Version	0.4
Date	2023-06-21
Firmware	1.1.4

D20 eStation IoT Device Developer Manual

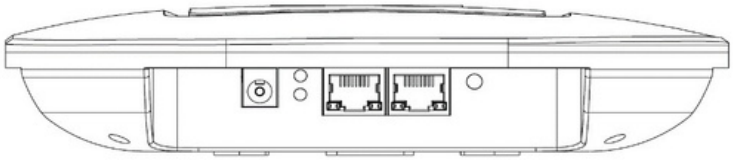
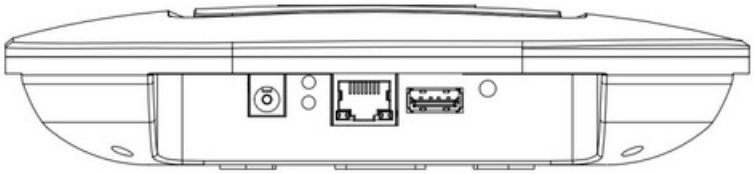
Contents

1 Introduce

1.1 Background

This eStation IoT Device, type code AP-04, which is designed for system integrator to develop quickly for their business project, using MQTT protocol.
Before you start with eStation, please carefully know that difference between AP03 and eStation (AP04). AP03 is the previous generation, it works with the middleware Cronus project, developers should work with the middleware via WebAPI or code level integration. eStation does not have any software level API, it only provides a hardware level API with MQTT protocol. In another world, AP03 works with Cronus, eStation directly works with your application.

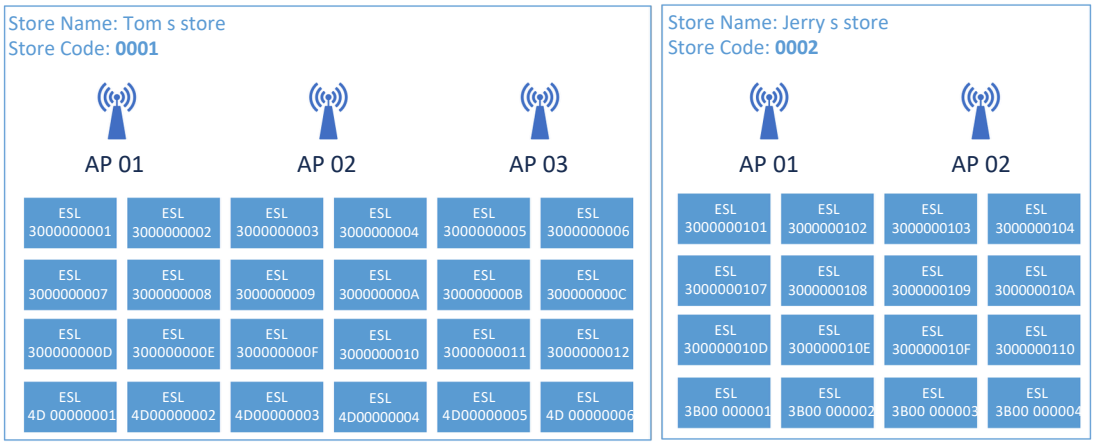
The appearance of AP03 and eStation:

AP03	
eStation (AP04)	

eStation has a WPF demo project in GitHub. The address is: https://github.com/andersonhwang/estation.
And will also provide Rust, Python and Java console demo.
Before you start to use eStation, you should understand bellowing keywords:

1. **Group**: If you have multiple APs in a store, and you have more than one stores. If one ESL communication failed, you may need call all AP in the store to try communication with the failed ESL. In this case, you should put these APs in one store. You should manage the AP and Group relationship in your application.
2. **AP**: The radio frequency (RF) access point which connect to the labels.
3. **AP ID**: The identification of AP, eStation always use MAC as its ID.
4. **Tag**: The electronical shelf label (ESL) or Pick-to-Light (PTL).
5. **Base64String**: Convert an image to Base64 string. For more information about Base64 string, please refer to Base64 Encode and Decode - Online.
6. **MQTT**: eStation (AP04) is using the MQTT communication protocol.
7. **MessagePack**: eStation (AP04) is using the MessagePack data format to reduce data package length. For more information about MessagePack, please refer to MessagePack: It's like JSON. but fast and small. (msgpack.org).

For example, your project has two stores, looks like:



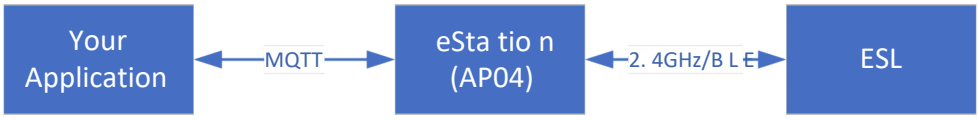
The data structure should be:

Store Code	AP ID	Tag ID
0001	01	30000001, 30000002, 30000007, 30000008, 3000000D …
	02	30000003, 30000004, 30000009, 3000000A, 3000000F …
	03	30000005, 30000006, 3000000B, 3000000C, 30000011 …
0002	01	30000101, 30000102, 30000107, 30000108, 3000010D …
	02	30000103, 30000104, 30000109, 3000010A, 3000010F …

Note: Normally, both tags and APs are belonging to a store, but tags are not static belonging to any AP. The tag just tries to remember the last default AP ID which has successfully send data to it.

1.2 System Architecture of eStation

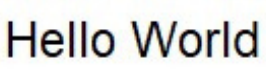
Basically, your project sends a task which include a tag ID and an image data (Base-64 string) to the eStation, and the eStation will send the image data to the exactly label with 2.4GHz radio frequency (RF), also the eStation returns the result to your project. In another words, the eStation is between your application and the labels.



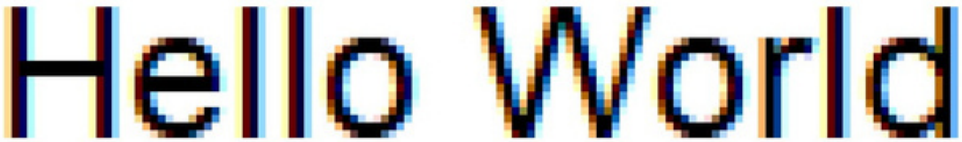
This document will describe 2 sides: [1]your application side and [2]eStation side. Remember, if your application and the eStation are not in a same private network, youshould add a strong password to protect the connection. And its better to addan X.509certificate.

1.3 About the Image

The ESL screen is a Black-White or Black-White-Red color screen, no gray color. For example, your image looks like:



Zoom in 800% looks like:



And after remove gray scale parts, it looks like:

The larger the font size, the clearer the jagged shape.

So, it's not a good idea to using labels to show images, if you want to do that, you may need dithering the image to looks like smoothly.

An algorithm for dithering images using C# is: Even more algorithms for dithering images using C# - Articles and information on C# and .NET development topics • Cyotek

For example, an image looks like:

With dithering and without dithering the image effect looks like:

2 Start with eStation

As said before, there is no software-level integration with eStation. You should manage the connection, communication in your application.

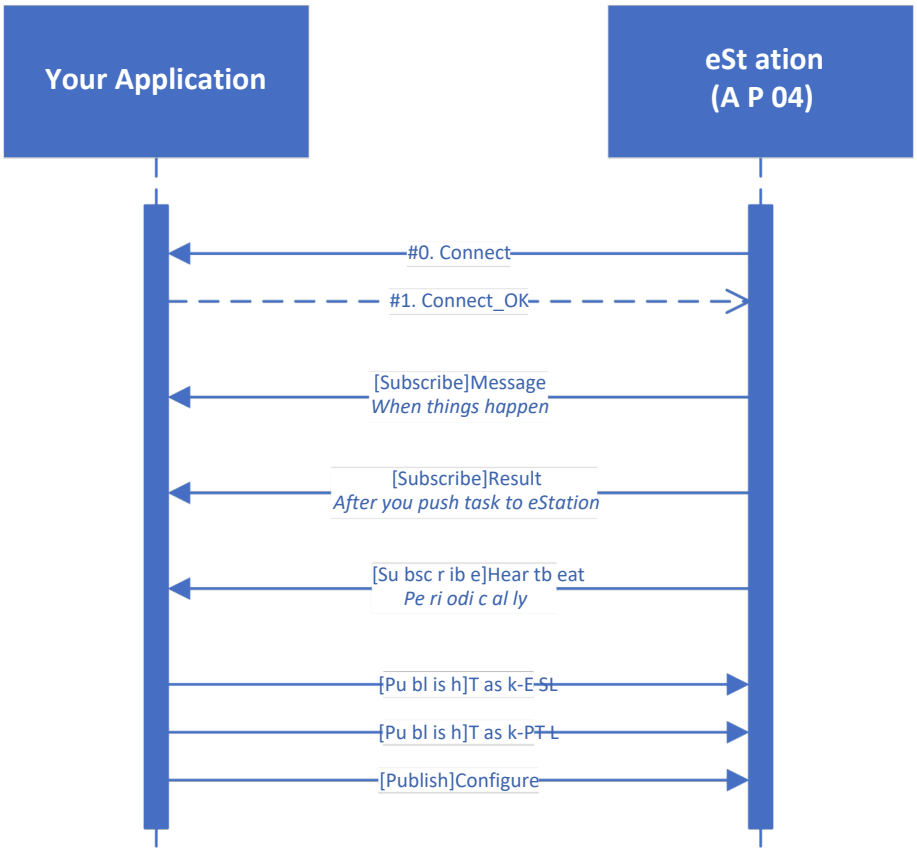
There are 5 topics, 3 topics are published from eStation side, and 6 topics are published from your application side.

From eStation side, your application should subscribe below topics:

- **Message**: eStation will return some messages when things happened.
- **Result**: eStation will return the result after its communication with ESL.
- **Heartbeat**: eStation will periodically return currently status.

From your application side, the eStation will subscribe below topics:

- **Task ESL**: You can publish ESL tasks with this topic.
- **Task PTL**: You can publish PTL tasks with this topic.
- **Configure**: You can configure eStation with this topic.



Note: Publish/Subscribe is from your application side

2.1 Connect to eStation

eStation is working as a IoT device client, it will try to connect to your application (serve side) after it is powered on.

The default parameters of eStation are:

Default target server address	192.168.1.92:9080
Default user name	test
Default password	123456

Note: Please change user name and password as soon as possible.

After connect to one eStation, you can use configuration topic to modify it (Please refer to 2.7 Publish Configuration Information).

However, if you cannot connect to the eStation, or you forget its target server address, or you forget the user name/password, you can press the reset button for a few seconds, after you see the REST and boot display, it will restore to default target server IP address, user name and password.

Note: if you have reset the eStation, but it still cannot connect to your application, please check your computer firewall and the ethernet cable line.

2.2 Receive eStation message

Topic: /estation/{ID}/message

ID: The ID of eStation.

PayloadSegment: Bytes of message, ASCII encoding. After you received this message, you should use ASCII decoding to get the message.

This topic is a help type topic, you can display the message in your application page.For example, if long time no ESL task received, eStation will send message "*Long_Time_No_Task*" with this topic.

2.3 Receive eStation result

Topic: /estation/{ID}/result
ID: The ID of eStation.
PayloadSegment: Bytes of TaskResult object, MessagePack Serialize. This TaskResult object contains a list of TagResult object.

The **TaskResult** property is:

#	Property	Type	Remark
0	ID	String	AP ID, AP MAC
1	TotalCount	Int	Total tasks count in the cache
2	SendCount	Int	Sending tasks count in the antenna module
3	Tags	List<TagResult>	Results list

And the **TagResult** properties are:

#	Property	Type	Remark
0	TagID	String	Tag ID
1	Version	Int	ESL firmware version, keep in future
2	Screen		Screen factory code, keep in future
3	RfPower		RF power, dBm, null/-256 means no data
4	Battery		Battery, Voltage, null/0 mean no data
5	Temperature	Int	Temperature, ℃, null means no data
6	Token	Int	Token, data close cycle, refer to 2.5 Token

Note: The eStation has a queue inside, when your application sends task data into eStation, it will send to the antenna immediately if the antenna is idle, or it will wait untilthe antenna is idle if it is working. So, the communication between your application and eStation is asynchronously.

2.4 Receive eStation heartbeat

Topic: /estation/{ID}/heartbeat
ID: The ID of eStation.
PayloadSegment: Bytes of **eStationInfor** object, MessagePack Serialize.
The **eStationInfor** property is:

#	Property	Type	Remark
0	ID	String String	ID, read-only
1	MAC	String	Device MAC address, read-only
2	Alias	Int Endpoint	Alias name of device, friendly view
3	ClientType	String String	Default is 2
4	ServerAddress	Array	Server address, Endpoint(IP + port) Default is: 192.168.1.192:9080
5	Parameters		Connection parameters, user name, and password.
6	LocalIP	IP String	Local IP address, empty means auto obtain an IP address from router.
7	SubnetMask	IP String	Subnet mask
8	Gateway	String	Gateway
9	Heartbeat	String	Heartbeat speed
10	DummyVersion	String	Communication engine version
11	AppVersion	String	eStation firmware version
12	TotalCount	String	Total count of tasks in cache
13	SendCount	Int	Total count of tasks in antenna

2.5 Publish ESL tasks

Topic: /estation/{ID}/ taskESL
ID: The ID of eStation.
PayloadSegment: Bytes of ESLEntity object, MessagePack Serialize.
The **ESLEntity** property is:

#	Property	Type	Remark
0	TagID	String	AP ID, AP MAC
1	Pattern	Int, Pattern	Pattern code
2	PageIndex	Int, PageIndex	Page index
3	R	Bool	Red color LED light
4	G	Bool	Green color LED light
5	B	Bool	Blue color LED light
6	Times	2 Bytes	LED light flashing times, second
7	Token	2 Bytes	Token, will return in 2.3 Token
8	OldKey	8 Bytes	Old key, keep empty if use default key
9	NewKey	8 Bytes	New key, keep empty if not need to set
10	Base64String	String	Image Base64 string

Overwrite: For one ESL ID, if your application does not wait its previous task result feedback, and continue send task data to it, the eStation will drop its previous task.

For more information of old key and new key, please refer to 2.6 Security of ESL.

2.6 Security of ESL

Basically, if some other project uses their eStation (AP) to control your project ESL, unexcepted and unaccepted result will happen. In this case, it is recommended that you should set a secret key for your ESL.

When you received new ESLs from your sales manager, the default secret key of ESL is empty (FFFFFFFFFFFFFFFF), you can keep it or set both old key and new key. After that, your communication should use the new key.

1. The fixed length of secret key is 16 (8 Bytes).
2. Only hexadecimal characters (0123456789ABCDEF) are acceptable.
3. If old key is empty, eStation will use default secret key (FFFFFFFFFFFFFFFF) to replace.
4. If new key is empty, eStation will not change the secret key. If new key is not empty, eStation will change the secret key.
5. VERY IMPORT: Please safety storage your secret key, if you forget the secret, you will lose control your ESLs.

2.7 Publish PTL tasks

Topic: /estation/{ID}/ taskPTL
ID: The ID of eStation.
PayloadSegment: Bytes of PTLEntity object, MessagePack Serialize.
The **ESLEntity** property is:

#	Property	Type	Remark
0	TagID	String	AP ID, AP MAC
1	R	Bool	Red color LED light
2	G	Bool	Green color LED light
3	B	Bool	Blue color LED light
4	Beep	Bool	Beeper
5	Times	2 Bytes	LED light flashing times, second
6	Token	2 Bytes	Token, will return in 2.3 Token

Overwrite: For one PTL ID, if your application does not wait its previous task result feedback, and continue send task data to it, the eStation will drop its previous task.

2.8 Publish configure information

Topic: /estation/{ID}/config
ID: The ID of eStation.
PayloadSegment: Bytes of **eStationInfor** object, MessagePack Serialize.
The **eStationInfor** property is:

#	Type Property		Remark
0	ID	String String	ID, read-only
1	MAC	String	Device MAC address, read-only
2	Alias	Int Endpoint	Alias name of device, friendly view
3	ClientType	String String	Default is 2
4	ServerAddress	Array	Server address, Endpoint (IP + port) Default is: 192.168.1.192:9080
5	Parameters		Connection parameters, user name, and password.
6	LocalIP	IP String	Local IP address, empty means auto obtains an IP address from router.
7	SubnetMask	IP String	Subnet mask, empty if LocalIP is empty
8	Gateway	String	Gateway, empty if LocalIP is empty
9	Heartbeat	String	Heartbeat speed, default 15, seconds.
10	DummyVersion	String	Communication engine version, read-only
11	AppVersion	String	eStation firmware version, read-only
12	TotalCount	String	Total count of tasks in cache, read-only
13	SendCount	Int	Total count of tasks in antenna, read-only

3 Reference

3.1 ESL Gen 3.0 Type List

Type	Size (Inch)	Screen	Color	Pixels (H*W)	Freezing
ET0154-30	1.54	Eink	B/W/R	152*152	-
ET0154-31	1.54	Eink	B/W/Y	152*152	-
ET0154-32	1.54	Eink	B/W	152*152	-
ET0154-33	1.54	Eink	B/W/R	200*200	-
ET0213-36	2.13	Eink	B/W/R	250*122	-
ET0213-39	2.13	Eink	B/W	250*122	Y
ET0266-3A	2.66	Eink	B/W/R	296*152	-
ET0266-5B	2.66	Eink	B/W	296*152	Y
ET0290-3D	2.90	Eink	B/W/R	296*128	-
ET0290-3F	2.90	Eink	B/W	296*128	-
ET0290-54	2.90	Eink	B/W	296*128	Y
ET0420-40	4.20	Eink	B/W/R	400*300	-
ET0420-43	4.20	Eink	B/W/R	400*300	-
ET0750-44	7.50	Eink	B/W/R	800*480	-
ET0430-4C	4.30	Eink	B/W/R	522*122	-
ET0580-4F	5.80	Eink	B/W/R	648*480	-
ET0350-55	3.50	Eink	B/W/R	384*184	-
ET1250-58	12.50	Eink	B/W/R	1304*984	-
ET1250-59	12.50	Eink	B/W/Y	1304*984	-
ET1250-5A	12.50	Eink	B/W	1304*984	-
ET420-5D	4.20	TFT	B/W/R	400*300	-
ET0266-5E	2.66	Eink	B/W/R/Y	296*152	-
ET0730-5F	7.30	Eink	7 Colors*	800*400	-
ET0213-60	2.13	Eink	B/W/R/Y	250*122	-
ET0290-61	2.90	Eink	B/W/R/Y	296*128	-
ET0130-62	1.30	TFT	B/W/R	200*144	-
ET1020-64	10.20	Eink	B/W/R	960*640	-

Note: In color column, B means black, W means white, R means red, Y means yellow.

3.2 Pattern of ESL

Pattern Code	Description
0- UpdateDisplay	Update and display
1- UpdatePart	Update part of screen
2- Update	Update
3- Display	Display
4- DisplayInfor	Display label information, no return
5- Query	Query label information
6- Check	Check label if exist
7- LED	Flashing LED lights

3.3 Page Index of ESL

Page Code	Description
0- P0	The 1st page index
1- P1	The 2nd page index
2- P2	The 3rd page index
3- P3	The 4th page index
4- P4	The 5th page index
5- P5	The 6th page index
6- P6	The 7th page index
7- P7	The 8th page index

Note: Currently all ESL only have 8-page data cache.

3.4 Data structure definition (C# language)

eStationInfor:for eStation configuration, parameters and status. Properties ID、MAC、AppVersion、DummyVersion、TotalCount、SendCount are read-only. Its C# language definition looks like:

```
using MessagePack;
[MessagePackObject]
public class eStationInfor
{
        /// <summary>
        /// Client ID - readonly
        /// </summary>
        [Key(0)]
        public string ID { get; set; }
        /// <summary>
        /// MAC address - readonly
        /// </summary>
        [Key(1)]
        public string MAC { get; set; }
        /// <summary>
        /// Alias, only for display
        /// </summary>
        [Key(2)]
        public string Alias { get; set; }
        /// <summary>
        /// Client type
        /// </summary>
        [Key(3)]
        public int ClientType { get; set; }
        /// <summary>
        /// Server IP + port address
        /// </summary>
        [Key(4)]
        public string ServerAddress { get; set; }
        /// <summary>
        /// Connection parameters with connection type
        /// 0 - NULL
        /// 1 - Certial name, password
        /// 2 - User ID, password
        /// </summary>
        [Key(5)]
        public string[] Parameters { get; set; }
        /// <summary>
```

```
        /// Local IP address, keep empty means auto obtain an IP address
        /// If local IP address is empty, will ignore subnet mask and gateway
        /// </summary>
        [Key(6)]
        public string LocalIP { get; set; }
        /// <summary>
        /// Subnet mask, will use empty when LocalIP is empty
        /// </summary>
        [Key(7)]
        public string SubnetMask { get; set; }
        /// <summary>
        /// Gateway, will use empty when LocalIP is empty
        /// </summary>
        [Key(8)]
        public string Gateway { get; set; }
        /// <summary>
        /// Heartbeat, less than 15 is auto reset to 15
        /// </summary>
        [Key(9)]
        public int Heartbeat { get; set; } = 15;
        /// <summary>
        /// [Readonly]Dummy version
        /// </summary>
        [Key(10)]
        public string DummyVersion { get; set; }
        /// <summary>
        /// [Readonly]App version
        /// </summary>
        [Key(11)]
        public string AppVersion { get; set; }
        /// <summary>
        /// [Readonly]Task in working channel
        /// </summary>
        [Key(12)]
        public int TotalCount { get; set; } = 0;
        /// <summary>
        /// [Readonly]Task in cache
        /// </summary>
        [Key(13)]
        public int SendCount { get; set; } = 0;


}
```

ESLEntity:ESL task entity, C# definition code looks like:

```
using MessagePack;
/// <summary>
/// ESL entity
///                 </summary>
[MessagePackObject]
public class ESLEntity {


        /// <summary>
        /// Tag ID
        /// </summary>
        [Key(0)]
        public string TagID { get; set; } = "";
        /// <summary>
        /// Pattern, default is Update + Display
        /// </summary>
        [Key(1)]
        public Pattern Pattern { get; set; } = Pattern.UpdateDisplay;
        /// <summary>
        /// Page index, default is the 1st page data cache
        /// </summary>
        [Key(2)]
        public PageIndex PageIndex { get; set; } = PageIndex.P0;
        /// <summary>
        /// Red LED light
        /// </summary>
        [Key(3)]
        public bool R { get; set; } = false;
        /// <summary>
        /// Green LED light
        /// </summary>
        [Key(4)]
        public bool G { get; set; } = false;
        /// <summary>
        /// Blue LED light
        /// </summary>
        [Key(5)]
        public bool B { get; set; } = false;
        /// <summary>
        /// LED light flashing times
        /// </summary>
        [Key(6)]
        public int Times { get; set; } = 0;
        /// <summary>
```

```
        /// Service code, 0~65535(0XFFFF)
        /// </summary>
        [Key(7)]
        public int Token { get; set; } = 0;
        /// <summary>
        /// The old key
        /// </summary>
        [Key(8)]
        public string OldKey { get; set; } = string.Empty;
        /// <summary>
        /// The new key
        /// </summary>
        [Key(9)]
        public string NewKey { get; set; } = string.Empty;
        /// <summary>
        /// Image data, Base64 string
        /// </summary>
        [Key(10)]
        public string Base64String { get; set; } = string.Empty;

}
```

PTLEntity:PTL task entity, C# definition code looks like:

```
///    <summary>
/// PTL entity ///
</summary>
 [MessagePackObject]
public class PTLEntity
{
        /// <summary>
        /// Tag ID
        /// </summary>
        [Key(0)]
        public string TagID { get; set; } = "";
        /// <summary>
        /// Red LED light
        /// </summary>
        [Key(1)]
        public bool R { get; set; } = false;
        /// <summary>
        /// Green LED light
        /// </summary>
        [Key(2)]
        public bool G { get; set; } = false;
        /// <summary>
        /// Blue LED light
        /// </summary>
        [Key(3)]
        public bool B { get; set; } = false;
        /// <summary>
        /// Beep
        /// </summary>
        [Key(4)]
        public bool Beep { get; set; } = false;
        /// <summary>
        /// LED light flashing times
        /// </summary>
        [Key(5)]
        public int Times { get; set; } = 0;
        /// <summary>
        /// Service code, 0~65535(0XFFFF)
        /// </summary>
        [Key(6)]
        public int Token { get; set; } = 0;


}
```

TaskResult:A group of ESL/PTL communication results list:

```
using MessagePack;
/// <summary>
/// Task result entity
/// </summary>
 [MessagePackObject]
public class TaskResult
{
        /// <summary>
        /// AP ID
        /// </summary>
        [Key(0)]
        public string ID { get; set; } = string.Empty;
        /// <summary>
        /// Total count
        /// </summary>
        [Key(1)]
        public int TotalCount { get; set; }
        /// <summary>
        /// Send count
        /// </summary>
        [Key(2)]
        public int SendCount { get; set; }
        /// <summary>
        /// Tag results list
        /// </summary>
        [Key(3)]
        public List<TagResult> Tags { get; set; } = new List<TagResult>();

}
```

TagResult: A single ESL/PTL communication result.

```
usingMessagePack;
[MessagePackObject] ///
<summary>
/// Tag result
/// </summary>
public class TagResult {

        /// <summary>
        /// Tag ID
        /// </summary>
        [Key(0)]
        public string TagID { get; set; }
        /// <summary>
        /// Version
        /// </summary>
        [Key(1)]
        public string Version { get; set; }
        /// <summary>
        /// Screen
        /// </summary>
        [Key(2)]
        public string Screen { get; set; }
        /// <summary>
        /// RSSI
        /// </summary>
        [Key(3)]
        public int RfPower { get; set; } = -256;
        /// <summary>

        /// Battery    Power
        /// </summary>
        [Key(4)]
        public int Battery { get; set; } = 0;
        /// <summary>
        /// Temperature
        /// </summary>
        [Key(5)]
        public int Temperature{get;set;}=0;
        /// <summary>
        /// Token
        /// </summary>
        [Key(6)]
        public int Token { get; set; } = 0;

}
```