uDDS 产品系列

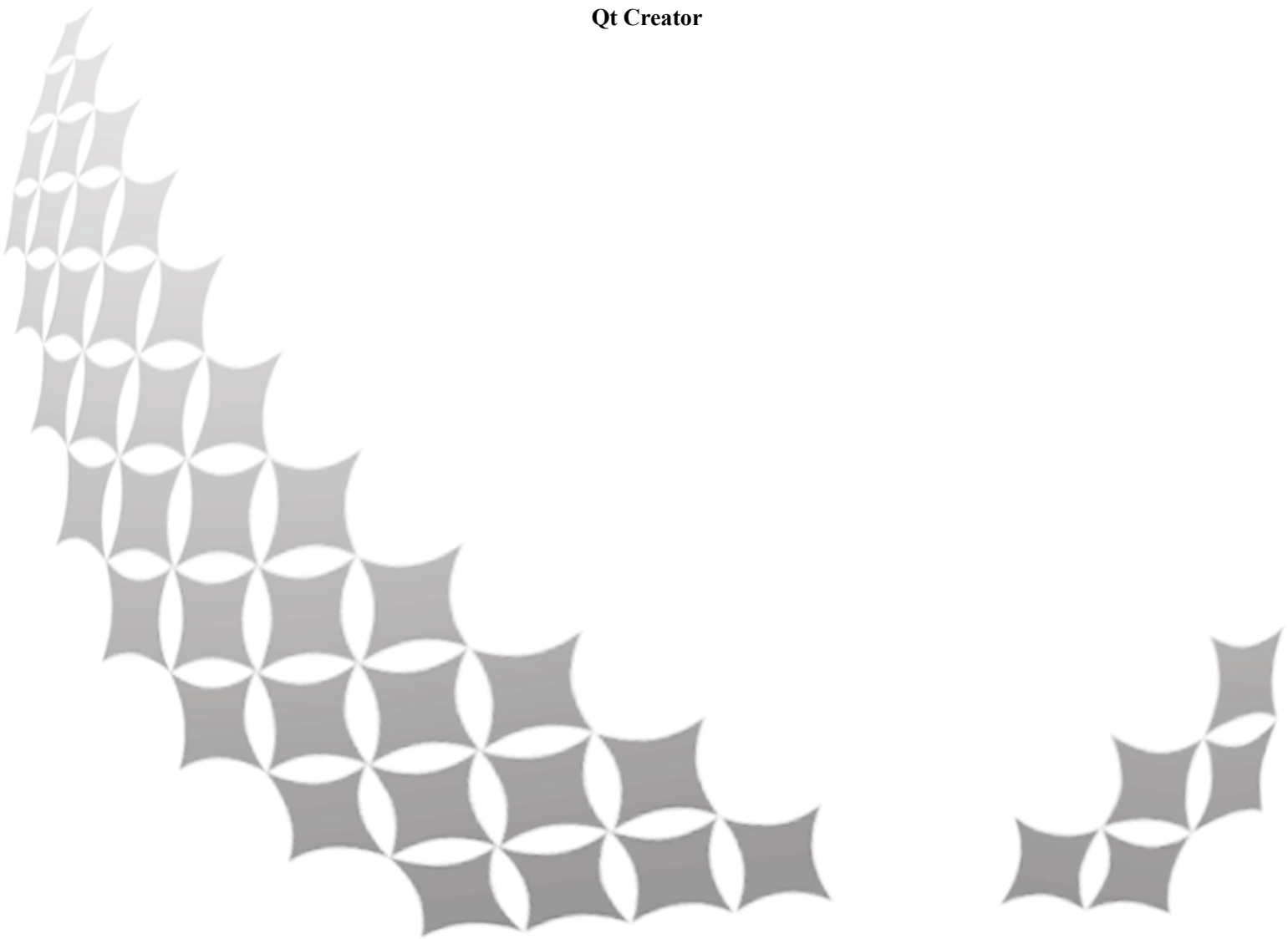
uDDS

开发者指南

Developer Manual

for Kylin

Qt Creator



目　录

第一章

产品简介

概述

uDDS（Blue Data Connect Solution）是由中国船舶工业系统工程研究院研发的面向未来计算机网络应用系统需求的新一代高性能数据交互解决方案。该解决方案由一套逻辑相互关联、功能各有侧重的中间件及工具软件组成，构建在分布式系统网络化硬件平台上，面向分布式系统应用领域需求、系统集成特点，结合中国船舶工业系统工程研究院多年来在海军舰艇作战系统、电子信息系统、武器系统等领域系统集成的丰富经验，为用户提供了高效、可靠的开发/部署/集成大规模分布式实时系统的手段。

说明文件

uDDS 的说明文件包括以下内容：

- ✓ uDDS *安装指南*：介绍如何在您的系统上安装 uDDS。
- ✓ uDDS *版本说明*：介绍 uDDS 能够支持的软硬件平台版本，uDDS 版本编号规范，以及当前版本的更新内容。
- ✓ uDDS *开发者指南*：介绍如何 uDDS 开发应用软件，并给出一个完整的示例程序。
- ✓ uDDS *程序员参考*：介绍如何使用 uDDS 支持的数据类型，以及如何使用 uDDS 所提供的 API 接口。

适用对象

本文档适用于在 UOS 平台下使用 Qt Creator 进行应用软件开发和运行的用

户。

第二章

理解 uDDS

▶什么是 DDS

数据分发服务（Data Distribution Service）是 OMG 的有关分布式实时系统中数据分发的一个规范。该规范标准化了分布式实时系统中数据发布、传递和接受的接口和行为，定义了以数据为中心的发布/订阅（Data-Centric Publish-Subscribe）机制，提供了一个与平台无关的数据模型。

▶什么是 uDDS

uDDS 基于 OMG 组织制订的 DDS1.2 规范，使用订阅/发布机制进行应用功能软件间的信息传输通道组织和信息分发，使应用无需关心信息的来源和去向等信息传输细节，以统一的信息交互接口实现按需信息分发、按需信息获取，根据不同的系统信息流设计模式和要求，提供质量保证策略接口，保证高质量的实时信息传输。

以实时数据分发服务为核心，uDDS 还提供了系统状态监视、数据记录、辅助开发、组件检测等工具套件，为系统提供实时、高效、灵活的数据交互手段，并在应用系统设计、开发、调试、运行的各个环节提供完整的支撑服务。

▶uDDS 特性

uDDS 具有以下部分中描述的一些关键特性。

● 完整的开发环境

uDDS 为分布式系统的开发者提供了完整的开发环境。用户在应用程序开发

阶段可使用 IDL 语言编译器及辅助开发工具实现中间件相关代码的便捷嵌入；在系统集成阶段可使用应用程序检测工具对程序进行预检测，以降低集成测试的故障发生概率和定位成本；系统实际运行阶段可使用系统监控、数据记录工具来获取系统的运行状态，并能够对数据进行离线分析。使用 uDDS 提供的完整开发环境，可以极大程度地降低分布式应用系统设计、实现、测试和故障定位的复杂性。

- 高性能通信服务

uDDS 通过高效运行的核心代码和灵活多样的 QoS 策略，保证应用系统在苛刻的应用场景下能够实现良好的数据交互。

- 应用数据过滤

uDDS 提供多种应用数据过滤方式。在应用系统中进行一对多的数据传输时，不同的订阅者对同一主题的数据内容、数据频率需求不尽相同。针对这一应用需求，uDDS 可根据用户配置，自动过滤掉订阅者不关心的数据，以减少应用程序的数据处理工作。

- 可靠性

uDDS 为大规模应用系统中的应用程序提供 P2P 的信息交互服务，在系统中不存在集中式的代理或服务器进程，从而保证整个系统服务不存在单点故障的风险。

- 独立升级和移植

uDDS 提供多个操作系统的版本，均采用动态库的形式提供应用程序使用，并为不同操作系统下的应用程序提供统一的服务调用接口。应用系统和 uDDS 可分别进行独立升级，且应用系统在不同操作系统上移植时几乎无需针对 uDDS 相关的代码进行改动。

- 扩展性

uDDS 使用“订阅/发布”机制进行数据交互，建立全局的虚拟数据空间，在通信层面将应用逻辑与节点的物理信息解耦合。应用程序可根据需要在系统中的任意节点上部署或迁移，uDDS 均能够自动完成发现过程，保证应用程序实现原有的通信功能。当系统由于链路故障或人为需要而物理上被划分成多个子系统时，各子系统内仍能够按照原有的交互关系进行通信；当多个子系统存在物理连

接时，也同样能够完成自动化集成。

- 丰富的 QoS 策略

uDDS 提供丰富的 QoS 策略。QoS 策略能够独立或组合使用，以满足用户不同的应用场景下对通信质量灵活而复杂的需求。

▶uDDS 兼容性

uDDS 符合对象管理组织(OMG)的 DDS 规范(v1.2)和 RTPS 规范(v2.1)。有关详情，请参阅位于 http://www.omg.org/中的相应规范。

▶uDDS 开发环境

uDDS 作为一套成熟的高性能数据交互解决方案，由一套逻辑相互关联、功能各有侧重的中间件及工具软件组成。其中，与用户开发相关的工具主要包括：

- uDDS 数据定义文件编译工具（IDLC.exe）；
- uDDS 数据交互中间件（uDDS）；

▶uDDS 数据定义文件编译器（uDDS IDLC）

uDDS 数据交互中间件支持用户通过 IDL 文件格式定义用户数据类型，以便在数据收发过程中对应用数据进行封装和解析。在进行应用软件开发之前，请使用 IDL 文件编译器（idlc.exe）对 IDL 文件进行编译，将生成的 C++代码文件添加到应用代码工程中（详见本文第三章）。

▶uDDS 数据交互中间件（uDDS DDS）

uDDS 数据交互中间件以动态库的形式提供应用软件调用，为分布式实时系统提供实时的数据分发服务。为提高应用灵活性，中间件的部分使用参数在配置文件（uDDS.xml）中进行设置，用户可使用文本编辑器对配置文件进行修改（详见本文第三章）。

▶IDL 至 C++的映射

uDDS 符合 OMG IDL/C++语言映射规范。有关通过 idlc 编译程序实现的当前 uDDS IDL 至 C++语言的映射的汇总，请参阅 uDDS 程序员参考。对于每个 IDL 构造，都存在描述相应的 C++构造以及代码示例的小节。

有关映射规范的详情，请参阅 OMG IDL/C++语言映射规范。

第三章

开发实例应用

本节使用实例来描述创建 C++ 下的实时分布式应用程序的开发过程。该实例应用的 C++ 代码可以在 uDDS 软件包安装完成后所在位置的 demos 目录下找到。

开发过程

当您使用 uDDS 开发分布式应用时，您必须首先确定应用程序需要进行发布/订阅的数据类型。以下是开发该实例所采取的步骤的概要描述：

1 使用接口定义语言（IDL）为每种数据类型编写说明；

2 使用 IDL 编译器生成用户代码；

3 将生成的代码添加到应用软件的开发工程中；

4 设置必须的应用软件开发工程属性；

5 为应用软件编辑对应的配置文件；

6 编写应用软件功能代码。

步骤一：定义数据类型

使用 uDDS 创建应用的第一步是使用 OMG 的 接口定义语言 （IDL）说明您需要进行发布/订阅的数据类型，然后您可以使用 idlc 编译器生成辅助开发代码。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（语法与 C/C++基本相同，支持的数据类型见程序员参考）。

例如需要创建的 IDL 文件名为 UserDataType.idl，内容见图 1。

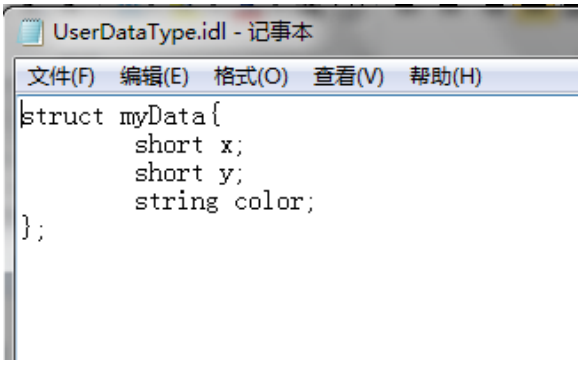


图 1 用户自定义数据类型示例

步骤二：生成辅助文件

在安装目录下的 IDL-tools 文件夹找到名为 idlc-XX.exe 的 IDL 编译器，名字中的 XX 为对应操作系统版本，例如 idlc-windows.exe/ idlc-vxworks.exe/ idlc-kylin.exe，请注意一定要使用对应的操作系统版本（该文件夹中其他文件为编译器的依赖库，请确保与 exe 放在同一目录下）。见图 2。

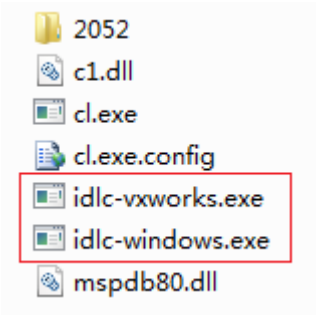


图 2 确认 idlc 编译器

随后，使用控制台进入 IDL-tools 目录并执行 idlc 程序，对用户定义的 idl 文件进行编译。例如 IDL 编译器名称为 idlc.exe，用户定义的 idl 文件名称为 UserDataType.idl，则编译过程见图 3。

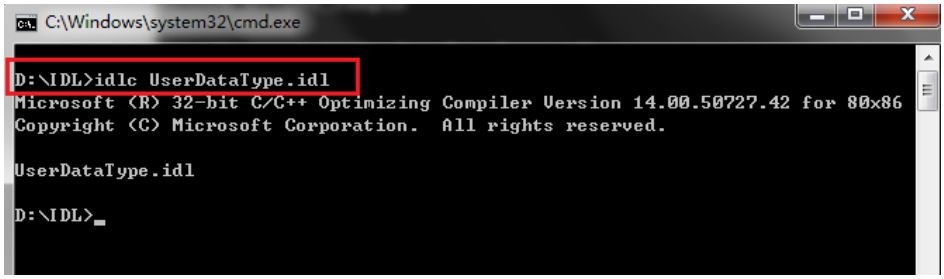


图 3 使用 idlc.exe 编译用户自定义数据类型

如果输出结果为用户编译的 idl 文件名称，如上图所示，则说明编译成功，

编译器将在 IDL-tools 目录下产生十个文件，见图 4。

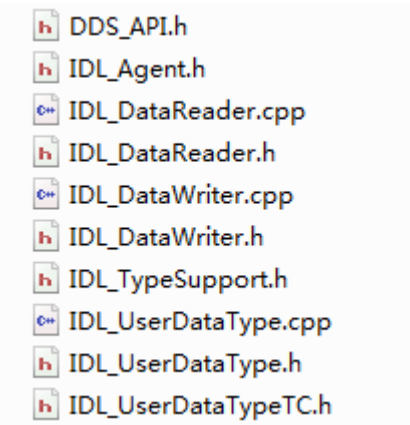


图 4 使用 IDL 编译器编译生成的文件

这十个文件的主要内容如下：

- DDS_API.h：用户接口定义
- IDL_Agent.h：使用 uDDS 库需要头文件及类
- IDL_DataReader.h：UserDataType 数据类型订阅接口定义
- IDL_DataReader.cpp：UserDataType 数据类型订阅接口实现
- IDL_DataWriter.h：UserDataType 数据类型发布接口定义
- IDL_DataWriter.cpp：UserDataType 数据类型发布接口实现
- IDL_TypeSupport.h：UserDataType 数据类型实现需要头文件及类
- IDL_UserDataType.h：UserDataType 数据类型的定义
- IDL_UserDataType.cpp：UserDataType 数据类型的函数实现
- IDL_UserDataTypeTC.h：UserDataType 数据类型的 TypeCode 方法

注意：

1、使用系统内置类型_DDS_STRING 时，首先需要使用 IDL 编译器生成文件，否则无法使用；

2、如果用户在 Visual Studio 中创建工程时选择使用预编译头，需要在生成的 IDL 文件中的三个 cpp 文件中添加#include ”stdafx.h”。

▶步骤三：应用软件环境配置

使用步骤二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十个

文件添加进来，并分别添加发布端/订阅端示例程序。见图 5 和图 6。

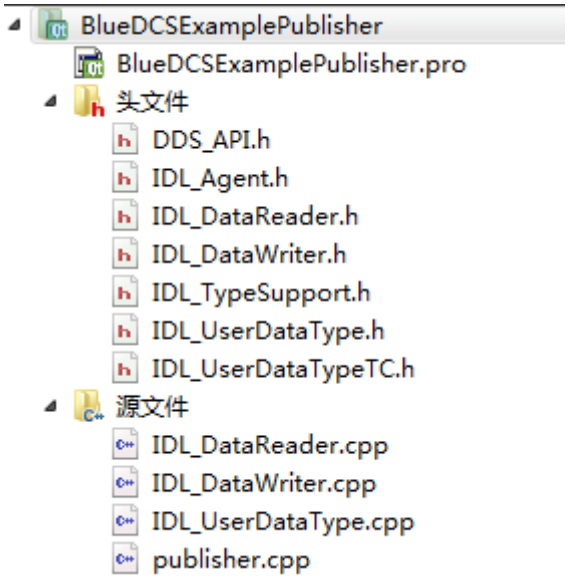


图 5 将生成的文件和发布端例子程序加入到发布端用户工程当中

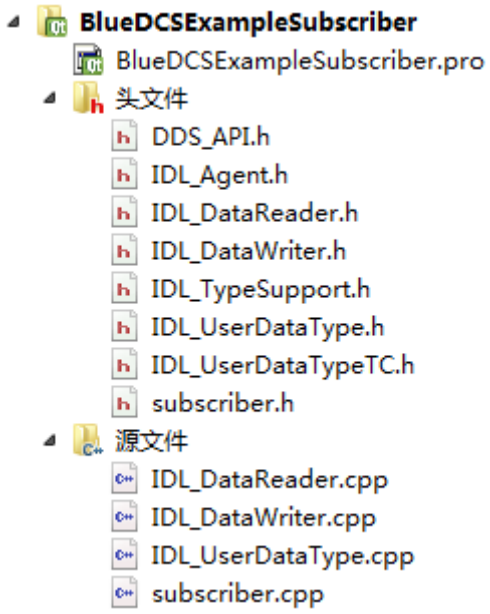


图 6 将生成的文件和订阅端例子程序加入到订阅端用户工程当中

将 uDDS 的头文件路径、库文件路径和库名称添加进工程（.pro 文件），并添加宏定义，见图 7。（请注意，下图只是说明要添加哪些属性项，具体路径以在您电脑上的真实情况为准）

```
43
44 DEFINES += KYLIN
45 LIBS += -L /opt/apps/org.gnome.uDDS/files/lib -luDDS
46 INCLUDEPATH +=/opt/apps/org.gnome.uDDS/files/include
47
```

图 7 将 uDDS 头文件路径、库文件路径和库名称添加到工程

再将 libuDDS.so 和 common 文件夹拷到应用软件可执行程序的同一目录下（一般是 Debug 目录），并将 common 文件夹同时拷贝到应用软件工程目录下。

说明：Kylin 版本的 uDDS 核心库默认在“当前目录”下读取 common 文件夹以获取运行配置。Qt Creator 在执行“CTRL+R”或“F5”时（即在工具内点击绿色三角按钮运行程序时），“当前目录”是图 8 所示的目录，所以需要在此目录下也拷贝一份配置文件以便调试运行。否则将只支持在 Debug 目录下双击可执行程序运行应用软件。

图 8 确认软件工作目录并将 uDDS 配置文件拷贝到该目录下

步骤四：配置文件设置

根据用户的实际需求，进行配置文件的适应性修改，一般需要确认以下部分：

```
<?xml version="1.0" encoding="UTF-8"?>
<uDDS Version="2.3.0" Description="Qos XML Config">
    <common>
        <LocalIp EthName="enp1s0">192.168.1.120</LocalIp>
        <!--OLD VERSION <HostIP EthName="eno16777736">192.168.1.120</H
        <CacheLength>1200</CacheLength>
        <BroadcastAddr>198.1.106.253</BroadcastAddr>
        <MulticastDiscovery UseMulticastDiscovery="1">
            <MultiAddr>239.255.0.1</MultiAddr>
        </MulticastDiscovery>
        <Multicast UseMulticast="1">
            <MultiAddrLow>239.3.3.3</MultiAddrLow>
            <MultiAddrNumber>1</MultiAddrNumber>
        </Multicast>
        <MultiNetworkCard UseMultiNetworkCard="0">
            <MultiNetworkCardCount>2</MultiNetworkCardCount>
            <NetworkCardIP>198.1.106.55</NetworkCardIP>
            <NetworkCardIP>192.168.1.8</NetworkCardIP>
        </MultiNetworkCard>
        <LeaseDuration>
            <PeriodSecond>50</PeriodSecond>
            <PeriodNanosec>0</PeriodNanosec>
        </LeaseDuration>
        <SpecifiedAddr UseSpecifiedAddr="0">
            <SpecifiedIP>192.0.0.0</SpecifiedIP>
        </SpecifiedAddr>
        <PortReadd UsePortReadd="0">0</PortReadd>
```

图 9 配置文件需注意内容

其中，红色标注的部分请按照实际运行机器的网卡名称进行配置。

说明：关于配置文件的具体说明，及更多情况下的配置使用方法，请参见《uDDS 程序员参考》。

▶ 步骤五：应用软件开发

根据用户的实际需求，进行应用软件代码的开发。开发人员可参考以下示例程序。关于示例程序中 uDDS 相关函数的具体定义，请参见《uDDS 程序员参考》。

▶ 发布端示例程序

```
/*********************************************************
*****************发布端程序publisher.cpp******************
```

```
**********************************************************/
#include "dds_api.h"
#include <time.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>

#define compName     "DDSTest"
#define domainId     1
#define topicName    "myDataTest"
using namespace std;


int main()
{
    _RETURNCODE_T ret;

    //应用组件名为DDSTest加入到数据域的初始化过程
    ret = DomainInit(domainId,compName);

    if (ret != RETURNCODE_OK) //出错处理
    {
        cout<<"Domain: "<<domainId <<" Init failed"<<endl;
        return 0;
    }

    //应用组件QoS设置
    _DATA_WRITER_QOS *qos = new _DATA_WRITER_QOS();

    qos->Reliability.Kind = BEST_EFFORT;


    //创建发布者
    DataWriter * dataWriter = NULL;
    dataWriter = CreateDataWriter(compName,domainId, topicName, "myData", NULL, qos);
    if (dataWriter == NULL)
    {
        cout<<"Create Data Writer failed"<<endl;
        DomainRelease(domainId);
        return 0;
    }

    //安全的类型转换
```

```
myDataDataWriter *myDataTypeDataWriter = NULL;
myDataTypeDataWriter = myDataDataWriter::Narrow(dataWriter);

if (myDataTypeDataWriter == NULL)
{
        cout<<"myDataDataWriter Narrow failed"<<endl;
        DeleteDataWriter(dataWriter);
        DomainRelease(domainId);
        return 0;
}

myData data;

data.color = new char[30];

cout<<"Please enter the point and color you want to send,such as 3 4 red"<<endl;
while(1)
{
        cin>>data.x>>data.y>>data.color;

        //将用户输入的数据调用Write接口发送
        ret = myDataTypeDataWriter->Write(data);

        //返回发送结果
        if (ret != RETURNCODE_OK)
        {
                cout <<"Write error: " << ret << endl;
        }
}

//安全删除dataWriter
DeleteDataWriter(dataWriter);

//安全退出参与数据域
DomainRelease(domainId);

return 0;
}
```

► 订阅端示例程序

```
/*************************************************************
```

```
*****************订阅端程序subscriber.cpp *****************
************************************************************/
#include "subscriber.h"

_RETURNCODE_T HelloListener::On_Data_Available(DataReader* dataReader)
{
	_RETURNCODE_T ret = RETURNCODE_OK;

	myDataDataReader* myDataReader = NULL;
	myDataReader = myDataDataReader::Narrow(dataReader);

	if (myDataReader == NULL)
	{
		cout << "Narrow failed at On_Data_Available of HelloListener" << endl;
		return -1;
	}

	myData data;

	ret = myDataReader->Read_Next_Sample(data);

	if (ret != RETURNCODE_OK)
	{
		return -1;
	}

	cout<<"Received "<< data.x<<" "<<data.y<<" "<<data.color<<endl;

	return RETURNCODE_OK;
}


int main()
{
	_RETURNCODE_T ret;
	DataReader *dataReader = NULL;

	ret = DomainInit(domainId,compName);

	if (ret != RETURNCODE_OK)
	{
		cout<<"Domain:"<<domainId<<" Init failed"<<endl;
		return 0;
	}
```

```
    _DATA_READER_QOS *qos = new _DATA_READER_QOS();

    qos->Reliability.Kind = BEST_EFFORT;

    HelloListener *listener = new HelloListener(0);

    dataReader = CreateDataReader(compName, domainId, topicName, "myData", listener, qos);


    if (dataReader == NULL)
    {
        cout<<"Create Data Reader failed"<<endl;
        DomainRelease(domainId);
        return 0;
    }

    cout<<"Press any button to exit..."<<endl;
    int a;
    cin >> a;

    return 0;
}

/************************************************************
*****************订阅端程序subscriber.h ******************
************************************************************/
#ifndef _SUBSCRIBER_H
#define _SUBSCRIBER_H

#include "dds_api.h"
#include <time.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>

#define compName     "DDSTest"
#define domainId     1
#define topicName    "myDataTest"
using namespace std;

class HelloListener : public DataReaderListener
{
```

```
public:
    HelloListener(long num)
    {
        receiveCount = num;
    }

    _RETURNCODE_T On_Data_Available(DataReader* dataReader);

    long receiveCount;

};
#endif
```

地址：北京市海淀区丰贤东路 1 号
邮编：100094
传真：010-59516800
联系人：杨猛（18911990524）