rtslabelscale dll 说明

1. 说明

1.1. 回调函数说明

//上传 PLU 及销售流水的回调函数 sResult：数据 JSON 格式 iRecNO:第几条记录 ACount 总记录数

TScaleCallBackProc= procedure(sResult: PAnsiChar; iRecNO: Integer; ACount: Integer); **stdcall**;

1.2. json 字段结构说明

PLU 的 json

```
{
 "PackageWeight": 0,//包装重量
 "PackageType": 0,  //包装类型
 "Ice": 0,          //含冰量
 "Message2": 0,     //信息 2 0~255
 "Message1": 0,     //信息 1 0~10000
 "BarCode": 2,      //条码类型 条码类型,0-99,101-110
 "WeightUnit": 4,     //质量单位  0~12 0:50g,  1:g   ,2:10g,   3:100g ,
                        4:Kg,5:Oz,      6:LB,         7:500g,      8:600g
                        9:,PCS(g),10:PCS(kg),  11:PCS(oz),   12: PCS(Lb)
 "HotKey": 1,         //热键
 "LabelId": 0,        //标签
 "Tolerance": 0,      //误差
 "UnitPrice": 9000,   //单价 无小数模式,0-9999999
 "QtyUnit": 0,        //数量单位 填 0
 "LFCode": 10001,     //生鲜码 1-999999,唯一识别每一种生鲜商品
 "IsLock": false,     //是否锁定价格
 "Rebate": 0,         //折扣 0~127：自定义折扣率
                        //129~138 排程：折扣模式 1~折扣模式 10
 "Deptment": 2,       //部门 0~99
 "Tare": 0,           //去皮  0~ 15.000
 "Name": "pluName1001", //品名 36 个字符  Name, 36 characters
 "Code": "10001",       //货号 10 位数字,用来组成条码
 "Recomdays": 0,        //推荐天数 0~4095
 "ShlefTime": 15,      //保质期 -9999~+9999
}
```

2. DLL 函数说明

2.1. 加载配置文件

function rtscaleLoadIniFile(cfgFileName:PAnsiChar): Integer; stdcall;
功能:　　加载配置文件名
参数:　　cfgFileName:PAnsiChar　包含路径 如 d:\bin\system.cfg
返回值:　　Integer 0:成功 -1:失败

文件格式：

```
[Setting]
DecimalDigits=2　　//小数位数
LfcodeLen=6;　　//生鲜码，默认 6 位

[Function]
 isUseMessage2=0　//是否启用 Message2　　1:启用　0：关闭(默认)
 isShowRecomdays=0 //是否启用推荐天数 1:启用　0：关闭(默认)
  IsDownloadPLUName2UKR=1 //是否下载 52 个字符（包克兰专用） 1:启用　0：关闭(默认)
  WeightRatio_500g=1; //yuntan 要为 1000
```

2.2. 连接秤

function rtscaleConnect(Addr: PAnsiChar; BaudRate:integer; var Connid:Integer): Integer; stdcall;
功能：　　连接秤
参数:　　Addr: PAnsiChar;Ip 或 COM 如 192.168.1.78 或 COM1
BaudRate:integer　　波特率，连接方式为 RS232,要设置，连接方式为 IP，此值设为 0;
Connid: 返回当前连接的 id　该值要传给其他函数使用
返回值:　　Integer :0 连接成功，-1：失败，-3：异常

2.3. 断开秤

function rtscaleDisConnect(Connid:Integer): Integer; stdcall;
功能：　　断开秤
参数:　　Connid: 连接 id
返回值:　　Integer :0 连接成功，-1：失败

2.4. 下发数据

function rtscaleDownLoadData(Connid:Integer; Data:PAnsiChar; len:Integer): Integer; stdcall; far; external 'rtslabelscale.dll'; //下载数据　功能:　　下发数据

参数:　　Connid:连接 id　Data:下发的数据　len: 数据长度

返回值:　　Integer :0 成功，-1：失败

2.5. 上传数据

function rtscaleUploadData(Connid:Integer; Data:PAnsiChar; len:Integer;　Retdata: PAnsiChar): Integer; stdcall; //上传数据

功能:　　下发数据到秤上

参数:　　Connid:连接 id　Data:PAnsiChar 下发的数据　len:数据长度　Retdata:返回 265byte 数据，不做任何处理

返回值:　　Integer　0 成功，-1：失败

2.6. 清除 PLU 和 Message 数据

function rtscaleClearPLUData(Connid:Integer): Integer; stdcall; far; external 'rtslabelscale.dll';

功能：　　清除秤上的 PLU 及 Message 数据

参数:　　无

返回值:　　Integer　0:成功 -1：失败

2.7. 下载 PLU 数据

function rtscaleDownLoadPLU(Connid:Integer; PluJson:PAnsiChar; ipack:Integer): Integer; stdcall;

功能:　　传输 PLU 信息 ,4 组的 PLU

参数:　　Connid:连接 id　PluJson:PAnsiChar　//4 组的 json 数据　ipack:当前第几包 从 0 开始 ,4 条记录为一包

返回值:　Integer :0 成功，-1：失败

举例：json 数据如下


```
[
{
  "PackageWeight": 0,
  "PackageType": 0,
  "Message2": 0,
  "Message1": 0,
  "BarCode": 22,
  "WeightUnit": 4,
```

```
  "PluName": "pluName1001",
  "LabelId": 0,
  "Tolerance": 0,
  "UnitPrice": 9000,
  "LFCode": 10001,
  "Rebate": 0,
  "Deptment": 2,
  "Tare": 0,
  "Code": 10001,
  "ShlefTime": 15,
 },{
  "PackageWeight": 0,
  "PackageType": 0,
  "Message2": 0,
  "Message1": 0,
  "BarCode": 22,
  "WeightUnit": 4,
  "PluName": "pluName1002",
  "LabelId": 0,
  "Tolerance": 0,
  "UnitPrice": 9000,
  "Reserved2": 0,
  "LFCode": 10002,
  "Rebate": 0,
  "Deptment": 2,
  "Tare": 0,
  "Code": 10002,
  "ShlefTime": 15,
 },{
  "PackageWeight": 0,
  "PackageType": 0,
  "Message2": 0,
  "Message1": 0,
  "BarCode": 22,
  "WeightUnit": 4,
  "PluName": "pluName1003",
  "LabelId": 0,
  "Tolerance": 0,
  "UnitPrice": 9000,
  "LFCode": 10003,
  "Rebate": 0,
  "Deptment": 2,
  "Tare": 0,
  "Code": 10003,
```

```
  "ShlefTime": 15,
  "Reserved": 0
 },{
  "PackageWeight": 0,
  "PackageType": 0,
  "Message2": 0,
  "Message1": 0,
  "BarCode": 22,
  "WeightUnit": 4,
  "PluName": "pluName1004",
  "LabelId": 0,
  "Tolerance": 0,
  "Account": 0,
  "UnitPrice": 9000,
  "LFCode": 10004,
  "Rebate": 0,
  "Deptment": 2,
  "Tare": 0,
  "Code": 10004,
  "ShlefTime": 15,
 }]
```

Delphi Example

```
procedure DownLoadPlu;
var
  jo,aSuperArray: ISuperObject;
  s:string;
  i,J:Integer;
  Result: Integer;
begin
  for J := 0 to 150 do
  begin
    aSuperArray := SA([]);
    for i:=0 to 3 do    //一次只能传 4 条  Can only pass 4 at a time
    begin
      jo := SO();
      s := 'pluName' + IntToStr(J * 4 + I + 1+1000);
      jo.S['PluName'] := s; //品名,36 个字符    Name, 36 characters
      jo.I['LFCode'] :=    J * 4 + I + 1 + 10000; //生鲜码,1-999999,唯一识别每一种生鲜商品
fresh code, 1-999999, uniquely identifies each fresh product
      jo.I['Code'] := J * 4 + I + 1 + 10000; //货号，10 位数字,用来组成条码    goods no, 10 digits
      jo.I['BarCode'] := 22; //条码类型,0-99      barcode type, 0-99
      jo.I['UnitPrice'] :=9000;//单价,无小数模式,0-9999999        unit price, no decimal mode,
0-9999999
```

```
        jo.I['WeightUnit'] := 4;       //称重单位  Weighing Units    0-12
        jo.I['Deptment'] := 2; //部门,2 位数字,用来组成条码   Department, two digits
        jo.D['Tare'] :=0;   //去皮质量,逻辑换算后应在 15Kg 内      Tare, logical conversion should be within 15Kg
        jo.I['ShlefTime'] :=15;   //保存期,0-365    Shelf life, 0-365
        jo.I['PackageType'] :=0;//包装类型      Pack Type    0:正常 1:定重 2：定价 3:定重定价 4:二维码      Package Type 0: Normal 1: Fixed Weight 2: Pricing 3: Fixed Price 4: QR Code
        jo.D['PackageWeight'] := 0;// //包装重量/限重重量,逻辑换算后应在 15Kg 内   Package weight, logical conversion should be within 15Kg
        jo.I['Tolerance'] :=0; //包装误差,0-20      Packaging error, 0-20
        jo.I['Message1'] := 0; //信息 1,0-10000 Message 1,0- 10000
        jo.I['Message2'] := 0;         //信息 2, 0-255
        jo.I[LabelId] := 0;;      //标签类型 1,2,4,8,16,32,64,128,,3,12 分别对应标签编缉器 RTLabel.exe 的标签类型(A0,A1,B0,B1,C0,C1,D0,D1,E0,E1)
                                          // Label type 1,2,4,8,16,32,64,128,,3,12 correspond to the label types of the label editor RTLabel.exe (A0, A1, B0, B1, C0, C1, D0, D1, E0, E1)
        jo.I['Rebate'] := 0;      //折扣,0-99      discounts, 0-99
        aSuperArray.AsArray.Add(jo);
      end;
       //Transfer one PLU
      lblDispPrgress.Caption := IntToStr((J+1) * 4);
      lblDispPrgress.Refresh;

      s := aSuperArray.AsJSon(True,False);
      ShowMessage(s);
      aSuperArray := nil;
      Result := rtscaleDownLoadPLU(Connid,PAnsiChar(s));
      if (Result < 0) then
         exit;
    end;
```

2.8. 上传 PLU

function rtscaleUploadPluData(Connid:Integer; ACallBack: TScaleCallBackProcEx): Integer;stdcall;//上传 PLU 数据

功能:　　上传 PLU

参数:　　ACallBack: TScaleCallBackProcEx　上传 PLU 的回调 包含热键及商品信息

返回值:　　Integer：　0:成功 -1:失败

上传的 json 格式，一次只传回一条记录

格式如下

```
{
 "PackageWeight": 0,//包装重量
 "PackageType": 0,  //包装类型
```

```
 "Ice": 0,           //含冰量
 "Message2": 0,      //信息 2
 "Message1": 0,      //信息 1
 "BarCode": 22,       //条码类型
 "WeightUnit": 4,     //质量单位
 "HotKey": 1,         //热键
 "LabelId": 0,        //标签
 "Tolerance": 0,      //误差
 "UnitPrice": 9000,   //单价
 "QtyUnit": 0,        //数量
 "LFCode": 10001,     //生鲜码
 "IsLock": false,     //锁定
 "Rebate": 0,         //折扣
 "Deptment": 2,       //部门
 "Tare": 0,           //去皮
 "PluName": "pluName1001", //品名
 "Code": "10001",       //货号
 "Recomdays": 0,        //推荐天数
 "ShlefTime": 15,      //保质期
}
```

2.9. 下载热键

function rtscaleDownLoadHotkey(Connid:Integer; HotkeyTable: PHotkeyTable; TableIndex: Integer): Integer; stdcall;

数据结构定义

THotkeyTable = array[0..83] of LongInt; //每一个数组元素为一条 PLU 的生鲜码,
//84 个元素对应 84 条 PLU

PHotkeyTable = ^THotkeyTable;

功能: 用于传送热键表到标签秤，当按下某个键时，可以找到对应的 PLU

输入参数： **HotkeyTable**：热键表，84 个元素对应 84 条 PLU 的生鲜码

如 HotkeyTable[0]=10001, HotkeyTable[1]=10002; ….. 当标签秤按下 1 键时，会显示生鲜码为 10001 的 plu，当标签秤按下 2 键时，会显示生鲜码为 1002 的 plu……

TableIndex:Integer //热键传送表号，112 键两页的布局要分 3 次来下载， 如热键号为 1-84，TableIndex=0, 热键号为 85-168, TableIndex=1, 热键号为 169-224, TableIndex=2 返回值: Integer 0:成功 1：失败

2.10. 更新价格

function rtscaleUpDatePrice(Connid:Integer; PluPriceJson:PAnsiChar): Integer; stdcall;

功能:　更新价格
参数:　Connid:连接 id　PluPriceJson: 36 组的生鲜+价格, json 格式
返回值:　Integer　0:成功 1：失败

[{
"UnitPrice": 100,　//单价
"LFCode": 10001 //生鲜码
},{
"UnitPrice": 200,
"LFCode": 10002
},
..... 36 组....
]

2.11. 清除 PLU 营养信息数据

function rtscaleClearPluNutritionData(Connid:Integer): Integer; stdcall;
功能:　清除秤上的 PLU 营养信息数据
参数:　Connid:Integer 连接 id
返回值:　Integer　-1：失败　0：成功

2.12. 下载营养元素表

function rtscaleDownLoadNutritionTable(Connid:Integer; sTableJson:PAnsiChar): Integer; stdcall;
功能:　传输营养元素基本表到秤上
参数:　Connid:Integer;连接 id　sTableJson:PAnsiChar　营养表的 json 格式，12 组的数据
返回值:　Integer　-1：失败　0：成功

Json 格式如下
[
{
"iUnit": 1,　//单位:1:kj,2:g*10,3:mg*10,4:ug*100
"Name": "VB"
},{
"iUnit": 2,
"Name": "VC"
},,{
"iUnit": 4,
"Name": "VE"
}
.....// 12 组

]

2.13. 下载单品营养表

function rtscaleDownLoadPluNutrition(Connid:Integer; sNutriDetailJson:PAnsiChar): Integer; stdcall;

功能: 下载单品营养表到秤上

参数: Connid:连接 id; sNutriDetailJson: 单品营养信息 4 个 PLU 为一组下发

返回值: Integer 0:成功 -1：失败

```
[
{
 "percent": [50,51,52,53,54,55,56,57,58,59,60,61],//NRV%  12 个
 "value": [30,31,32,33,34,35,36,37,38,39,40,41],//含量,注意：下发时，要转成整数，根据单位:1:kj*1,2:g*10,3:mg*10,4:ug*100 乘于对于的基数
 "LFCode": 10001, //生鲜码
 "GroupTitle": "pluname1001",//营养抬头
 "isPrint": [ true,true,true,true,true,true,true,true,true,true,true,true] //是否打印
},{
 "percent": [
  50,51,52,53,54,55,56,57,58,59,60,61],
   "value": [
  30,31,32,33,34,35,36,37,38,39,40,41],
 "LFCode": 10002,
 "GroupTitle": "pluname1002",
 "isPrint": [
  true,true,true,true,true,true,true,true,true,true,true,true]
},
.....4 组....
{}
]
```

2.14. 清除 Message

function rtscaleClearMessage(Connid:Integer):Integer;far; external 'rtslabelscale.dll'; // 清除 Message

功能: 清除 message 用于新的下位机

参数: Connid:Integer; 连接 id

返回值: Integer 0:成功 1：失败

2.15. 下载 Message

function rtscaleDownLoadMessage(Connid:Integer; MsgId: Integer; PMessage: PAnsiChar; DataLen: Integer; var iLongMsg:Integer): Integer; stdcall;

功能: 下发 Message, 用于打印标签时，可以把单品的附加信息，如产地信息打印出来

参数: Connid:Integer; 连接 id MsgId: Integer; 信息 id:0~10000 PMessage: PAnsiChar;信息正文

DataLen: Integer 信息正文长度 最多 246

iLongMsg: 第 iLongMsg 条长消息，该值要赋初值 0，

返回值: Integer 0:成功 1：失败

iLongMsg：返回当前第 n 条长消息，下次再调用此函数时，要代入此值。该值是累加的，直到所有消息发送结束。

在同一个 Message 的包数计算方式为前面 250 byte 为 1 包，后面的字符 256byte 为 1 包。一个长消息的包数至少为 2 包。

2.16. 上传 Message

function rtscaleUploadMessage(Connid:Integer; ACallBack: TScaleCallBackProc): Integer;stdcall;//上传 PLU 数据

功能: 上传 Message

参数: Connid:Integer; 连接 ID ACallBack: 回传 Message 一次只传回一条记录

返回值: Integer

Integer 0:成功 -1：失败

上传的 json 格式

```
{
 "MsgText": "testmessage0
}
```

2.17. 下载质量单位

function rtscaleDownLoadWeightUnit(Connid:Integer; AWeightUnitJson:PAnsiChar): Integer;

功能: 下载质量单位

参数: Connid:Integer; 连接 ID AWeightUnitJson:PAnsiChar 单位 json 格式

```
{

 "WeightUnit0": "50g",
  "WeightUnit1": "g",
```

"WeightUnit2": "10g",
"WeightUnit3": "100g",
"WeightUnit4": "Kg",
"WeightUnit5": "Oz",
"WeightUnit6": "LB",
"WeightUnit7": "500g",
"WeightUnit8": "600g",
"WeightUnit9": "pcs(g)",
"WeightUnit10": "pcs(kg)"
"WeightUnit11": "pcs(oz)",
"WeightUnit12": "pcs(Lb)",
"WeightUnit13": "",
"WeightUnit14": "",
"WeightUnit15": "",
"WeightUnit16": "",
"WeightUnit17": "",
"WeightUnit18": "",
"WeightUnit19": "",
"WeightUnit20": "",
"WeightUnit21": "",
"WeightUnit22": "",
"WeightUnit23": "",
"WeightUnit24": "元",

//}
// 返回值: Integer 0:成功 -1：失败

2.18. 上下传广告标语的头部信息

下传头部信息

function rtscaleDownLoadAdHead(Connid:Integer; AdInfotxt:PAnsiChar; len:Integer): Integer; stdcall;

功能: 下载广告标语的头部信息，在秤的屏屏蔽会显示，标签放入 Head 组件，会打印出来

参数: Connid:连接 ID; AdInfotxt:头部信息; len: AdInfotxt 的长度，最多 255

返回值: Integer 0:成功 -1：失败

上传头部信息

function rtscaleUploadDataAdHead(Connid:Integer; Retdata: PAnsiChar): Integer;

功能: 上传广告标语的头部数据

参数: Connid:Integer;连接 ID Retdata: PAnsiChar 返回数据

返回值:　　Integer 0:成功 -1：失败

2.19. 上下载广告标语的尾部信息

下载尾部信息

function rtscaleDownLoadAdTail(Connid:Integer; AdInfotxt:PAnsiChar; len:Integer): Integer; stdcall;

功能:　　下载广告标语的尾部信息 标签放入 Bottom 组件，会打印出来,一般用来打印一些广告标语，如门店信息如电话，地址

参数:　　Connid:连接 ID; AdInfotxt:尾部信息，; len: AdInfotxt 的长度，最多 255

返回值:　　Integer 0:成功 -1：失败

上传尾部信息

function rtscaleUploadDataAdTail(Connid:Integer; Retdata: PAnsiChar): Integer;

功能:　　上传广告标语的头部数据

参数:　　Connid:Integer;连接 ID　Retdata: PAnsiChar　返回数据

返回值:　　Integer 0:成功 -1：失败

2.20. 上传交易流水

function rtscaleUploadSaleData(Connid:Integer; AIsClearData: Boolean; AScaleCallBackProc: TScaleCallBackProc):Integer;//

功能:　从秤上上传销售数据

参数：Connid: 连接 ID; AIsClearData: 是否清除秤上数据；AScaleCallBackProc: TScaleCallBackProcEx　回调销售数据，一条条调用

返回值:　　Integer -1:失败　>=0 返回记录条数

```
{
"Weight": 365, //重量
"Quantity": 1, //数量
"WeightUnit": 4, //单位
"PluName": "大白菜 22", //品名
"TotalPrice": 365, //总价
"UnitPrice": 1000, //单价
"OnlineTime": "20200908155100", //上次清除流水的时间
"LFCode": 1,//生鲜码
"Rebate": 80,//折扣
"SaleTime": "20200908170700",//销售时间
"UserID": 80358,//用户 id, 没用
```

"Clerk": 0 //收银员
}

2.21. 上下传折扣数据

下传折扣

function rtscaleDownLoadDisCountSchedule(Connid:Integer; DisCountInfoJson:PAnsiChar): Integer;stdcall;

功能: 下载折扣信息
参数: Connid:Integer; 连接 id DisCountInfoJson:PAnsiChar 折扣的信息，json 格式
一个模式 5 个时段，10 个模式，100 个时段
json 格式如下：

```
//                [
//          {
//            "Hour": 8,
//            "Minute": 30,
//            "Discount": 0
//         }
//         ,
//         {     "Hour": 9,
//               "Minute": 30,
//               "Discount": 0,
//          },
//          {
//                "Hour": 10,
//               "Minute": 30,
//               "Discount": 0,
//          }
//          {}，{} ....100 个时段
//
```

// 返回值: Integer
0： 成功 -1 失败

上传折扣

function rtscaleUpLoadDisCountSchedule(Connid:Integer; ACallBack: TScaleCallBackProc): Integer; stdcall;

功能: 从秤上上传打折排程信息
输入参数: Connid:Integer; 连接 ID
输出参数： ACallBack:TScaleCallBackProc; 折扣数据通过回调返回，返回 10 个模式的数据
回调函数如下

procedure UploadDiscountCallback(sResult: PAnsiChar; iRecNO: Integer; ACount: Integer); stdcall;

sResult：折扣数据,json 格式和下传的一样格式

iRecNO: 返回 0

ACount: sResult 的长度

返回值: Integer 0： 成功 -1 失败

2.22. 获取重量

function rtscaleGetPluWeight(Connid:Integer; var dWeight: Double): Integer; stdcall; //获取重量

功能: 获取秤上的重量

输入参数: Connid：连接 id

输出参数： dWeight: Double 返回的重量

返回值: Integer 0： 成功 -1 失败

2.23. 下载自定义条码

function rtscaleDownLoadCustomBar(Connid:Integer; CusBarJson:PAnsiChar): Integer; stdcall;

功能: 下载自定义条码 101~110

参数: Connid:Integer; 连接 id CusBarJson:PAnsiChar 自定义条码格式

返回值: Integer 0： 成功 -1 失败

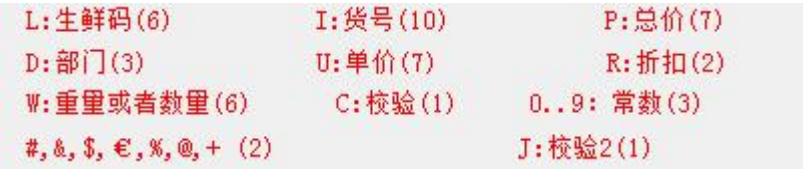
CusBarJson 条码的 json 格式如下， 注意，要按 101~110 的顺序传进参数

```
[
{
  "BarKind": 13, //条码类型为 13 码
  "OffsetLen": 0, //重量或数量的位数
  "BarFormat": "DDLLLLLUUUUUC",    //条码字段格式
  "BarID": 101   //条码编号
 },{
  "BarKind": 18,
  "OffsetLen": 3,//重量的位数为 3
  "BarFormat": "18LLLLLWWWWWPPPPPC",
  "BarID": 102
 }]
```

条码编号: 设置条码的类型编号：固定位 101-110

条码类别:设置条码的长度

条码格式:设置条码格式,每个字母对应字段如下，“(6)”里面的数字，代表最长只能几位，如 L:LF Code(6) 代表最长只能输入 6 个”L”,即 LLLLLL，条码打印时将打印 6 位的生鲜码



重量/数量小数位: 设置重量或数量的小数位数

2.24. 升级固件程序

function rtscaleUpgradeFirmware(AFileName: PAnsiChar; Addr: PAnsiChar; BaudRate:integer; var Connid:Integer; UpgradeBack: TUpgradeFirmwareCallBackProc):integer;stdcall;

功能: 升级秤上的固件程序
参数: AFileName: 要升级的文件名.hex 或.bin 格式
Addr: 秤的 ip 地址或 COM 口; BaudRate:波特率;
var Connid 返回连接 id
UpgradeBack: TUpgradeFirmwareCallBackProc 升级时的回调函数
返回值: Integer 0： 成功 -1 失败

更新固件程序的回调函数说明
TUpgradeFirmwareCallBackProc = procedure(sTips: PAnsiChar; status:Integer; iRecNO: Integer; ACount: Integer); stdcall;
// sTips：提示信息 status:当前状态 iRecNO:第几个包 ACount 总包数
status:状态 的常量定义
Upgrade_Status_StartCommandFail = 0; //发送升级命令失败
Upgrade_Status_upgrading = 1; //正在更新
Upgrade_Status_upgradeFail = 2;//更新失败
Upgrade_Status_EndCommandFail = 3;//发送升级结束命令失败
Upgrade_Status_upgradeOk = 4; // 升级成功

2.25. 下载部门信息

function rtscaleDownLoadDepartment(Connid:Integer; depinfoJson:PAnsiChar; isDownGst:Boolean) :integer;stdcall;
功能: 下载部门信息到秤上
参数: Connid:Integer; 连接 id depinfoJson:PAnsiChar; 部门信息,json 格式
isDownGst:Boolean 是否下载 GST 信息
返回值: Integer -1：失败 0：成功
注意：最多下载 59 条信息，一次性全部上传
Json 格式如下

```
  [
 {
  "IsPrintDeptName": true, //是否打印部门名称
  "DeptId": 1,//部门编号
  "SGST": 1.12,   //最多两位小数，值范围 0~99.99
  "DeptName": "Depart1", //部门名称  14 个 byte
  "CGST": 1.5,//最多两位小数，值范围 0~99.99
  "IsPrintDeptId": true //是否打印部门编号
 },{
  "IsPrintDeptName": true,
  "DeptId": 2,
  "SGST": 2,
  "DeptName": "Depart2",
  "CGST": 2.5,
  "IsPrintDeptId": true
 },
  ......
]
```

2.26. 上传部门信息

function rtscaleUpLoadDepartment(Connid:Integer; ACallBack: TScaleCallBackProc; isUploadGst:Boolean): Integer;

功能: 从秤上上传部门信息，一次性全部上传

参数: Connid:Integer; 连接 id ACallBack: TScaleCallBackProc 回调 isDownGst:是否上传 Gst，

返回值: Integer： -1 失败 其他：成功上传记录数

回调的函数 //上传 PLU 及销售流水的回调函数 sResult：数据 JSON 格式 iRecNO:第几条记录 ACount 总记录数

TScaleCallBackProc= procedure(sResult: PAnsiChar; iRecNO: Integer; ACount: Integer); **stdcall**;

sResult: 所有部门信息，json 格式，请参考下载的 json 格式

iRecNO:固定为 1

ACount：总的记录数

2.27. 选择字符集

function rtscaleGetFontCharset(sFontName:PAnsiChar; var iCharset:Integer):Boolean;stdcall; external 'rtslabelscale.dll';//选择字符集

功能: 选择字符集，生成字体使用，为了给没用弹出字符集选择对话框的程序语言使用，

如 QT
参数: sFontName:PAnsiChar; 字体名 var iCharset:Integer 返回的字符集，也可以作为输入参数

返回值: Boolean： true 有选择 false 没有选择

2.28. 加载简体汉字字库

function rtscaleLoadHZCode(isFullCode:Boolean):Boolean;
功能: 加载简体汉字字库，下载中文库前调用一次即可（加载后，后续不用再加载）
参数: isFullCode:true 加载 HzwordFull.dat, false, Hzword.dat(默认)
返回值: boolean: true:成功 false 失败(一般是文件不存在)

2.29. 创建字库

function rtscaleCreateFontLib(ParamJson:PAnsiChar; CallBackProc: TScaleCallBackProc): Integer;
功能: 生成字库文件
'8x16.fnt', '8x24.fnt', '16x32.fnt', '12x24.fnt', '8H16.fnt', '16H16.fnt',
'Other.FNT', '24L24.FNT', '12L24.FNT','HZWORD.FNT'
参数: Connid:Integer;连接 id ParamJson:PAnsiChar; 字体的参数(json 格式如下)

```
 {
 "DoubleFontName": "宋体",//双字节字体名
 "DoubleWeight": 4,//双字节字体粗细
 "DoubleFntCharSet": 1,//双字节字符集


"SingleFontName": "Calibri", //单字节字体名
 "SingleFntCharSet": 1, //单字节字体粗细
"SingleWeight": 4, //单字节字体粗细程度
 "FntStart": 0, //要下载字体的开始
 "FntEnd": 9  // 要下载字体的结束
 "isDownDoubleFont": true,//是否下载双字节字体
}
```

CallBackProc: TScaleCallBackProc 回调函数
TScaleCallBackProc= procedure(sResult: PAnsiChar; iRecNO: Integer; ACount: Integer); **stdcall**;
sResult：当前正在生成的字体名
iRecNo: 当前字体的第几个字符
ACount:当前字体的字符总数
返回值: Integer 0:成功 -1：失败

2.30. 查询秤类型

function rtscaleGetScaleType(Connid:Integer; sRetJson: PAnsiChar; len:Integer):Integer;

功能: 查询秤的类型，如秤的协议

输入参数: Connid：连接 id len:Integer sRetJson 分配的长度

输出参数: sRetJson： 返回 json 格式

返回值: Integer 0: 成功，-1:失败

sRetJson 格式如下

```
{
"Protocol": 1      //0:旧协议   //1:新协议  RONGTA_PV2.0
 }
```