SUNNY YU

TCP 通信应用开发工具

协议包定义说明 v1.0

Sunny Yu

SunnyYu2000@gmail.com

QQ 群: 328452359

2013-7-12

本文档对 TCP 网络通信应用开发工具的协议包定义加以说明，阐述在定义文件中可以使用的协议包数据类型以及协议包字段的定义方式

目录

数据包定义说明.....3
1 基本名称和函数定义.....3
2 数据类型定义.....3
内部数据类型.....4
长度类型.....4
3 数据包头定义.....5
包头字段类型.....5
4 数据包定义.....5
包体字段定义.....6
字段条件说明.....6
条件字段名.....6
条件运算符.....6
条件值类型.....7
5 包定义范例.....7
生成的数据包类申明范例.....9
Pascal 代码.....9
C++代码 10

数据包定义说明

数据包定义主要目的是根据实际通讯需要，对通讯中使用的数据包结构做描述，描述包括包头，包体所含的字段类型大小等信息，然后代码生成器会根据这里的描述生成相关的数据包处理的目标代码。

本描述语言适合定义符合如下结构的网络协议数据包：

包头数据				**包体数据**
包大小	**包类型/命令字**	包序号	其他	对应包头包类型/命令字的具体包体结构

1 基本名称和函数定义

用于定义生成的数据包类前缀，代码生成的包定义文件中所引用到的基类文件名，使用到的基类数据包类名称的其中的几个方法名称，对于 class 和 fun 两类设置可以在后面用括号指定对应的 PAS 设置（将 PAS 代码和 C++代码名称作区分）

定义参数	**类型**	**说明**	**值范例**
target_path		生成代码保存的路径，相对于定义文件	.\src
使用到的基类名称设置			
class_base		包体的基类，生成包体代码由此类派生	PacketBase
class_packet		包含包头包体的数据包类名称	MyPacket
class_filename		存放数据包定义的文件名（不要后缀）	MyPackets
class_prefix		要生成的数据包类定义的前缀	My
使用到的基类函数名设置			
fun_pack		数据包编码的函数名	packData
fun_depack		数据包解码的函数名	depackData
fun_getpacketsize		获取数据包大小的函数名	getPacketSize
fun_tostring		输出数据包文本描述的函数名	toString
cmd_respmask		应答包掩码值	80000000
cmd_idvaltype		数据包命令字数据类型(int | str)	int
head_onlybodylen		包头的长度字段是否只包括包体的大小	0

基类需要自己做实现，以对应自己习惯的后端代码。

2 数据类型定义

数据类型定义由 **datatypes** 开头花括号引起来本份协议所需要的数据类型定义

```
datatypes '{' [ DataTypeDef [';'] ] ... '}'
```

用户定义数据包使用的基本数据类型

<table>
<tr><td colspan="5">DataTypeName DataType '(' DataLenType ')' ['code_type' ':' DataTypeCodeType] ['code_fun' ':' '{' DataTypeCodeFun/';'... '}'] [PadData]</td></tr>
<tr><td>DataType</td><td>DataType</td><td>DataLenType</td><td></td><td>PadData</td></tr>
<tr><td>定义本份描述文件中要用到的类型名</td><td>该类型对应的内部类型，可用得内部类型可见内部数据类型表</td><td>长度类型，是定长 fix 还是变长 var</td><td></td><td>对齐方式，定长数据的对齐方式，左对齐或右对齐，一般用在定长字符串类型上</td></tr>
<tr><td>code_type</td><td colspan="4">指明该定义对应的代码数据类型</td></tr>
<tr><td rowspan="4">code_fun</td><td colspan="4">指明读写该类型数据所要使用的函数名</td></tr>
<tr><td>r/read</td><td>w/write</td><td>tostr</td><td>size</td></tr>
<tr><td>读出数据</td><td>写入输入</td><td>输出的文本描述</td><td>获得类型大小</td></tr>
<tr><td colspan="4">可在这儿指定自己的数据读写函数替代默认的读写方法名</td></tr>
</table>

内部数据类型

内部定义的数据类型的说明以及读写该类型数据的默认函数名，默认函数名可以在定义协议包数据类型时替换成自己的函数名。

类型	说明	默认读写方法名
int	数值	getFixLenInt
nint	网络顺序数值	getFixLenNetInt
hint	HEX 字符串数值	getFixLenHexInt
sint	字符串数值	getFixLenStrInt
bcd	BCD 数值字符串	getFixLenBCDInt
str	字符串	getFixLenStr
date	时间字符串	getFixLenStr
hex	HEX 表示的数据	getFixLenHex
bin	BIN 表示的数据	getFixLenBin
arr	数组，主要用于指定某个字段的重复	

长度类型

数据字段在存取时占用的字节数方式作描述

类型	说明	
fix	定长，指定长度	可以是 1，2，4 等明确的数字
var	变长，字段的最前面字节指定实际长度	可用附加长度字段的定长字段来描述

3 数据包头定义

包头定义由 header 开始，花括号引起来包头字段定义

```
header '{' [HeadFieldDef]... [';'] '}'
```

包头字段定义

包头字段类型	字段名	定义的数据类型	字段长度

数据包头定义，一般要求必须有 **field_len,field_cmd** 几个字段的定义

包头字段类型

包头字段类型指出某个包头字段在包头定义中的作用，主要为**长度**和**命令字**，可以任意顺序

类型	说明	
field_len	指定数据包长度的字段	可以包含/不包含包头大小
field_cmd	表示数据包命令字的字段	用来确定包体数据类的字段
field_seq	用以区别数据包的序号字段	一般用于异步通讯中对应哪个序号的数据包，比如无应答做数据包重发机制的一个依据
field_other	其他可选字段	

4 数据包定义

一个数据包定义包含**包名称**、**包名称标识**、**标识常量值**以及**包体**几个部分组成。

```
PacketName ':' PacketID '=' PacketIDVal '{' [Body]... '}'
```

包体定义，区分正常包体 **body** 和应答包体 **respbody**，后面方括号括起具体字段定义

```
( 'body' | 'respbody' ) '[' [FieldDef [';']]... ']'
```

包体字段定义

包体结构中的字段定义，不限制字段数量，可以通过 SubFields 作一级子结构定义，不能在 SubFields 中再嵌套定义

FieldName ':' FieldType ['(' FieldLen ')'] [SizeField] [SubFields] [':' CodeType] ['<' OptCond '>']						
FieldName	**FieldType**	**FieldLen**	**SizeField**	**SubFields**	**CodeType**	**OptCond**
字段名	定义的字段类型	字段长度	长度字段名	子字段定义，由花括号限定范围	代码类型，输出代码中的代码类型，可以指定自己的代码类型	使该字段起作用的条件

字段条件说明

条件用于指定这一个字段在什么情况下起作用，比如当在这个字段前面的某一个字段为 1 的时候这个字段才会实际起作用出现在数据包中，否则这个数据包中不存在该字段信息。

条件的表示方式：**条件字段名** **条件运算符** **条件值**

条件字段名

条件中使用的条件字段名必须出现在这个字段之前，可以使当前字段域或上一层字段名（当为子字段定义时）

条件运算符

条件运算符	**说明**	
=	等于	
==	等于	
>	大于	
>=	大于等于	
<	小于	
<=	小于等于	
<>	不等于	
!=	不等于	

条件值类型

- 标识（可以指向数据包中的另外一个字段名或变量名， 标识前面带点号，表示是当前子结构体中的字段标识），
- 数字
- 字符串

包含条件字段的包结构范例

```
SendCard : SEND_CARD = 02 {
 body [
      data_nums : BCD(4);
      cards        : ARR data_nums {
            typ                  : BCD(1);
            card_id              : STR(18);
            tone_type            : BCD(1);
            tone_info            : STR(30) < .tone_type = 3 >;
            reserve              : STR(10) < .tone_type = 3 >
      } : SendCardInfo
   ]
}
```

5 包定义范例

```
// 设定协议名称为 HTM
packets HTM {
[
      target_path       : "..\protocol";
      class_base        :   CMyPacketBase(TPacketBase);
      class_packet      :   CMyPacket(TNetPacket);
      class_prefix      :   CS(My);
      class_filename    :   CMyHTMPackets(HTMPackets);

      fun_depack        : depackData;
      fun_pack          : packData;
      fun_getpacketsize: getPacketSize;
      fun_tostring      : toString;

      cmd_respmask      : 8000;
      cmd_idvaltype     : int;
]
```

```
// 定义在协议中要使用的数据类型
datatypes {
   INT    int(fix);
   STR    str(fix);
   ARR    arr(var)
}

// 这样后面的包体结构中可以使用 INT,STR 和 ARR 三个类型


// 定义数据包的包头信息
header {
   field_len :    datalen INT(4);
   field_cmd :    cmd INT(2);
   field_seq :    seq INT(4)
}

// 定义数据包包体信息
// 地址请求数据包
ReqUrl : REQ_URL = 02 {
   body [
      urlsize          : INT(4);
      url              : STR(200) urlsize
   ]
   respbody [
      res              : INT(1);
      urlsize          : INT(4);
      open_url         : STR(100) urlsize
   ]
}

// 结果反馈数据包
FeedBack    : FEED_BACK = 03 {
   body [
      urlsize          :    INT(4);
      url               : STR(18) urlsize;
      htmlsize         : INT(4);
      html              : STR(3) htmlsize
   ]
   respbody [
      res              : INT(1)
   ]
}
```

```
}
```

生成的数据包类申明范例

Pascal 代码

```
TMyBHOHeader = class(TPacketBase)
public
  datalen                 :   Integer; //4
  cmd                     :   AnsiString; //2
  seq                     :   Integer; //4
public
  function packData: Boolean; override;
  function depackData: Boolean; override;
  function getPacketSize: Integer; override;
  function toString: String; override;
end;

{ ReqUrl }
TMyBHOReqUrlBody = class(TPacketBase)
public
  urlsize                 :   Integer; //4
  url                     :   AnsiString; //urlsize
public
  function packData: Boolean; override;
  function depackData: Boolean; override;
  function getPacketSize: Integer; override;
  function toString: String; override;
end;

TMyBHOReqUrlRespBody = class(TPacketBase)
public
  res                     :   Integer; //1
  urlsize                 :   Integer; //4
  open_url                :   AnsiString; //urlsize
public
  function packData: Boolean; override;
  function depackData: Boolean; override;
  function getPacketSize: Integer; override;
  function toString: String; override;
end;
```

C++代码

```
class CSBHOHeader : public CSPacketBase{
public:
    long      datalen;                    // 4
    string cmd;                           // 2
    long      seq;                        // 4
public:
    bool packData();
    bool depackData();
    int    getPacketSize();
    string toString();
};

/// ---ReqUrl ---
class CSBHOReqUrlBody : public CSPacketBase{
public:
    long      urlsize;                    // 4
    string url;                           // urlsize
public:
    bool packData();
    bool depackData();
    int    getPacketSize();
    string toString();
};

class CSBHOReqUrlRespBody : public CSPacketBase{
public:
    long      res;                        // 1
    long      urlsize;                    // 4
    string open_url;                      // urlsize
public:
    bool packData();
    bool depackData();
    int    getPacketSize();
    string toString();
};
```