# DDS接口文档

|  |  |
| --- | --- |
| 函数名 | DDS\_ParticipantFactory\_init |
| 功能描述 | 资源初始化 |
| 输入参数 | 无 |
| 返回 | VOID |
| 注释 | 进程内只需要初始化一次 |

|  |  |
| --- | --- |
| 函数名 | DDS\_DomainParticipantFactory\_get\_defaul\_participant\_qos |
| 功能描述 | 获取域参与者默认qos |
| 输入参数 | DDS\_DomainParticipantQos\* 参与者qos指针 |
| 返回 | DDS\_ReturnCode\_t 返回码：成功|失败 |
| 注释 | 需要先定义参与者qos变量，传入该地址 |

|  |  |
| --- | --- |
| 函数名 | DDS\_Participant\_bind\_network |
| 功能描述 | 域参与者绑定网卡，选择指定IP网卡通信 |
| 输入参数 | DDS\_DomainParticipantQos\* pstDomPartQos/参与者qos指针；const CHAR\* IP地址，不调用此函数即默认使用第一个网卡 |
| 返回 | VOID |
| 注释 | 以太网中使用，RAPDIO中该接口无效 |

|  |  |
| --- | --- |
| 函数名 | DDS\_DomainParticipantFactory\_create\_participant |
| 功能描述 | 创建参与者 |
| 输入参数 | SHORT域ID，不同的域ID通信隔离；DDS\_DomainParticipantQos\* 参与者qos指针，为NULL即默认qos |
| 返回 | DDS\_DomainParticipant\* 参与者指针 |
| 注释 | 创建失败返回NULL |

|  |  |
| --- | --- |
| 函数名 | DDS\_DomainParticipant\_create\_topic |
| 功能描述 | 创建主题 |
| 输入参数 | DDS\_DomainParticipant\* 参与者指针；const CHAR\* 主题名；const CHAR\* 主题类型；DDS\_TopicQos\* 主题qos指针，为NULL默认qos |
| 返回 | DDS\_Topic\* 主题指针 |
| 注释 | 创建失败返回NULL，同一个参与者创建多个不同主题名的Topic，相同主题名会失败。 |

|  |  |
| --- | --- |
| 函数名 | DDS\_Topic\_get\_default\_datawriter\_qos |
| 功能描述 | 获取默认写入器qos |
| 输入参数 | DDS\_Topic\* 主题指针；DDS\_DataWriterQos\* 写入器qos指针 |
| 返回 | DDS\_ReturnCode\_t返回码 |
| 注释 | 获取失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_Topic\_create\_datawriter |
| 功能描述 | 创建本地写入器 |
| 输入参数 | DDS\_Topic\* 主题指针；DDS\_DataWriterQos\* 写入器qos指针，为NULL默认qos；CALLBACKTIME 用户发送时间戳回调函数，为NULL默认使用系统时间函数 |
| 返回 | DDS\_DataWriter\* 写入器指针 |
| 注释 | 创建失败返回NULL |

|  |  |
| --- | --- |
| 函数名 | DDS\_Topic\_get\_default\_datareader\_qos |
| 功能描述 | 获取默认阅读器qos |
| 输入参数 | DDS\_Topic\* 主题指针；DDS\_DataReaderQos \* 阅读器qos指针 |
| 返回 | DDS\_ReturnCode\_t返回码 |
| 注释 | 获取失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_Topic\_create\_datareader |
| 功能描述 | 创建本地阅读器 |
| 输入参数 | DDS\_Topic\* 主题指针；DDS\_DataReaderQos \* 阅读器qos指针，为NULL默认qos；CALLBACKFUN 回调函数，不能为NULL。 |
| 返回 | DDS\_DataReader\* 阅读器指针 |
| 注释 | 创建失败返回NULL |

|  |  |
| --- | --- |
| 函数名 | DDS\_StringDataWriter\_write |
| 功能描述 | 写入器发送数据 |
| 输入参数 | DDS\_DataWriter\* 写入器指针；const CHAR\* 用户数据；UINT32 数据长度 |
| 返回 | DDS\_ReturnCode\_t 返回码 |
| 注释 | 发送失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_StringDataReader\_take |
| 功能描述 | 阅读器接收数据 |
| 输入参数 | DDS\_DataReader\* 阅读器指针；DataSeq 数据包 |
| 返回 | DDS\_ReturnCode\_t 返回码 |
| 注释 | 发送失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_StringSeq\_get |
| 功能描述 | 接收数据解析 |
| 输入参数 | DataSeq\* 数据包；UINT32 数据包编号；DataLen 数据长度 |
| 返回 | CHAR\* 数据首地址指针 |
| 注释 | 失败返回NULL |

|  |  |
| --- | --- |
| 函数名 | DDS\_TimeSeq\_get |
| 功能描述 | 接收数据时间戳解析 |
| 输入参数 | DataSeq\* 数据包；UINT32 数据包编号；Time\_t\*时间戳 |
| 返回 | 布尔 |
| 注释 | 失败返回FALSE |

|  |  |
| --- | --- |
| 函数名 | DDS\_Topic\_delete\_datareader |
| 功能描述 | 删除指定主题下的本地阅读器 |
| 输入参数 | DDS\_Topic\* 主题指针 |
| 返回 | DDS\_ReturnCode\_t 返回码 |
| 注释 | 失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_Topic\_delete\_datawriter |
| 功能描述 | 删除指定主题下的本地写入器 |
| 输入参数 | DDS\_Topic\* 主题指针 |
| 返回 | DDS\_ReturnCode\_t 返回码 |
| 注释 | 失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_DomainParticipant\_delete\_topic |
| 功能描述 | 删除指定参与者下的主题 |
| 输入参数 | DDS\_DomainParticipant\* 参与者指针；DDS\_Topic\* 主题指针 |
| 返回 | DDS\_ReturnCode\_t 返回码 |
| 注释 | 失败返回error |

|  |  |
| --- | --- |
| 函数名 | DDS\_DomainParticipantFactory\_delete\_participant |
| 功能描述 | 删除参与者 |
| 输入参数 | DDS\_DomainParticipant\* 参与者指针 |
| 返回 | DDS\_ReturnCode\_t 返回码 |
| 注释 | 失败返回error |

示例：

/\*---------------------------------------------------------------------------------

- 函 数 名: GetWinTime

- 功能描述: 用户自定义写数据时间戳

- 输 入: 时间类型结构体

- 返 回:

- 全局变量:

- 注 释: 用户自定义写数据时间戳

- 修改记录:

- 修改日期 版本 修改人 修改原因及内容

==================================================================================

- 2019\8\31 1.0 Hunter Created

----------------------------------------------------------------------------------\*/

VOID STACALL GetWinTime(Time\_t\* pstTimeInfo)

{

pstTimeInfo->sec = time(NULL);

}

/\*---------------------------------------------------------------------------------

- 函 数 名: ReceiveData

- 功能描述: 接收数据回调函数

- 输 入: 本地阅读器

- 返 回:

- 全局变量:

- 注 释: 接收处理用户数据报文

- 修改记录:

- 修改日期 版本 修改人 修改原因及内容

==================================================================================

- 2019\8\31 1.0 Hunter Created

----------------------------------------------------------------------------------\*/

VOID STACALL ReceiveData(DDS\_DataReader\* pstDataReader)

{

UINT32 dataNum = 0;

DataSeq stDataSeq;

DataLen dataLen;

Time\_t timeInfo;

char timeStr[32];

/\* 收数据 \*/

DDS\_ReturnCode\_t retCode = DDS\_StringDataReader\_take(pstDataReader, &stDataSeq);

if (DDS\_RETCODE\_NO\_DATA != retCode)

{

for (dataNum = 0; dataNum < stDataSeq.length; dataNum++)

{

DDS\_TimeSeq\_get(&stDataSeq, dataNum, &timeInfo);

TimeToStr(timeInfo.sec, "%Y-%m-%d %H:%M:%S", timeStr);

printf("TIME:%s, MSG: %s \n", timeStr, DDS\_StringSeq\_get(&stDataSeq, dataNum, &dataLen));

}

/\* 释放缓存 \*/

DDS\_StringDataReader\_return\_loan(pstDataReader, &stDataSeq);

}

}

/\*---------------------------------------------------------------------------------

- 函 数 名: main

- 功能描述: 主函数

- 输 入:

- 返 回:

- 全局变量:

- 注 释: 测试数据收发

- 修改记录:

- 修改日期 版本 修改人 修改原因及内容

==================================================================================

- 2019\8\31 1.0 Hunter Created

----------------------------------------------------------------------------------\*/

void main()

{

UINT32 i = 0;

CHAR\* buff = NULL;

UINT32 uiLen = 256;

DDS\_ReturnCode\_t retureCode;

DDS\_DomainParticipantQos stDomPartQos;

DDS\_TopicQos stTopicQos;

DDS\_DataWriterQos stWriterQos;

DDS\_DataReaderQos stReaderQos;

/\* 资源初始化 \*/

DDS\_ParticipantFactory\_init();

/\* 获取参与者qos \*/

retureCode = DDS\_DomainParticipantFactory\_get\_defaul\_participant\_qos(&stDomPartQos);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Data writer setup error !");

return;

}

/\* 绑定网卡，不设置默认第一个网卡 \*/

DDS\_Participant\_bind\_network(&stDomPartQos, "192.168.100.118");

/\* 创建参与者 \*/

DDS\_DomainParticipant\* pstParticipant = DDS\_DomainParticipantFactory\_create\_participant(78, &stDomPartQos);

if (NULL == pstParticipant)

{

printf("ERROR! Failed to create participant!");

return;

}

/\* 创建主题 \*/

DDS\_Topic\* pstTopic = DDS\_DomainParticipant\_create\_topic(pstParticipant, "HELLO", "HELLO", &stTopicQos);

if (NULL == pstTopic)

{

printf("ERROR! Failed to create topic!");

return;

}

/\* 获取写入器qos \*/

retureCode = DDS\_Topic\_get\_default\_datawriter\_qos(pstTopic, &stWriterQos);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Data writer setup error !");

return;

}

/\* 设置写入器可靠性 \*/

stWriterQos.reliability.kind = BEST\_EFFORT\_RELIABILITY\_QOS;

stWriterQos.reliability.kind = RELIABLE\_RELIABILITY\_QOS;

/\* 创建写入器 \*/

DDS\_DataWriter\* pstDataWriter = DDS\_Topic\_create\_datawriter(pstTopic, &stWriterQos, GetWinTime);

if (NULL == pstDataWriter)

{

printf("ERROR! Failed to create dataWriter!");

return;

}

/\* 获取阅读器qos \*/

retureCode = DDS\_Topic\_get\_default\_datareader\_qos(pstTopic, &stReaderQos);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Data reader setup error !");

return;

}

/\* 设置阅读器可靠性 \*/

stReaderQos.reliability.kind = BEST\_EFFORT\_RELIABILITY\_QOS;

stReaderQos.reliability.kind = RELIABLE\_RELIABILITY\_QOS;

/\* 创建阅读器 \*/

DDS\_DataReader\* pstDataReader = DDS\_Topic\_create\_datareader(pstTopic, &stReaderQos, ReceiveData);

if (NULL == pstDataReader)

{

printf("ERROR! Failed to create dataReader!");

return;

}

/\* 内存监控 \*/

DDS\_MemoryUsage();

/\* 待发送数据内存分配 \*/

buff = (CHAR\*)DDS\_STATIC\_MALLOC(uiLen);

if (NULL == buff)

{

printf("ERROR! Insufficient memory space.\n");

return;

}

for (i = 0; i < 5; i++)

{

sprintf\_s(buff, uiLen, "Hello World from bao ! (%d)\n", i);

/\* 发数据 \*/

retureCode = DDS\_StringDataWriter\_write(pstDataWriter, buff, uiLen);

if (DDS\_RETCODE\_ERROR == retureCode)

{

/\* 内存监控 \*/

DDS\_MemoryUsage();

printf("ERROR! Failed to send message, Insufficient memory space!\n");

}

Sleep(1000);

}

DDS\_STATIC\_FREE(buff);

/\* 删除阅读器 \*/

retureCode = DDS\_Topic\_delete\_datareader(pstTopic);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Failed to delete dataReader!");

return;

}

printf("OK! successfully delete dataReader!\n");

Sleep(1000);

/\* 删除写入器 \*/

retureCode = DDS\_Topic\_delete\_datawriter(pstTopic);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Failed to delete dataWriter!\n");

return;

}

printf("OK! successfully delete dataWriter!\n");

Sleep(1000);

/\* 删除主题 \*/

retureCode = DDS\_DomainParticipant\_delete\_topic(pstParticipant, pstTopic);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Failed to delete topic!\n");

return;

}

printf("OK! successfully delete topic!\n");

Sleep(2000);

/\* 删除参与者 \*/

retureCode = DDS\_DomainParticipantFactory\_delete\_participant(pstParticipant);

if (DDS\_RETCODE\_ERROR == retureCode)

{

printf("ERROR! Failed to delete participant!\n");

return;

}

printf("OK! successfully delete participant!\n");