|  |
| --- |
|  |
| **信息平台研发** |
| **项目编号：2300-K1170024-01** |
| **数据总线用户手册** |
| **(版本号：V0.0.1)** |
|  |
|  |
| **2017年05月**  **内部资料 注意保密** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本记录** | | | | |
| **版本号** | **日期** | **修改章节** | **修改内容及说明** | **编制者** |
| V0.0.1 | 20170531 | 全部 | 创建文档 | 王华伟 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **评审单位** | **签字** | **日期** |
| **编制者：** |  |  |
| **审核者：** |  |  |
| **项目负责人：** |  |  |
| **承担单位总工程师：** |  |  |
| **技术管理中心：** |  |  |
| **公司总工程师：** |  |  |

**目 录**

1. 简介 5

1.1. 软件产品简介 5

1.2. 文档目的和范围 5

2. 软件介绍 6

2.1. 软件总体结构 6

2.2. 通信逻辑地址描述 7

2.3. 应用环境及条件 8

3. 总线服务配置 9

3.1. NetRouterConfigure.xml配置文件 9

3.1.1. 示例 9

3.1.2. 广域网配置详细说明 10

3.2. log4cplus.properties 12

4. 应用开发 13

4.1. 配置文件描述 13

4.1.1. Configure.xml 13

4.1.2. log4cplus.properties 14

4.2. Java适配 14

4.2.1. 数据结构描述 14

4.2.2. 接口描述 15

4.3. C++适配 17

4.3.1. 数据结构描述 17

4.3.2. 接口描述 18

# 简介

## 软件产品简介

数据总线定义为一套跨平台、可移植、能实现寻址发送、订阅发布和多播组播的软件功能库。

## 文档目的和范围

本文档用于给出数据总线软件的使用说明，及使用客户端接口库进行二次开发过程的数据结构描述和接口描述。该文档的目标读者是使用数据总线进行相关应用程序设计开发人员，用户开展相关应用软件设计开发工作。

# 软件介绍

## 软件总体结构

数据总线由总线服务NetRouter可执行程序和用户接入库NetRouterClient组成。

总体架构可以比拟成平时使用的物理网络以及网络中的路由器，如图1所示。Net Router可以理解为软件定义的路由器。Net1、Net2、Net3和Net4代表4个小的网络（网络是可扩展的，这里为说明方便，只用4个网络作为说明），4个小的网络组成一个大的通信网。每个小网络都有一个Net Router，在组成大的网络的同时，每个小网络也有自己的子网。Net Router不仅负责网络间的数据传输的管理，也负责子网数据传输的管理。一个个Net Router就像真实网络中的一个个实体路由器，路由器将网络中的计算机连接起来，而Net Router则将我们的应用进程连接起来。

应用程序调用客户端接口库，连接至本网的NetRouter，通过Net Router，网络中的任何一个应用能够将数据传递给网络中任意的另外的一个应用，无论它在同一个子网中还是在另外一个网络中。这就实现了数据的一对一传输、数据的一对多传输和数据的一对一个网络域的传输。同时又与真实的路由器不同的是，Net Router不仅能够传递数据，又通过保存每个应用的注册信息，最终能够实现数据的注册转发等功能。

图1 总线总体结构



## 通信逻辑地址描述

数据总线采用逻辑地址来进行通信局域区分，寻址和组播。

通信地址格式设计如下：

（局码、单位类型、单位ID、设备类型、设备ID）。从概念上讲，按铁路系统的管理体系来形成地址的各级单元，上级单元包含了下级单元，每级单元还附带一个类型来形成附加描述用以扩展，比如单位与单位类型。下面是地址的各级单元的说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址单元 | 字节 | 取值 | 说明 |
| 局码 | 1 | 0：全体  FFH：全体 | 铁路局代码 |
| 单位类型 | 1 | 0：全体  1：车站  2：中心  FFH：全体 | 1-车站，2-中心 |
| 单位ID | 2 | 0：中心服务器  FFFFH：全体  站码或调度台ID | 单位的概念指的是铁路管理单位的意思，是某个职责岗位所包含的设备和人员的集合。  当单位类型为车站时为站码，当单位类型为中心时为调度台ID，中心服务器的单位ID为0 |
| 设备类型 | 1 | 0：全体  1：服务器  2：调度台  3：车站终端  FFH：全体 | 1-服务器，2-中心调度台，3-车站终端 |
| 设备ID | 2 | 0：全体  FFFFH：全体 | 一个设备可以理解为一台（或一对漂移IP的双机）计算机。  设备ID用以区分某单位范围内某设备类型下不同设备 |

地址单元取值为全体意味着值不确定，用作匹配的话，能匹配该地址单元所有有效值。最好都不用0这个值，而使用FFH、FFFFH、FFFFFFFFH来表示全体。

## 应用环境及条件

数据总线软件库产品，其代码开发具有通用性和可移植性，可面向Windows平台和Linux平台进行直接编译或交叉编译，支持C++、Java、C#开发语言。针对各开发语言所发布的组件见下表：

|  |  |  |
| --- | --- | --- |
| 编程语言 | NetRouter | NetRouterClient |
| C++ | 对应平台可执行程序 | 1、C++动态链接库 |
| JAVA | 对应平台可执行程序 | 1. C++动态链接库 2. Java package接口包 |
| C# | 可执行程序 | 1. C++动态链接库 2. C#动态链接库 |

# 总线服务配置

本节用来描述总线服务NetRouter可执行程序的配置。

NetRouter的配置文件位于可执行文件的相同目录下的etc文件夹当中，分为NetRouterConfigure.xml应用程序配置文件和log4cplus.properties日志配置文件。

## NetRouterConfigure.xml配置文件

### 示例

<NetRouter>

<LocalAddr>//本网的局域地址

<bureau\_code>8</bureau\_code>//局码

<unit\_type>center</unit\_type>//单位类型

<unit\_id>0</unit\_id>//单位ID

</LocalAddr>

<InNeT>//局域网配置

<NetAddr>//局域网总线服务地址，双网

<ip1>191.168.99.98</ip1>

<port1>9003</port1>

<ip2>10.2.48.123</ip2>

<port2>9005</port2>

</NetAddr>

<Config>

<receive\_thread\_num>8</receive\_thread\_num>//接收线程数

<send\_thread\_num>4</send\_thread\_num>//发送线程数，最大为4

<client\_links>100</client\_links>//接入的客户端数量最大值

<patrol\_duration>3</patrol\_duration>//断开检测间隔，单位是秒

</Config>

</InNeT>

<OutNet>//广域网配置，可配置单双网

<svrIP>191.168.99.98</svrIP>//本机服务ip

<svrPort>9001</svrPort>//本机服务port

<remote>//若需连入多个，则需要配置多个remote

<!--必须配置，所要连往的服务器ip和端口号 -->

<server\_ip> 191.168.99.98 </server\_ip>

<server\_port> 9002 </server\_port>

</remote>

</OutNet>

</NetRouter>

### 广域网配置详细说明



图2 总体结构图

上图是数据总线的总体结构图，以Net1和Net2为例。

* 1. 首先看LocalAddr局域地址的作用

假设Net1的局域地址为：

<LocalAddr>//本网的局域地址

<bureau\_code>8</bureau\_code>

<unit\_type>station</unit\_type>

<unit\_id>1</unit\_id>

</LocalAddr>

Net2的局域地址为：

<LocalAddr>//本网的局域地址

<bureau\_code>8</bureau\_code>

<unit\_type>station</unit\_type>

<unit\_id>2</unit\_id>

</LocalAddr>

则所有连接至Net1的应用程序发往8,1,2地址的数据，都会由Net2的NetRouter发往Net2局域网的应用程序。而反过来亦然。

* 1. 其次看OutNet广域网配置的作用

假设Net1的配置为：

<OutNet>//广域网配置，可配置单双网

<svrIP>191.168.99.98</svrIP>//本机服务ip

<svrPort>9001</svrPort>//本机服务port

<remote>

<!--必须配置，所要连往的服务器ip和端口号 -->

<server\_ip> 191.168.99.23 </server\_ip>

<server\_port> 9002 </server\_port>

</remote>

</OutNet>

Net2的配置为：

<OutNet>//广域网配置，可配置单双网

<svrIP>191.168.99.23</svrIP>//本机服务ip

<svrPort>9002</svrPort>//本机服务port

<remote>

<!--必须配置，所要连往的服务器ip和端口号 -->

<server\_ip> 191.168.99.98 </server\_ip>

<server\_port> 9001 </server\_port>

</remote>

</OutNet>

则两个网的NetRouter可以互相连接和通信。若需多子网互联，需要加入相应个数的remote标签配置。

## log4cplus.properties

使用默认值。

# 应用开发

本节针对用户使用NetRouterClient接口库进行应用开发进行描述。具体分为配置文件描述、数据结构描述和接口描述。

## 配置文件描述

NetRouterClient的配置文件位于应用程序同目录下的config文件夹中，有Configure.xml和log4cplus.properties两个。

### Configure.xml

如果不使用Configure.xml文件进行初始化，而是应用程序自己初始化的话，不需要对其进行配置，只需要保留这个文件即可(必须保留)，无需更改其配置。

<FCMP>

<server>//局域网总线服务地址，双网

<ip1>191.168.99.98</ip1>

<port1>9003</port1>

<ip2>191.168.99.98</ip2>

<port2>9005</port2>

</server>

<address>//逻辑地址标识

<!--必须配置，线路号-->

<bureau\_code>8</bureau\_code>

<!--必须配置，单位类型：center，station, station:0x01, center:0x02-->

<unit\_type>center</unit\_type>

<!--必须配置，单位ID：调度台ID，站码，中心服务器的单位ID为0 -->

<unit\_id>0</unit\_id>

<!--必须配置，设备类型：svr，ddt，term，other, svr:0x01, ddr:0x02, term: 0x03, other: 0x04 -->

<dev\_type>svr</dev\_type>

<!--必须配置，设备ID -->

<dev\_id>3</dev\_id>

</address>

<in\_condition>//注册信息

<rec>

<protocol418\_condition>

<type\_func>0xff,0xff</type\_func>

</protocol418\_condition>

</rec>

</in\_condition>

<config>

</config>

</FCMP>

### log4cplus.properties

暂时使用默认配置。

## Java适配

### 数据结构描述

1. Address

每个通信客户端的逻辑通信地址设计如下：

Class Address{

1: byte bureauCode, //表示局码

2: byte unitType, //表示站或是中心

3: short unitId, //表示站或是中心的id

4: byte devType, //表示设备类型

5: short devId //表示设备id

}

1. SendMessage

Class SendMessage{

1: List<Address> destAddrs\_, //消息目的地址

2: String msg\_ //发送的消息

}

1. RecvMessage

Class RecvMessage{

1: Address srcAddrs\_, //消息源地址

2: String msg\_ //接收到的消息

}

### 接口描述

1. NetRouterClient类

用户程序通过这个类与本网的NetRouter进行交互。

|  |  |
| --- | --- |
| NetRouterClient（） | |
| 返回参数 |  |
| 参数列表 | String appName |
| 描述 | 构造函数，用此函数构造对象则需要对配置文件进行配置，不推荐。 |

|  |  |
| --- | --- |
| NetRouterClient（） | |
| 返回参数 |  |
| 参数列表 | * 1. String appName //应用的名称   2. String ip1 //网1服务的Ip地址   3. int port1 //网1服务的端口号   4. String ip2 //网2服务的ip地址   5. int port2 //网2服务的端口号   6. Address localAddr //本程序的逻辑通信地址   7. String reginfo //注册信息，可填空，默认注册全信息 |
| 描述 | 构造函数，用此函数构造对象则不需要对配置文件进行配置，推荐。  Reginfo的注册信息示例如下：  string reginfo =  "<in\_condition>\n"+  "<rec>\n"+  "<protocol418\_condition>\n"+  "<type\_func>0x0A,0x0B</type\_func>\n"+  "<type\_func>0x0A,0x0C</type\_func>\n"+  "</protocol418\_condition>\n"+  "</rec>\n"+  "</in\_condition>\n";  表示向NetRouter订阅类型为0A和子类型为0B、0C的数据。 |

|  |  |
| --- | --- |
| start（） | |
| 返回参数 | boolean |
| 参数列表 |  |
| 描述 | 开启通信服务，进行双网连接，成功返回true，失败返回false |

|  |  |
| --- | --- |
| sendMessage（） | |
| 返回参数 | boolean |
| 参数列表 | SendMessage sendMsg |
| 描述 | 发送消息，成功返回true，失败返回false |

|  |  |
| --- | --- |
| receiveMessage（） | |
| 返回参数 | boolean |
| 参数列表 | RecvMessage recvMsg |
| 描述 | 接收消息，成功返回true，失败返回false. |

|  |  |
| --- | --- |
| isNet1Connected（） | |
| 返回参数 | boolean |
| 参数列表 |  |
| 描述 | 检查网1是否连接，连接返回true，断开返回false. |

|  |  |
| --- | --- |
| isNet2Connected（） | |
| 返回参数 | boolean |
| 参数列表 |  |
| 描述 | 检查网2是否连接，连接返回true，断开返回false. |

|  |  |
| --- | --- |
| close（） | |
| 返回参数 | boolean |
| 参数列表 |  |
| 描述 | 断开网络连接，成功返回ture，失败返回false |

### 集成方法

在运行目录下放入config文件夹和NetRouterCppClient.dll。将NetRouterClient.jar集成到应用程序。

## C++适配

### 数据结构描述

1. NetAddress

每个通信客户端的逻辑通信地址设计如下：

Class NetAddress{

1: uinsigned char bureauCode, //表示局码

2: uinsigned char unitType, //表示站或是中心

3: uinsigned short unitId, //表示站或是中心的id

4: uinsigned char devType, //表示设备类型

5: uinsigned short devId //表示设备id

}

1. SentMessage

Class SentMessage{

1: vector<NetAddress> destAddrs\_, //消息目的地址

2: string msg\_ //发送的消息

}

1. RecvMessage

Class RecvMessage{

1: NetAddress srcAddrs\_, //消息源地址

2: string msg\_ //接收到的消息

}

### 接口描述

1. NetRouterClient类

用户程序通过这个类与本网的NetRouter进行交互。

|  |  |
| --- | --- |
| NetRouterClient（） | |
| 返回参数 |  |
| 参数列表 | string appName |
| 描述 | 构造函数，用此函数构造对象则需要对配置文件进行配置，不推荐。 |

|  |  |
| --- | --- |
| NetRouterClient（） | |
| 返回参数 |  |
| 参数列表 | 1. string appName //应用的名称 2. string ip1 //网1服务的Ip地址 3. int port1 //网1服务的端口号 4. string ip2 //网2服务的ip地址 5. int port2 //网2服务的端口号 6. NetAddress &localAddr //本程序的逻辑通信地址 7. string reginfo //注册信息，可不填，默认注册全信息 |
| 描述 | 构造函数，用此函数构造对象则不需要对配置文件进行配置，推荐。  Reginfo的注册信息示例如下：  string reginfo =  "<in\_condition>\n"  "<rec>\n"  "<protocol418\_condition>\n"  "<type\_func>0x0A,0x0B</type\_func>\n"  "<type\_func>0x0A,0x0C</type\_func>\n"  "</protocol418\_condition>\n"  "</rec>\n"  "</in\_condition>\n";  表示向NetRouter订阅类型为0A和子类型为0B、0C的数据。 |

|  |  |
| --- | --- |
| init（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 初始化，成功返回true，失败返回false |

|  |  |
| --- | --- |
| start（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 开启通信服务，进行双网连接，成功返回true，失败返回false |

|  |  |
| --- | --- |
| sendMessage（） | |
| 返回参数 | bool |
| 参数列表 | SentMessage &sendMsg |
| 描述 | 发送消息，成功返回true，失败返回false |

|  |  |
| --- | --- |
| receiveMessage（） | |
| 返回参数 | bool |
| 参数列表 | RecvMessage &recvMsg |
| 描述 | 接收消息，成功返回true，失败返回false. |

|  |  |
| --- | --- |
| isNet1Connected（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 检查网1是否连接，连接返回true，断开返回false. |

|  |  |
| --- | --- |
| isNet2Connected（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 检查网2是否连接，连接返回true，断开返回false. |

|  |  |
| --- | --- |
| close（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 断开网络连接，成功返回ture，失败返回false |

### 集成方法

在运行目录下放入config文件夹和NetRouterCppClient.dll。

## C#适配

由于C#调用C++动态库需要进行两种语言间参数的匹配，因此逻辑通信地址使用的是结构体。

### 数据结构描述

1. NETADDR

每个通信客户端的逻辑通信地址设计如下：

Struct NETADDR{

1: byte bureauCode, //表示局码

2: byte unitType, //表示站或是中心

3: UInt16 unitId, //表示站或是中心的id

4: byte devType, //表示设备类型

5: UInt16 devId //表示设备id

}

1. SENDMSG

Class SENDMSG{

1: List< NETADDR > destAddrs\_, //消息目的地址

2: string msg\_ //发送的消息

}

1. REVMSG

Class REVMSG{

1: NETADDR srcAddrs\_, //消息源地址

2: string msg\_ //接收到的消息

}

### 接口描述

1. NetRouterClientFactory类

简单工厂类，用户通过这类来生成具体的NetRouterClient对象。

|  |  |
| --- | --- |
| CreateNetRouterClient（） | |
| 返回参数 | INetRouterClient |
| 参数列表 | string appName |
| 描述 | 用此函数构造对象则需要对配置文件进行配置，不推荐。 |

|  |  |
| --- | --- |
| CreateNetRouterClient（） | |
| 返回参数 | INetRouterClient |
| 参数列表 | 1. string appName //应用的名称 2. string ip1 //网1服务的Ip地址 3. int port1 //网1服务的端口号 4. string ip2 //网2服务的ip地址 5. int port2 //网2服务的端口号 6. ref NETADDR localAddr //本程序的逻辑通信地址 7. string reginfo //注册信息，可不填，默认注册全信息 |
| 描述 | 用此函数构造对象则不需要对配置文件进行配置，推荐  Reginfo的注册信息示例如下：  string reginfo =  "<in\_condition>\n"+  "<rec>\n"+  "<protocol418\_condition>\n"+  "<type\_func>0x0A,0x0B</type\_func>\n"+  "<type\_func>0x0A,0x0C</type\_func>\n"+  "</protocol418\_condition>\n"+  "</rec>\n"+  "</in\_condition>\n";  表示向NetRouter订阅类型为0A和子类型为0B、0C的数据。 |

1. INetRouterClient接口

用户程序通过此接口类与本网的NetRouter进行交互。

|  |  |
| --- | --- |
| start（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 开启通信服务，进行双网连接，成功返回true，失败返回false |

|  |  |
| --- | --- |
| sendMessage（） | |
| 返回参数 | bool |
| 参数列表 | Ref SENDMSG sendMsg |
| 描述 | 发送消息，成功返回true，失败返回false |

|  |  |
| --- | --- |
| receiveMessage（） | |
| 返回参数 | bool |
| 参数列表 | Ref REVMSG recvMsg |
| 描述 | 接收消息，成功返回true，失败返回false. |

|  |  |
| --- | --- |
| isNet1Connected（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 检查网1是否连接，连接返回true，断开返回false. |

|  |  |
| --- | --- |
| isNet2Connected（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 检查网2是否连接，连接返回true，断开返回false. |

|  |  |
| --- | --- |
| close（） | |
| 返回参数 | bool |
| 参数列表 |  |
| 描述 | 断开网络连接，成功返回ture，失败返回false |

### 集成方法

在运行目录下放入config文件夹和NetRouterCppClient.dll、NetRouterClient.dll。其中NetRouterClient.dll是C#接口库，将其引入项目。