文档版本	1.16
.NET 版本	1.5.0
.NET Core 版本	2.2.0
释放日期	2020-03-09

D11 电子标签系统开发者手册

用户编号: _____
项目编号: _____

技术支持：huanghaipeng@etag-tech.com

版本历史

版本号	日期	描述	作者	审阅
1.0	2017-01-06	文档初始化	黄海鹏	黄海鹏
1.1	2017-02-22	新版协议对 LED 灯闪烁次数的修改	黄海鹏	黄海鹏
1.2	2017-04-18	兼容码字、点阵发送方式全屏、局部刷新 重新封装 SDK 接口	黄海鹏	黄海鹏
1.3	2017-04-20	增加新版协议对航道灯的支持	黄海鹏	黄海鹏
1.4	2017-10-27	增加新版协议对彩色屏的支持	黄海鹏	黄海鹏
1.5	2018-01-12	针对商超零售等场景，增加绑定基站的说明 改变二维码显示方式	黄海鹏	黄海鹏
1.6	2018-04-21	增加对于下发数据超载的检查 增加错误代码描述	黄海鹏	黄海鹏
1.7	2018-04-28	增加新版基站（带门店号）支持 增加图像处理抖动算法 增加灰阶阈值二分法可设定阈值，RGB 可设定阈值 增加矩形框	黄海鹏	黄海鹏
1.8	2018-06-28	更新字体（价格部分）	黄海鹏	黄海鹏
1.9	2018-09-19	注意：部分不向前兼容!!! 增强属性约束范围 增加.NET Core 版本	黄海鹏	黄海鹏
1.10	2018-11-13	变更：坐标系参数重命名命名 增加对按键标签、主动侦听基站的支持	黄海鹏	黄海鹏
1.11	2019-05-05	增加数据溢出检查 增加组合最优解（贪心算法） 增加 DataEntity 构造注入 增加 PTL290、ESL1250R 支持	黄海鹏	黄海鹏
1.12	2019-07-03	增加 SDK 工作模式 增加基站类型	黄海鹏	黄海鹏
1.13	2019-07-25	新增 4 款标签尺寸	黄海鹏	黄海鹏
1.14	2019-10-18	新增按键反馈类型和 PTL 拣选数量	黄海鹏	黄海鹏
1.15	2019-12-16	细节优化，版本一致性确认	黄海鹏	黄海鹏
1.16	2020-03-09	增加 PTL290X 支持 优化默认值	黄海鹏	黄海鹏

目录

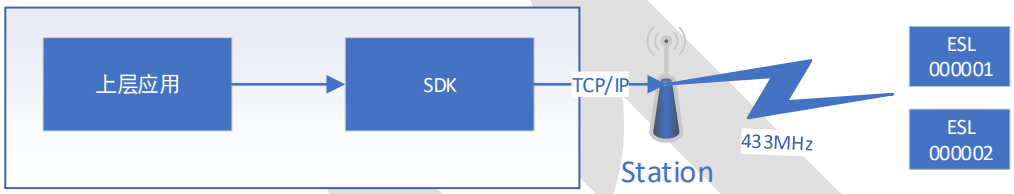
1 简述

本文档适用于基于.NET Framework 4.0 或者更高版本的项目。
本文档适用于基于.NET Core 2.1 或者更高版本的项目。

1.1 系统架构

电子标签系统可以简单分为三个部分：上层应用、SDK 和硬件设备。上层应用是指面向于用户业务逻辑的应用程序程序，诸如 WMS、SAP/ERP、HIS 等等；SDK 负责抽象硬件功能并提供给上层应用，同时将上层应用数据打包后传递给硬件；硬件包括无线通信基站、电子标签和相关网络设备。

电子标签系统整体结构按上述划分的视图如下：



特定的名词解释：

基站，是指电子标签系统中的无线发射器，SDK 中通常称为 Station，或者 AP。基站包含两个身份属性：门店编号（ShopCode）和 ID，门店编号取值范围为 0x0000~0xFFFF，ID 取值范围为 0x00~0xFF。同一个门店编号下，基站 ID 不能出现重复，否则会出现冲突。

门店编号应用的场景，适用于连锁店，或者多个仓库，在这种情况下，是可以驱动 SDK 对不同的门店进行并发操作的。

标签，是指电子标签（ESL）。标签包含一个身份属性：ID。标签 ID 为 6 位或者 8 位 16 进制 ID，其中，8 位 ID 前两位为标签类型，具体请参考附录。

注意：SDK 并不会与标签直接联系，而是与基站联系。

1.2 关于 SDK

SDK 目前提供.NET 和.NET Core 两个版本，皆在 NuGet 上发布：

1. .NET 版本：

```
Install-Package eTag.SDK
```

2. .NET Core 版本：

```
Install-Package eTag.SDK.Core
```

如无特别注明，本文档中的相关内容皆适用于.NET 版本和.NET Core 版本。

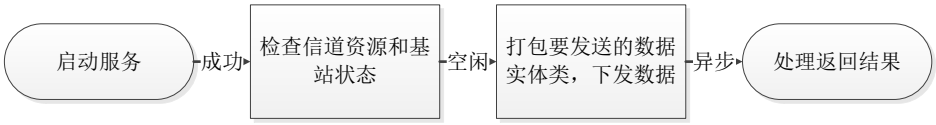
SDK 通过单实例类模式，由 `Server.Instance` 向开发者提供所有访问方法/属性。

SDK 向开发者提供如下必要的概念：

1. 启动服务；

2. 下发给电子标签数据的接口；
3. 通过注册事件的方法异步接受基站返回的数据；
4. 电子标签屏幕数据的布局(Layout)概念；
5. SDK 相关属性。

以下是一个典型的发送流程：



SDK 主要强调两个概念：异步、多线程。异步是指 SDK 与硬件之间的通信，是异步的。多线程是指，SDK 与多个基站之间通信，是多线程的。但是在同一环境下，受无线电电磁波冲撞制约，基站与标签之间的通信时单线程的。

1.3 开始之前

在使用 SDK 进行开发之前，需要确认：

- 请与销售人员/代理商/技术支持人员确认 SDK 版本、电子标签固件版本、基站固件版本；
- 硬件设备和网络环境已经准备就绪，包括诸如设备参数已经配置正确，网络环境正常，无线电电磁环境许可，电子标签与无线基站的通信距离在合理范围之内等；
- 操作系统环境配置正确，并且计算机上所安装的安全软件和防火墙允许上层应用对端口（默认为 1234）的访问。

2 初始化

2.1 启动服务

启动服务的代码很简单：

```
Server.Instance.Start(int port = 1234)
```

以默认方式启动 SDK 服务，默认端口值为 1234。

如果你的网络配置 SDK 监听端口为其他端口，需要修改端口值。

启动服务默认的返回值为：

- Resuslt.OK：启动成功；
- Result.StartFailed：启动失败。

2.2 注册事件

由于 SDK 是异步的返回基站和标签数据的，因此通过注册事件的办法来通知上层应用接受基站和标签的返回数据。

对于基站而言，基站注册事件为：

```
public event EventHandler<StationEventArgs> StationEventHandler
```

对于标签而言，标签结果返回事件为：

```
public event EventHandler<ResultEventArgs> ResultEventHandler
```

有关注册事件的具体处理，请参见 4. 处理返回数据。

2.3 基站状态

基站共有三种状态：离线（Offline）、空闲（Online）、工作中（Working）。

SDK 分别用 GetStationWorkingList 和 GetStationOnlineList 两个方法表示当前正在工作的基站列表和当前空闲的基站列表。离线状态不会被列举，由上层系统自行处理。

只有基站处于空闲状态时，上层应用才可以通过 SDK 与其通信。

1. 查询当前正在工作的基站列表（一般返回单个）

public string GetStationWorkingList(string shop = "")		
参数	类型	注释
shop	string	门店编号，必须指定
返回	工作基站 ID，空表示没有	

2. 查询当前处于空闲状态的基站列表

public List<string> GetStationOnlineList(string shop = "")		
参数	类型	注释
shop	string	门店编号，必须指定
返回	基站 ID 列表	

注意：门店编号为空的时候，并不是指全部门店，而是不带门店编号的基站。

SDK 也提供字典（Dictionary）属性，用于支持带门店号（ShopCode）的基站列表，满足其树形结构的数据特性。属性分别为：

工作基站字典：Dictionary<string, string> StationWorkingDict

在线基站字典：Dictionary<string, List<string>> StationOnlineDict

其 Key 为门店号（ShopCode），Value 为基站 ID 列表。

3 发送数据

3.1 检查数据

在下发数据前，需要确认一下细节：

1. 正确的基站 ShopCode 和 ID，并且基站处于空闲状态；
2. 正确的标签 ID，如果是多个标签 ID，偏移量（跨度）不能超过 0XFFFF（65536），单次通信标签数量存在限制（数据长度限制），具体请参阅 3.3 发送数据中关于标签数量的限制；
3. 正确的标签数据，具体请参见 3.2 准备数据。

3.2 准备数据

上层应用与标签进行通信，下发数据，主要是将文本，图像，和闪灯的指令下发给电子标签，因此SDK为上层应用提供了标签以及标签数据的实体封装类，分别是TagEntity和DataEntity。

标签实体类： TagEntity

TagEntity 为标签的实体抽象类，主要定义如下：

属性	类型	注释
TagID	string	标签 ID
TagType	ESLType	标签类型（参见 6.6 标签类型）
StationID	string	基站 ID
Pattern[1]	Pattern	通信模式（参见 6.2 Pattern）
PageIndex[2]	PageIndex	页码：共 4 页（参见 6.3PageIndex）
DataList	List<DataEntity>	标签数据列表
Status[3]	TagStatus	标签状态
R[5]	bool	红色 LED 灯
G[5]	bool	绿色 LED 灯
B[5]	bool	蓝色 LED 灯
Times[4]	int	闪灯次数
Before	bool	是否先闪灯后刷新屏幕
Token	int	服务号

其中，特定的定义属性如下：

1. Pattern 通信模式是指上层应用需要对标签进行何种通信动作，如：Update1 是指刷新屏幕第一缓存数据，UpdatePart1 是指局部刷新屏幕第一缓存数据（屏幕原有图像数据不会抹除）；
2. PageIndex 页面，标签共有 4 个页面缓存，可以指定具体的页面缓存，而不影响其他页面的图像数据。
3. Status 标签状态，用于 SDK 在返回标签通信结果时使用，主要是成功或者失败，上层应用下发数据时无需对该值赋值；
4. Times 闪灯次数，为-1~32765 次，-1 为常亮，其他为闪烁次数（闪烁频率约为 1 次/秒）；
5. R/G/B 为布尔值，True 为对应颜色的 LED 灯闪烁，False 为对应颜色的灯不闪烁，与 Times 组合使用；
6. Token 为服务号，服务号是指上层应用在下发数据给某个标签的时候，传递了一个值（Flag），比如 1000，标签在通信返回/按键反馈等数据回传时，会带上该值，形成一个数据闭环。服务号的范围 0~65535 次，需要注意的是：如果同一标签的连续两次通信，服务号一致的话，第二次标签不会做任何操作，立即返回成功。

数据实体类： DataEntity

DataEntity 为标签数据的实体抽象类，该类是一个虚类，主要定义如下：

属性	类型	注释
ID[1]	uint	序号
Top	uint	高度坐标（左上为起始点 1）
Left	uint	宽度坐标（左上为起始点 1）

Field[2]	abstract FieldType	数据类型，只读（readonly）
Color	FontColor	颜色（参见 6.7 颜色）
Data[3]	abstract object	数据
InvertColor[4]	bool	是否反色

其中，特定的定义属性如下：

1. ID 序号，用于文本模式下，多个元素的层次定义。如要实现一个带矩形框包围的文本效果，需要定义一个矩形框实体类，和一个文本实体类，并且矩形框的 ID 要小于文本的的 ID；
2. Field 数据类型，该类型为只读类型，由具体派生子类实现赋值；
3. Data 数据，尽管为 object 类型，但是 SDK 存在装箱操作，因此赋值需要遵守实际的字段类型的约束，如 ImageEntity，其 Data 复制只有可能是一个 Bitmap 图像类，
4. InvertColor 对于文本类型 TextEntity 来说，是进行反色设置，对条形码 Barcode 来说，是是否显示条形码文本值。

具体继承 DataEntity 的实体类如下：

类	适用
BarcodeEntity[1]	条形码数据
ImageEntity[2]	点阵图像数据
ExImageEntity[3]	带图像增强处理的点阵图形数据
LineEntity	直线
RectangleEntity[4]	矩形
PriceEntity	价格类型数据
QrcodeEntity	二维码图像数据
TextEntity	文本类型数据

3.3 数据实体

1. 关于 **BarcodeEntity** 的特有属性：

属性	类型	注释
Height	int	高度为 16 像素~88 像素。默认为 16 像素

仅当设置 BarcodeType 为 Code128Ext,Code39Ext,EAN13Ext,EAN8Ext 的时候，该属性才会生效。

2. 关于 **ImageEntity** 的特有属性：

属性	类型	注释
Gray	int	灰阶阈值（0-255），默认值为 125
R	int	红色（Red）阈值
G	int	绿色（Green）阈值
B	int	蓝色（Blue）与之

通过设置 Gray 属性，可以调整对黑白图片灰阶的阈值；

通过设置 R、G、B 属性，可以调整对彩色图片 RGB 三色的灰阶与之。

3. 关于 **ExImageEntity** 的特有属性：

属性	类型	注释
Gray	int	灰阶阈值（0-255），默认值为 125
R	int	红色（Red）阈值

G	int	绿色（Green）阈值
B	int	蓝色（Blue）与之
ExImage	FiledType	图像抖动算法，枚举类型

通过设置 Gray 属性，可以调整对黑白图片灰阶的阈值；
通过设置 R、G、B 属性，可以调整对彩色图片 RGB 三色的灰阶与之；
通过设置 ExImage 属性，可以调整图像抖动算法的方式。
如下图所示，是三个 2.13 英寸标签（参见附录 6.8 图像抖动算法）的图片显示效果：

005EF7、005E91 带有图像抖动算法，005E90 采用黑白灰阶阈值二分法

附：原图

4. 关于 **RectangleEntity** 的特有属性：

属性	类型	注释
Height	int	矩形框高度（像素）
Width	int	矩形框宽度（像素）

特别的，可以设置 Data 属性，其格式为“高度|宽度”，如高度为 90 像素，宽度为 70 像素，那么 Data 设置为“90|70”（不含引号），其效果与设置 Height=90, Width=70 的效果是一致的。SDK 优先检查 Data 属性，再检查 Height 和 Width 属性。

如下图所示，是一个 2.9 英寸标签（参见附录 6.6 标签类型）的布局示例效果图：

共包含 3 个 DataEntity 元素：

Index	Data	Field	Font	InvertColor	Top	Left	Color
1	12345abcde	Text	u12px	False	1	1	Black
2	电子标签 Hello World	Text	u16px	False	20	20	Black
3	1234567890	Barcode	Code128	False	50	1	Black

注：电子标签条形码 Code128 所生成的最后一个字符 A 是检验符号，条码扫描枪会自动忽略该字符，不影响实际使用。

上层应用借助于 SDK 公布的屏幕布局属性，达到用户所期待的显示效果。

5. 关于 **PickNumberEntity** 的特有属性：

属性	类型	注释
Data	int	PTL4:限定 0~9999 四位十进制数值 PTL290X:限定数值范围 0~65535，当超过 65535 的时候，按键反馈不能准确反馈数值。
AutoClean	bool	按 OK 键后是否自动清屏
LorR	bool	文本方向：左对齐（true）或者右对齐（false）

注：需要搭配 PTL 无线通信基站，和按键监听基站。

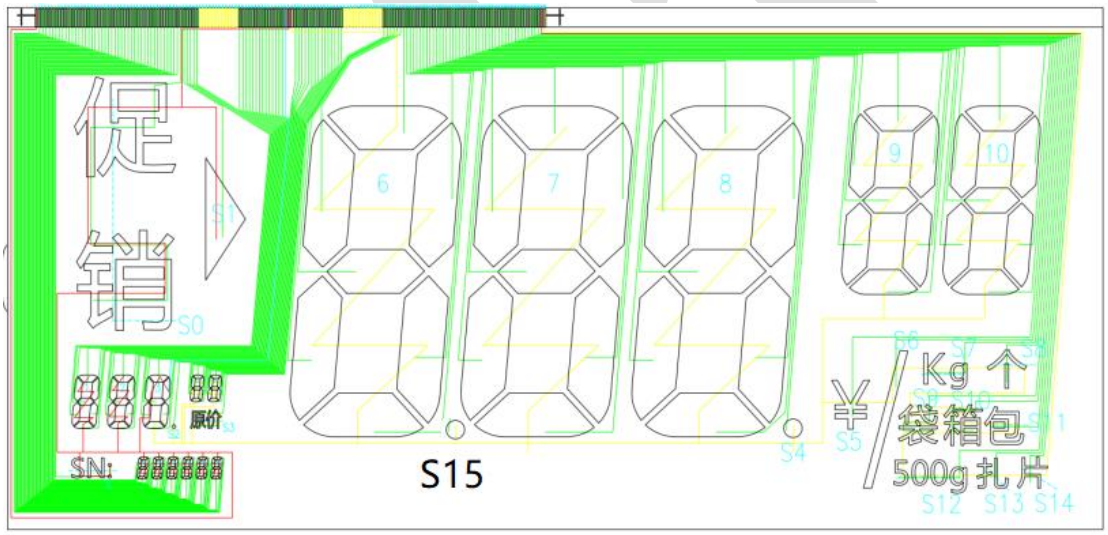
注：PTL 属于快速标签，其闪灯次数为 0.25/秒，受硬件限制，速度不可调整。

6. 关于 **SEG680Entity** 的特有属性：

属性	类型	注释
BigPrice	string	大价格区域显示内容
SmallPrice	string	小价格区域显示内容
Number	string	序列号数字区域显示内容
BigPriceVisible	bool	大价格区域显示方式：True 显示，False 隐藏
SmallPriceVisible	bool	小价格区域显示方式：True 显示，False 隐藏

NumberVisible	bool	序列号区域显示方式：True 显示，False 隐藏
SN	bool	序列号格区域 SN 显示方式：True 显示，False 隐藏
促销	bool	“促销”符号显示方式：True 显示，False 隐藏
三角	bool	“三角”符号显示方式：True 显示，False 隐藏
原价	bool	“原价”符号显示方式：True 显示，False 隐藏
RMB	bool	“RMB”符号显示方式：True 显示，False 隐藏
Kg_盒	bool	促销符号显示方式：True 显示，False 隐藏
g500	bool	“500g”符号显示方式：True 显示，False 隐藏
个	bool	“个”符号显示方式：True 显示，False 隐藏
袋	bool	“袋”符号显示方式：True 显示，False 隐藏
箱	bool	“箱”符号显示方式：True 显示，False 隐藏
包	bool	“包”符号显示方式：True 显示，False 隐藏
扎	bool	“扎”符号显示方式：True 显示，False 隐藏
片_捆	bool	“片”符号显示方式：True 显示，False 隐藏
Slash	bool	反斜杠符号显示方式：True 显示，False 隐藏
AutoBigger	bool	大价格区域当价格小于 10.00 时，自动使用大数字

注：SEG680 属于区域定制化产品，仅限中文市场。



Kg变为盒
片变为捆

3.4 发送数据

在使用本节列出的方法之前，需要符合 3.1，3.2 的约束。

对于点阵（或局部点阵），SDK 会对传入的图像进行剪裁处理，使其满足指定标签类型的屏幕像素大小，同时也会进行去灰阶处理（固定阈值法二值化）。因此最终显示到标签屏幕上的数据可能会与传入图像有所偏差，特别是出现明显的锯齿和部分灰色像素点的丢失。

对于码字，由于受硬件的制约，目前只支持中文/英文内容。

需要注意的是，在进行数据发送时，由于实际通信能力的限制，即 SDK 在对传入的数据进行打包后，会检查整个数据长度是否超过范围。一般来说，2.9 寸标签纯文本模式下发数据，其上限在 800 个，2.9 寸标签在全屏点阵图像模式下发数据，其上限约在 20 个。

当单个标签（码字模式下）数据量过大的时候，SDK 会返回 Overflow 错误。

当下发数据超过基站通信单次能力的时候，SDK 会返回 OutOfMemory 错误。

上层应用在开发过程中，建议根据实际应用场景选择单个或多个发送，单个发送，可以降低整体标签的功耗，并且无线电受噪音干扰概率更低，通信成功率更高，对于大面积整体通信（如商超价签集体调价），则多个发送速度更快。

1. 可选的：信道检测，如果当前区域（该区域是指必须同一 ShopCode 的基站部署在同一个局域网内）存在无线电通信，此时需要检测是否通信繁忙。

`public bool IsBusy(string shopCode)`		
参数	类型	注释
shopCode	string	当前区域门店编号
返回	bool	

2. 可选的：基站是否在线状态检测。

`public bool IsOnline (string shopCode, string stationID)`		
参数	类型	注释
shopCode	string	当前区域门店编号
stationID	string	需检测状态的基站 ID
返回	bool	

3. 可选的：最大分组检测，如果价签数量过大（特别是点阵模式），建议使用此方法获得最大装载能力的组合。

`public List<TagEntity> TryMaxGroup(List<TagEntity> lst)`		
参数	类型	注释
lst	List<TagEntity>	待发标签集合
返回	List<TagEntity>	

4. 单个标签（无门店编号）

`public Result Send(string stationID, TagEntity tagEntity, bool bindStation = false, bool higherPower = false)`		
参数	类型	注释
stationID	string	基站 ID
tagEntity	TagEntity	单个标签实体类
bindStation	bool = false	是否绑定基站，默认否
higherPower	bool = false	是否高功率，默认否
返回	Result	

5. 多个标签（无门店编号）

`public Result Send(string stationID, List<TagEntity> tagList, bool bindStation = false, bool higherPower = false)`		
参数	类型	注释
stationID	string	基站 ID
tagList	List<TagEntity>	多个标签实体类列表 s
bindStation	bool = false	是否绑定基站，默认否

higherPower	bool = false	是否高功率，默认否
返回	Result	

6. 单个标签（带门店编号）

```
public Result Send(string shopCode, string stationID, TagEntity tagEntity, bool bindStation = false, bool higherPower = false)
```

参数	类型	注释
shopCode	string	门店编号
stationID	string	基站 ID
tagEntity	TagEntity	单个标签实体类
bindStation	bool = false	是否绑定基站，默认否
higherPower	bool = false	是否高功率，默认否
返回	Result	

7. 多个标签（带门店编号）

```
public Result Send(string shopCode, string stationID, List<TagEntity> tagList, bool bindStation = false, bool higherPower = false)
```

参数	类型	注释
shopCode	string	门店编号
stationID	string	基站 ID
tagList	List<TagEntity>	多个标签实体类列表 s
bindStation	bool = false	是否绑定基站，默认否
higherPower	bool = false	是否高功率，默认否
返回	Result	

3.5 广播

在使用本节列出的方法之前，需要符合 3.1，3.2 的约束。

注意：广播指令无视基站与标签是否存在逻辑绑定关系，也无通信数量制约，会在基站无线电信号覆盖范围内操作所有标签。

关于 BroadcastOption 广播指令选项的所有选项，请参见 6.9 BroadcastOption。

1. 无门店编号

```
public Result Broadcast(string stationID, BroadcastOption option, bool isHighPower = false)
```

参数	类型	注释
stationID	string	基站 ID
option	BroadcastOption	广播指令选项。
higherPower	bool = false	是否高功率，默认是
返回	Result	

2. 有门店编号

```
public Result Broadcast(string shop, string stationID, BroadcastOption option, bool isHighPower = false)
```

参数	类型	注释
Shop	string	门店编号

stationID	string	基站 ID
option	BroadcastOption	广播指令选项。
higherPower	bool = false	是否高功率，默认是
返回	Result	

3.6 反馈/按键

（本节适用于装载按键的电子标签装载无线电监听模块的通信基站）

在使用本节列出的方法之前，需要符合 3.1，3.2 的约束。

如果选配主动按键反馈基站，则无需显式使用该方法，即可通过结果返回事件 ResultEventHandler 自动获得按键数据（ResultType.Monitor）。

1. 无门店编号

public Result Feedback(string stationID)		
参数	类型	注释
stationID	string	基站 ID
返回	Result	

2. 有门店编号

public Result Feedback(string shop, string stationID)		
参数	类型	注释
shop	string	门店编号
stationID	string	基站 ID
返回	Result	

特别的，反馈方法无需占用无线电信道资源，而且是并发的。

3.7 绑定基站

注意：该方法已经过时，现有的硬件节能水平已经可以忽略该方法。推荐使用 Send 方法里的参数 bindStation= true 代替。

在某些场景下，无线通信基站 AP 和电子标签的数量都很多。正常通信时，为了不会影响其他标签（主要处于节约电量考虑），标签只会与当前绑定的基站通信。

因此，需要遵循如下步骤操作：

1. 设置 Pattern 为 Bind，不含数据，与标签通信；
2. 标签通信成功之后，使用 Update1 等正常发送模式携带数据下发；

备注：与 3.5 节的区别在于，3.3 节中的 bindStation 参数不会改变当前通信标签的基站 ID（无视绑定模式），此处的 Pattern=Bind 会改变当前标签的基站 ID。

当前标签的基站 ID 是指，每个标签的固件参数里，有一个参数用于配置基站的 ID，只有是该 ID 的基站与标签通信，标签才会通信成功。

4 返回数据

4.1 基站事件

基站在上线和离线时都会触发基站事件，上层应用可以通过注册该事件，能够及时掌握场内基站状况，合理调度基站进行通信。

StationEventArgs 主要包括如下属性：

属性	类型	描述
ShopCode	string	门店编号
StationID	string	基站 ID
IP	IPAddress	基站 IP 地址
Port	Int	基站 TCP Socket 端口号
Online	bool	是否上线，True 为上线，False 为离线
StationType	StationType	基站类型，Data 为数据基站，Monitor 为监听基站

注意：如果你的应用场景搭载监听基站，需要借助 StationType 来区分基站类型。监听基站不可用于主动发送数据，只能用于被动接受按键反馈数据。如果向监听基站发送数据，将会导致基站异常离线。

4.2 标签数据事件

在上层应用通过 SDK 与基站通信之后，基站会异步地返回标签数据。主要包括标签是否成功响应，以及服务号、温度、电量和信号强度。

ResultEventArgs 主要包括如下属性：

属性	类型	描述
ShopCode	string	门店编号
StationID	string	基站 ID
ResultType	ResultType	结果返回类型，参见 6.9 ResultType
ResultList	IList<ResultEntity>	单个标签返回结果列表

对于单个标签返回数据 ResultEntity 包括如下属性：

属性	类型	描述
StationID	string	基站 ID
TagID	string	标签 ID
TagStatus	TagStatus	标签状态
Signal	int	信号强度（-256~0，-256 意味着失败）
Temperature	int	环境温度
PowerLevel	Power	电池电量等级（分为 Empty/ Lower/ Half/ High/ Full，一般来说，电量等级为 Empty 或者 Lower 的时候需要更换电池）
PowerValue	decimal	电池电量
Token	int	服务号
KeyType	KeyType	按键反馈类型，None 为未按,OK 为 OK 按键,FN 为 FN 按键

PtlNumber	int	仅限 PTL 标签，搭载可返回数据

一般来说，上层应用需要关注 TagStatus 为失败的标签，重新调度补发（有限次数）。

对于搭载有按键反馈的标签，如果调用广播指令，获取按键反馈的数据（Feedback）时，同一个标签至多可以返回 6 个内容一致的重复数据。

特别的，对于群控广播指令等通信，基站会返回一个 ID 为 000000 的标签，表明基站应答成功。

5 日志

日志功能是对 log4net 的简单封装，所有方法和级别亦遵循 log4net 的定义。

只有当启动项目中的 App.config/Web.config 或者其他配置文件中配置了 log4net 的部分时，日志功能才会生效。

具体方法如下：

方法	参数
LogHelper.Warn(string message)	message 消息内容
LogHelper.Error(string message, Exception ex)	message 消息内容 ex 异常
LogHelper.Error(string message)	message 消息内容
LogHelper.Infor(string message)	message 消息内容
LogHelper.Debug(string message)	message 消息内容

具体请参考：http://logging.apache.org/log4net/

项目可以通过 NuGet 命令获取：Install-Package log4net

6 附录

6.1 Result

SDK 所有返回 Result 类型的函数，其对应的返回值如下：

Result 值	可能情况
InvalidStationID	不正确的基站 ID
InvalidTagID	不正确的标签 ID
EmptyData	发送数据为空
InvalidCount	发送的标签数量超出范围
InvalidServiceCode(InvalidToken)	不正确的服务号
InvalidFlashCount	不正确的 RGB LED 灯闪烁次数
DuplicateTagID	重复的标签 ID
InvalidOffset	不正确的偏移量，最小标签 ID 与最大标签 ID 偏移量超过 0xFFFF
UnregisteredStation	没有注册的基站，试图与一个没有上线的基站发送数据
AccessDenied	访问已经被拒绝

StationBusy	基站忙，当前存在正在工作的基站
ServerBusy	没有空余的信道资源
Fail	发送失败
UnexpectedFailure	基站状态异常
OutOfMemory	下发数据量过大，请减少装载的标签数据量
Overflow	下发数据量过大，请减少单个标签的数据量
OK	成功

6.2 Pattern

适用于开发者使用的 Pattern 具体如下：

Pattern 值	说明
UpdateDisplay	更新屏幕，并刷屏
UpdateDisplayPart	局部更新屏幕，并刷新
Update	更新换成，不刷屏
Display	显示第一缓存
Frosted	磨砂
Bind	绑定基站（见 3.5 节）
DisplayInfor	显示标签信息
Clean	清除屏幕内容
NoChange	不作任何更变，只通信一次（点名/点卯）

6.3 PageIndex

标签共有 4 页缓存，分别定义为：

- P0　　第 1 页
- P1　　第 2 页
- P2　　第 3 页
- P3　　第 4 页

6.4 标签状态

标签状态有如下几种状态，通常使用的只有 Success 和 Failed：

- Unknown　　未知状态
- Success　　成功
- Failed　　失败
- Timeout　　超时
- ERRE1　　错误 1
- ERRE2　　错误 2
- ERRE3　　错误 3
- ERRE4　　错误 4
- ERRE5　　错误 5

- ERRE6　　错误 6

6.5 字段类型

字段	字段类型	备注
TextSize	u7px	仅支持英文和数字 CJK is none, others 7*5
	u12px	CJK is 12*12; others 12 dot unequal width
	u16px	CJK is 16*16; others 16 dot unequal width
	u24px	CJK is 24*24; others 24 dot unequal width
	u32px	CJK is 32*32; others 32 dot unequal width
	u48px	CJK is 48*48; others 48 dot unequal width
	u64px	CJK is 64*64; others 64 dot unequal width
	u96px	CJK is 96*96; ASCII 96 dot unequal width, only in 12.48inch
	u128px	CJK is 128*128; ASCII 128 dot unequal width, only in 12.48inch
	u160px	CJK is 160*160; ASCII 160 dot unequal width, only in 12.48inch
	u192px	CJK is 192*192; ASCII 192 dot unequal width, only in 12.48inch
	u256px	CJK is 256*256; ASCII 256 dot unequal width, only in 12.48inch
	u320px	CJK is 320*320; ASCII 320 dot unequal width, only in 12.48inch
	u384px	CJK is 384*384; ASCII 384 dot unequal width, only in 12.48inch
PriceSize	p24_12px	(24*12)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p28_14px	(28*14)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p32_16px	(32*16)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p36_18px	(36*18)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p40_20px	(40*20)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p48_24px	(48*24)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p56_28px	(56*28)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p24_9px	(24*9)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p32_12px	(32*12)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p40_15px,	(40*15)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p48_18px	(48*18)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p56_21px,	(56*21)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p64_24px,	(64*24)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p80_30px,	(80*30)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p96_36px,	(96*36)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p112_42px,	(112*42)-0/1/2/3/4/5/6/7/8/9/./¥/$/€

	p128_48px,	(128*48)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p144_48px,	(144*48)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p160_48px,	(160*48)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
	p192_48px,	(192*48)-0/1/2/3/4/5/6/7/8/9/./¥/$/€
BarcodeType	EAN8	26*85
	EAN13	26*113
	Code128	32 dot unequal width
	Code39	32 dot unequal width
	EAN8D	Enlarged to double
	EAN13D	Enlarged to double
	Code128D	Enlarged to double
	Code39D	Enlarged to double
	EAN8Ext	bar length define by next data
	EAN13Ext	bar length define by next data
	Code128Ext	bar length define by next data
	Code39Ext	bar length define by next data
LineType	Hline	Horizontal Line-Length define by data
	Vline	Vertical Line-Length define by data
ImageType	Image	Bitmap
	ImageX2	Bitmap with double zoom
	ImagePart	Bitmap(part)
QrcodeType	Qrcode	Qrcode
	QrcodeX2	Qrcode with double zoom
NumberSize	n32_16px	(32*16)-0/1/2/3/4/5/6/7/8/9
	n48_24px	(48*24)-0/1/2/3/4/5/6/7/8/9
	n64_24px	(64*24)-0/1/2/3/4/5/6/7/8/9
	n64_32px	(64*32)-0/1/2/3/4/5/6/7/8/9
	n128_48px	(128*48)-0/1/2/3/4/5/6/7/8/9
	n144_48px	(144*48)-0/1/2/3/4/5/6/7/8/9
	n160_48px	(160*48)-0/1/2/3/4/5/6/7/8/9
	n192_48px	(192*48)-0/1/2/3/4/5/6/7/8/9

6.6 标签类型

上层应用将数据发送给电子标签之前，还需要考虑不同类型电子标签屏幕的显示范围(SEG680 属于段码屏，显示内容固定)：

ESLType	颜色	屏幕尺寸（英寸）	高 H（px）	宽 W （px）
ESL154R ESL154	黑黑白白红	1.54	152	152
ESL213 ESL213R ESL213Y	黑白 黑白红 黑白黄	2.13	104	212

ESL290 ESL290Y ESL290R PTL290 PTL290X	黑白 黑白黄 黑白红 黑白 黑白	2.90	128	296
ESL420 ESL420R ESL420Y	黑白 黑白红 黑白黄	4.20	300	400
ESL750R ESL750Y ESL750	黑白红 黑白黄 黑白	7.50	384	640
ESL1250R	黑白红	12.50	894（码字） 492（点阵）	1304(码字) 652(点阵)
ESL154RH	黑白红	1.50	200	200
ESL213RH	黑白红	2.13	122	250
ESL266R	黑白红	2.66	152	296
ESL580R	黑白红	5.80	480	684
PTL4	-	2.90	-	-
BEC01	-	-	-	-

备注：ESL1250R 需要区分码字和点阵两种发送方式，并且基站固件版本必须为 71 或更高版本。

6.7 图像抖动算法

枚举 ExImageType 支持如下图像抖动算法：

- FloydSteinbergDithering 有序抖动（佛洛依德算法）
- BurksDithering 伴有巴克斯误差扩散的色彩抖动
- JarvisJudiceNinkeDithering 贾维斯司法
- StuckiDithering 斯图基抖动
- Sierra3Dithering 锯齿 X3
- Sierra2Dithering 锯齿 X2
- SierraLiteDithering 雪崩处理
- AtkinsonDithering 阿特金森（推荐）
- RandomDithering 上述算法随机选择

6.8 BroadcastOption

广播会控制基站通信能力范围之内的所有标签，并且不会返回具体每个标签的通信结果。

- DisplayInfor 所有标签显示当前版本信息
- Clean 所有标签清空屏幕，并不抹除数据
- DisplayCache0 所有标签显示第1页数据
- DisplayCache1 所有标签显示第2页数据
- DisplayCache2 所有标签显示第3页数据

- DisplayCache3　　所有标签显示第4页数据
- DisplayID　　所有标签显示当前的ID（Code39条码）
- Feedback　　获取监听/按键反馈数据

6.9 ResultType

- SendData　　常规发送数据返回
- Broadcast　　广播返回
- Feedback　　查询侦听返回
- Monitor　　主动侦听返回
- Heartbeat　　心跳