

uDDS 数据分发服务软件 开发手册-VS2017

南京磐优信息科技有限公司

2022 年 12 月

修订记录

版本号	修订状态	简要说明修订内容和范围	修订日期	修订人
V1.0	A	创建文件	20221118	蒋梦杰
	A	添加进阶开发	20221121	蒋梦杰

注：修订记录在文件提交后换版时使用，修订状态栏填写：A—增加，M—修改，D—删除

目 录

1 准备工作	3
2 目录介绍	3
3 快速开发过程	3
4 进阶开发过程	6
4.1 生成辅助文件	6
4.2 创建发布端/订阅端。	7
5 错误排查	11
5.1 拷贝文件失败	12
5.2 找不到指定文件	12
5.3 SDK 版本错误.....	13
5.4 Write error 200	14
6 发布端示例程序	14
7 订阅端示例程序	19

1 准备工作

- 1) Windows 操作系统。
- 2) Visual Studio 2017。
- 3) 试用版 uDDS (uDDS-Trial) , 可在[下载-磐优 \(udds.cn\)](http://udds.cn) 获取。

2 目录介绍

uDDS-Trial 文件内容介绍:

uDDS_VS2017/:vs2017 版本 uDDS 库, 头文件。

进阶开发资源/uDDSGen/:处理 IDL 文件, 生成辅助代码, 配合 uDDS 库使用。

进阶开发资源/示例代码/: 开发手册中发布和订阅的示例代码。

快速开发资源/DemoTool/:DDS 示例自动生成工具,快速生成 uDDS 示例项目,可直接编译运行。

用户开发手册/:适用于 Windows 平台 Visual Studio 2017 的 uDDS 开发手册。

3 快速开发过程

- 1) 进入 “快速开发资源/DemoTool” 文件。

如需更改 idl 内容请见[生成辅助文件](#),更改后的 xxx.idl 可以拷贝到该目录下,后续用 xxx.idl 来代替 UserDataTypes.idl 生成项目,继续快速开发。

- 2) 在 DemoTool 文件目录下打开 cmd，或者直接打开 cmd 转到 DemoTool 目录下，并执行如下代码

```
DemoTool.exe UserDataTypes.idl vs2017
```

如图所示，“1”表示 DemoTool 程序，“2”表示 idl 文件名，“3”表示 vs 版本信息。

```
E:\uDDS_VS2017_Trial\DemoTool>DemoTool.exe UserDataTypes.idl vs2017
格式命令正确,继续运行
start init resource..
***** 拷贝 resource_vs2017 文件夹成功! *****
已复制 1 个文件。
已复制 1 个文件。
Start Create Pub TypeFiles ...;
UserDataTypes.idl

/*****
/*
/*          Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*          Update time 2022.09.19
/*
*****/
用于 x64 的 Microsoft (R) C/C++ 优化编译器 19.16.27045 版
版权所有 (C) Microsoft Corporation。保留所有权利。

IDL_UserDataTypes.idl
Start Create Sub TypeFiles ...;
UserDataTypes.idl

/*****
/*
/*          Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*          Update time 2022.09.19
/*
*****/
用于 x64 的 Microsoft (R) C/C++ 优化编译器 19.16.27045 版
版权所有 (C) Microsoft Corporation。保留所有权利。

IDL_UserDataTypes.idl
Start Delete cl ...;
*****创建目标文件夹成功!*****
Start init project set ...;
创建成功,请在本目录下寻找 PubSub_vs2017 文件夹!
End ...;

**** Demo 创建完成 ! ****

E:\uDDS_VS2017_Trial\DemoTool>
```

图 1 生成项目

- 3) 进入 PubSub_vs2017 文件后,鼠标右击 Sub_Pub.sln 适用 Visual Studio 2017 打开。如图所示。

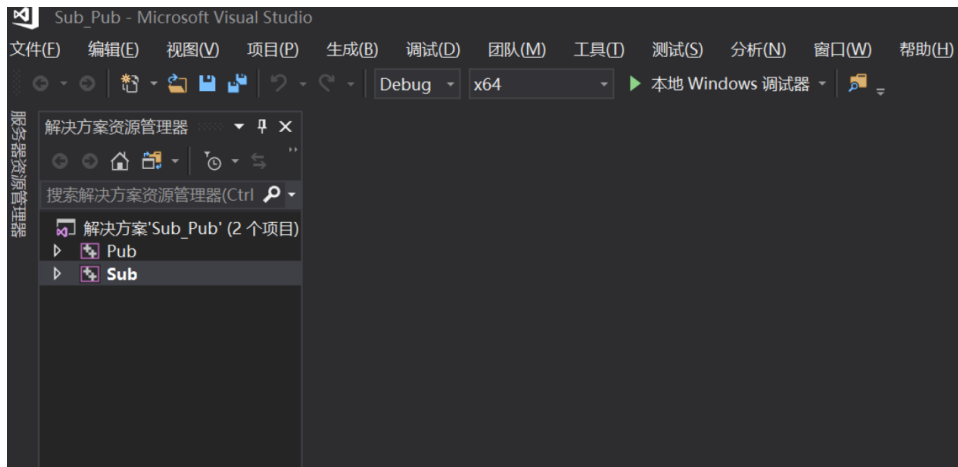


图 2 进入 sln

- 4) 鼠标分别右击 Pub 和 Sub 的解决方案->点击生成,稍后就会出现生成成功的提示。[如果出现“找不到 SDK 版本”的错误请跳转 SDK 版本错误进行解决,解决后,返回继续接下来的步骤。](#)
- 5) 打开 PubSub_vs2017\bin 文件,进入生成对应的解决方案配置版本(debug, release)和平台版本(x86, x64)文件即可。
- 6) 双击 Pub.exe 和 Sub.exe 即可分别运行数据发布和数据接收两个演示程序,运行结果如图所示。

```

*****
/*
/*      Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*      TRIAL
/*      The number of domains is limited to 3
/*      The number of topics is limited to 3
/*
/*      Update time 2022.04.19
/*
/*
*****
DDSLogConfig.xml does not exist,Use default configuration!
网卡名称: {EFA9442B-1420-4A91-907B-5394782D8449}
IP 地址 : 10.0.0.36
网卡名称 : 以太网 2

Create TransmitRun
SPDP: add Remote Participant 10.0.0.36.0.0.0.0.134.26.0.0.0.1.1
SPDP: add Remote Participant 10.0.0.36.0.0.0.0.206.10.0.0.0.1.1
SEDP: add remote DataReader Example UserDataTypes 10.0.0.36.0.0.0.0.1.1
0 : write successfully . .
1 : write successfully . .
2 : write successfully . .
3 : write successfully . .
4 : write successfully . .

网卡名称: {EFA9442B-1420-4A91-907B-5394782D8449}
IP 地址 : 10.0.0.36
网卡名称 : 以太网 2

Create TransmitRun
SPDP: add Remote Participant 10.0.0.36.0.0.0.0.134.26.0.0.0.1.1
SPDP: add Remote Participant 10.0.0.36.0.0.0.0.206.10.0.0.0.1.1
SEDP: add remote DataWriter Example UserDataTypes 10.0.0.36.0.0.0.0.1.1
UserDataTypes:
x = 0
y = 0
color =
UserDataTypes:
x = 0
y = 0
color =
UserDataTypes:
x = 0
y = 0
color =
UserDataTypes:
x = 0
y = 0
color =

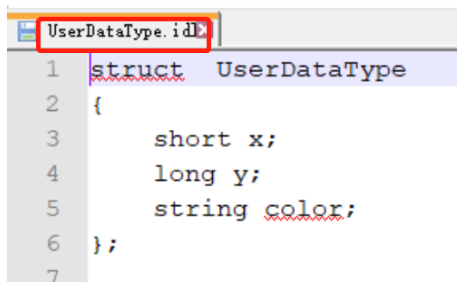
```

图 3 运行结果

4 进阶开发过程

4.1 生成辅助文件

- 1) 进入 uDDSGen 文件，创建一个 idl 文件并在 idl 文件里面添加需要使用的数据类型，数据类型的定义形式为 IDL 文件，用户可以创建一个文本文件，然后将扩展名修改为 “.idl”，并使用文本编辑器进行数据类型定义，如图所示。



```

1 struct UserDataTyp.e
2 {
3     short x;
4     long y;
5     string color;
6 };
7

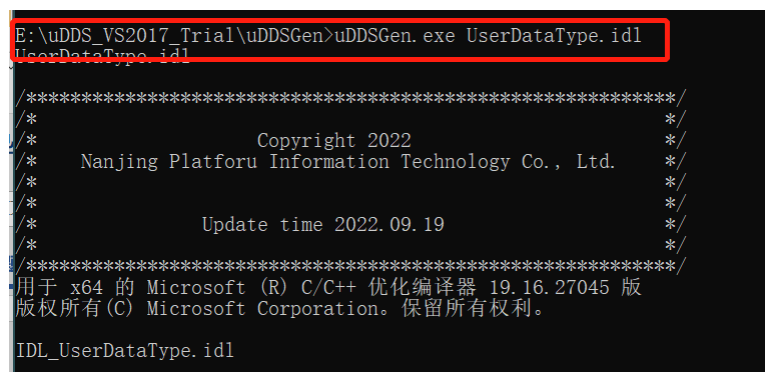
```

图 4 用户定义数据类型

- 2) 使用控制台进入 uDDSGen 目录并执行 uDDSGen.exe 程序，对用户定义 idl 文件进行编译。例如编译器名称为 uDDSGen.exe，用户定义 idl 文件名称为 UserDataTyp.e.idl，执行代码如下

```
uDDSGen.exe UserDataTyp.e.idl
```

如图所示。



```

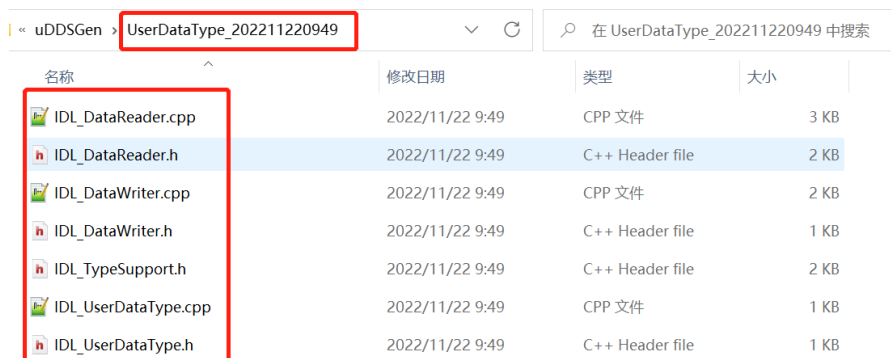
E:\uDDS_VS2017_Trial\uDDSGen>uDDSGen.exe UserDataTyp.e.idl
UserDataTyp.e.idl

/*****
/*
/*      Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*
/*      Update time 2022.09.19
/*
/*
*****/
用于 x64 的 Microsoft (R) C/C++ 优化编译器 19.16.27045 版
版权所有 (C) Microsoft Corporation。保留所有权利。
IDL_UserDataTyp.e.idl

```

图 5 编译 idl 文件

- 3) 在当前目录下会生成一个以 IDL 名字+当前本地年月日时分组合的名字文件，在这个文件里面有 7 个辅助文件，如图所示。



名称	修改日期	类型	大小
IDL_DataReader.cpp	2022/11/22 9:49	C++ 文件	3 KB
IDL_DataReader.h	2022/11/22 9:49	C++ Header file	2 KB
IDL_DataWriter.cpp	2022/11/22 9:49	C++ 文件	2 KB
IDL_DataWriter.h	2022/11/22 9:49	C++ Header file	1 KB
IDL_TypeSupport.h	2022/11/22 9:49	C++ Header file	2 KB
IDL_UserDataType.cpp	2022/11/22 9:49	C++ 文件	1 KB
IDL_UserDataType.h	2022/11/22 9:49	C++ Header file	1 KB

图 6 辅助文件

这七个文件的主要内容如下：

- IDL_DataReader.h: UserDataTypes 数据类型订阅接口定义
- IDL_DataReader.cpp: UserDataTypes 数据类型订阅接口实现
- IDL_DataWriter.h: UserDataTypes 数据类型发布接口定义
- IDL_DataWriter.cpp: UserDataTypes 数据类型发布接口实现
- IDL_TypeSupport.h: UserDataTypes 数据类型实现需要头文件及类
- IDL_UserDataType.h: UserDataTypes 数据类型的定义
- IDL_UserDataType.cpp: UserDataTypes 数据类型的函数实现

4.2 创建发布端/订阅端。

使用生成的辅助代码进行用户程序开发，新建工程将上面生成的七个文件添加进来，并分别添加发布端/订阅端示例程序。

- 1) 创建解决方案 uDDS_demo，在解决方案下创建项目 publisher 和 subscriber。打开 visual studio 2017，按如下步骤进行操作：“文件”（界面左上角）->“新建”->“项目”【或者使用快捷键 Ctrl+shift+N】。选择“Visual C++”里的“空项目”，并对解决方案和项目进行命名。【注：一个解决方案中可以有多个项目，每个项目都是一个单独的程序】在此处先创建解决方案 uDDS_demo 和项目 publisher，如图所示。

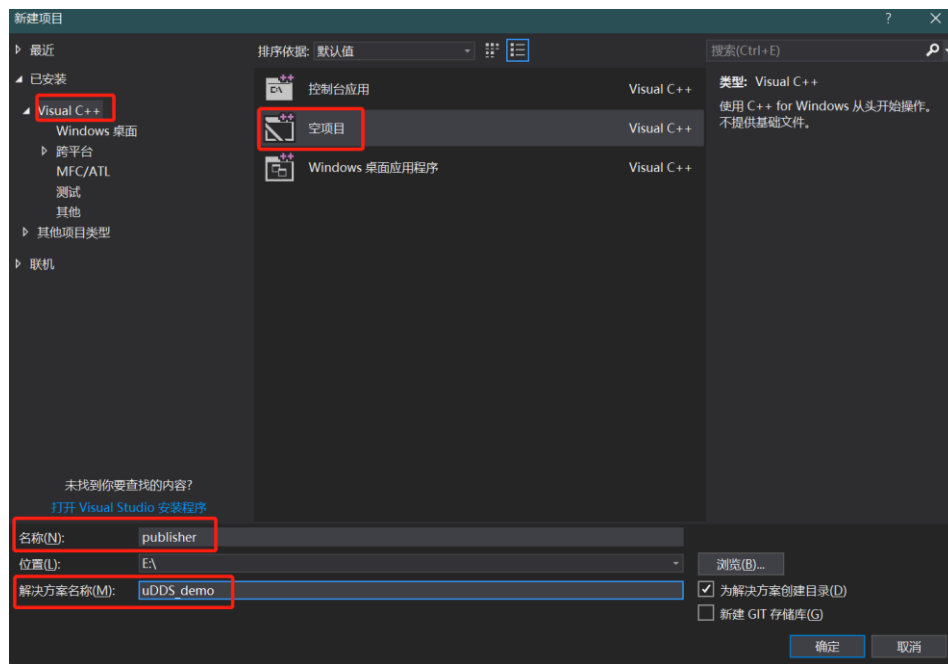


图 7 创建解决方案

- 2) 返回主界面，打开解决方案资源管理器，再创建第二个项目 subscriber。右击“解决方案资源管理器”中的解决方案“uDDS_demo”->“添加”->“新建项目”。此时界面中只有设置项目名的输入框，输入项目名“subscriber”。点击确定，创建项目 subscriber。再次观察解决方案管理器，解决方案 uDDS_demo 下会有两个项目，如图所示。

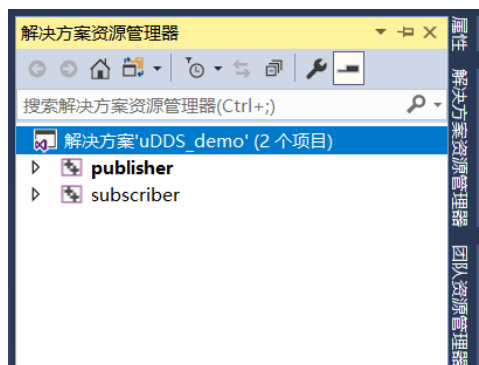


图 8 publisher 和 subscriber 项目

- 3) 将 IDL 编译器 uDDSGen.exe 生成的七个文件各复制一份到两个项目对应的本地目录中（~\uDDS_demo\publisher 目录和 ~\uDDS_demo\subscriber 目录，本地目录添加完成后，还需要在“解决方案资源管理器”的项目里手动导入，“右击头文件”->“添加现有项”->“选择所有.h 文件”，“右击源文件”->“添加现有项”->“选择所有.cpp 文件”，将每个项目对应目录里的文件导入（也可以直接把头文件和源文件拖入），同时我们提供了 pub.cpp 和 sub.cpp 发布和订阅代码模板参考，在“uDDS_VS2017_Trial\示例代码\ExampleCode\winodws”您可以找到他们并加入到您的项目中，如图所示。

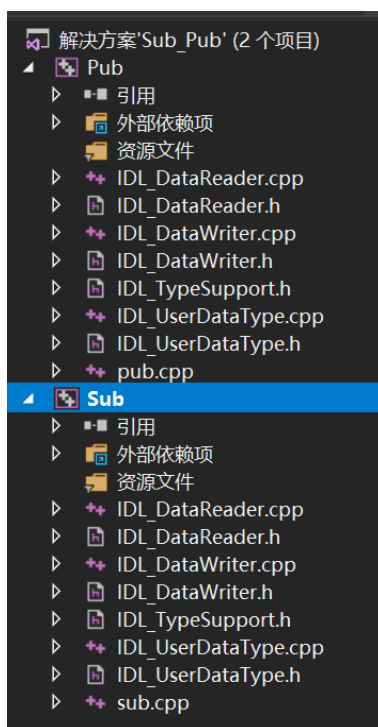


图 9 导入辅助文件

- 4) 在配置项目属性前，x86 和 x64 的环境配置是独立，以下所有配置在切换平台之后都要重新配置，如图所示。



图 10 注意平台对应

- 5) 为项目添加 uDDS 的头文件路径。在解决方案资源管理器中右击项目 publisher 点击属性。选择“C/C++”->“常规”中的“附加包含目录”，设置头文件所在路径。（请注意，具体路径以在您电脑上的真实情况为准，主要就是把 uDDS 目录中的 include 目录添加进去）。在为

publisher 项目完成如上操作后，为 subscriber 项目进行相同的头文件路径添加操作。

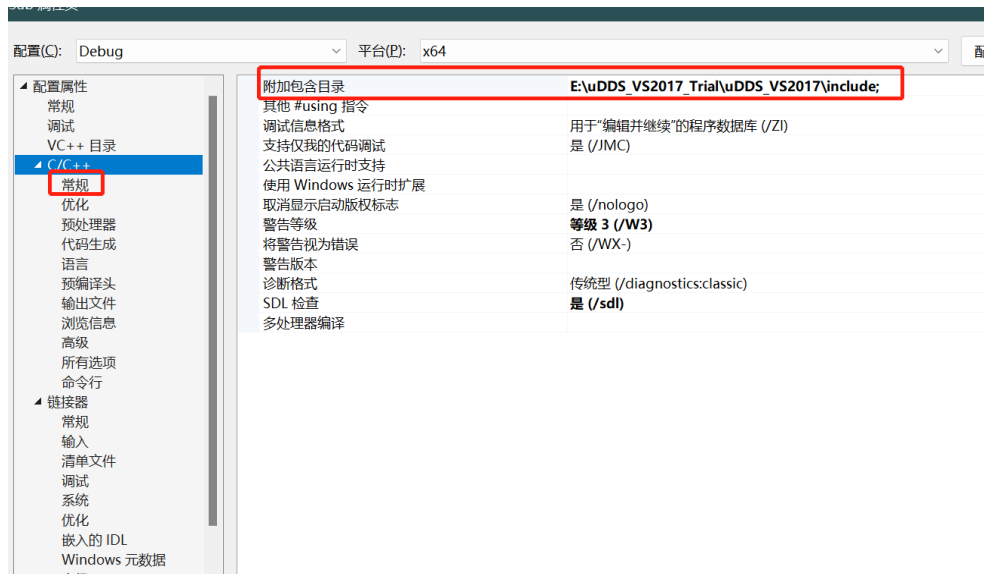


图 11 添加“附加包含目录”

- 6) 添加 uDDS 动态库，debug 版本的动态库名字为 uDDSD.lib，release 版本的动态库名字为 uDDS.lib 。

首先添加 lib 文件所在路径，右击项目名 publisher 或 subscriber，点击“属性”。选择“链接器” -> “常规” -> “附加库目录”，在其中设置 lib 文件所在路径。

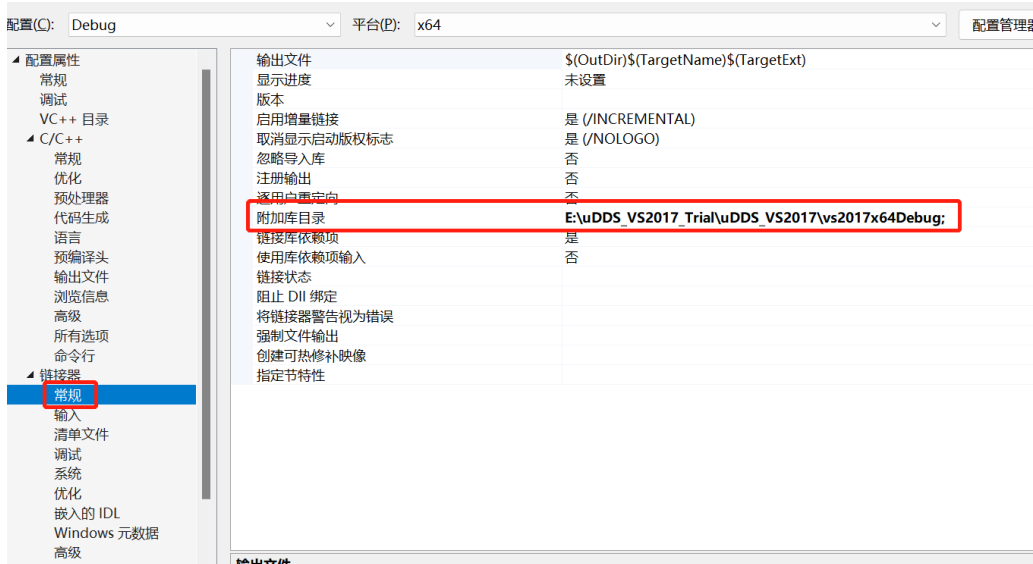


图 12 添加 lib 路径

添加 lib 库名字。右击项目名，点击“属性”，选择“链接器” -> “输入” -> “附加依赖项”，在其中手动输入 lib 的名字。（debug 版本的动态库名字

为 uDDSD.lib，release 版本的动态库名字为 uDDS.lib 。) 如图所示。

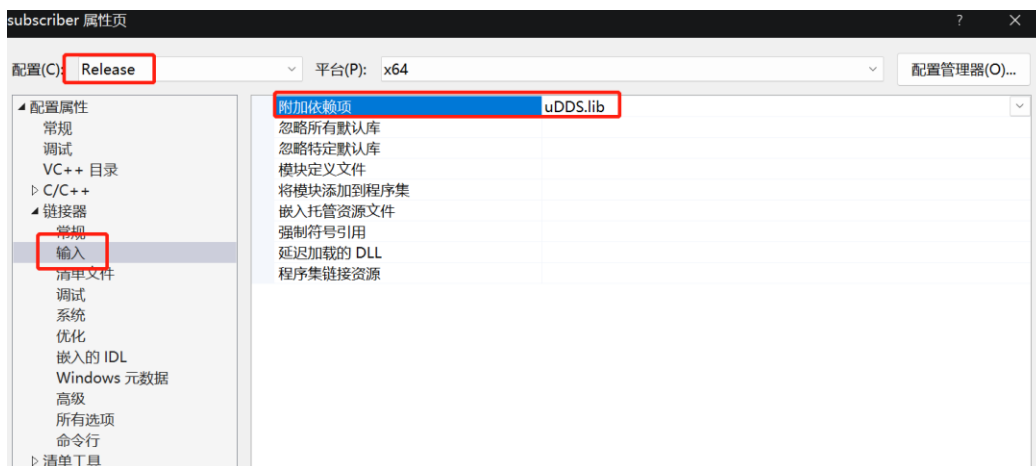


图 13 release 版本 lib



图 14 debug 版本 lib

7) 为程序添加 uDDS 动态库

将 uDDS.dll 拷贝到应用软件可执行程序的同一下目录下, 如果您需要调试请在调试目录下添加动态库文件。

5 错误排查

下面内容有助于解决一些问题, 这些问题是在您拿到压缩文件到运行演示结束, 过程中可能会遇到。如果这一阶段您没有出现问题, 可以忽略。

5.1 拷贝文件失败

如图所示。

```
E:\uDDS_VS2017_Trial\DemoTool>DemoTool.exe UserDataType.idl vs2017
格式命令正确,继续运行
start init resource..
找不到文件 - *.*
***** 拷贝 resource_vs2017 文件夹失败! *****
Start init project set ..;
创建成功,请在本目录下寻找 PubSub_vs2017 文件夹!
End ..;

**** Demo 创建完成 ! ****

E:\uDDS_VS2017_Trial\DemoTool>
```

图 15 拷贝 vs 版本失败

问题原因：输入的 vs 版本和下载的 uDDS-Trial 的 vs 版本不对应。

解决方法：输入正确的 vs 版本或者去[下载-磐优 \(udds.cn\)](http://udds.cn)下载您需要的 vs 版本。

5.2 找不到指定文件

如图所示。

```
E:\uDDS_VS2017_Trial\DemoTool>DemoTool.exe mydata.idl vs2017
格式命令正确,继续运行
start init resource..
***** 拷贝 resource_vs2017 文件夹成功! *****
系统找不到指定的文件。
系统找不到指定的文件。
Start Create Pub TypeFiles ..;
mydata.idl
can not open!Start Create Sub TypeFiles ..;
mydata.idl
can not open!Start Delete cl ..;
*****创建目标文件夹成功! *****
Start init project set ..;
创建成功,请在本目录下寻找 PubSub_vs2017 文件夹!
-
```

图 16 找不到指定文件

问题原因：idl 文件输入有误。

解决方法：确认一下 DemoTool 文件下，用户输入的 idl 文件名字是否与 UserDataType.idl 对应。

5.3 SDK 版本错误

如图所示。

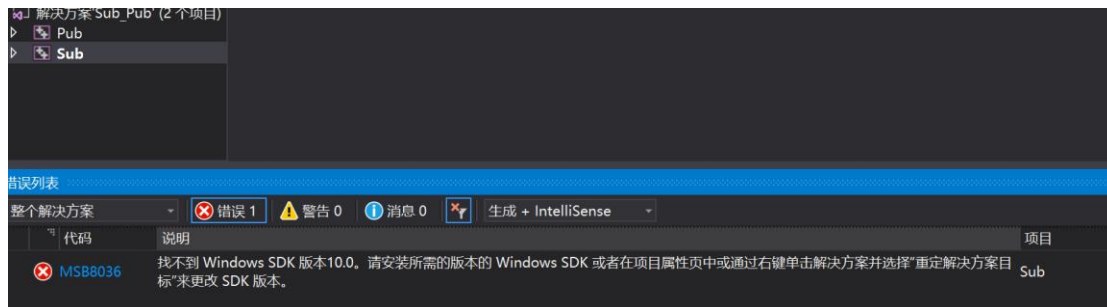


图 17 SDK 版本错误

问题原因：用户使用的 SDK 版本和预先设定的 SDK 版本不统一。

解决方法：右击项目属性->配置属性->常规->Windows SDK 版本，选择一版您那边所拥有的即可，如图所示。



图 18 解决 SDK 问题

5.4 Write error 200

如图所示。

```

/*****
/*
/*      Copyright 2022
/*      Nanjing Platform Information Technology Co., Ltd.
/*
/*      TRIAL
/*      The number of domains is limited to 3
/*      The number of topics is limited to 3
/*
/*      Update time 2022.04.19
/*
*****/
DDSLogConfig.xml does not exist,Use default configuration!
网卡名称: {EFA9442B-1420-4A91-907B-5394782D8449}
IP 地址  : 10.0.0.36
网卡名称 : 以太网 2

Create TransmitRun
SPDP: add Remote Participant 10.0.0.36.0.0.0.0.164.26.0.0.0.1.193. total count = 1
write error 200
write error 200
write error 200
write error 200

```

图 19 write error 200

问题原因：没有对应的订阅端演示程序。

解决方法：启动一个订阅端演示程序即可。

6 发布端示例程序

```

/*****
*****发布端程序pub.cpp*****
*****/
#include <stdio.h>
#include <stdlib.h>
#include <cstring>

/* IDL_TypeSupport.h中包含所有依赖的头文件 */
#include "IDL_TypeSupport.h"

/* 删除所有实体 */
static int publisher_shutdown(DomainParticipant *participant)
{
    ReturnCode_t retcode;
    int status = 0;

    if (participant != NULL) {
        retcode = participant->delete_contained_entities();
        if (retcode != RETCODE_OK) {
            fprintf(stderr, "delete_contained_entities
error %d\n", retcode);
            status = -1;
        }

        retcode =
DomainParticipantFactory::get_instance()->delete_participant(participant);
        if (retcode != RETCODE_OK) {

```

```

        fprintf(stderr, "delete_participant error %d\n",
retcode);
        status = -1;
    }
}

return status;
}

/* 发布者函数 */
extern "C" int publisher_main(int domainId, int sample_count)
{
    DomainParticipant *participant = NULL;
    Publisher *publisher = NULL;
    Topic *topic = NULL;
    DataWriter *writer = NULL;
    UserDataWriter * UserDataWriter = NULL;
    UserData *instance = NULL;
    ReturnCode_t retcode;
    InstanceHandle_t instance_handle = HANDLE_NIL;
    const char *type_name = NULL;
    int count = 0;

    /* 1. 创建一个participant, 可以在此处定制participant的QoS */
    /* 建议1: 在程序启动后优先创建participant, 进行资源初始化*/
    /* 建议2: 相同的domainId只创建一次participant, 重复创建会占用大量资源 */
    /*
    participant =
DomainParticipantFactory::get_instance()->create_participant(
        domainId, PARTICIPANT_QOS_DEFAULT/* participant默认QoS */,
        NULL /* listener */, STATUS_MASK_NONE);
    if (participant == NULL) {
        fprintf(stderr, "create_participant error\n");
        publisher_shutdown(participant);
        return -1;
    }

    /* 2. 创建一个publisher, 可以在创建publisher的同时定制其QoS */
    /* 建议1: 在程序启动后优先创建publisher */
    /* 建议2: 一个participant下创建一个publisher即可, 无需重复创建 */
    publisher = participant->create_publisher(
        PUBLISHER_QOS_DEFAULT /* 默认QoS */,
        NULL /* listener */, STATUS_MASK_NONE);
    if (publisher == NULL) {
        fprintf(stderr, "create_publisher error\n");
        publisher_shutdown(participant);
        return -1;
    }

    /* 3. 在创建主题之前注册数据类型 */
    /* 建议1: 在程序启动后优先进行注册 */

```



```

/* 建议2: 一个数据类型注册一次即可 */
type_name = UserDataDataTypeSupport::get_type_name();
retcode = UserDataDataTypeSupport::register_type(
    participant, type_name);
if (retcode != RETCODE_OK) {
    fprintf(stderr, "register_type error %d\n", retcode);
    publisher_shutdown(participant);
    return -1;
}

/* 4. 创建主题, 并定制主题的QoS */
/* 建议1: 在程序启动后优先创建Topic */
/* 建议2: 一种主题名创建一次即可, 无需重复创建 */
topic = participant->create_topic(
    "Example UserDataType"/* 主题名 */,
    type_name /* 类型名 */, TOPIC_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (topic == NULL) {
    fprintf(stderr, "create_topic error\n");
    publisher_shutdown(participant);
    return -1;
}

/* 5. 创建datawriter, 并定制datawriter的QoS */
/* 建议1: 在程序启动后优先创建datawriter */
/* 建议2: 创建一次即可, 无需重复创建 */
/* 建议3: 在程序退出时再进行释放 */
/* 建议4: 避免打算发送数据时创建datawriter, 发送数据后删除, 该做法消耗
资源, 影响性能 */
writer = publisher->create_datawriter(
    topic, DATAWRITER_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (writer == NULL) {
    fprintf(stderr, "create_datawriter error\n");
    publisher_shutdown(participant);
    return -1;
}
UserDataWriter = UserDataWriter::narrow(writer);
if (UserDataWriter == NULL) {
    fprintf(stderr, "DataWriter narrow error\n");
    publisher_shutdown(participant);
    return -1;
}

/* 6. 创建一个数据样本 */
/* 建议: 该数据为new出来的, 使用后用户需要调用delete_data进行释放内存
*/
instance = UserDataDataTypeSupport::create_data();
if (instance == NULL) {
    fprintf(stderr, "UserDataDataTypeSupport::create_data
error\n");

```

```

        publisher_shutdown(participant);
        return -1;
    }

    //此处可以修改数据的值
    instance->x = 3;
    instance->y = 4;
    instance->color = const_cast<char *>("red");

    /* 7. 主循环 , 发送数据 */
    for (count = 0; (sample_count == 0) || (count < sample_count); ++count)
    {
        retcode = UserDataWriter->write(*instance,
instance_handle);
        if (retcode != RETCODE_OK) {
            fprintf(stderr, "write error %d\n", retcode);
        }
        else
            fprintf(stderr, "%d : write successfully . . \n",
count);

#ifdef ULINX_LINUX
            sleep(1);
#elif (defined( ULINX_WIN) || defined( _WIN32))
            Sleep(1000);
#endif //沉睡一秒
    }

    /* 8. 删除数据样本 */
    retcode = UserDataWriterSupport::delete_data(instance);
    if (retcode != RETCODE_OK) {
        fprintf(stderr, "UserDataWriterSupport::delete_data
error %d\n", retcode);
    }

    /* 9. 删除所有实例 */
    return publisher_shutdown(participant);
}

int main(int argc, char *argv[])
{
    int domain_id = 0;
    int sample_count = 0; /* 无限循环 */

    if (argc >= 2) {
        domain_id = atoi(argv[1]); /* 发送至域domain_id */
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]); /* 发送sample_count次 */
    }
}

```

```
}  
    return publisher_main(domain_id, sample_count);  
}
```

7 订阅端示例程序

```

/*****
*****订阅端程序sub.cpp*****
*****/
#include <stdio.h>
#include <stdlib.h>
#include <cstring>

/* IDL_TypeSupport.h中包含所有依赖的头文件 */
#include "IDL_TypeSupport.h"

/* UserDataListener继承于DataReaderListener,
   需要重写其继承过来的方法on_data_available(), 在其中进行数据监听读取操作 */
class UserDataListener : public DataReaderListener {
public:
    virtual void on_data_available(DataReader* reader);
};

/* 重写继承过来的方法on_data_available(), 在其中进行数据监听读取操作 */
void UserDataListener::on_data_available(DataReader* reader)
{
    UserDataReader *UserDataReader = NULL;
    UserDataSeq data_seq;
    SampleInfoSeq info_seq;
    ReturnCode_t retcode;
    int i;

    /* 利用reader, 创建一个读取UserData类型的UserDataReader */
    UserDataReader = UserDataReader::narrow(reader);
    if (UserDataReader == NULL) {
        fprintf(stderr, "DataReader narrow error\n");
        return;
    }

    /* 获取数据, 存放至data_seq, data_seq是一个队列 */
    retcode = UserDataReader->read(
        data_seq, info_seq, DDS_LENGTH_UNLIMITED, DDS_ANY_SAMPLE_STATE,
        DDS_ANY_VIEW_STATE, DDS_ANY_INSTANCE_STATE);
    if (retcode == RETCODE_NO_DATA) {
        return;
    }
    else if (retcode != RETCODE_OK) {
        fprintf(stderr, "take error %d\n", retcode);
        return;
    }

    /* 打印数据 */
    /* 建议1: 避免在此进行复杂数据处理 */

```

```

/* 建议2: 将数据传送到其他数据处理线程中进行处理 */
/* 建议3: 假如数据结构中有string类型, 用完后需手动释放 */
for (i = 0; i < data_seq.length(); ++i) {
    UserDataSupport::print_data(&data_seq[i]);
}
}

/* 删除所有实体 */
static int subscriber_shutdown(
    DomainParticipant *participant)
{
    ReturnCode_t retcode;
    int status = 0;

    if (participant != NULL) {
        retcode = participant->delete_contained_entities();
        if (retcode != RETCODE_OK) {
            fprintf(stderr, "delete_contained_entities error %d\n",
retcode);
            status = -1;
        }

        retcode =
DomainParticipantFactory::get_instance()->delete_participant(participant);
        if (retcode != RETCODE_OK) {
            fprintf(stderr, "delete_participant error %d\n",
retcode);
            status = -1;
        }
    }
    return status;
}

/* 订阅者函数 */
extern "C" int subscriber_main(int domainId, int sample_count)
{
    DomainParticipant *participant = NULL;
    Subscriber *subscriber = NULL;
    Topic *topic = NULL;
    UserDataListener *reader_listener = NULL;
    DataReader *reader = NULL;
    ReturnCode_t retcode;
    const char *type_name = NULL;
    int count = 0;
    int status = 0;

    /* 1. 创建一个participant, 可以在此处定制participant的QoS */
    /* 建议1: 在程序启动后优先创建participant, 进行资源初始化*/
    /* 建议2: 相同的domainId只创建一次participant, 重复创建会占用大量资源 */
    participant = DomainParticipantFactory::get_instance()->create_participant(
        domainId, PARTICIPANT_QOS_DEFAULT/* participant默认QoS */,

```

```

        NULL /* listener */, STATUS_MASK_NONE);
if (participant == NULL) {
    fprintf(stderr, "create_participant error\n");
    subscriber_shutdown(participant);
    return -1;
}

/* 2. 创建一个subscriber, 可以在创建subscriber的同时定制其QoS */
/* 建议1: 在程序启动后优先创建subscriber*/
/* 建议2: 一个participant下创建一个subscriber即可, 无需重复创建 */
subscriber = participant->create_subscriber(
    SUBSCRIBER_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (subscriber == NULL) {
    fprintf(stderr, "create_subscriber error\n");
    subscriber_shutdown(participant);
    return -1;
}

/* 3. 在创建主题之前注册数据类型 */
/* 建议1: 在程序启动后优先进行注册 */
/* 建议2: 一个数据类型注册一次即可 */
type_name = UserDataTypesupport::get_type_name();
retcode = UserDataTypesupport::register_type(
    participant, type_name);
if (retcode != RETCODE_OK) {
    fprintf(stderr, "register_type error %d\n", retcode);
    subscriber_shutdown(participant);
    return -1;
}

/* 4. 创建主题, 并定制主题的QoS */
/* 建议1: 在程序启动后优先创建Topic */
/* 建议2: 一种主题名创建一次即可, 无需重复创建 */
topic = participant->create_topic(
    "Example UserDataTypesupport"/* 主题名, 应与发布者主题名一致 */,
    type_name, TOPIC_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (topic == NULL) {
    fprintf(stderr, "create_topic error\n");
    subscriber_shutdown(participant);
    return -1;
}

/* 5. 创建一个监听器 */
reader_listener = new UserDataTypesupportListener();

/* 6. 创建datareader, 并定制datareader的QoS */
/* 建议1: 在程序启动后优先创建datareader*/
/* 建议2: 创建一次即可, 无需重复创建 */
/* 建议3: 在程序退出时再进行释放 */

```

```

/* 建议4: 避免打算接收数据时创建datareader, 接收数据后删除, 该做法消耗资源,
影响性能 */
reader = subscriber->create_datareader(
    topic, DATAREADER_QOS_DEFAULT/* 默认QoS */,
    reader_listener/* listener */, STATUS_MASK_ALL);
if (reader == NULL) {
    fprintf(stderr, "create_datareader error\n");
    subscriber_shutdown(participant);
    delete reader_listener;
    return -1;
}

/* 7. 主循环 , 监听器会默认调用on_data_available() 监听数据 */
for (count = 0; (sample_count == 0) || (count < sample_count); ++count) {
    //保持进程一直运行
#ifdef ULINX_LINUX
        sleep(1);
    #elif (defined( ULINX_WIN) || defined(_WIN32))
        Sleep(1000);
    #endif //沉睡一秒
}

/* 8. 删除所有实体和监听器 */
status = subscriber_shutdown(participant);
delete reader_listener;

return status;
}

int main(int argc, char *argv[])
{
    int domain_id = 0;
    int sample_count = 0; /* 无限循环 */

    if (argc >= 2) {
        domain_id = atoi(argv[1]);/* 发送至域domain_id */
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]);/* 发送sample_count次 */
    }
    return subscriber_main(domain_id, sample_count);
}

```



uDDS 数据分发服务软件

让智能系统的数据交互更简单

联系人：胡敬羽

联系方式：18100610350

E-mail: hujingyu@platforu.com

公司地址：南京市江北新区星火路 15 号智芯科技大厦 7 楼