Document version	1.17
.NETversion	1.5.1
.NET Coreversion	2.3.3
Release date	2020-03-10

D11 Electronic Shelf Label System Developer's Manual

User ID:
Item Number:

SDK developer subscription plan：

www.sertag.com leaker@sertag.com
Our agents and third-party developers can email your project information and contact information to the above address in order to get timely information such as the SDK version update.

Version history

Version number	Date	Description	Author	Review
1.0	2017-01-06	Document initialization	Haipeng Huang	Haipeng Huang
1.1	2017-02-22	The new version ofprotocol to modify the number of LEDlight sparkling	Haipeng Huang	Haipeng Huang
1.2	2017-04-18	Compatible codeword, dot matrix transmission mode full screen, partial refresh Repackage the SDK interface	Haipeng Huang	Haipeng Huang
1.3	2017-04-20	The new version of protocoladd supports for the channel lights	Haipeng Huang	Haipeng Huang
1.4	2017-10-27	The new version of protocoladd supports for the color screens	Haipeng Huang	Haipeng Huang
1.5	2018-01-12	Response to the super or retail scenario, instructions for adding a bonded base station change the way QR code is displayed	Haipeng Huang	Haipeng Huang
1.6	2018-04-21	Increase the check for the data overload Add error code description	Haipeng Huang	Haipeng Huang
1.7	2018-04-28	Add a new version of the base station (with store number) support Increase image processing dithering algorithm Increase the grayscale threshold dichotomy to set the threshold, RGB can set the threshold Add a rectangle	Haipeng Huang	Haipeng Huang
1.8	2018-06-28	Update font (price part)	Haipeng Huang	Haipeng Huang
1.9	2018-09-19	Note: Some are not forward compatible!!! Enhanced attribute constraint range Add .NET Core version	Haipeng Huang	Haipeng Huang
2.0	2018-11-13	Alteration: Coordinate system parameter rename Increase support for button labels and active listening base stations	Haipeng Huang	Haipeng Huang
1.11	2019-05-05	Added data overflow checks Add combinatorial optimal solution (greedy algorithm) Adds DataEntity construct injection Added PTL290, ESL1250R support	Haipeng Huang	Haipeng Huang
1.12	2019-07-03	Add SDK work model Add type of base station	Haipeng Huang	Haipeng Huang
1.13	2019-07-25	add 4 sizes of label	Haipeng Huang	Haipeng Huang
1.14	2019-10-18	Add key feedback type and PTL pick number	Haipeng Huang	Haipeng Huang

1.15	2019-12-16	Details optimization, version consistency confirmation	Haipeng Huang	Haipeng Huang
1.16	2020-03-09	Add PTL290X function Optimization default value	Haipeng Huang	Haipeng Huang
1.17	2020-03-26	Add ESL437R, ESL750RP, ESL1160R	Haipeng Huang	Haipeng Huang

Content

1 Brief

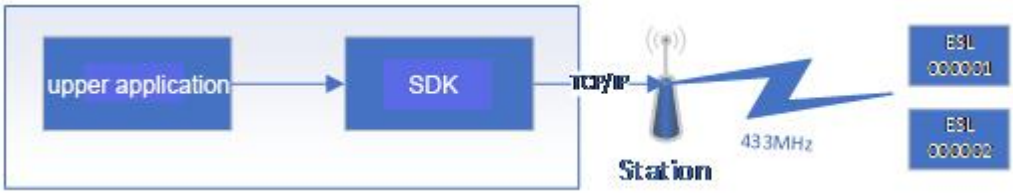
This document is for projects based on the .NET Framework 4.0 or higher.

This document is for projects based on .NET Core 2.1 or higher.

1.1System structure

The electronic label system can be easily divided into three parts: the upper application, the SDK and the hardware device. The upper application refers to the application program oriented to the user's business logic, such as WMS, SAP/ERP, HIS, etc.; the SDK is responsible for abstracting hardware functions and providing them to the upper application, and packaging the upper application data to the hardware; the hardware includes wireless communication base stations, electronic labels and related network devices.

The overall structure of the electronic label system is divided into the following views as above-mentioned.



Specific explanation of nouns:

A base station is a wireless transmitter in an electronic label system. The SDK is commonly referred to a Station, or an AP. The base station contains two identity attributes: ShopCode and ID. The ShopCode ranges from 0x0000 to 0xFFFF, and the ID ranges from 0x00 to 0xFF. Under the same ShopCode , the base station ID cannot be duplicated, otherwise there will be conflicted.

The scenario of the ShopCode application applies to a multiple shop or multi-warehouses. In this case, the SDK can be driven to concurrent operations on different stores.

A label is an electronic label (ESL). The label contains an identity attribute: ID. The label ID is 6-bit or 8-bit hexadecimal ID. The first two digits of the 8-bit ID are label types. For details, please refer to the appendix.

Note: The SDK does not contact with the label directly, but instead contacts with the base station.

1.2 About the SDK

The SDK currently offers two versions of .NET and .NET Core, all posted on NuGet:

1. .NET version:

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

2. .NET Core version:

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

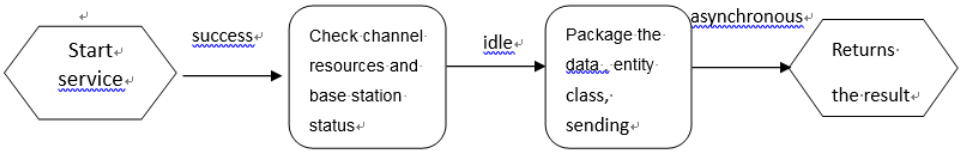
Unless otherwise noted, the relevant content in this document applies to both the .NET version and the .NET Core version.

The SDK provides all access methods/properties to the developer by Server.Instance through the single instance class pattern.

The SDK provides developers with the necessary concepts as follows:

1. Start the service;
2. An interface issued to the electronic label data;
3. Asynchronously accept the data reverse back from the base station by registering the event;
4. Layout concept of electronic label screen data;
5. SDK related properties

The following is a typical sending process,



The SDK mainly emphasizes two concepts: asynchronous, multi-threaded. Asynchronous refers to the communication between the SDK and the hardware, which is asynchronous. Multi-threading means that the SDK communicates with multiple base stations is multi-threaded. However, in the same environment, it is subject to radio electromagnetic wave collision, and the communication between the base station and the label is single-threaded.

1.3 Before the start

Before using the SDK for development, you need to confirm:

- Please confirm the SDK version, electronic label firmware version and base station firmware version with the sales personal/agent/technical support personnel.;
- The hardware device and network environment are ready, including, such as, the device parameters have been configured correctly, the network environment is normal, the radio electromagnetic environment is permitted, and the communication distance between the electronic label and the wireless base station is within a reasonable range;
- The operating system environment is configured correctly, and the security software and firewall installed on the computer allow the upper-layer application to access the port (default setting is 1234).

2 Initialization

2.1 Start service

The code for start the service is simple:

```
Server.Instance.Start(intport = 1234)
```

SDK service is started by the default mode, with a default port value of 1234.

If your network configuration SDK listens for other ports, you need to change the port value.

The default return value for starting the service is:

- `Resuslt.OK`：The startup is successful;
- `Result.StartFailed`：The startup failed.

2.2 Registration event

Since the SDK returns the base station and the label data asynchronously,so the upper layer application is notified by the registration event to accept the return data of the base station and the label.

For a base station, the base station registration event is:

```
publiceventEventHandler<StationEventArgs> StationEventHandler
```

For labels, the label result return event is:

```
publiceventEventHandler<ResultEventArgs> ResultEventHandler
```

For details on the registration event, see 4. Processing Return Data.

2.3 Base station status

The base station has three states: Offline, Online, and Working.

The SDK uses the GetStationWorkingList and GetStationOnlineList methods to indicate the list of currently working base stations and the list of currently free base stations.

Only when the base station is free can the upper application communicate with it through the SDK.

1. Query the list of base stations currently working (generally returning a single)

`public List<string>GetStationWorkingList(string shop = "")`		
parameter	Types	Comment
shop	`string`	Store number, Must be specified
return	Base station ID list, Empty means no	

2. Query the list of base stations that are currently free

`public List<string>GetStationOnlineList(string shop = "")`		
Parameter	Types	Comment

shop	string	Store number, Must be specified
return	Base station ID list, Empty means no	

The SDK also provides a Dictionary attribute to support a list of base stations with a ShopCode that satisfies the data characteristics of its tree structure. The attributes are:

```
Dictionary<string, List<string>> StationWorkingDict
Working base station dictionary
Dictionary<string, List<string>> StationOnlineDict
Online base station dictionary
```

The Key is the store code (ShopCode) and the Value is the base station ID list.

3 Sending data

3.1 Checking the data

Before you send the data, you need to confirm the details:

1. The correct base station ShopCode and ID, and the base station is free;
2. The correct label ID, if it is multiple label IDs, the offset (span) cannot exceed 0XFFFF (65536), and there is a limit on the number of single-communication labels (data length limit). For details, please refer to 3.3 Sending data about the number limit of the label;
3. Correct label data. For details, see 3.2 Preparing Data.

3.2 Preparing data

The upper layer application communicates with the label, and the data is sent, and the text, the image, and the flashing instructions are sent to the electronic label mainly. Therefore, the SDK provides the entity wrapper class of the label and the label data for the upper layer application, respectively, TagEntity and DataEntity.

Tag entity class： `TagEntity`

TagEntity is the entity abstract class of the tag, mainly defined as follows

Attributes	Types	Comment
TagID	string	Tag ID
TagType	ESLType	Label type (see 6.6 Label Type)
StationID	string	Base station ID
Pattern[1]	Pattern	Communication mode (see 6.2 Pattern)
PageIndex[2]	PageIndex	Page number: 4 pages in total (see 6.3PageIndex)
DataList	List<DataEntity>	label data list
Status[3]	TagStatus	Label status
R[5]	bool	Red LED light

G[5]	bool	Green LED light
B[5]	bool	Blue LED light
Times[4]	int	Number of flashes
Before	bool	Whether to refresh the screen after flashing first
Token	int	Service number

Among them, the specific definition attributes are as follows:

1. The Pattern communication mode refers to the communication action that the upper layer application needs to perform on the label. For example, Update1 refers to refreshing the first cache data of the screen, and UpdatePart1 refers to partially refreshing the first cache data of the screen (the original image data of the screen will not be erased);
2. PageIndex page, the label has a total of 4 page cache, you can specify a specific page cache, without affecting the image data of other pages.
3. Status label status, used by the SDK to return the label communication result, mainly success or failure, the upper layer application does not need to assign a value to it when sending the data;
4. Times flashing times, from -1 to 32,765 times, -1 is always on, others are flashing times (flashing frequency is about 1 time/second)；
5. R/G/B is a Boolean value, True is the LED of the corresponding color flashing, and False is the corresponding color of the light not flashing, and used in combination with Times;
6. Token is the service number. The service number is when the upper application sends a value to a label, and a value (Flag) is sent, such as 1000. The label is carried when the data is returned in the communication return/key feedback. This value forms a closed loop of data. The service number ranges from 0 to 65535. It should be noted that if the same label has two consecutive communication and the service number is the same, the second label will not perform any operation and will return immediately.

Data entity class:DataEntity

DataEntity is the entity abstract class of the label data. This class is a virtual class and is mainly defined as follows

Attributes	Types	Comment
ID[1]	uint	Serial number
Top	uint	Height coordinate (top left is the starting point 1)
Left	uint	Width coordinate (top left is the starting point 1)
Field[2]	abstractFieldType	Data type, read only
Color	FontColor	Color (see 6.7 colors)
Data[3]	abstractobject	data
InvertColor[4]	bool	Whether it is reversed

Among them, the specific definition attributes are as follows

1. ID serial number, used in text mode, hierarchical definition of multiple elements. To implement a text effect surrounded by a rectangular box, you need to define a rectangular box entity class, and a text entity class, and the ID of the rectangle is smaller than the ID of

the text;

2. Field It is data type, which is read-only and is assigned by a concrete derived subclass.

3. Data , although it is an object type, but the SDK has a boxing operation, so the assignment needs to comply with the actual field type constraints, such as ImageEntity, its Data copy can only be a Bitmap image class.

4. InvertColor is the inverse color setting for the text type TextEntity, and whether the barcode text value is displayed for the Barcode.

The entity class that inherits DataEntity specifically is as follows

class	Applicable scenario
BarcodeEntity[1]	Barcode data
ImageEntity[2]	Dot image data
ExImageEntity[3]	Dot matrix graphic data with image enhancement processing
LineEntity	straight line
RectangleEntity[4]	rectangle
PriceEntity	Price type data
QrcodeEntity	QR code image data
TextEntity	Text type data

3.3 Data entity

1. About the unique properties of BarcodeEntity

Attributes	Types	Comment
Height	int	The height is 16 pixels to 88 pixels. The default is 16 pixels

This property only takes effect when the BarcodeType is set to Code128Ext, Code39Ext, EAN13Ext, EAN8Ext.

2. About the unique properties of ImageEntity

Attributes	Types	Comment
Gray	int	Grayscale threshold (0-255), default is 125
R	int	Red threshold
G	int	Green threshold
B	int	Blue threshold

By setting the Gray property, you can adjust the threshold of grayscale for black and white images.

By setting the R, G, and B properties, you can adjust the grayscale threshold of the three colors of the color picture RGB.

3. About the unique properties of ExImageEntity

Attributes	Types	Comment
Gray	int	Grayscale threshold (0-255), default is 125
R	int	Red threshold
G	int	Green threshold
B	int	Blue threshold

ExImage	FiledType	Image dithering algorithm, enumerated type

By setting the Gray property, you can adjust the threshold of the grayscale of the black and white image;

By setting the R, G, and B properties, you can adjust the grayscale threshold of the three colors of the color picture RGB.

By setting the ExImage property, you can adjust the way the image dithering algorithm works.

As shown in the figure below, the picture display effect of three 2.13 inch labels (see Appendix 6.8 Image Dithering Algorithm):

The label marked 005EF7, 005E91 are deal with image dithering algorithm, and the label marked 005E90 is deal with black and white gray-scale threshold dichotomy

Attachment: the original image

4. Unique properties about RectangleEntity

Attributes	Types	Comment
Height	int	Height of rectangle (pixels)
Width	int	Width of Rectangle (pixels)

In particular, you can set the Data property, the format is "height | width", such as height is 90 pixels, width is 70 pixels, then Data is set to "90|70" (without quotes), the effect setting Height = 90 and Width=70 is the same. The SDK first checks the Data property and then checks the Height and Width properties.

As shown in the figure below, it is a layout example of a 2.9 inch label (see Appendix 6.6 Label Type):

Here are 3 DataEntity elements

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

Note: The last character A generated by the electronic label barcode Code128 is the check symbol, and the barcode scanner will automatically ignore the character without affecting the actual use.

The upper application uses the screen layout properties what is published by the SDK to achieve the desired display effect.

5. Unique properties about **PickNumberEntity**:

Attributes	Types	Comment
Data	int	PTL4: Limit to 0-9999 four - digit decimal values PTL290X: Limit the value range to 0~65535, when more than 65535,

		the key feedback can not accurately feedback the value.
AutoClean	bool	Press OK to clear the screen automatically
LorR	bool	Text orientation: align left (true) or align right (false)

Note: PTL wireless communication base station and key monitoring base station are needed.

Note: PTL is a fast label, the number of flash is 0.25/ SEC, the speed can not be adjusted due to the hardware limit.

3.4 Sending data

Before using the methods listed in this section, you need to comply with the constraints of chapter 3.1, 3.2.

For dot matrix (or local dot matrix), the SDK clips the incoming image to fit the screen pixel size of the specified tag type, and also performs grayscale processing (fixed threshold binarization). Therefore, the data that is ultimately displayed on the label screen may deviate from the incoming image, especially the apparent sawtooth and the loss of some gray pixels.

For code words, due to hardware constraints, only Chinese/English content is supported currently.

It should be noted that when data transmission is performed, due to the limitation of the actual communication capability, the SDK checks whether the entire data length exceeds the range after packaging the incoming data. In general, the 2.9 inch label is sent in plain text mode with an upper limit of 800. The 2.9 inch label sends data in full-screen bitmap mode, with an upper limit of about 20.

When a single tag (in code mode) has too much data, the SDK returns Overflow errors.

When the data is sent out of the base station communication single-capacity, the SDK will return an Out Of Memory error.

In the development process of the upper layer, it is recommended to select single or multiple transmissions according to the actual application scenario, and a single transmission can reduce the power consumption of the overall tag, and the radio is less subject to noise interference, and the communication success rate is higher, for large-area overall communication. (If the merchant over-prices collectively adjust the price), multiple transmissions are faster.

1. 1. Optional: channel detection. If there is radio communication in the current area (which means that the base station of the same ShopCode must be deployed in the same LAN), it is necessary to detect whether the communication is busy.

public bool IsBusy(string shopCode)		
parameter	Types	Comment
shopCode	string	The current region Shop code
return	bool	

2. Optional: check whether the base station is online.

public bool IsOnline (string shopCode, string stationID)		
parameter	Types	Comment
shopCode	string	The current region Shop code

stationID	string	Need to detect the base station ID
return	bool	

3. Optional: maximum group detection. If the number of price tags is too large (especially in lattice mode), it is recommended to use this method to obtain the combination of maximum loading capacity.

public List<TagEntity> TryMaxGroup(List<TagEntity> lst)		
parameter	Types	Comment
lst	List<TagEntity>	collection of tags for sending
return	List<TagEntity>	

4. Single label (no shop code)

publicResult Send(string stationID, TagEntity tagEntity, bool bindStation = false, bool higherPower = false)		
parameter	Types	Comment
stationID	string	Base station ID
tagEntity	TagEntity	Single label entity class
bindStation	bool = false	bind basestation, default is no
higherPower	bool = false	High power, default is no
return	Result	

5. Multiple labels (no shop code)

publicResult Send(string stationID, List<TagEntity> tagList, bool bindStation = false, bool higherPower = false)		
parameter	Types	Comment
stationID	string	Base station ID
tagList	List<TagEntity>	Multiple labels entity class
bindStation	bool = false	bind base station, default is no
higherPower	bool = false	High power, default is no
return	Result	

6. Single label (with shop code)

publicResult Send(string shopCode, string stationID, TagEntity tagEntity, bool bindStation = false, bool higherPower = false)		
parameter	Types	Comment
shopCode	string	Shop code
stationID	string	Base station ID
tagEntity	TagEntity	Single label entity class
bindStation	bool = false	bind basestation, default is no
higherPower	bool = false	High power, default is no
return	Result	

7. Multiple labels (with shop code)

publicResult Send(string shopCode, string stationID, List<TagEntity> tagList, bool bindStation = false, bool higherPower = false)		
parameter	Types	Comment

shopCode	string	Shop code
stationID	string	Base station ID
tagList	List<TagEntity>	Multiple labels entity class
bindStation	bool = false	bind base station, default is no
higherPower	bool = false	High power, default is no
返回	Result	

3.5 Broadcast

Before using the methods listed in this section, you need to comply with the constraints of chapter 3.1, 3.2.

Note: The broadcast command ignores whether the base station and the tag have a logical binding relationship, and there is no restriction for communication quantity, and all labels are operated within the coverage of the base station radio signal.

For all command options of the BroadcastOption, please see the chapter 6.9 Broadcast Option.

1. No shop code

publicResultBroadcast(string stationID, BroadcastOption option, bool isHighPower = false)		
parameter	Types	Comment
stationID	string	Base station ID
option	BroadcastOption	Option of Broadcast
higherPower	bool = false	High power, default is no
return	Result	

2. With shop code

publicResultBroadcast(stringshop, string stationID, BroadcastOption option, bool isHighPower = false)		
Parameter	Types	Comment
Shop	string	Shop code
stationID	string	Base station ID
option	BroadcastOption	Option of Broadcast
higherPower	bool = false	High power, default is no
return	Result	

3.6 Feedback / Button

(This section is available to the base station where the labels with button and loaded monitoring module)

Before using the methods listed in this section, you need to comply with the constraints of chapter 3.1, 3.2.

If an active keypad feedback base station is selected, this method is not required to be explicitly used to automatically obtain keypad data (resulttype.monitor) through the result return event ResultEventHandler.

1. No shop code

publicResultFeedback(string stationID)		
parameter	Types	Comment
stationID	string	Base station ID
return	Result	

2. With shop code

publicResultFeedback(stringshop, string stationID)		
parameter	Types	Comment
shop	string	Shop code
stationID	string	Base station ID
return	Result	

In particular, the feedback method does not require radio channel resources and it is parallel.

3.7 Binding the base station

Note: this method is out of date, and the current level of hardware savings is negligible. The parameter bindStation= true in the Send method is recommended instead.In some scenarios, the number of base stations (APs)and labels is large. In normal condition, in order to avoid affect the other labels (mainly in the case of saving power), the label will only communicate with the currently bound base station.

Therefore, you need to follow the steps below:

1. Set Pattern to Bind, without data, and communicate with the tag;
2. After the label is successfully communicated, the data is sent using the normal sending mode such as Update1.

Note: The difference from chapter 3.5 is that the bindStation parameter in chapter 3.3 does not change the base station ID of the current communication label (ignoring the binding mode). Here, Pattern=Bind will change the base station ID of the current label.

The base station ID of the current label means that one of the firmware parameters of each label is used to configure the base station ID only this base station communicates with the label, the label will be successfully communicated.

4 Return data

4.1 Base station event

The base station will triggers the base station event when it goes online and offline. The upper layer application can register the event, and can timely grasp the status of the base station in the field and reasonably schedule the base station to communicate.

StationEventArgs mainly includes the following attributes:

Attributes	Types	Description

ShopCode	string	Shop code
StationID	string	Base station ID
IP	IPAddress	Base station IP address
Port	Int	Base station TCP Socket port number
Online	bool	Whether it is online, True is online, False is offline
StationType	StationType	Base station type: Data is Data base station, and Monitor is listening base station

Note: if your application scenario is equipped with a listening base station, you will need to distinguish the base Station type by StationType. The listening base station cannot be used to send data actively, but can only be used to passively accept the feedback data of keys. If data is sent to the listening base station, it will cause the base station to go offline.

4.2 Label Data Events

After the upper layer application communicates with the base station through the SDK, the base station asynchronously returns the label data. It mainly includes whether the label responds successfully, as well as the service number, temperature, power, and signal strength.

ResultEventArgs mainly includes the following attributes

Attributes	Types	Description
ShopCode	string	Shop code
StationID	string	Base station ID
ResultType	ResultType	The result return type, see 6.9 ResultType
ResultList	IList<ResultEntity>	Single label returns result list

Returning data for a single label ResultEntity includes the following

Attributes	Types	Description
StationID	string	Base station ID
TagID	string	Label ID
TagStatus	TagStatus	Label status
Signal	int	Signal strength (-256~0, -256 means failure)
Temperature	int	Ambient temperature
PowerLevel	Power	Battery level (divided into Empty / Lower / Half / High / Full, in general, the battery needs to be replaced when the power level is Empty or Lower)
PowerValue	decimal	Battery power

Token	int	Service number
KeyType	KeyType	Key feedback type: None is not pressed,OK is OK, and FN is FN
PtlNumber	int	PTL tag only, with return data

In general, the upper application needs to pay attention to the TagStatus as a failed label and reschedule the reissue.

For a label equipped with button feedback, if a broadcast command is called to obtain the feedback data (Feedback), the same label can return at most 6 duplicate data with the same content.

In particular, for communication such as group control broadcast commands, the base station returns a label with an ID number 000000, indicating that the base station responds successfully.

5 Logs

This chapter applies to the distribution of the SDK with the log function, and it need to reference eTagTech.Logger.

The logging function is a simple wrapper around log4net, and all methods and levels follow the definition of log4net.

The log function will only take effect when the log4net part is configured in the App.config/Web.config or other configuration files in the startup project.

The specific method is as follows:

method	parameter
LogHelper.Warn(string message)	message
LogHelper.Error(string message, Exception ex)	message ex abnormal
LogHelper.Error(string message)	message
LogHelper.Infor(string message)	message
LogHelper.Debug(string message)	message

For details, please refer to:http://logging.apache.org/log4net/

The project can be obtained by the NuGet command: Install-Package log4net

6 Appendix

6.1 Result

The SDK returns all functions of the Result type, and the corresponding return values are as follows:

Result Value	Possible situation
InvalidStationID	Incorrect base station ID
InvalidTagID	Incorrect label ID
EmptyData	Send data is empty
InvalidCount	The number of tags sent is out of range
InvalidServiceCode(InvalidToken)	Incorrect service number
InvalidFlashCount	Incorrect RGB LED flashing times
DuplicateTagID	Duplicate tag ID
InvalidOffset	Incorrect offset, minimum tag ID and maximum tag ID offset exceed 0xFFFF
UnregisteredStation	Unregistered base station, trying to send data with a base station that is not online
AccessDenied	Access has been denied
StationBusy	The base station is busy, there is currently a working base station
ServerBusy	No free channel resources
Fail	Failed to send
UnexpectedFailure	Base station status is abnormal
OutOfMemory	The amount of data sent is too large, please reduce the amount of tag data loaded.
Overflow	The amount of data sent is too large, please reduce the amount of tag data loaded.
OK	success

6.2 Pattern

The pattern for developers to use is as follows:

Pattern value	Description
UpdateDisplay	Update the screen and swipe
UpdateDisplayPart	Partially update the screen and refresh
Update	Update to change, do not swipe
Display	Display first cache
Frosted	Scrub
Bind	Bind the base station (see chapter 3.5)
DisplayInfor	Display label information
Clean	Clear screen content
NoChange	Do not make any changes, only communicate once (Call the roll)

6.3 PageIndex

The label has a total of 4 pages of cache, which are defined as:

- P0 Page 1
- P1 Page 2
- P2 Page 3
- P3 Page 4

6.4 Label Status

The tag status has the following statuses. Usually Success and Failed are used only:

- Unknown
- Success
- Failed
- Timeout
- ERRE1
- ERRE2
- ERRE3
- ERRE4
- ERRE5
- ERRE6

6.5 Field type

Field	Field type	Remarks
TextSize	u7px	Only English and numbers are supported 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 Label Type

Before the upper application sends the data to the electronic label, it also needs to consider the display range of different types of electronic label screens (SEG680 belongs to the segment code screen, and the display content is fixed):

ESL Type	color	Screen size (inches)	High H(px)	Width W (px)
ESL154R ESL154	Black white red Black white	1.54	152	152
ESL213 ESL213N ESL213R ESL213Y	Black white Black white Black white red Black white yellow	2.13	104	212
ESL290 ESL290Y ESL290R PTL290 PTL290X	Black white Black white yellow Black white red Black white Black white	2.90	128	296
ESL420 ESL420R ESL420Y	Black white Black white red Black white yellow	4.20	300	400
ESL437	Black white red	4.37	176	480
ESL750R ESL750RP ESL750Y ESL750	Black white red Black white red Black white yellow Black white	7.50	384	640
ESL1160R	Black white red	11.60	640	960
ESL1250R	Black white red	12.50	894（code） 492（pix）	1304(code) 652(pix)
ESL154RH	Black white red	1.50	200	200
ESL213RH	Black white red	2.13	122	250
ESL266R	Black white red	2.66	152	296

ESL580R	Black white red	5.80	480	684
PTL4	-	2.90	-	-
BEC01	-	-	-	-

Note: ESL1250R needs to distinguish between codeword and bitmap, and the base station firmware version must be 71 or higher.

Developers need to verify that the label is a black and white screen or a color screen.

6.7 Image Jitter Algorithm

The enumeration ExImageType supports the following image dithering algorithms:

- `FloydSteinbergDithering`
- `BurksDithering`
- `JarvisJudiceNinkeDithering`
- `StuckiDithering`
- `Sierra3Dithering`
- `Sierra2Dithering`
- `SierraLiteDithering`
- `AtkinsonDithering`
- `RandomDithering`

6.8 BroadcastOption

The broadcast controls all labels within the range of communication capabilities of the base station and does not return the communication results for each of the labels.

- `DisplayInfor` All labels display current version information
- `Clean` All labels clear the screen and do not erase data
- `DisplayCache0` All labels display the data of page 1
- `DisplayCache1` All labels display the data of page 2
- `DisplayCache2` All labels display the data of page 3
- `DisplayCache3` All labels display the data of page 4
- `DisplayID` All labels display `the label ID（Code39 barcode）`
- `Feedback` Get listening/keystroke feedback data

6.9 ResultType

- `SendData`
- `Broadcast`
- `Feedback`
- `Monitor`
- `Heartbeat`