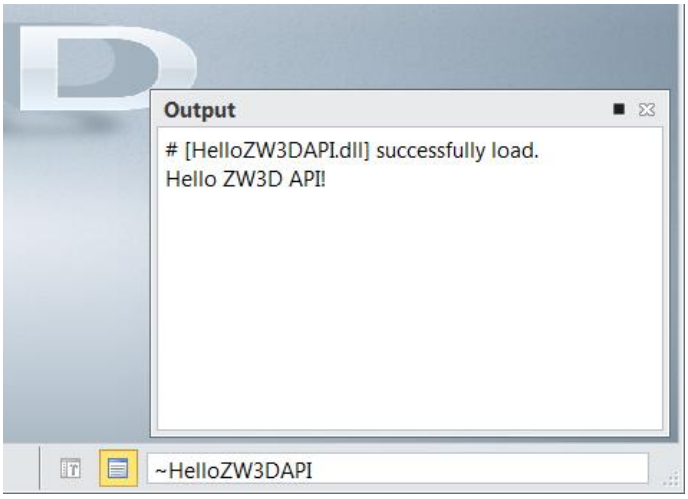



b) 复制 HelloZW3DAPI.dll 至安装文件夹："C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs"，然后启动 ZW3D。

***注意：****ZW3D 会自动加载".\apilibs" 下的应用程序。*

10. 运行该应用程序。

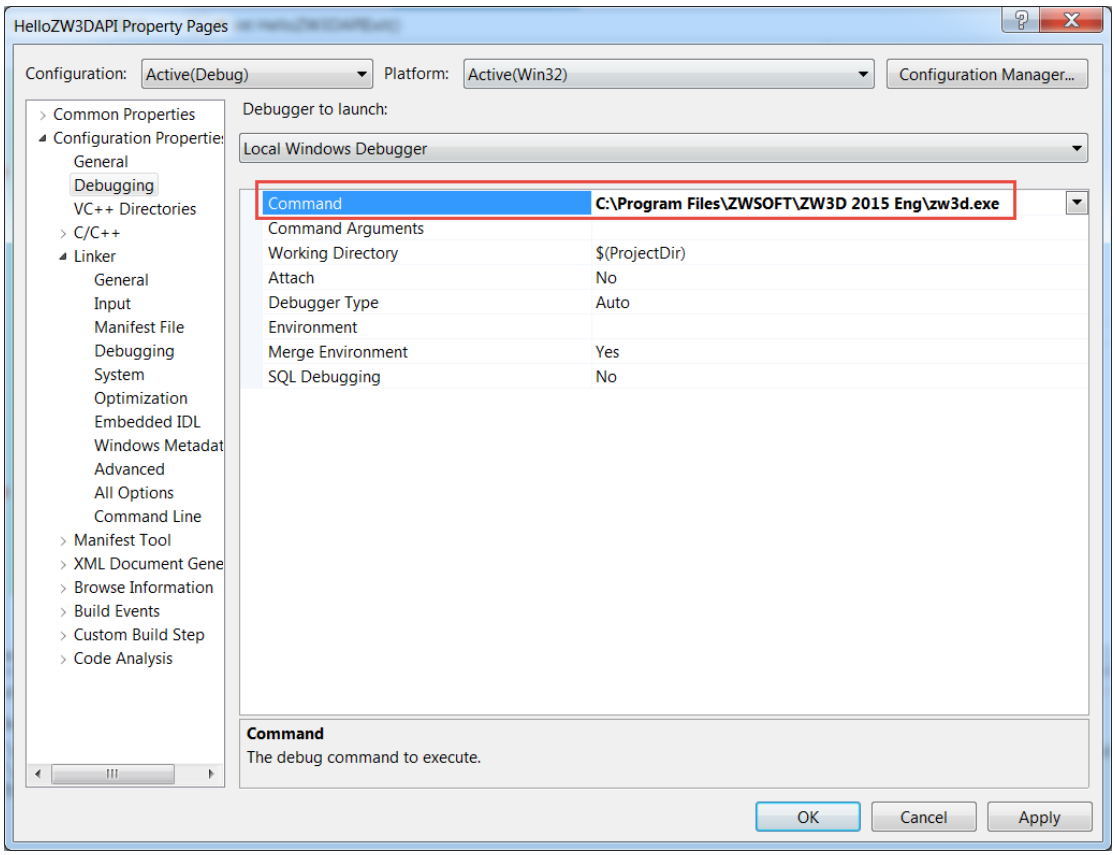
在命令行中输入"~HelloZW3DAPI"，按下 Enter。您会发现"Hello ZW3D API!"显示在***输出***对话框中。



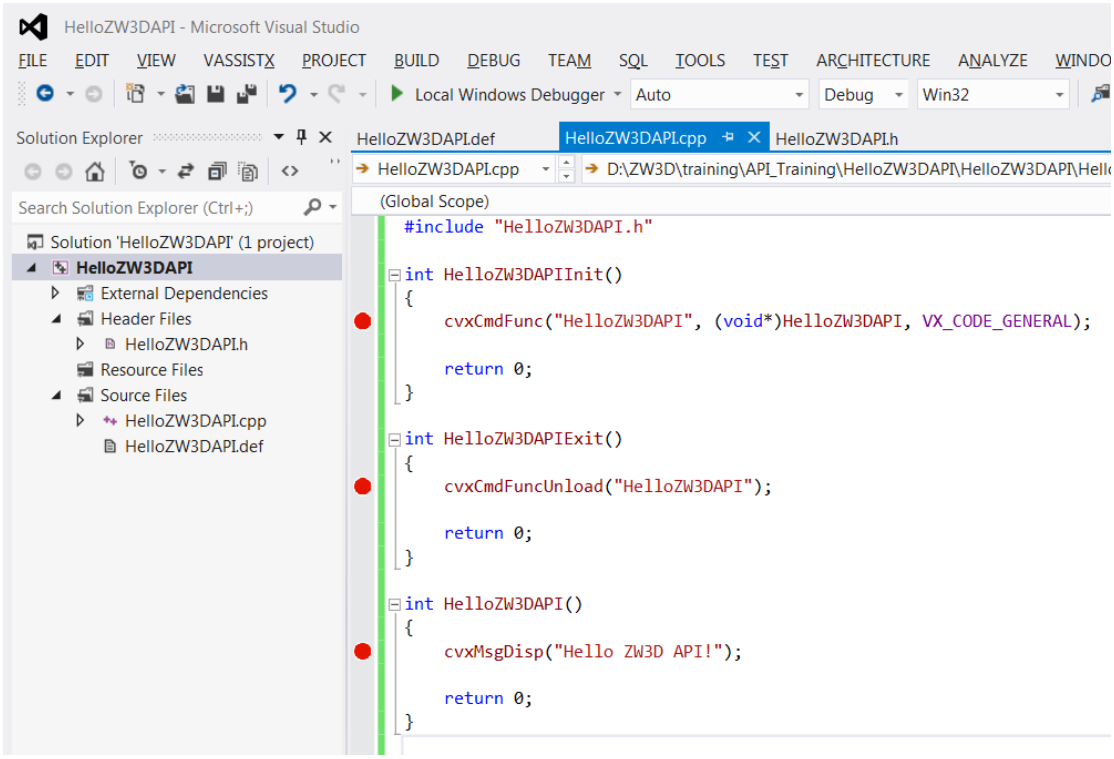

***注意：****函数名称可以定义为任何名称，不需要与项目相同。但将其名称定义与项目名称相同会是个好习惯。*

11. 调试应用程序。

a) 右键点击此项目，按如下设置调试命令。然后运行***本地 Windows 调试器***或按 F5 启动调试器。




b) 为每个函数设置断点。

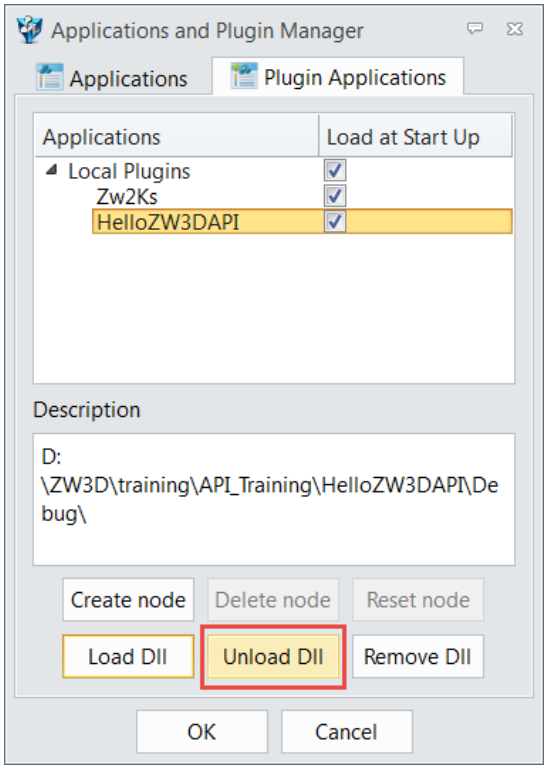



c) 参考步骤 9→ a)加载 DLL，您会发现有以下情况：

i. 当加载 DLL 时，调试器会进入 HelloZW3DAPIInit()。

ii. 当运行“~HelloZW3DAPI” (步骤 10)时，调试器进入 HelloZW3DAPI()。

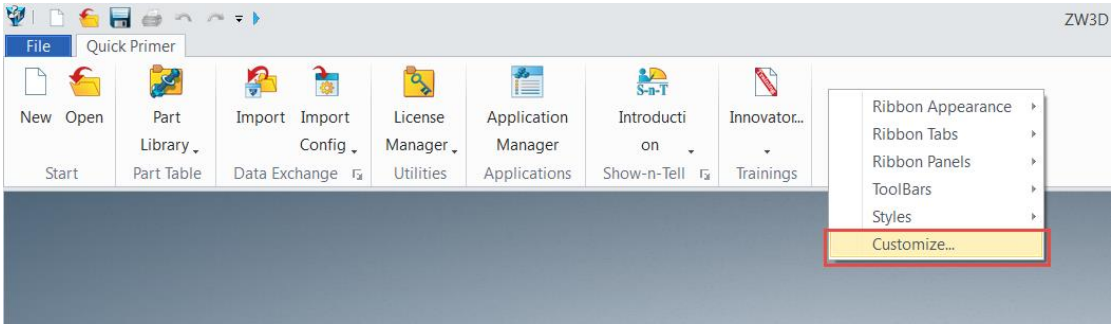
iii. 当卸载该应用程序时，调试器会进入 HelloZW3DAPIExit()。



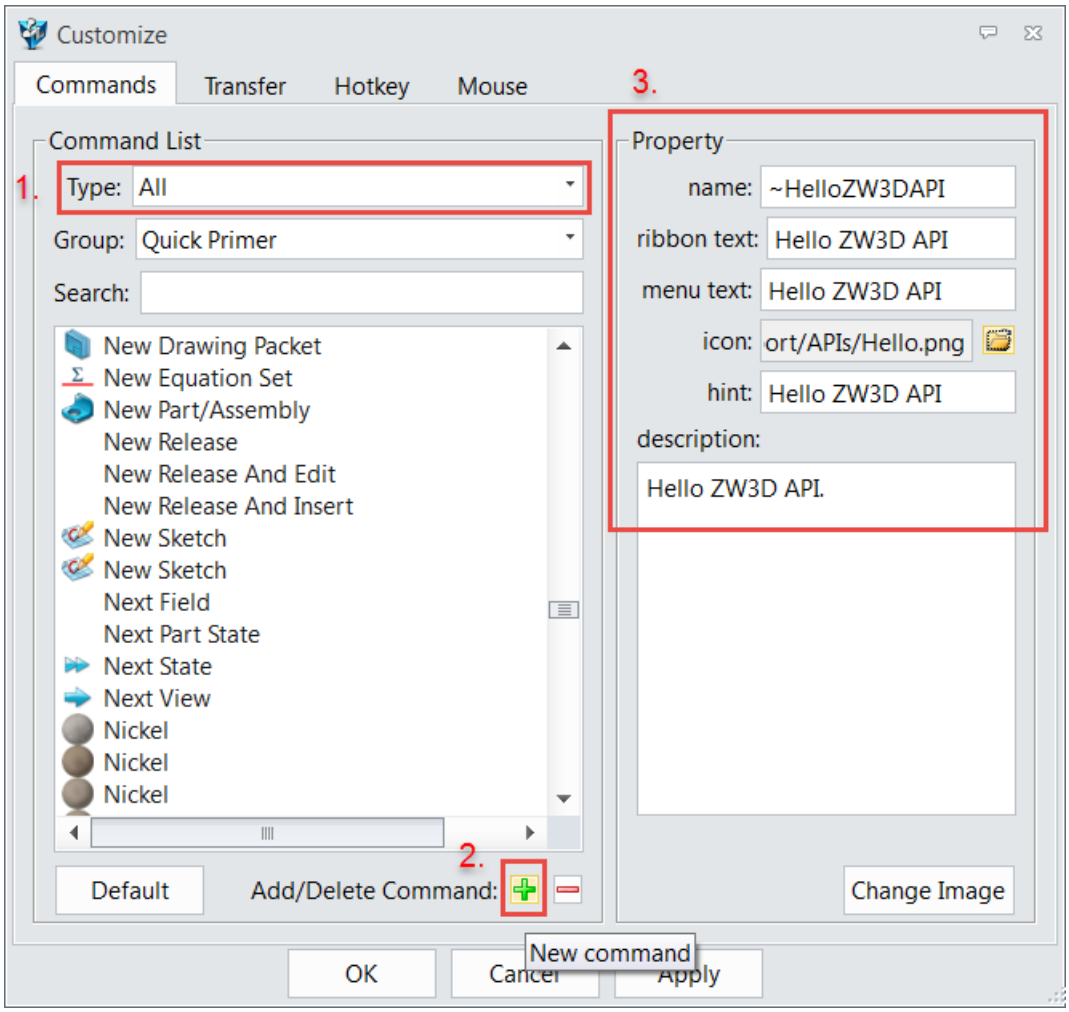
# 第二章自定义菜单/Ribbon 栏/工具栏

虽然我们在 ZW3D 中创建了第一个自定义命令，但输入命令是很复杂的。在这章我们会介绍如何将命令置于菜单/Ribbon 栏/工具栏。

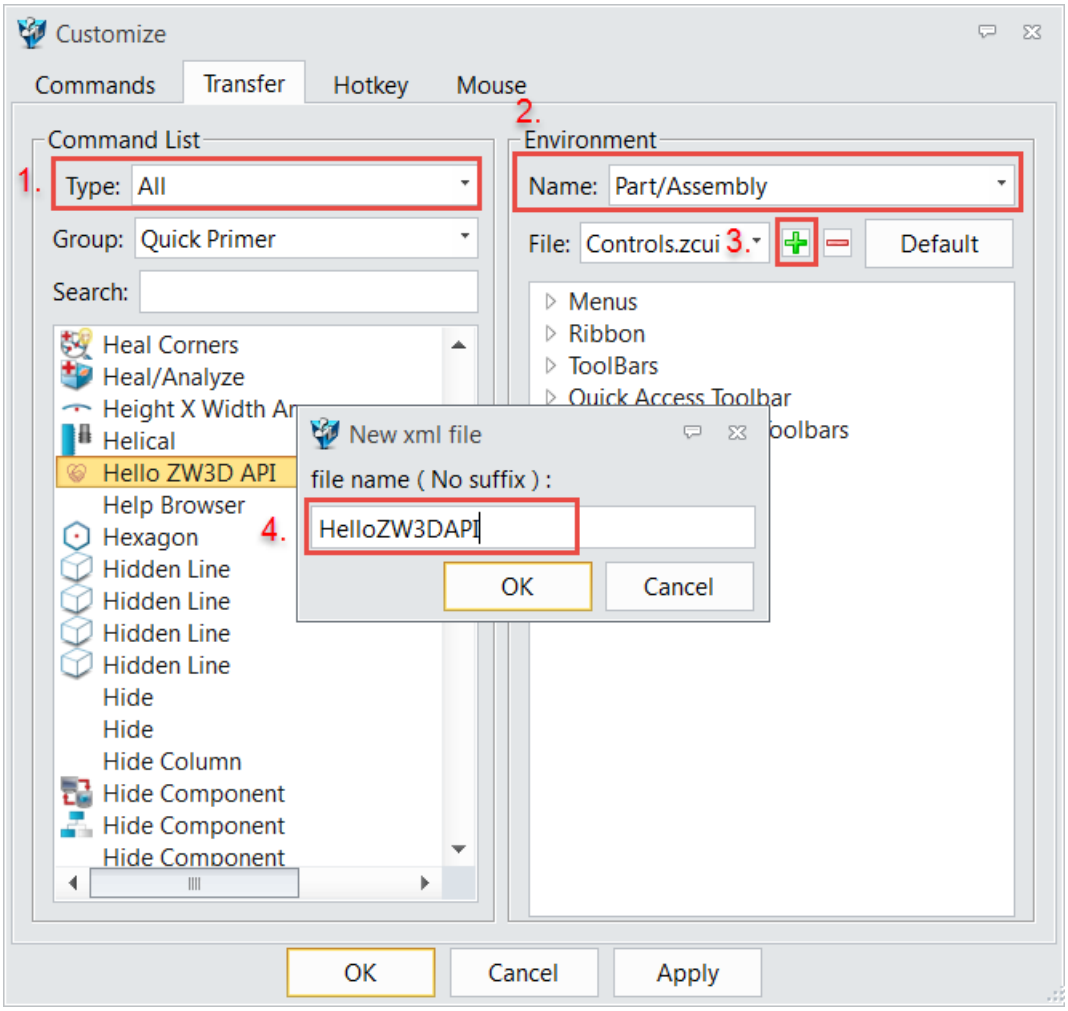
1. 打开 ZW3D，右键点击 Ribbon 栏空白区域，然后点击*自定义…*



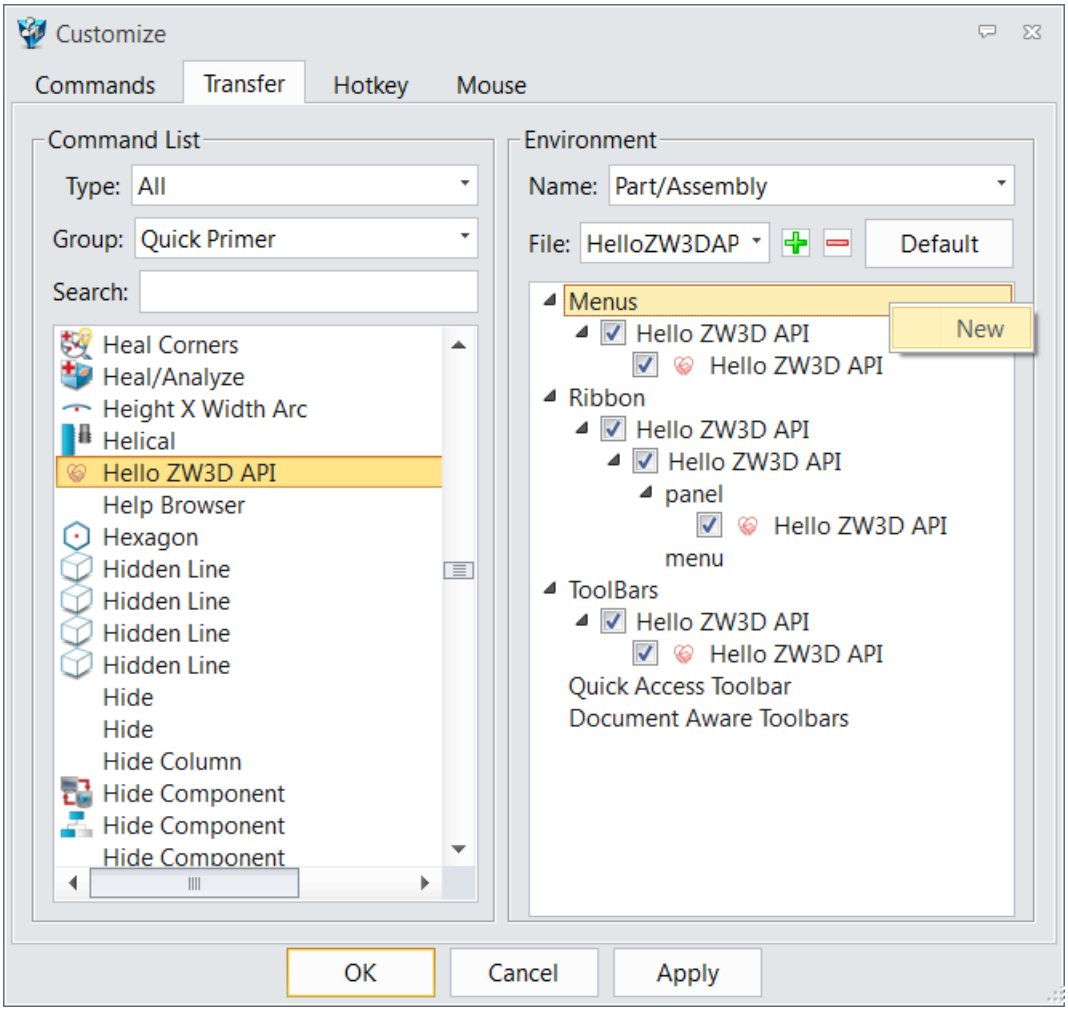

2. 定义命令。
   a) 将类型改成*所有*。
   b) 点击“+”按钮添加新命令。
   c) 修改新命令的属性。



3. 切换至 Transfer 属性页，创建您自己的菜单/Ribbon 栏/工具栏。
   a) 将类型改成***所有***。
   b) 将环境改成***零件/装配***。
   c) 点击“+”按钮，新建 xml 文件。
   d) 文件命名 **HelloZW3DAPI**。

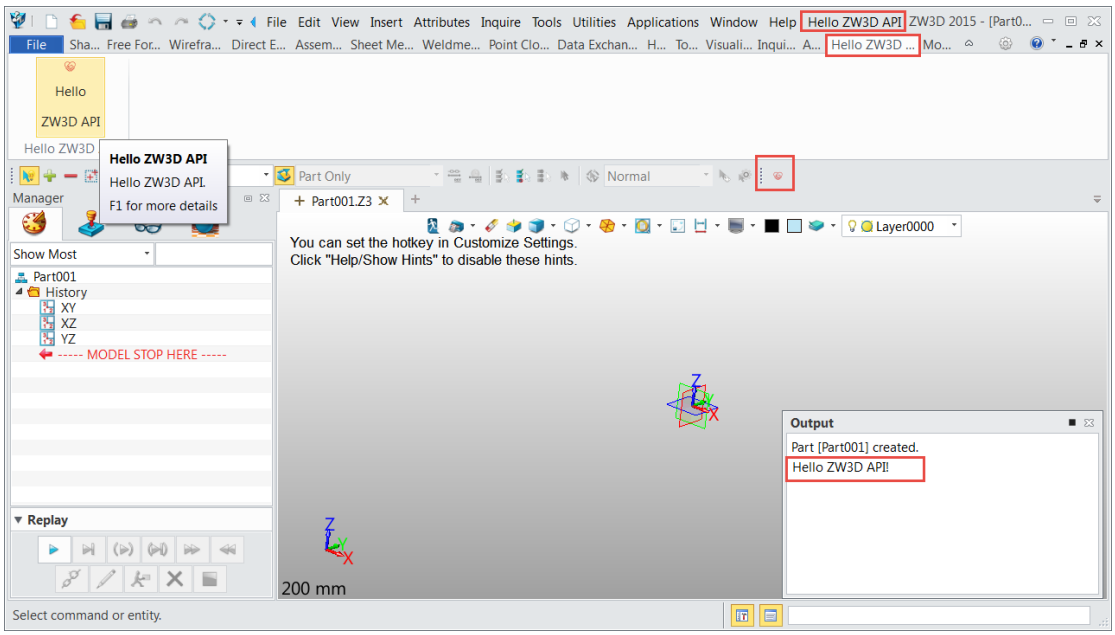



   e) 自定义新 xml 文件。
      i. 右键点击菜单/Ribbon 栏/工具栏，分别新建选项。
      ii. 将命令“Hello ZW3D API”从左边拖拽至您刚刚创建的选项中。
      iii. 请参考以下图片，创建相同的菜单/Ribbon 栏/工具栏。




f) 点击***确定***，完成自定义。

g) 新建一个零件，您就能找到这个菜单/Ribbon 栏/工具栏，按下按钮可运行此命令。



***注意：****如果您看到提示，"ALERT: Unable to find HelloZW3DAPI: No such symbol."这意味着在 ZW3D 里找不到您的命令，您需要先加载 DLL。详情可参考第一章步骤 9 加载 DLL。*

4. 与其他人共享自定义。

a) 从用户文件夹的路径下获取自定义的 UIXML 文件(HelloZW3DAPI.zcui)。

**版本 2020 之前的获取路径：**

%appdata%\ZW3D 2015Eng\profiles\Default\Environment-2\Controls

**版本 2020（含 2020 版）之后的获取路径：**

%appdata%\ZWSOFT\ZW3D\ZW3D 2022Eng\custom\profiles\Default\Environment-2\Controls

b) 从用户文件夹的路径下获取命令 XML 文件(User.zcui)。

**版本 2020 之前的获取路径：**

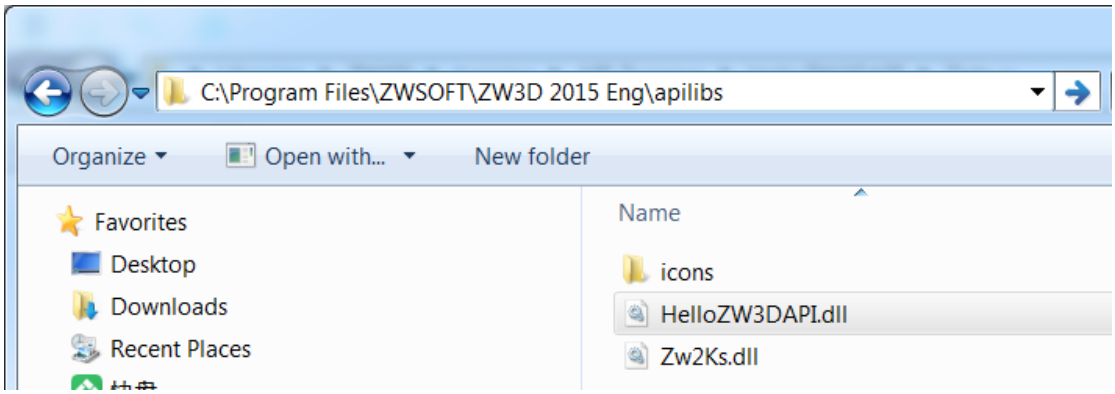
%appdata%\ZW3D 2015Eng\profiles\Default\Action

**版本 2020（含 2020）之后的获取路径：**

%appdata% \ZWSOFT\ZW3D \ZW3D 2022Eng\custom\profiles\Default\Action

c) 将这些文件复制至另一台电脑的相同目录下。

d) 复制 DLL 和图标至安装目录下，图标必须放在命名为 icons 的文件夹里。
C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs




***注意：****您可以重命名 XML 为任何其他的名称。但您必须将它们放在正确的目录中，ZW3D 会自动加载它们。*

# 第三章 ZW3D UI Designer 概述

1. 什么是 ZW3D UI Designer?

ZW3D UI Designer 是基于 QT 5.9.7 的一种 UI 设计工具。中望软件从 Digia 那购买了 QT 许可证。我们基于 QT 技术开发了我们自己的 UI 控件。ZW3D UI Designer 是基于 QT Designer 的一个插件，用户需要自行下载安装 QT，并将 ZW3D 提供的插件拷贝到 QT Designer 相关目录下，启动 QT Designer，即可看到 ZW3D 定义的 UI 控件。

***注意:** 您只能将 ZW3D UI Designer 用于 UI 设计。所有编码逻辑均不能使用 QT 技术，ZW3D UI Designer 不包含 QT 库。因此如果您想使用 QT 功能，您需要购买商业版本。*

(1) 具体操作如下：

a. 在安装包下的 plugins\\designer 找到以下 dll，拷贝到 QT 安装目录的 designer

CommonControlsPlugin.dll

QtnRibbonDsgn.dll

b. 在安装包找到以下 dll，拷贝到 QT 的 bin 目录下

CommonControls.dll

ResourceSystem.dll

logging.dll

QtnRibbon.dll

(2) 您也可以通过添加脚本的方式实现：

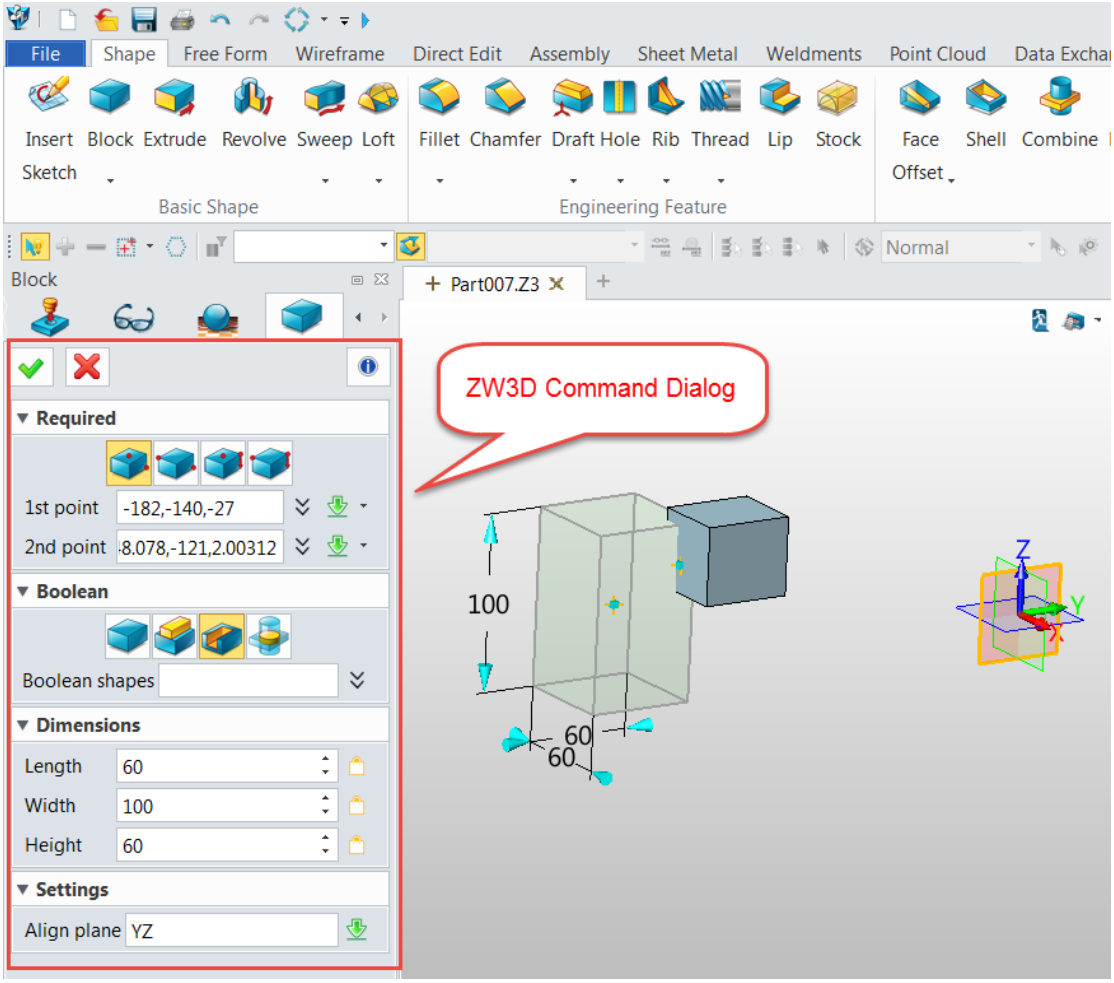
在安装路径的“api”文件夹中创建一个 CopyQTDll.bat 文件，复制以下内容，**注意修改 QTPath**

```
%@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. 什么是 ZW3D 命令对话框？

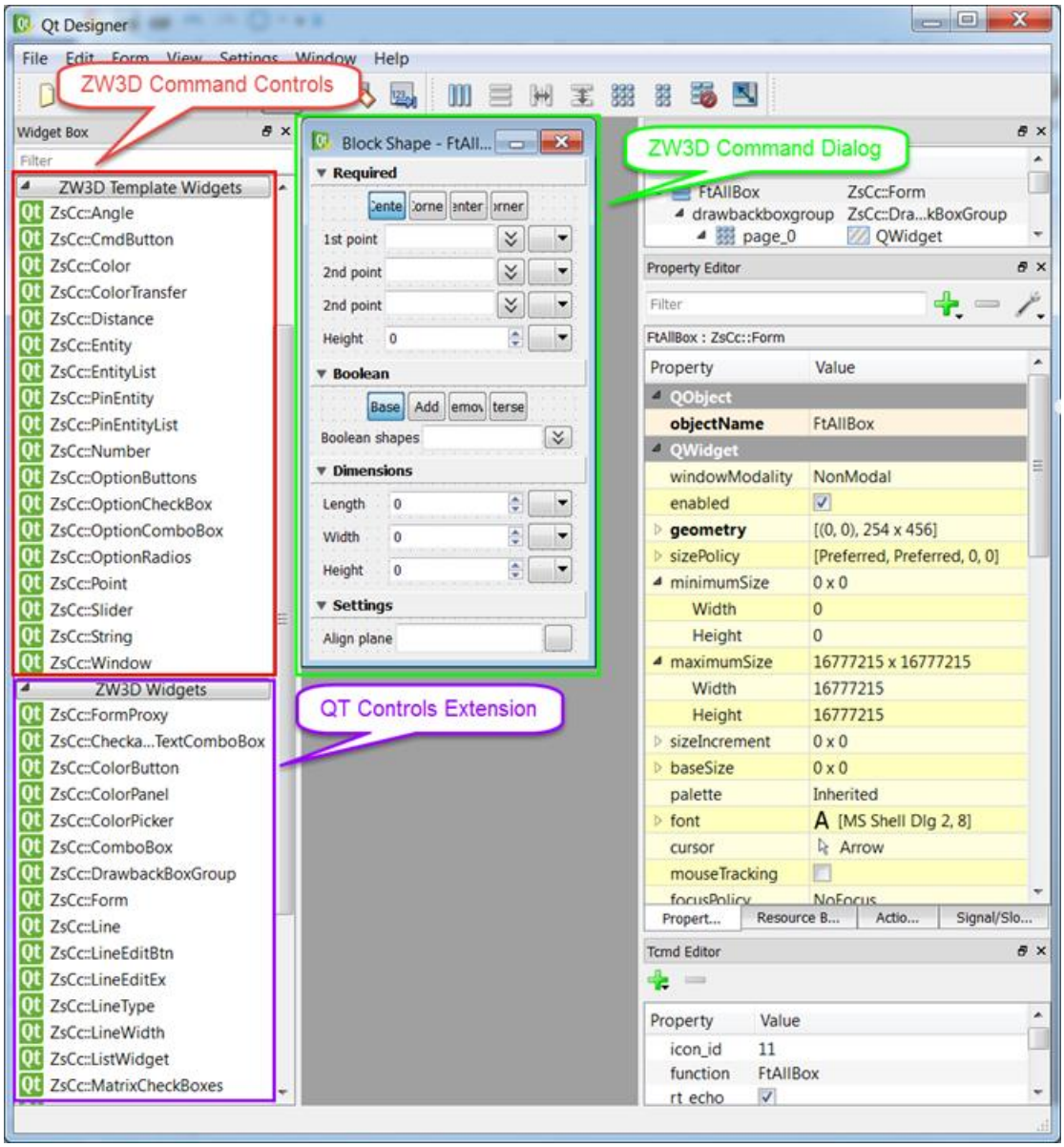
ZW3D 命令对话框是用户在运行 ZW3D 命令时进行交互的特殊对话框。ZW3D UI Designer 支持特殊控件从 ZW3D 建模区获取必要的值，或获取/设置只能在 ZW3D 使用的某些特殊值。



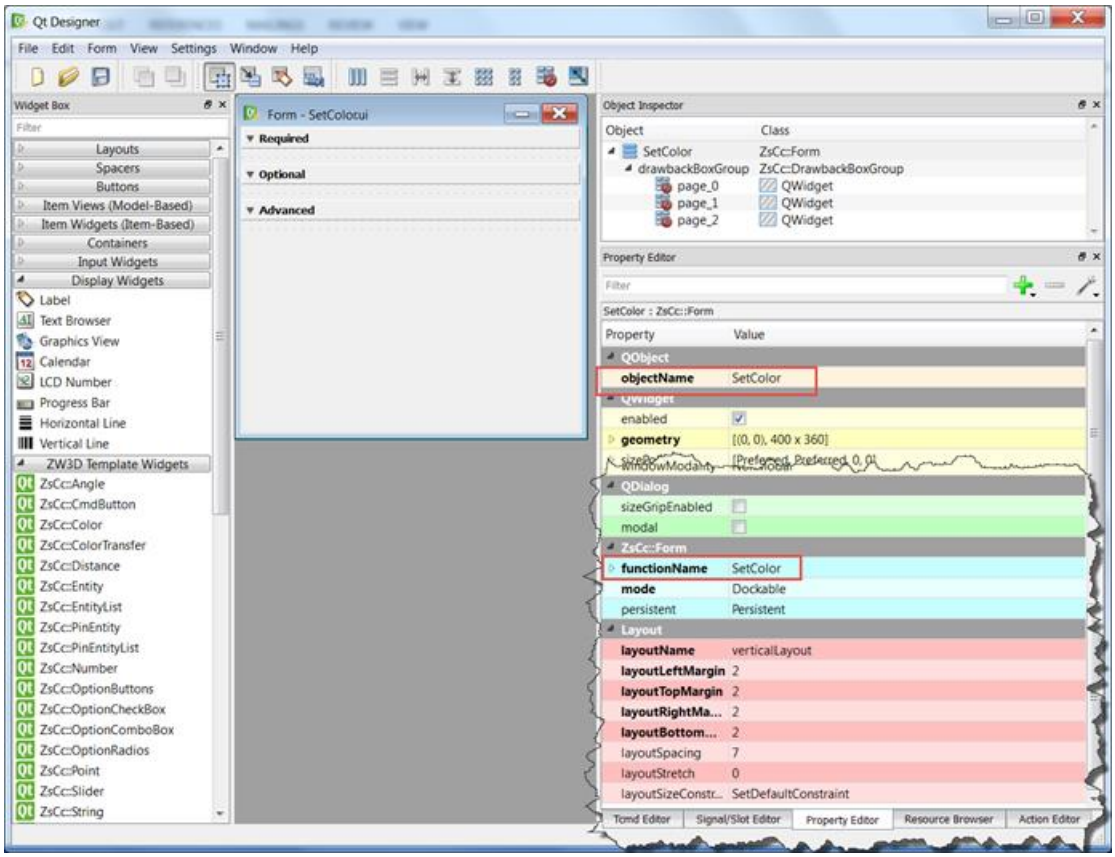

3. ZW3D UI Designer 简介。

由下图可知，您可以发现左边有两个扩展控件。红色标示框内的控件是 ZW3D 命令控件，用于 ZW3D 命令对话框的设计。

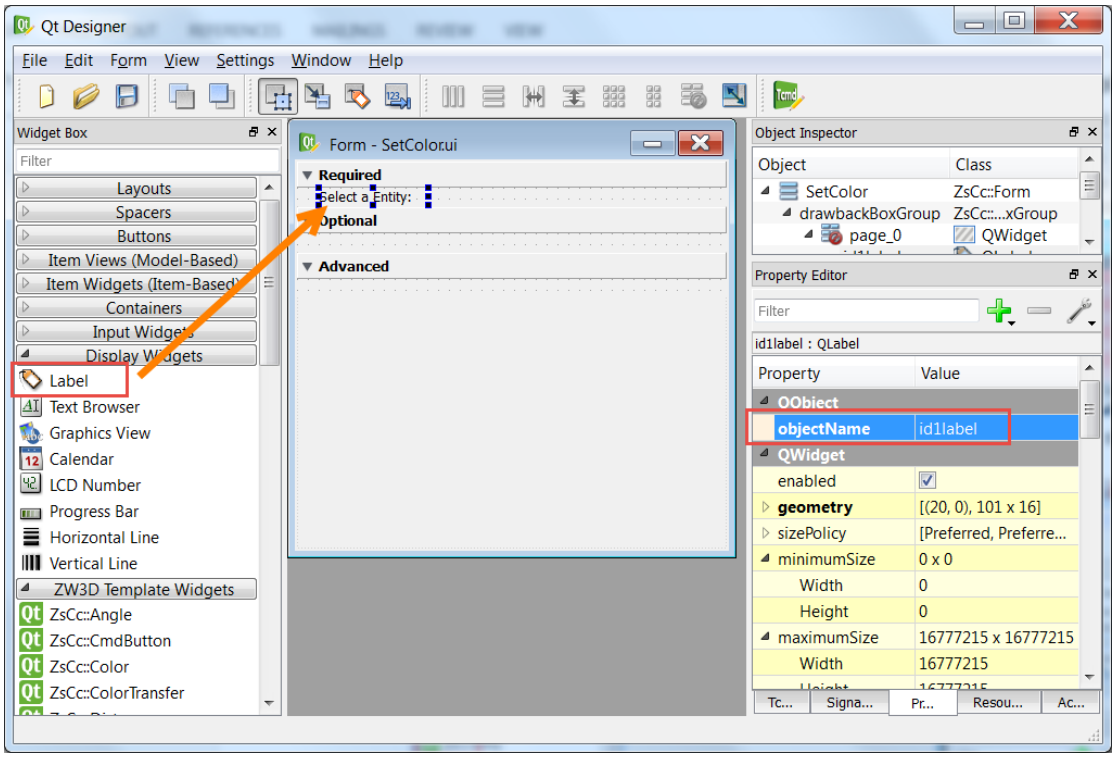
紫色标示框内的控件为 QT 的扩展控件，用于 QT 对话框设计。如无特殊要求，我们建议您不要使用 QT 控件。



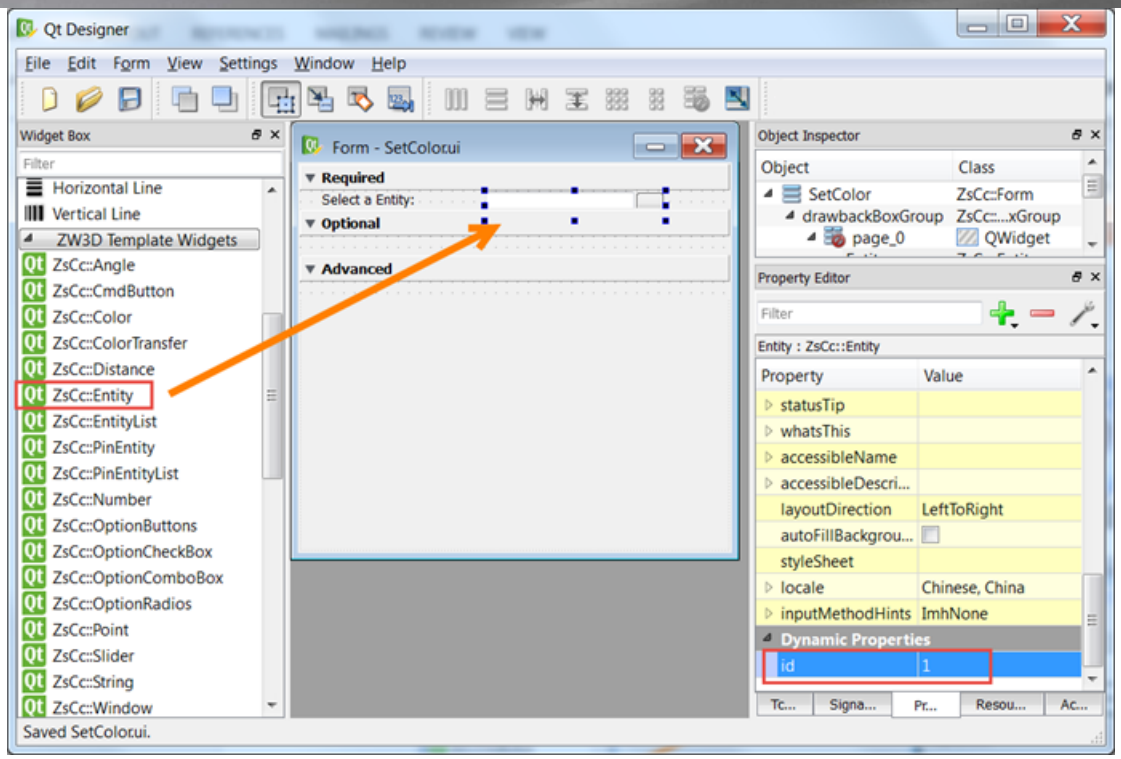

4. 如何创建 ZW3D 命令对话框？
   a) 打开 ZW3D UI Designer，新建基于 ZsCc::Form 的窗口。您可以根据下图得到基本的窗口，然后将 objectName 和 functionName 设置为 **SetColor**。



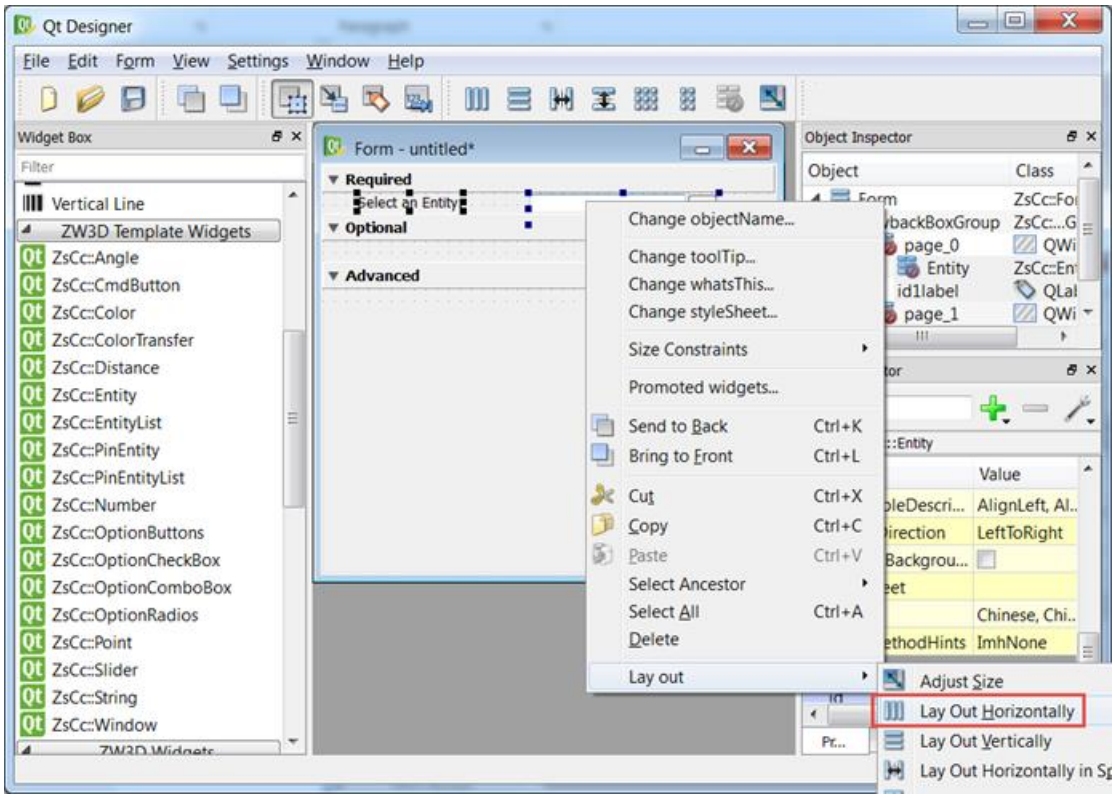

b) 将 Label 拖拽至 Required 区域，将文本改成 ***Select an Entity:***并将 **object Name** 设置为 **id1label**。



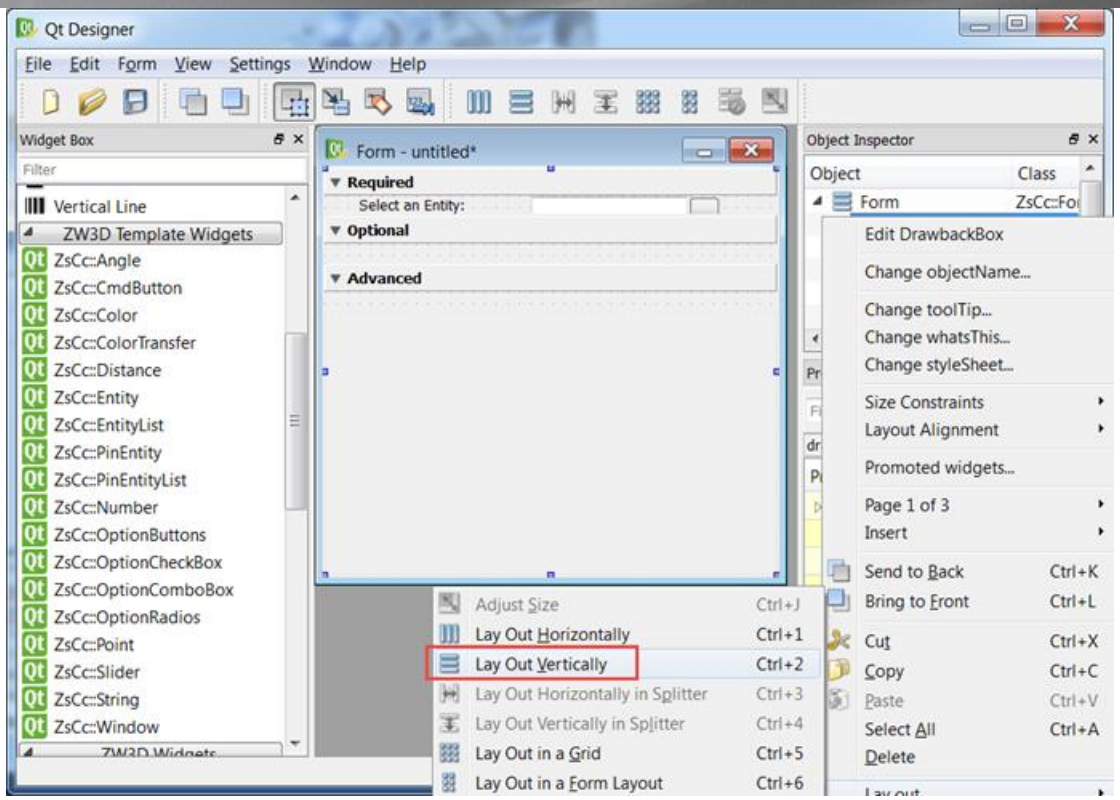

c) 将 Entity 控件拖拽至 Required 区域。设置 ID 为 1，ID 可以为任何数字，但必须与 Label 对象的数字保持一致，表示这两个控件是同一个群组。



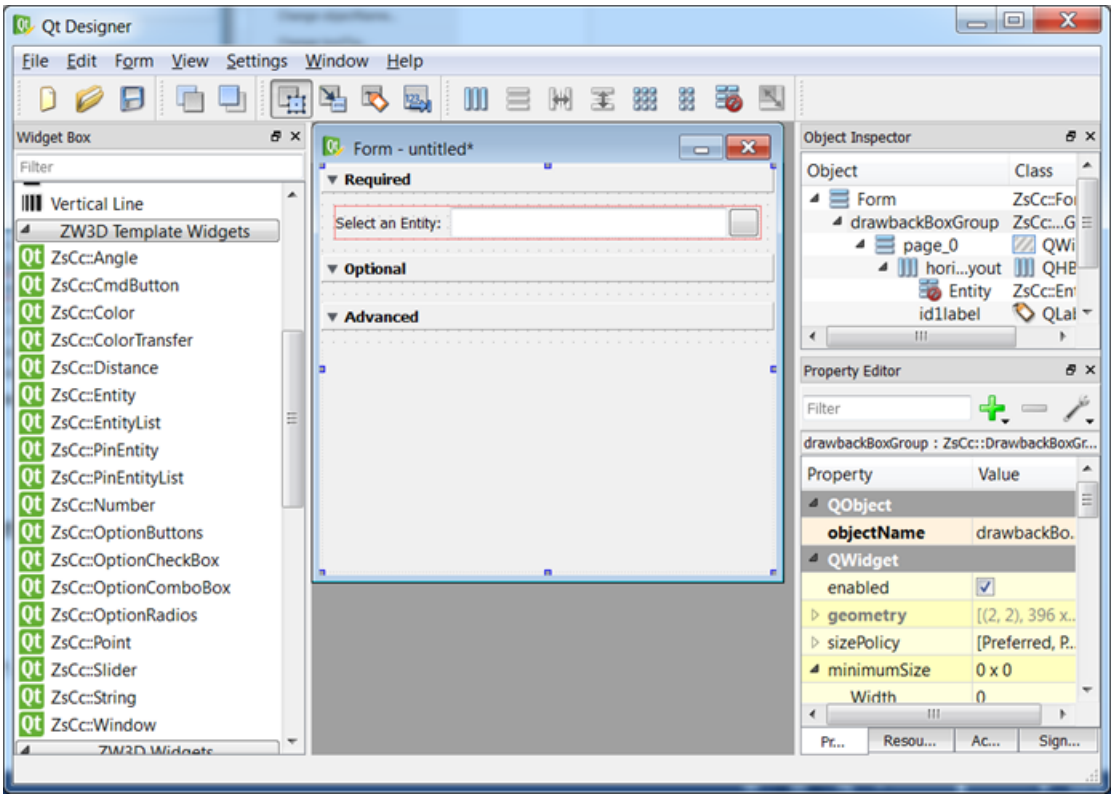

d) 使用 Qt 功能来布局这两个控件。同时选中这两个控件并右键选择 **Lay Out Horizontally**。




e) 右键点击 drawbackBoxGroup，选择 **Lay Out Vertically**。



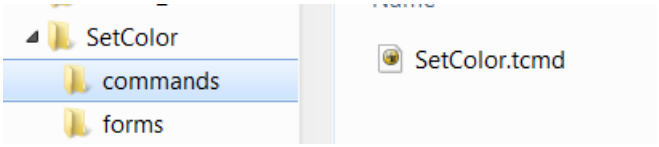

f) 您会得到如下的命令对话框：




g) 保存这个对话框至 **SetColor** 文件夹，命名为 **SetColor.ui**。对应的 **SetColor**.tcmd 需要您手动编辑，可参照 ZW3D 提供的平台模板文件，tcmd 中关键属性介绍详见第五章。

h) 将 **SetColor.ui** 放在 **forms** 文件夹里，将 **SetColor.tcmd** 放在 **commands** 文件夹。您

可以参考下图：

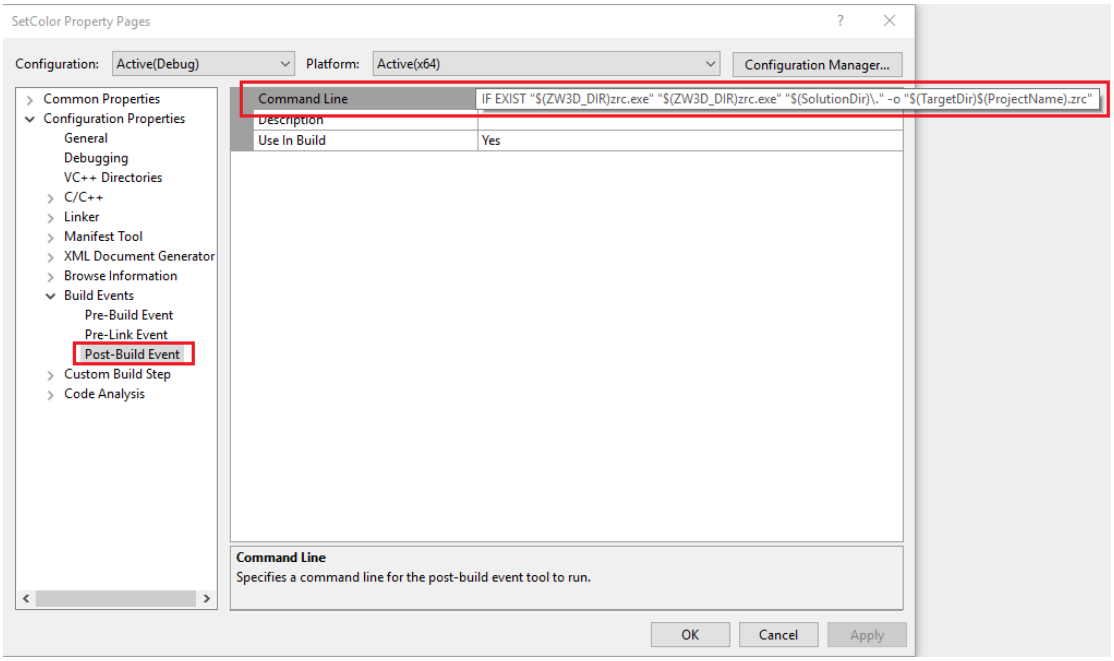



i) 要编译资源，您的项目需要添加命令行至：***生成事件→后期生成事件→命令行。***

**命令行内容如下所示：**

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

**（注意：ZW3D_DIR 是一个环境变量，其值是 ZW3D 的安装路径）**




# 第四章使用 ZW3D 命令对话框

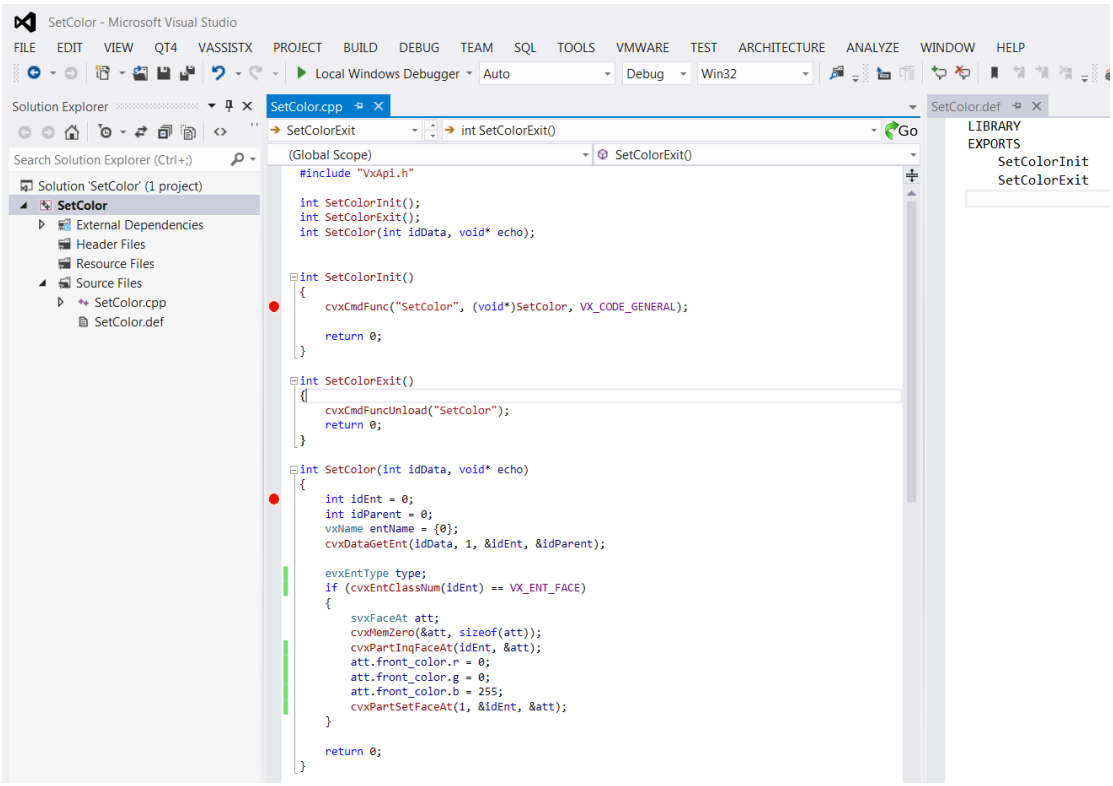
1. 参考第一章，创建一个空白项目，命名为 **SetColor**。
2. 在项目中添加一个 SetColor.cpp，然后输入以下代码：

```
#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;
}
```

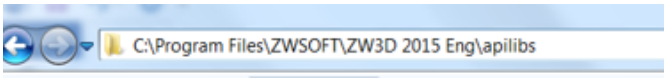
3. 添加模块定义文件 SetColor.def.复制如下代码：

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

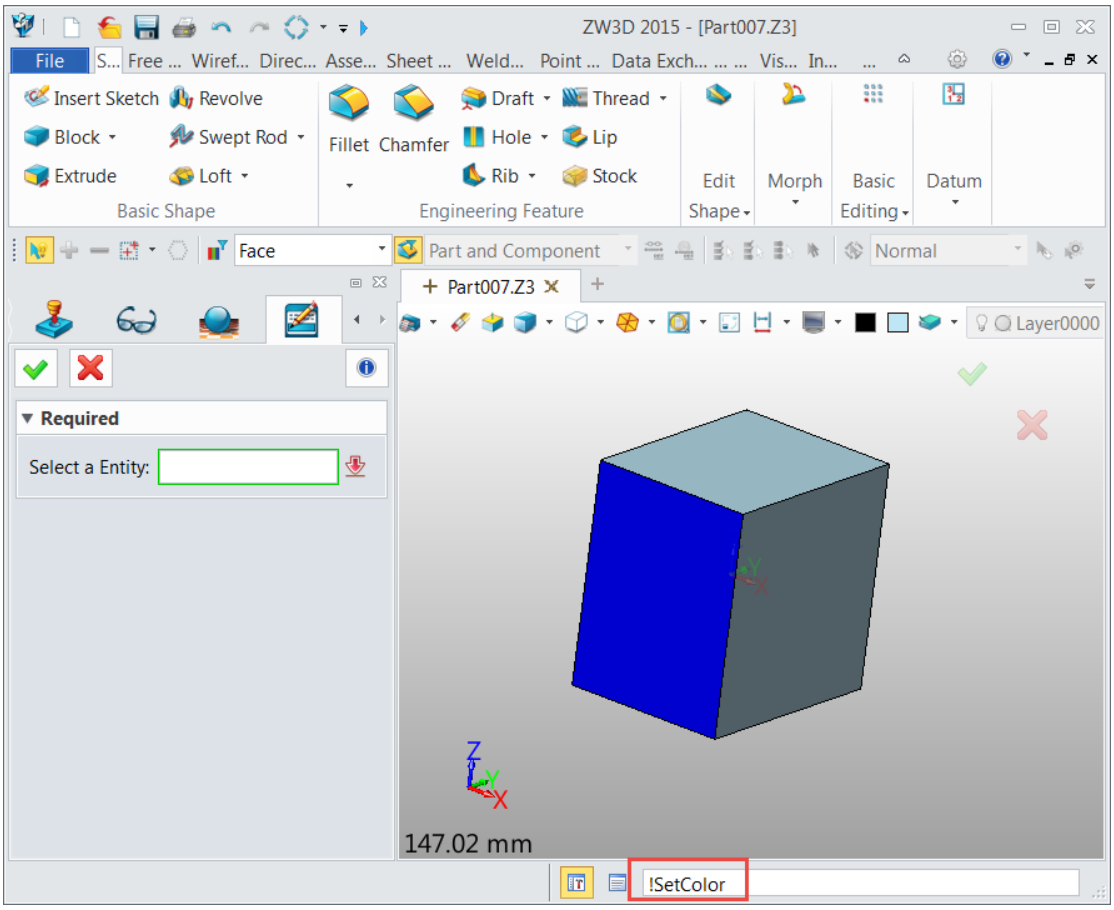



4. 参考第一章，设置项目配置，生成并得到 **SetColor.dll** 和 **SetColor.zrc**。现在，我们有 dll 和 zrc 文件，复制所有文件至 ZW3D 安装路径。

   ***注意：****复制文件前，您必须关闭 ZW3D。*

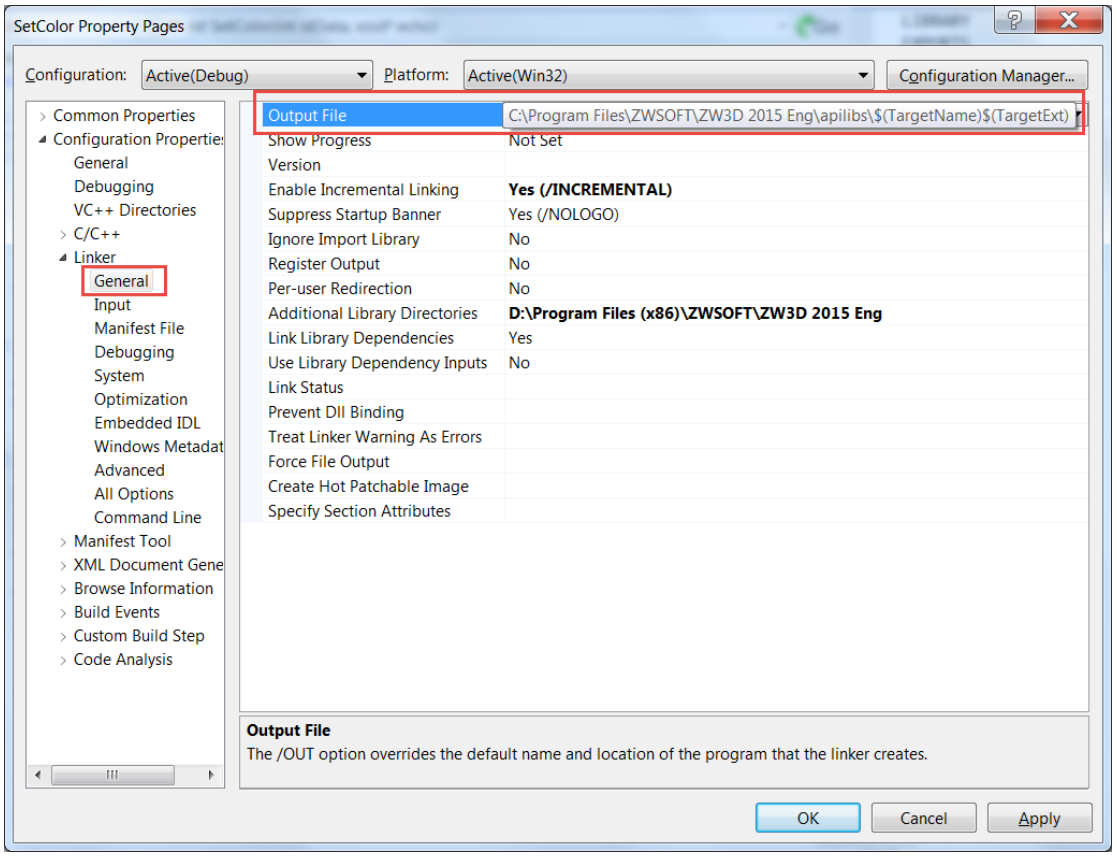



5. 启动 ZW3D，新建一个零件并画出一个六面体。在命令行中输入***!SetColor***。您会看到下图中的命令对话框。选择一个面，点击确定，您会发现曲面的颜色变成蓝色。




6. 调试项目。将输出文件设置于：
“C:\Program Files\ZWSOFT\ZW3D 2015 Eng\apilibs\$(TargetName)$(TargetExt)”。
您可以参考下图，然后重新生成项目。您可以设置断点以及调试该项目。

***注意：****ZW3D 会自动加载“apilibs”目录中所有的 DLL。*


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

# 第五章如何调用 ZW3D 函数

1. 获取 ZW3D 命令对话框。
   从以下链接下载所有 ZW3D 的命令对话框：
   https://dl.zwsoft.com/zw3d/Products/ZW3D_API/2023/zw3d-command-dialog_2023.rar
2. Tcmd 关键属性

(1) 模板命令符号定义

以 Extrude 命令为例，模板命令相关符号定义如下：

```
<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>
```

常用符号介绍如下

a. "template name" 即命令名

b. "function" 即命令的执行函数，通常与命令同名，函数定义：

```
int fun(int idData, int *idOut);
```
或
```
int fun(int idData);
```

**备注：其中 idData 为命令对应的参数容器，使用 cvxDataSet 向对应字段存储数据，cvxDataGet 获取对应字段数据。**

c. "echo_obj" 即命令的预览函数（可选），函数定义：

```
Int funEo(int idData);
```

d. "init" 命令初始化函数，可向参数容器初始化数据，init 调用时 form 还未构建，不可调用 ui 相关的接口获取界面数据。函数定义：

```
void fun_init(int idData);
```

e. "init_after" 命令初始化函数，此时 form 已构建，可初始化 ui 的一些状态。函数定义如下：

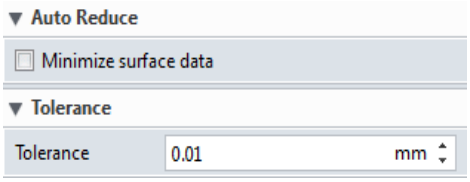
```
void fun_initAft(int idData);
```

f. "term" 命令退出函数，正常执行不会被调用。若 term 里面涉及如资源释放等，则可能需要显式调用 term 函数。函数定义如下：

```
void term();
```

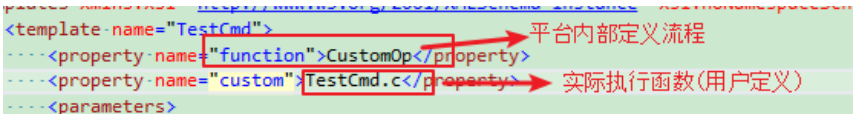
g. "multi_cmd" 多命令模式：在同一个命令里根据某个 option 类型 field 的值控制不同的字段对应控件的显示状态。例如 Block 命令，切换 8 这个字段的类型，界面上的其他控件的状态会相应改变。

h. "show_tol" 控制特征命令界面是否显示 Tolerance 页面，如下：



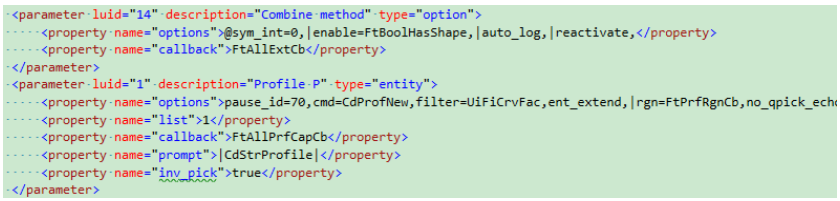

i. "esc_dlg" 当命令执行慢时是否自动弹出结束命令的弹框。

j. "custom"自定义特征命令，在用户自定义特征。如下：



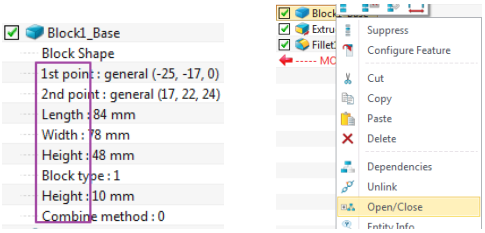

(2) 字段定义

字段参数定义如下：




a. **Parameters 自身属性：**

- **trigger**-命令执行标志，该标志之前的为必选输入，后面的为可选输入；同时，可以使得通过鼠标中键来结束命令。
- **description**- field 显示的字符，目前主要用于对特征节点进行 open 操作后显示的参数名称。



- **checker**-用户检查该 field 的值是否为空，若为空结束命令，用于 set-list。
- **type**-null, entity, point, option, number/distance/angle, form, command, continue, string.

b. **Option 通用的属性设置如下：**

- 允许输入为空：empty_ok
- field 可用条件：|enable=fun。与 callback 相比，enable 函调用的次数会频繁一点，其他控件参数变化都会触发调用。

  enable 函数定义如下：

  int funEn();

  备注：return 1，该字段可用，否则返回 0.
- no_qpick_echo：选择实体时不执行 echo 函数

c. **callback-回调函数，定义如下：**

- 回调函数：<property name="callback">funCb</property>

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

d. **next-该控件输入完毕之后跳转的下一个字段，属性定义如下：**

- 自动跳转：<property name="next">10</property>

表示该控件输入完成之后将自动跳转到 **id=10** 的控件。

(3) **entity**

获取实体输入，一般对应实体或实体列表控件(Entity/EntityList/EntityTable)

**实体控件常用的属性定义如下：**

a. **options**-控制控件行为的综合选项

常用行为包括：

- 过滤器：/shell/face/curve/edge/等，custom_filter=fun
  a. /face/curve/edge/为实体类型，以‘/’隔开，如果可以选择组件内的实体，可改为/Eface/Edge/・・・。
  b. custom_filter 为用户自定义过滤器设置，可设置一些条件限制选择，函数定义如下：

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

备注：return 1，idx_ent 可以被选择，否则返回 0.

c. 常用的实体过滤器如下表所示，用‘/’隔开：

| 实体对象 | 过滤器 |
|---|---|
| 造型 | shell |
| 面 | face |
| 边 | edge |
| 线 | curve |
| 点 | point |
| 曲线列表 | clist |
| 草图 | sketch |
| 基准面 | datum |
| 轴 | axis |
| 坐标系 | csys |
| 特征 | ftr |
| 组件 | comp |
| 文本 | text |
| 标注 | dim |

- 选择对象的检查与限制：chk_line，chk_plane（可与其他过滤器一起使用）
  a. chk_line:检查是否是直线
  b. chk_plane:检查是否是平面

b. list-是否是多选，1 表示是多选，单选可不设置，属性定义如下：

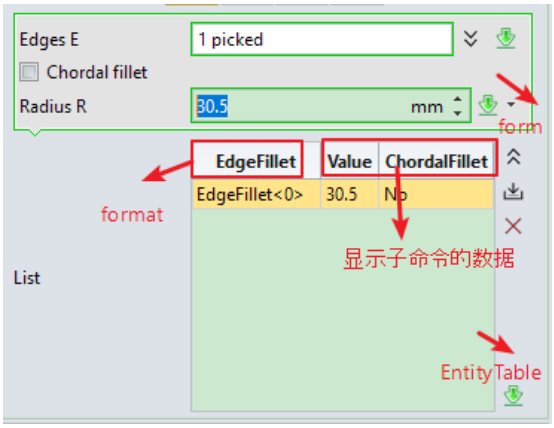
```
<property name="list">1</property>
```

c. comp--一般用于 setlist 的关联 form，与子模板命令的 form 字段成对使用，存放自模板命令的数据，属性定义如下：

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

a. SetList 定义子命令的方式如下图所示：


b. UI 界面显示如下：




(4) point

获取点输入，一般对应点控件(Point)

a. options-控制控件行为的综合选项

- 过滤器：/face/curve/edge/等，custom_filter=fun，与实体控件类似；或/direction/ 表示方向。
- ddd_drag：可动态拖拽点，控件作为点时可用，方向不可用
- on_ent：点必选在实体上
- hi_ent：被选中的实体高亮
- get_dir:表示方向时，会自动获取方向的值
- rev_dir:表示方向时，可通过鼠标点击方向箭头进行反转方向

(5) 2.3 option

获取用户选择的选项，一般对应 option 控件(OptionButtons/ OptionCheckBox/ OptionComboBox/ OptionRadios)

a. options-控制控件行为的综合选项

- 自动初始化数据：|auto_log（option 选项）
- 设置默认值：@sym_int=0

(6) number

获取数值输入，无单位，一般对应数值控件(Number)

a. options-控制控件行为的综合选项

- 设置默认值：@sym_dbl=0.0(会记录最后一次设置的值)，或者 val=0.0
- 设置最小值：min=-10000.0
- 设置最大值：max=10000.0
- 设置增量：inc=15.0,

(7) distance

获取数值输入，有单位，一般会生成标注，对应距离控件(Distance)

a. options-控制控件行为的综合选项

- 基本数值设置与 number 一样
- ddd 设置：property name="dim"，详见《动态拖拽标注 DDD 开发指引》

(8) angel

获取数值角度输入，一般对应角度控件(Angle)

a. options-控制控件行为的综合选项

- 基本数值设置与 number 一样
- ddd 设置：property name="dim"，详见《动态拖拽标注 DDD 开发指引》

(9) form

显示一个指定的 GUI form，对应(FormProxy)，用于 setlist 的情况，嵌套一个子模板命令，或者显示一个 GUI form。

对应于 setlist 的设置如下:

a. options-控制控件行为的综合选项

- 允许输入为空：empty_ok

b. comp--一般用于 setlist 的关联 entiy，详见实体控件中的 comp

- <property name="comp">2</property>，2 为实体控件的 id

c. Template-子模板设置

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

(10) continue

等待，一般用于中键结束命令，在必须项和非必须项中间设置。

a. options-控制控件行为的综合选项

- 终止命令：~CdSkipEnd,

b. prompt-提示

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

(11) string

获取字符串输入，一般对应字符控件(String)

(12) 字段值获取和设置

a. 设置

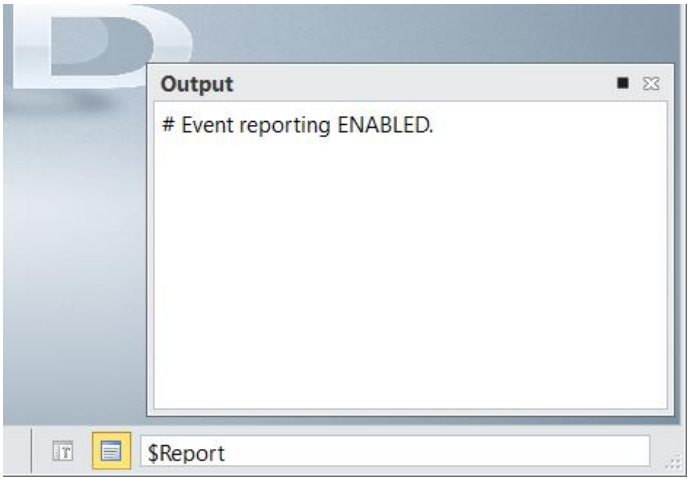
cvxDataSet 用于设置指定字段的数据

b. 获取

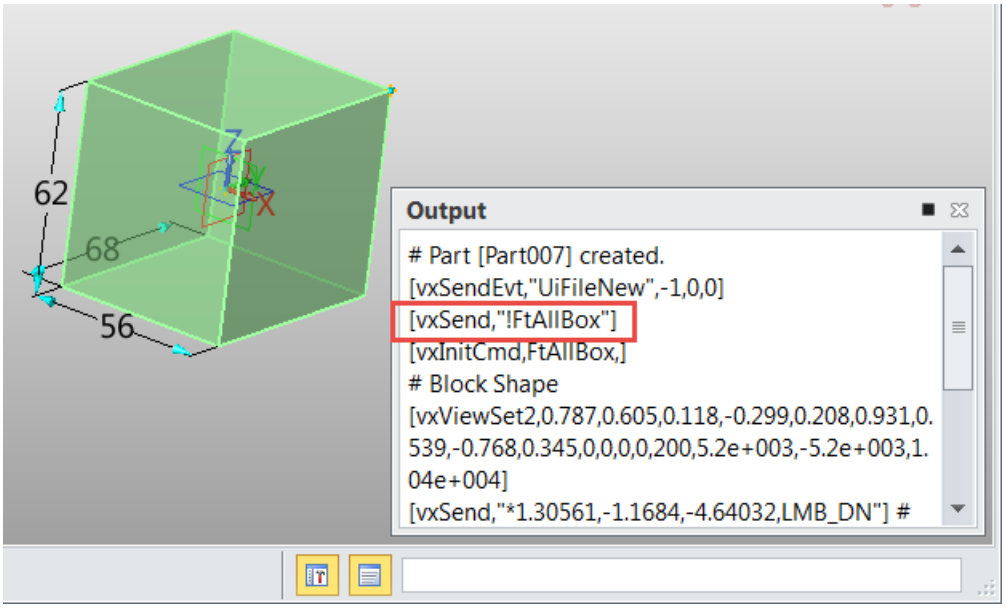
(13) cvxDataGet 用户获取指定字段的数据

3. 参考 ZW3D 的命令对话框。

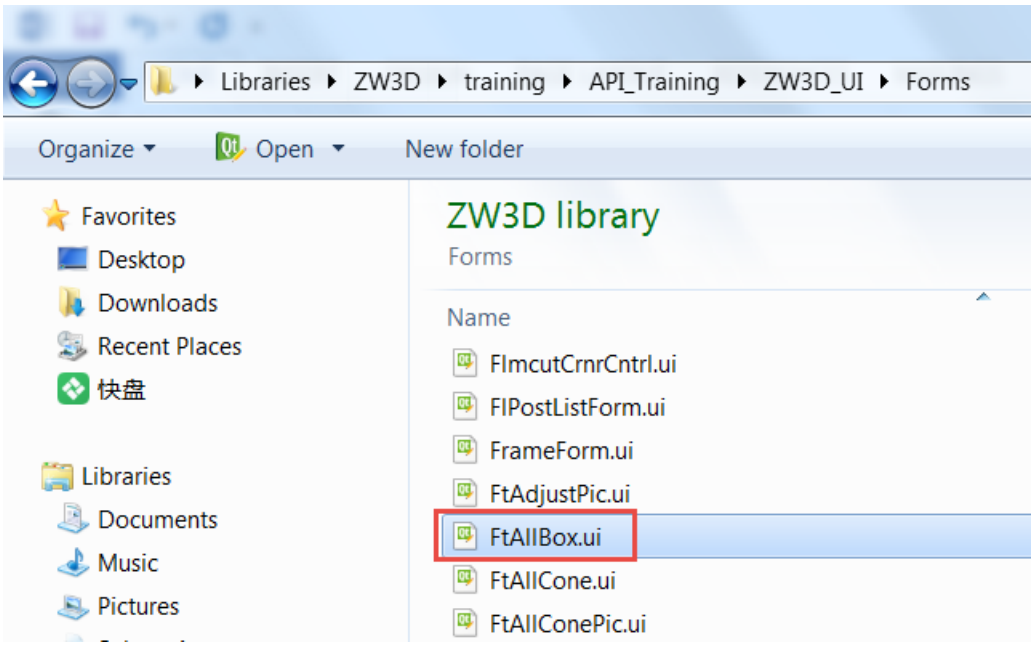
a) 在 ZW3D 命令行里运行***$Report***。
命令运行成功后，您会得到以下信息。



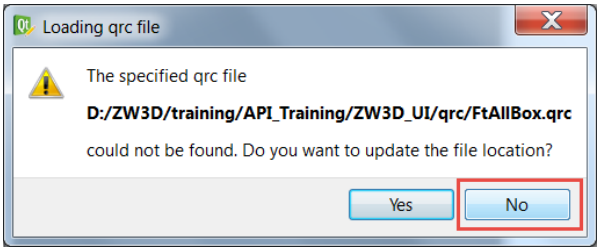

b) 运行您想要知道的命令，比如新建六面体的函数。
您会得到如下信息，“[vxSend,”!FtAllBox]”，表示 ZW3D 运行命令“!FtAllBox”创建六面体。



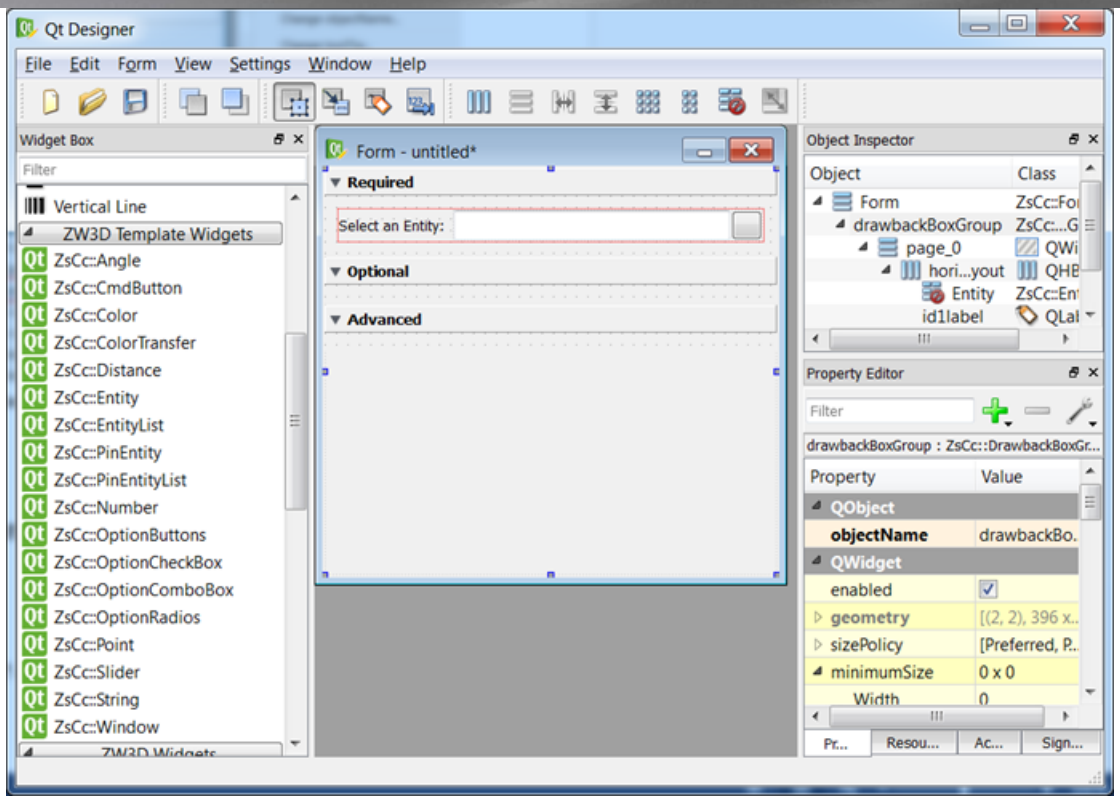

c) 在 ZW3D 命令对话框文件夹中，您可以找到命名为“FtAllBox.ui”的命令对话框。



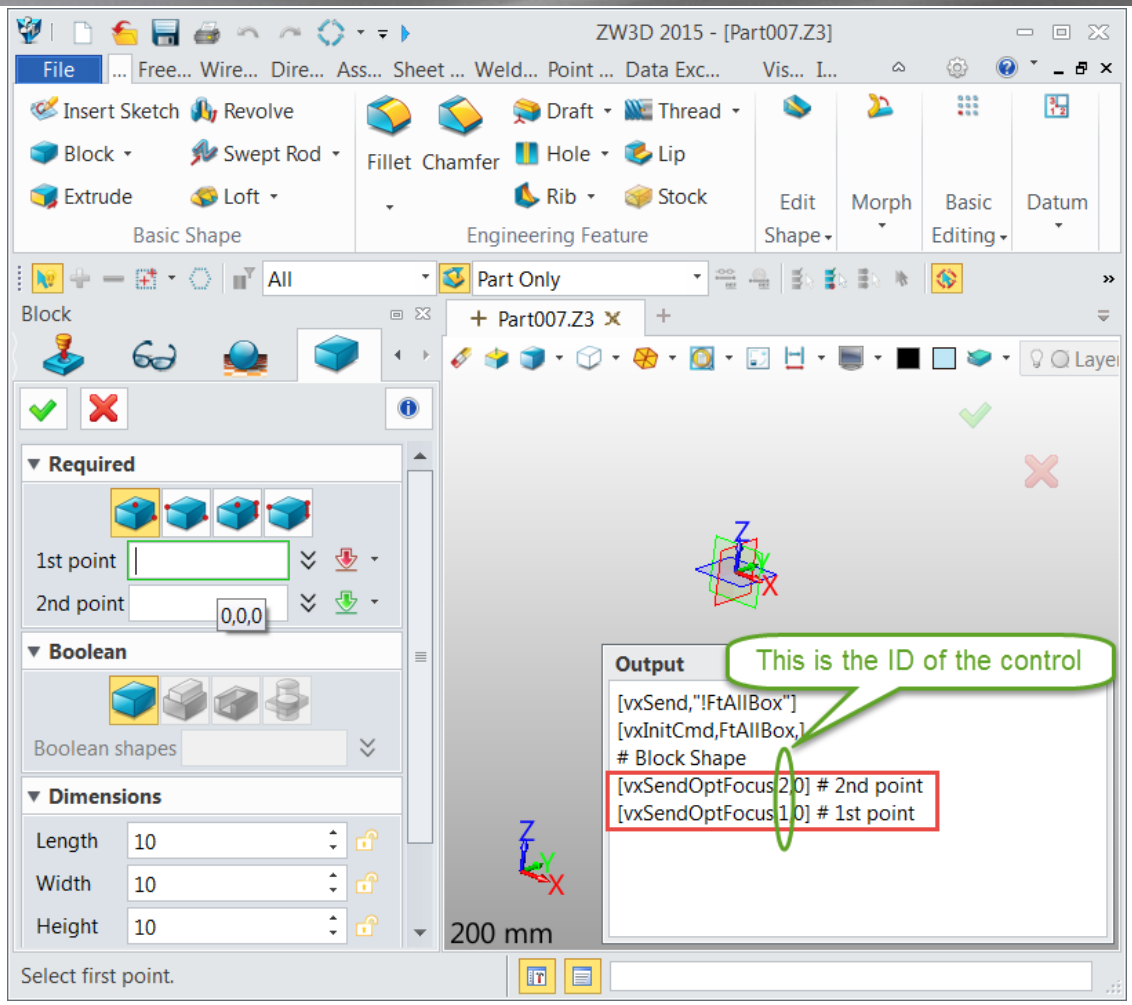

d) 使用 ZW3D UI Designer，打开这个对话框，您会得到一个提示信息。点击 **No**，忽略该信息。




e) 您将知道所有控件的 ID。




f) 您会看到有两个“2nd point”控件。至于它们的区别，参考步骤 b)了解此功能。在命令对话框的不同控件上切换鼠标焦点。您会发现将鼠标焦点设置到不同控件上输出消息区将显示不同的信息，从这些信息中，您可以找到控件的 ID。




4. 现在，您已经知道函数名称和控件 ID。下面我们使用 ZW3D 的功能在创建一个六面体，当然，ZW3D 也支持使用 API 函数[cvxPartBox()]创建六面体。

   a) 参考第一章，创建新项目，命名为“myBox”。

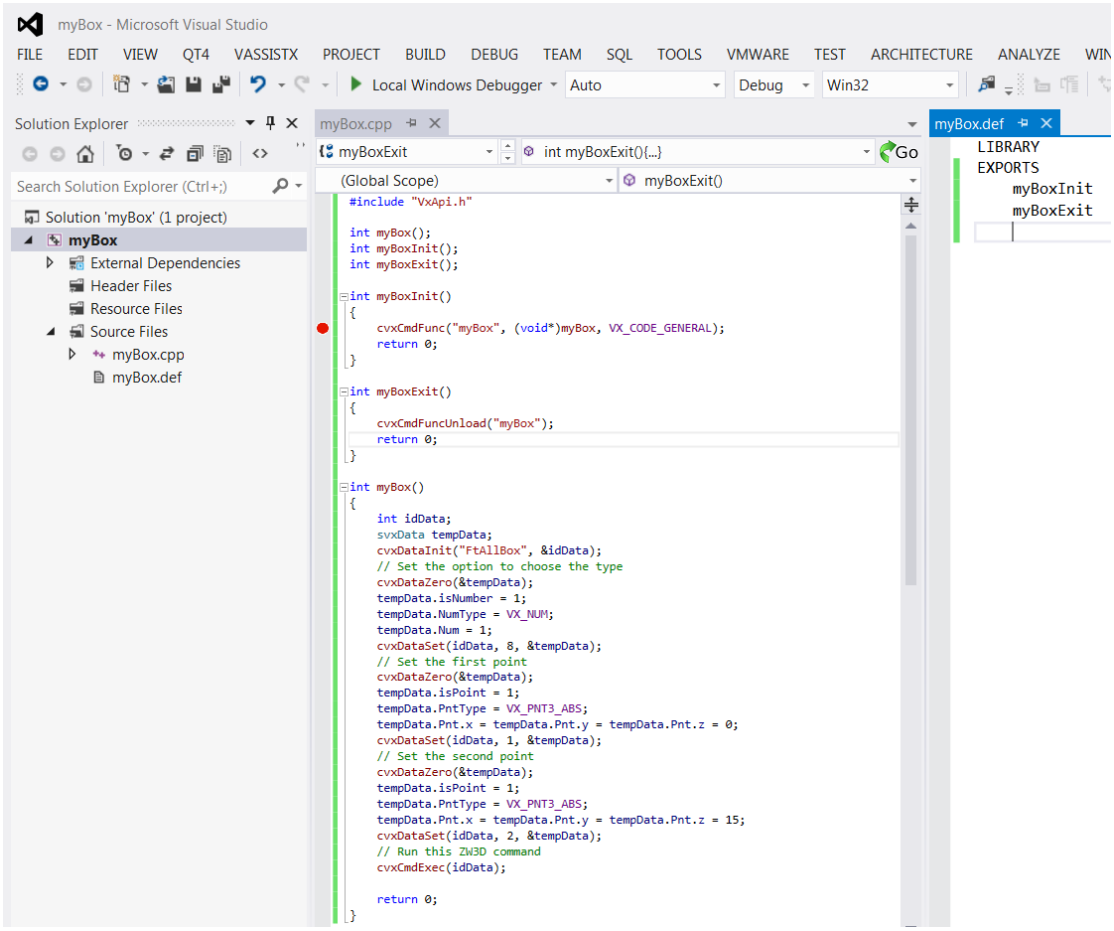
   b) 添加 myBox.cpp，复制以下代码：

```
#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) 添加模块定义文件 myBox.def，复制如下代码：

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

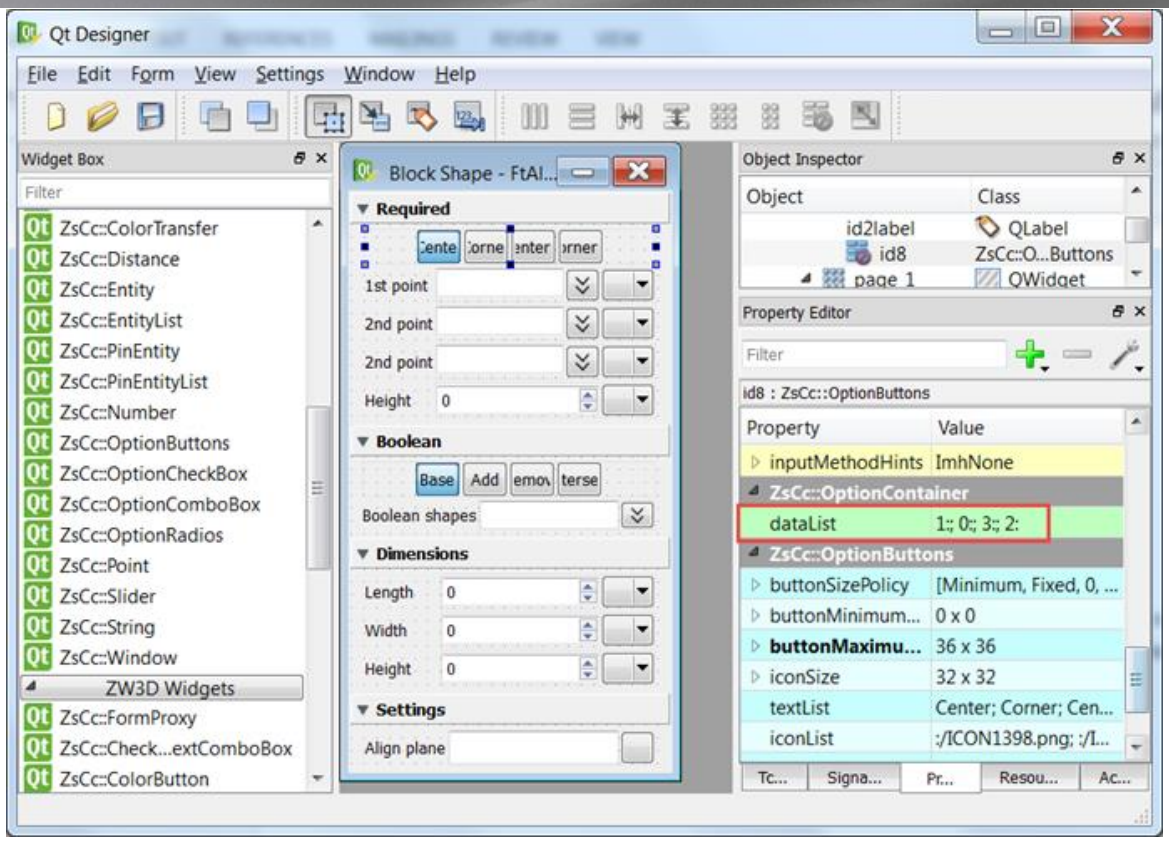



生成项目，在 ZW3D 中加载 DLL。新建零件后，在命令行中运行“~myBox”。这个命令会在建模区新建一个六面体。

5. ZsCc::optionbuttons 的值。

   点击控件后，您可以找到 **DataList** 的属性，控件的值的顺序应该是 1, 0, 3, 2。因此您需要在代码中设置正确的值。以下代码表示该项目使用第一按钮。如果您想要使用第二个按钮需将值设置为 0.

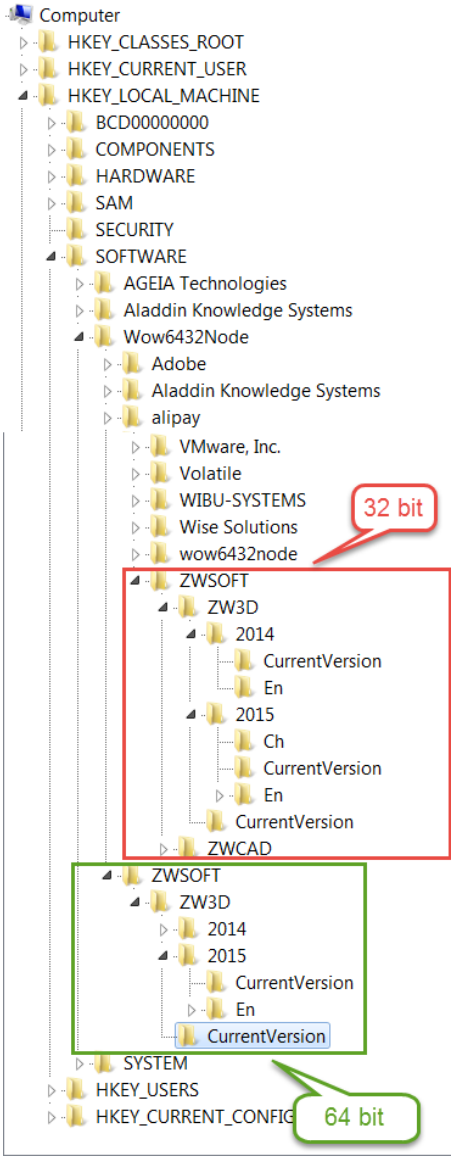
```
// Set the option to choose the type
    cvxDataZero(&tempData);
    tempData.isNumber = 1;
    tempData.NumType = VX_NUM;
    tempData.Num = 1;
    cvxDataSet(idData, 8, &tempData);
```

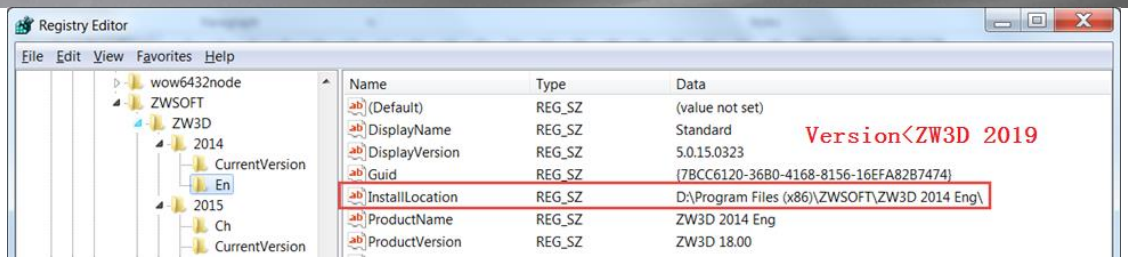

Qt Designer
File Edit Form View Settings Window Help
Widget Box
Filter
ZsCc::ColorTransfer
ZsCc::Distance
ZsCc::Entity
ZsCc::EntityList
ZsCc::PinEntity
ZsCc::PinEntityList
ZsCc::Number
ZsCc::OptionButtons
ZsCc::OptionCheckBox
ZsCc::OptionComboBox
ZsCc::OptionRadios
ZsCc::Point
ZsCc::Slider
ZsCc::String
ZsCc::Window
ZW3D Widgets
ZsCc::FormProxy
ZsCc::Check...extComboBox
ZsCc::ColorButton
Block Shape - FtAl...
Required
1st point
2nd point
2nd point
Height
Boolean
Base
Add
Boolean shapes
Dimensions
Length
Width
Height
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...

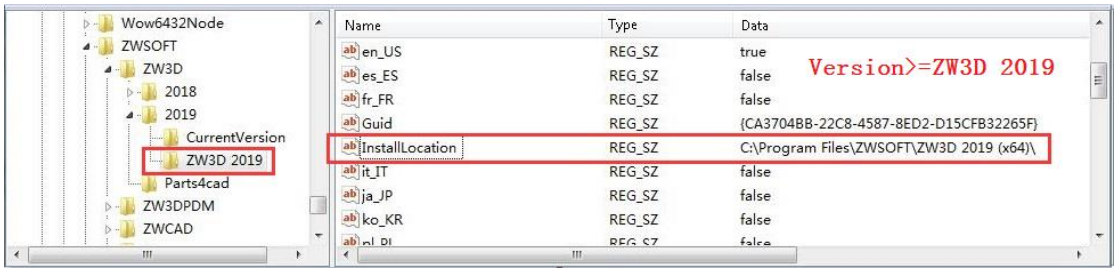
# 第六章 ZW3D 注册信息

ZW3D 的注册信息保存在：
64 位：[HKEY_LOCAL_MACHINE\SOFTWARE\ZWSOFT\ZW3D]
32 位：[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ZWSOFT\ZW3D]

根目录中的 **CurrentVersion** 将记住上次运行的版本。特殊版本目录里的 **CurrentVersion** 将记住上次运行的语言。
同样，您可以在每个语言目录下获取详细信息，比如安装路径。




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