**DATA STUDIO框架使用手册**

V1.00

目录

[1 概述 5](#_Toc517685511)

[2 核心部件 5](#_Toc517685512)

[2.1 通道传输部件（comm） 5](#_Toc517685513)

[2.1.1 结构 5](#_Toc517685514)

[2.1.2 模块 6](#_Toc517685515)

[2.1.3 用例 10](#_Toc517685516)

[2.2 协议结构部件（proto） 13](#_Toc517685517)

[2.2.1 结构 13](#_Toc517685518)

[2.2.2 模块 15](#_Toc517685519)

[2.2.3 用例 31](#_Toc517685520)

[2.3 协议解析部件（parser） 31](#_Toc517685521)

[2.3.1 结构 31](#_Toc517685522)

[2.3.2 模块 31](#_Toc517685523)

[2.3.3 用例 31](#_Toc517685524)

[2.4 数据传输部件（worker） 32](#_Toc517685525)

[2.4.1 结构 32](#_Toc517685526)

[2.4.2 模块 32](#_Toc517685527)

[2.4.3 用例 32](#_Toc517685528)

[2.5 窗口部件部件（widget） 32](#_Toc517685529)

[2.5.1 结构 32](#_Toc517685530)

[2.5.2 模块 32](#_Toc517685531)

[2.5.3 用例 33](#_Toc517685532)

[2.6 协议图表部件（chart） 33](#_Toc517685533)

[2.6.1 结构 33](#_Toc517685534)

[2.6.2 模块 33](#_Toc517685535)

[2.6.3 用例 33](#_Toc517685536)

[2.7 QML元子部件（meta） 33](#_Toc517685537)

[2.7.1 结构 33](#_Toc517685538)

[2.7.2 模块 34](#_Toc517685539)

[2.7.3 用例 34](#_Toc517685540)

[2.8 QML插件部件（plugins） 34](#_Toc517685541)

[2.8.1 结构 34](#_Toc517685542)

[2.8.2 模块 34](#_Toc517685543)

[2.8.3 用例 34](#_Toc517685544)

[2.9 其他部件 35](#_Toc517685545)

[2.9.1 jsoncpp 35](#_Toc517685546)

[3 Qbs编译库 35](#_Toc517685547)

[3.1 模板库 35](#_Toc517685548)

[3.1.1 Imports 35](#_Toc517685549)

[3.1.2 modules 36](#_Toc517685550)

[3.2 工程模板 37](#_Toc517685551)

[3.3 预编译 37](#_Toc517685552)

[3.4 依赖库导入 37](#_Toc517685553)

[3.4.1 3rdpart依赖库 38](#_Toc517685554)

[3.4.2 core依赖库 38](#_Toc517685555)

[3.5 3rdpart库编译 38](#_Toc517685556)

[3.6 core库编译 38](#_Toc517685557)

[3.7 app编译 38](#_Toc517685558)

[3.7.1 QtWidget App编译 38](#_Toc517685559)

[3.7.2 QtQuick App编译 39](#_Toc517685560)

[3.8 setup工程生成 39](#_Toc517685561)

[3.9 setup工程编译 39](#_Toc517685562)

[3.10 setenv编译 39](#_Toc517685563)

[4 用例说明 39](#_Toc517685564)

# 概述

本文档用于描述Data Studio框架核心组件、Qbs编译、框架用例等的功能和使用说明。

# 核心部件

核心组件包括通讯、协议封装和解析、数据传输处理、自定义窗口部件、协议图表控件、核心组件的QML包装、核心组件的QML插件等模块，是Data Studio框架软件的核心组成部分，涵盖大部分数据处理、UI相关二次开发需要的基础库。

## 通道传输部件（comm）

通道传输部件用于抽象文件、串口、网络等设备控制接口，实现数据传输功能。

### 结构

通道传输部件包括文件、UDP、串口三个模块，抽象为通道接口，支持通道的打开、关闭、发送、接收、属性读写等功能。

通道传输部件UML结构如图 1所示。



图 1通道传输部件UML关系

### 模块

公共属性、接口描述如表 2‑1所示。

表 2‑1公共接口、属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| channelType | 属性 | 通道类型标识 |
| config | 属性 | 通道配置信息 |
| identity | 属性 | 通道唯一标识号 |
| name | 属性 | 通道名称 |
| desc | 属性 | 通道描述信息 |
| relayer | 属性 | 转发通道 |
| lastError | 属性 | 最后运行错误信息 |
| lastErrorCode | 属性 | 最后运行错误编码 |
| sizeOfIn | 属性 | 输入缓冲区数据大小 |
| sizeOfOut | 属性 | 输出缓冲区数据大小 |
| open | 接口 | 打开通道 |
| close | 接口 | 关闭通道 |
| waitForReadyRead | 接口 | （阻塞）等待数据可读 |
| flush | 接口 | 刷新输出端数据到目标机 |

#### 文件通道

文件通道属性、接口描述如表 2‑2所示。

表 2‑2文件通道属性、接口

|  |  |  |
| --- | --- | --- |
| 表名称 | 类型 | 说明 |
| oepnMode | 属性 | 打开模式：  NotOpen（0）：不打开；  ReadOnly（1）：只读模式；  WriteOnly（2）：只写模式；  ReadWrite（3）：读写模式，默认项；  Append（4）：追加模式；  Truncate（8）：截断模式；  Text（16）：文本模式； |
| saveFormat | 属性 | 存储格式：  SaveFormatDomain（1）：包含表域名；  SaveFormatTimestamp（2）：包含时间戳；  SaveFormatNormal（3）：包含表域名和时间戳，默认项。 |
| filePath | 属性 | 文件名称（绝对路径/相对路径） |
| fileHeader | 属性 | 文件头。数据文件标识头，默认为“ICD\0”。 |
| domain | 属性 | 表域名。数据协议中的表域名，格式为：机型标识/系统标识/表标识。 |
| config | 属性 | 文件通道配置信息。  格式：“file: --filePath=x --openMode=[r][w] --id=x --name=x”。 |

#### 串口通道

串口通道属性、接口描述如表 2‑3所示。

表 2‑3串口通道属性、接口

|  |  |  |
| --- | --- | --- |
| 表名称 | 类型 | 说明 |
| portName | 属性 | 通道号。格式：COM{nnn}，其中，{nnn}为具体端口编号。 |
| baudRate | 属性 | 通道波特率。 |
| dataBits | 属性 | 通道数据位：  Data5（5）：5位；  Data6（6）：6位；  Data7（7）：7位；  Data8（8）：8位，默认项；  UnknownDataBits（-1）：未知。 |
| stopBits | 属性 | 通道停止位：  OneStop（1）：1位，默认项；  OneAndHalfStop（3）：1.5位；  TwoStop（2）：2位；  UnknownStopBits（-1）：未知。 |
| parity | 属性 | 通道校验位：  NoParity（0）：无校验，默认项；  EvenParity（2）：偶校验；  OddParity（3）：奇校验；  SpaceParity（4）：Space校验；  MarkParity（5）：Mark校验；  UnknownParity（-1）：未知。 |
| config | 属性 | 文件通道配置信息。  格式：“serial: --port=x --baudRate=x --dataRate=x --stopBits=x --parity=x --id=x --name=x”。 |

#### UDP通道

UDP通道属性、接口描述如表 2‑4所示。

表 2‑4UDP通道属性、接口

|  |  |  |
| --- | --- | --- |
| 表名称 | 类型 | 说明 |
| oepnMode | 属性 | 打开模式：  NotOpen（0）：不打开；  ReadOnly（1）：只读模式；  WriteOnly（2）：只写模式；  ReadWrite（3）：读写模式，默认项。 |
| localIP | 属性 | 本地地址 |
| localPort | 属性 | 本地端口 |
| remoteIP | 属性 | 目标地址 |
| remotePort | 属性 | 目标端口 |
| config | 属性 | 文件通道配置信息。  格式：“udp: --local=x --remote=x --openMode=[r][w] --id=xxx --name=xxx”。 |

### 用例

基本操作描述如表 2‑5所示。

表 2‑5通道基本操作

|  |  |
| --- | --- |
| 操作 | 用例 |
| 创建通道 | 1）类型方式创建：  Icd::Channel::createInstance(Icd::ChannelSerial);  2）配置信息方式创建：  // 创建串口通道。{xxx}为通道标识号  std::string config(“--portName=COM1 –baudRate=9600 --dataBits=8 --stopBits=1 --parity=even --id={xxx} --name=channel1”);  Icd::Channel::createInstance(config); |
| 开、关 | channel->open();  channel->close(); |
| 读写数据 | const buffer[1024];  channel->read(buffer, 1024); |
| 转发通道 | 提供数据转发功能，将本通道收到的数据发送给转发通道。  设置方式有两种：   1. 直接调用setRelayer接口，进行设置； 2. 通过在配置信息提供转发通道的标识号（id）进行设置，如：   // --relayer={xxx}部分为转发通道标识号属性  std::string config(“--portName=COM1 –baudRate=9600 --dataBits=8 --stopBits=1 --parity=even --id={xxx} --name=channel1 --relayer={xxx}”);  channel->setConfig(config); |

#### 文件通道

文件通道操作描述如表 2‑6所示。

表 2‑6文件通道操作

|  |  |
| --- | --- |
| 操作 | 用例 |
| 创建通道 | 1）类型方式创建：  Icd::Channel::createInstance(Icd::ChannelFile);  2）配置信息方式创建：  // 创建文件通道。{xxx}为通道标识号  std::string config(“--filePath=C:/folder1 --openMode=rw –saveFormat=dt --id={xxx} --name=channel1”);  Icd::Channel::createInstance(config); |
| 设置文件头 | // 设置文件头信息为：“ICD\0”  channel->setFileHeader(0x00444349U); |
| 设置表域名 | // 设置表域名为：机型标识/系统标识/表标识,{xxx}为GUID格式  channel->setDomain(“1001/2001/ICDTable\_{xxx}”); |

#### 串口通道

串口通道操作描述如表 2‑7所示。

表 2‑7串口通道操作

|  |  |
| --- | --- |
| 操作 | 用例 |
| 创建通道 | 1）类型方式创建：  Icd::Channel::createInstance(Icd::ChannelSerial);  2）配置信息方式创建：  // 创建文件通道。{xxx}为通道标识号  std::string config(“--filePath=C:/folder1 --openMode=rw –saveFormat=dt --id={xxx} --name=channel1”);  Icd::Channel::createInstance(config); |
| 设置文件头 | // 设置文件头信息为：“ICD\0”  channel->setFileHeader(0x00444349U); |
| 设置表域名 | // 设置表域名为：机型标识/系统标识/表标识,{xxx}为GUID格式  channel->setDomain(“1001/2001/ICDTable\_{xxx}”); |

#### UDP通道

UDP通道操作描述如表 2‑8所示。

表 2‑8UDP通道操作

|  |  |
| --- | --- |
| 操作 | 用例 |
| 创建通道 | 1）类型方式创建：  Icd::Channel::createInstance(Icd::ChannelUdp);  2）配置信息方式创建：  // 创建文件通道。{xxx}为通道标识号  std::string config(“--local=127.0.0.1:1001 --remote=127.0.0.1:1002 --openMode=rw --id={xxx} --name=channel1”);  Icd::Channel::createInstance(config); |

## 协议结构部件（proto）

协议结构部件用于包装各类常见协议为对象，主要功能如下：

1. 实现一系列相关操作，通过内存映射的方式实现数据的快速注入和取出功能，
2. 提供一系列各类数据的类型自动转换属性和接口实现数据便捷的可视化处理，
3. 提供数据协议对象序列化和反序列化接口功能，实现数据结构的快速导入和导出功能。

### 结构

协议结构部件主要数据类型如表 2‑9所示。

表 2‑9协议结构部件数据类型

|  |  |
| --- | --- |
| 类型 | 描述 |
| 根类型（root） | 协议结构的根节点（无实际意义，主要用于可视化处理）（可选） |
| 机型（vehicle） | 机型类型数据，协议结构实际根节点，挂在根节点下（可选，可多个） |
| 系统（system） | 系统类型，挂在机型节点下（可多个） |
| 边界（limit） | 边界类型，数值类型属性，表示数值默认范围 |
| 表（table） | 表类型，挂在系统、复合、帧等类型节点下 |
| 帧头（header） | 枕头类型，挂在表类型下（可多个） |
| 计数器（counter） | 计数器类型，挂在表类型下（一个） |
| 校验（check） | 校验类型，挂在表类型下（一个） |
| 帧识别码（framecode） | 帧识别码，挂在表类型下。可绑定一个帧类型（可选，可多个） |
| 数值（numeric） | 数值类型，挂在表类型下（可多个） |
| 数组（array） | 数组类型，挂在表类型下（可多个） |
| 时间（datetime） | 时间类型，挂在表类型下（可多个） |
| 比特映射（bitmap） | 比特映射类型，挂在表类型下（可多个） |
| 比特值（bitvalue） | 比特值类型，挂在表类型下（可多个） |
| 复合（complex） | 复合类型，挂在表类型下（可多个）。子节点为表类型（一个） |
| 子帧（frame） | 子帧类型，挂在表类型下（可多个）。子节点为表类型（可多个） |

协议结构部件UML结构如图 2所示。



图 2协议结构UML关系

### 模块

#### Object

Object型属性、接口描述如表 2‑10所示。

表 2‑10Object型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| objectType | 属性 | 元对象类型：   * ObjectInvalid（-1）：无效； * ObjectRoot（0）：Root类型； * ObjectVehicle（1）：Vehicle类型； * ObjectSystem（2）：System类型； * ObjectTable（3）：Table类型； * ObjectItem（4）：Item类型； * ObjectLimit（5）：Limit类型。 |
| id | 属性 | 唯一标识号 |
| domain | 属性 | 表域名。主要用于存储表类型域名，格式为：机型标识/系统标识/表标识，在文件通道模块中用于查询表域名。 |
| name | 属性 | 数名称 |
| mark | 属性 | 标识码（符合程序变量名格式） |
| desc | 属性 | 描述信息 |
| parent | 属性 | 父节点 |
| childCount | 属性 | 子节点个数 |
| objectTypeString | 接口 | 对象类型字符串化 |
| domainOfType | 接口 | 根据给定域名类型返回相应属性值。用于根据域名查找数据项的模块  域名类型有：   * DomainInvalid（-1）：无效； * DomainId（0）：标识号型域名； * DomainName（1）：名称型域名； * DomainMark（2）：标识码型域名。 |
| resetData | 接口 | 复位各数据项中的数据缓冲区为给定的默认值。 |
| clearData | 接口 | 将映射缓冲区内存数据复位为0。主要用于：当需要接收发送数据后的响应数据时，需要在接收前清空接收缓存数据（因为协接收端议映射缓冲区只有一个） |
| clone | 接口 | （深度）克隆操作 |
| operater= | 接口 | 赋值操作 |
| save/restore | 接口 | 对象序列化/反序列化 |
| typeName | 接口 | 对象类型名称 |
| codeName | 接口 | 对象标识码 |

#### Limit

继承自Object型。

Limit口描述如表 2‑11所示。

表 2‑11Limit型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| minimum | 属性 | 数值下限 |
| maximum | 属性 | 数值上线 |
| leftInf | 属性 | 下限有效性。true表示有下限 |
| rightInf | 属性 | 上限有效性。true表示有上限 |
| toString | 接口 | 获取数值范围信息的字符串形式 |

#### Root

继承自Object型。

Root型属性、接口描述如表 2‑12所示。

表 2‑12Root型数据、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| allVehicle | 属性 | 获取所有机型项 |
| appendVehicle | 接口 | 追加一个机型项 |
| insertVehicle | 接口 | 在给定位置插入一个机型项 |
| removeVehicle | 接口 | 删除给定位置的机型项 |
| removeVehicleByMark | 接口 | 删除给定标识码的机型项 |
| clearVehicle | 接口 | 清空所有机型项 |
| vehicleCount | 接口 | 获取机型项个数 |
| vehicleAt | 接口 | 获取给定位置的机型项 |
| vehicleByMark | 接口 | 获取给定标识码的机型项 |

#### Vehicle

继承自Vehicle型。

Vehicle型属性、接口描述如表 2‑13所示。

表 2‑13Vehicle型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| allSystem | 属性 | 获取所有系统项 |
| appendSystem | 接口 | 追加一个系统项 |
| insertSystem | 接口 | 在给定位置插入一个系统项 |
| removeSystem | 接口 | 删除给定位置的系统项 |
| removeSystemByMark | 接口 | 删除给定标识码的系统项 |
| clearSystem | 接口 | 清空所有系统项 |
| systemCount | 接口 | 获取系统项个数 |
| systemAt | 接口 | 获取给定位置的系统项 |
| systemByMark | 接口 | 获取给定标识码的系统项 |

#### System

继承自System型。

System型属性、接口描述如表 2‑14所示。

表 2‑14System型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| allTable | 属性 | 获取所有表项 |
| appendTable | 接口 | 追加一个表项 |
| insertTable | 接口 | 在给定位置插入一个表项 |
| removeTable | 接口 | 删除给定位置的表项 |
| removeTableByMark | 接口 | 删除给定标识码的表项 |
| clearTable | 接口 | 清空所有表项 |
| tableCount | 接口 | 获取表项个数 |
| tableAt | 接口 | 获取给定位置的表项 |
| tableByMark | 接口 | 获取给定标识码的表项 |

#### Table

继承自Object型。

Table型属性、接口描述如表 2‑15所示。

表 2‑15Table型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| itemOffset | 属性 | 位置偏移量（即在System型的表项队列中的位置，起始位为0） |
| bufferSize | 属性 | 映射内存占用大小 |
| bufferOffset | 属性 | 映射内存偏移量 |
| buffer | 属性 | 映射内存块 |
| sequence | 属性 | 表项时序（在System型中的表项的收发时序比例） |
| period | 属性 | 表项传输周期（单位：ms） |
| allItem | 属性 | 获取所有数据项 |
| appendItem | 接口 | 追加一个数据项 |
| appendItem | 接口 | 追加一个数据项 |
| insertItem | 接口 | 在给定位置插入一个数据项 |
| removeItem | 接口 | 删除给定位置的数据项 |
| removeItemById | 接口 | 删除给定标识号的数据项 |
| removeItemByMark | 接口 | 删除给定标识码的数据项 |
| clearItem | 接口 | 清空所有数据项 |
| itemCount | 接口 | 获取数据项个数 |
| itemAt | 接口 | 获取给定位置的数据项 |
| itemById | 接口 | 获取给定标识号的数据项 |
| itemByMark | 接口 | 获取给定标识码的数据项（当deep为true时，进行深度搜索） |
| tableByMark | 接口 | 获取给定标识码的表项（当deep为true时，进行深度搜索） |
| itemByDomain | 接口 | 获取给定域名的数据项（并且指定域名类型，默认为标识号型域名） |
| tableByDomain | 接口 | 获取给定域名的表项（并且指定域名类型，默认为标识号型域名） |
| isFrameTable | 接口 | 查询表项是否为子帧中的表项 |
| header | 属性 | 获取所有帧头项 |
| counterItem | 属性 | 获取计数器项 |
| isCheckValid | 属性 | 查询校验项有效性 |
| checkItem | 属性 | 获取校验项 |
| allFrameCode | 属性 | 获取所有帧码项 |
| updateSend | 接口 | 更新所有子帧数据。如果period为true，则循环更新子帧队列；否则只更新当前子帧数据。 |
| updateRecv | 接口 | 更新当前所有子帧数据 |
| resetSend | 接口 | 复位所有映射内存数据为给定默认值 |

#### Item

继承自Object型。

Item型属性、接口描述如表 2‑16所示。

表 2‑16Item型数据、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| itemOffset | 属性 | 位置偏移量（即在Table型的表项队列中的位置，起始位为0） |
| bufferSize | 属性 | 映射内存占用大小 |
| bufferOffset | 属性 | 映射内存偏移量 |
| buffer | 属性 | 映射内存块 |
| data | 属性 | 存取数据值（双精度型） |
| dataString | 接口 | 获取数据值的字符串形式 |
| defaultValue | 属性 | 存取数据默认值 |
| typeString | 接口 | 获取数据项类型的字符串形式 |
| stringType | 接口 | 转换给定字符串为数据项类型枚举值 |

#### Header

继承自Item型。

Header型属性、接口描述如表 2‑17所示。

表 2‑17Header型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| value | 属性 | 获取Header型的unsigned char型数据 |

#### Counter

继承自Item型。

Counter型属性、接口描述如表 2‑18所示。

表 2‑18Counter型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| value | 属性 | 获取Header型的unsigned char型数据 |

#### Check

继承自Item型。

Check型属性、接口描述如表 2‑19所示。

表 2‑19Check型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| checkType | 属性 | 校验类型：   * CheckInvalid（-1）：无效； * CheckNone（0）：无校验； * CheckSum8（1）：8位和校验（给定起止字段所有字节和取低八位）； * CheckSum16（2）：16位和校验（给定起止字段所有字节和取低16位）； * CheckCrc8（3）：CRC8校验（给定起止字段所有字节进行CRC8校验）； * CheckCrc16（4）：CRC16位检验（给定起止字段所有字节进行CRC16校验）； * CheckXor8（5）：8位异或检验（给定起止字段所有字节进行8位异或累加取低8位）； * CheckXor16（6）：16位异或检验（给定起止字段所有字节进行16位异或累加取低16位）。 |
| startPos | 属性 | 起始字节偏移量（从0开始） |
| endPos | 属性 | 终止字节偏移量（从0开始） |
| checkLength | 属性 | 起止字节个数（endPos – startPos + 1） |

#### Numeric

继承自Item型。

Numeric型属性、接口描述如表 2‑20所示。

表 2‑20Numeric属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| numericType | 属性 | 校验类型：   * NumericInvalid（-1）：无效； * NumericI8（0）：8位有符号整型； * NumericU8（1）：8位无符号整型； * NumericI16（2）：16位有符号整型； * NumericU16（3）：16位无符号整型； * NumericI32（4）：32位有符号整型； * NumericU32（5）：32位无符号整型； * NumericI64（6）：64位有符号整型； * NumericU64（7）：64位无符号整型； * NumericF32（8）：32位浮点数； * NumericF64（9）：64位浮点数。 |
| originalData | 接口 | 获取原始数值 |
| scale | 属性 | 数值比例尺:  value=offset + data \* scale |
| decimas | 属性 | 小数有效位（主要用于显示、字符串化） |
| offset | 属性 | 数值偏移量:  value=offset + data \* scale |
| limit | 属性 | 数值上下限（具体参考表 2‑11） |
| unit | 属性 | 数值单位 |
| specs | 接口 | 获取所有特殊描述信息 |
| addSpec | 接口 | 增加一组特殊描述信息 |
| specAt | 接口 | 获取给定键的一组特殊描述信息 |
| removeSpec | 接口 | 删除给定键的一组特殊描述信息 |
| clearSpec | 接口 | 清空特殊描述信息 |
| dataRange | 接口 | 获取原始数据（不经过偏移量和比例尺计算）的上下限 |
| valueRange | 接口 | 获取数值按偏移量和比例尺计算出来的上下限 |
| outOfLimit | 接口 | 获取数值溢出（超限）状态 |

#### Array

继承自Item型。

Array型属性、接口描述如表 2‑21所示。

表 2‑21Array属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| arrayType | 属性 | 校验类型：   * ArrayInvalid（-1）：无效； * ArrayI8（0）：8位有符号整型； * ArrayU8（1）：8位无符号整型； * ArrayI16（2）：16位有符号整型； * ArrayU16（3）：16位无符号整型； * ArrayI32（4）：32位有符号整型； * ArrayU32（5）：32位无符号整型； * ArrayI64（6）：64位有符号整型； * ArrayU64（7）：64位无符号整型； * ArrayF32（8）：32位浮点数； * ArrayF64（9）：64位浮点数。 |
| count | 属性 | 数组元素个数 |
| typeSize | 接口 | 数组类型大小（字节数） |
| toString | 接口 | 获取字符串形式的缓冲区数据 |
| i8、u8、i16、u16、i32、u32、i64、u64、f32、f64 | 接口 | 获取对应数组类型的缓冲区数据（缓冲区首地址） |

#### BitMap/BitValue

继承自Item型。

BitMap/BitValue型属性、接口描述如表 2‑22所示。

表 2‑22BitMap/BitValue属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| testBit | 接口 | 测试给定比特位的状态（0为false，1为true） |
| bitStart | 属性 | 比特位起始偏移量 |
| bitCount | 属性 | 比特位个数 |
| typeSize | 属性 | 数据实际占用字节数 |
| mask | 属性 | 数据掩码：  ((1ull << bitCount) - 1) << bitStart |
| specs | 接口 | 获取所有特殊描述信息 |
| addSpec | 接口 | 增加一组特殊描述信息 |
| specAt | 接口 | 获取给定比特位的一组特殊描述信息 |
| removeSpec | 接口 | 删除给定比特位的一组特殊描述信息 |
| clearSpec | 接口 | 清空特殊描述信息 |
| nameAt | 接口 | 获取给定比特位的名称 |
| nameOf | 接口 | 获取给定特殊描述信息（包括名称）中的名称 |
| descOf | 接口 | 获取给定比特位的描述信息 |

#### DateTime

继承自Item型。

DateTime型属性、接口描述如表 2‑23所示。

表 2‑23DateTime属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
|  |  |  |

#### FrameCode

继承自Item型。

FrameCode型属性、接口描述如表 2‑24所示。

表 2‑24FrameCode型属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| frameCodeType | 属性 | 校验类型：   * FrameCodeInvalid（-1）：无效； * FrameCodeU8（0）：8位无符号帧码； * FrameCodeU16（1）：16位无符号帧码； * FrameCodeU32（2）：32位无符号帧码； * FrameCodeU64（3）：64位无符号帧码。 |
| frameId | 属性 | 绑定的帧型数据标识号 |
| frame | 属性 | 绑定/获取的帧数据 |
| updateSend | 接口 | 更新所有帧数据。如果period为true，则循环更新帧队列；否则只更新当前帧数据。 |
| updateRecv | 接口 | 更新当前所有帧数据 |
| resetSend | 接口 | 复位所有映射内存数据为给定默认值 |

#### Frame

继承自Item型。

Frame型属性、接口描述如表 2‑25所示。

表 2‑25Frame属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| allTable | 属性 | 获取所有表项 |
| addTable | 接口 | 增加一个表项 |
| removeTable | 接口 | 删除给定帧码的表项 |
| clearTable | 接口 | 清空所有表项 |
| replaceCode | 接口 | 替换帧码 |
| tableCount | 接口 | 获取表项个数 |
| tableAt | 接口 | 获取给定帧码的表项 |
| itemAt | 接口 | 获取给定位置的数据项 |
| itemByMark | 接口 | 获取给定标识号的数据项 |
| tableByMark | 接口 | 获取给定标识码的表项（当deep为true时，进行深度搜索） |
| itemByDomain | 接口 | 获取给定域名的数据项（并且指定域名类型，默认为标识号型域名） |
| tableByDomain | 接口 | 获取给定域名的表项（并且指定域名类型，默认为标识号型域名） |
| updateSend | 接口 | 更新所有帧数据。如果period为true，则循环更新帧队列；否则只更新当前帧数据。 |
| updateRecv | 接口 | 更新当前所有帧数据 |
| resetSend | 接口 | 复位所有映射内存数据为给定默认值 |

#### Complex

继承自Item型。

Complex型属性、接口描述如表 2‑26所示。

表 2‑26Complex属性、接口描述

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| setBuffer | 接口 | 设置缓冲区 |
| setBufferSize | 接口 | 设置缓冲区大小（字节） |
| childCount | 接口 | 获取子项个数（递归） |
| itemByMark | 接口 | 获取给定标识号的数据项 |
| tableByMark | 接口 | 获取给定标识码的表项（当deep为true时，进行深度搜索） |
| itemByDomain | 接口 | 获取给定域名的数据项（并且指定域名类型，默认为标识号型域名） |
| tableByDomain | 接口 | 获取给定域名的表项（并且指定域名类型，默认为标识号型域名） |
| updateSend | 接口 | 更新所有帧数据。如果period为true，则循环更新帧队列；否则只更新当前帧数据。 |
| updateRecv | 接口 | 更新当前所有帧数据 |
| resetSend | 接口 | 复位所有映射内存数据为给定默认值 |

### 用例

请参见章节2.3。

## 协议解析部件（parser）

### 结构

### 模块

### 用例

## 数据传输部件（worker）

### 结构

### 模块

### 用例

## 窗口部件部件（widget）

### 结构

### 模块

### 用例

## 协议图表部件（chart）

### 结构

### 模块

### 用例

## QML元子部件（meta）

### 结构

### 模块

### 用例

## QML插件部件（plugins）

### 结构

### 模块

### 用例

## 其他部件

### jsoncpp

jsoncpp模块位于icdcore/3rdpart/jsoncpp中。……

# Qbs编译库

## 模板库

### Imports

#### 基本库

#### 预编译

#### Node

#### tools/EnvUtils

### modules

#### 3rdpart

#### core

#### desc

#### Node

#### NodeAddon

#### Pack

## 工程模板

## 预编译

## 依赖库导入

### 3rdpart依赖库

### core依赖库

## 3rdpart库编译

## core库编译

## app编译

### QtWidget App编译

### QtQuick App编译

## setup工程生成

## setup工程编译

## setenv编译

# 用例说明