File version	1.0
API version	1.0
eRetail version	3.0.0
Release Date	2022-03-10

D18 eRetail 3.0 System Integration Technical Manual

Version history

version	date	description	author	check
1.0	2022-03-10	Document initialization	anderson	anderson

Content

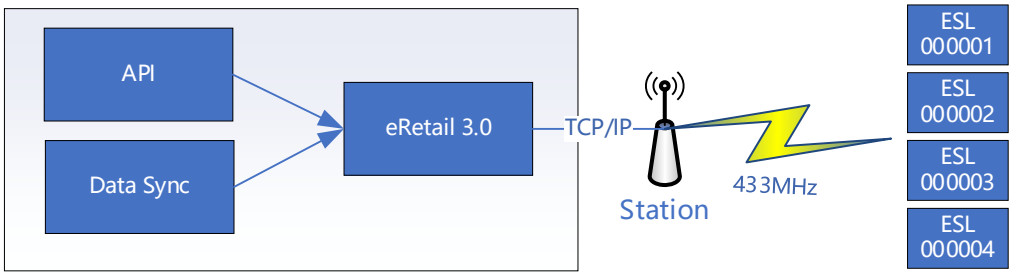
1 Resume

This document is applicable to based on Net 6.0 or later.
This document is applicable to the third generation ESL products (ESL gen3).

1.1 System Structure

eRetail 3.0 electronic shelf label system (eRetail 3.0) can be simply divided into three parts: [1]DataSync / API、[2]eRetail 3.0 和 [3]hardware devices. Among them, API is Restful style web API, which is responsible for providing interface for customer system to actively push commodity data; Data sync is a background service that actively grabs data from the customer system (such as database, WebAPI, Excel, etc.); SDK is responsible for managing commodity data, hardware equipment information, scheduling price change tasks etc. The hardware includes wireless communication base station, tag and related network equipment.

eRetail 3.0 overall structure is divided into the following views according to the above:



This document is a file that describing the API interface of eRetail 3.0.

2 Interface

It is assumed that the deployment address of eRetail 3.0 is 192.168.1.92 and the port is 5000.

Note: HTTPS is recommended instead of HTTP.

For details, see **Enforce HTTPS in ASP.NET Core | Microsoft Docs**。

2.1 Interface Test

The Purpose is to check whether eRetail 3.0 API can be accessed.

HTTP GET

URL: http:// 192.168.1.92:5000/api/hello

Respond: "OK"

Note: this interface is used for connectivity testing, informal scenarios.

2.2 Handshake interface

The purpose is that the interface is used for authentication. Access to all subsequent interfaces depends on the data obtained by this interface.

HTTP POST
URL：http:// 192.168.1.92:5000/api/login
Content-Type: application/json

Request parameter:

parameter	type	description
userName	String	User name
password	String	password

Return format：

parameter	type	description
code	Int	0：success，other：error
message	String	Success or error message
body	Json Node	Message body
token	String	subsequent sessions Token

Note: after obtaining the token, you need to add this content to the header of subsequent HTTP requests. as"Authorization: Bearer {token}"

example：
Request

```
{
  "userName": "admin",
  "password": "Pass99"
}
```

Response

```
{
  "code": 0,
  "message": "success",
  "body": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
}
```

2.3 Commodity Data Interface

The purpose is to add and update multiple commodity data.

HTTP POST

URL：http:// 192.168.1.92:5000/api/goods/save

Content-Type: application/json

Request parameter:

parameter	type	description
shopCode	String	We need to create corresponding stores in the system store list first.
template	String	We need to create a corresponding template in the template list first
items	Array String[26]	Commodity data attribute, temporarily fixed to 26 (extensible)

Note: for the creation of stores and templates, please contact sertag sales personnel or pre-sales / after-sales support engineers.

Return format：

parameter	type	description
code	Int	0：success，other：error
message	String	Success or error message
body	Json Node	Message body
token	String	subsequent sessions Token

Note: after obtaining the token, you need to add this content to the header of subsequent HTTP requests. as"Authorization: Bearer {token}"

example：

Request

```
[
  {
    "shopCode": "0001", // Shop code number
    "template": "REG", // tag template：REG-regular ，SAL-promotion，NOR-Out of stock，MER-member customer……
    "items": [
      "A050", // Customer store number
      "123456", // Commodity unique code: it can be commodity number or commodity barcode
      "Goods1", // Commodity name
      "123456789012", //UPC1: General product code
      "123456789013", // UPC2: General product code
      "123456789014", // UPC3: General product code
      "8.98", //price1：Retail price or original price
      "8.95", //price2：promotion price
      "8.96", //price3：member price
      "china", //origin
      "300ml", // Specifications
```

```
        "PC", //unit
        " Qualified ", //grade
        "2021/12/20", //promotion start date
        "2021/12/25", //promotion end date
        "http://www.baidu.com", //QR code
        "leaker", // Price clerk
        "5.1", //inventory
        " expand 1", // Extension field 1, no content left blank null ""
        " expand 2", // Extension field2
        " expand 3", // Extension field 3
        " expand 4",
        " expand 5",
        " expand 6",
        " expand 7",
        " expand 8",
        " expand 9",
        " expand 10"
    ]
  }
]
```

Response

```
{
  "code": 0,
  "message": "success",
  "body": "121f5151fdfffds21cdf"
}
```

Note: by default, the field properties in this example are fixed. In fact, the field content and template of commodity data can be set freely.

2.4 Tag entry interface

This interface is used for batch label entry. This is a special interface for system integration, which is used to integrate eRetail 3.0 into the customer system. If not necessary, you can directly use the background management system or APP of eRetail 3.0.

HTTP POST

URL: http:// 192.168.1.92:5000/api/tag/save

Content-Type: application/json

Request parameter:

parameter	type	description
shopCode	String	As chapter 2.3 named shop code

Id	String	Tag ID

Return format:

parameter	type	description
code	Int	0: success, other: error
message	String	Success or error message
body	String	Default is null

example:

Request


```
[
  {
    "id": "40000000EA39",
    "shopCode": "0001"
  },
  {
    "id": "40000000EA40",
    "shopCode": "0001"
  }
]
```


Response


```
{
  "code": 0,
  "message": "success",
  "body": ""
}
```


2.5 Bind Tag interface

This interface is used to bind the tag ID to the goods ID number. This is a special interface for system integration, which is used to integrate eRetail 3.0 into the customer system. If not necessary, you can directly use the background management system or APP of eRetail 3.0.

HTTP POST

URL: http:// 192.168.1.92:5000/api/tag/bind

Content-Type: application/json

Request parameter:

parameter	type	description
shopCode	String	As chapter 2.3 named shop code
ap	String	Base station ID(AP)
binds	Array	Bind array

tagID	Array String	Tag ID
goodsCode	Array String	Commodity code

Return format：

parameter	type	description
code	Int	0：success，other：error
message	String	Success or error message
body	String	Default is null

example：

Request

```
{
    "shopCode": "0001",
    "ap": "",
    "binds": [{
        "tagID": "40000000EA40",
        "goodsCode": "123456"
    }]
}
```

Response

```
{
  "code": 0,
  "message": "success",
  "body": ""
}
```

2.6 Unbind Tag interface

This interface is used to unbind the tag ID from the commodity number. This is a special interface for system integration, which is used to integrate eRetail 3.0 into the customer system. If not necessary, you can directly use the background management system or AOO of eRetail 3.0.

HTTP POST

URL：http:// 192.168.1.92:5000/api/tag/unbind

Content-Type: application/json

Request parameter:

parameter	type	description
shopCode	String	As chapter 2.3 named shop code
ap	String	Designated base station ID
unbindType	Int	Bind type，0-According to label ID，1-According to commodity code

idList	String Array	Tag ID or commodity number, depending on the unbound type

Return format:

parameter	type	description
code	Int	0: success, other: error
message	String	Success or error message
body	String	Default is null

example:

Request

```
{
   "shopCode": "0001",
   "ap": "",
   "idList": [
      "40000000EA40",
      "40000000EA39"
   ],
   "unbindType": 1//unbindType 1-tag ID, 2-Goods ID
}
```

Response

```
{
   "code": 0,
   "message": "success",
   "body": ""
}
```

2.7 Tag group Bind interface

This interface is used to combine and bind one tag ID with multiple commodity numbers, that is, multiple commodity information can be displayed on one tag. This is a special interface for system integration, which is used to integrate eRetail 3.0 into the customer system. If not necessary, you can directly use the background management system or APP of eRetail 3.0.

HTTP POST

URL: http:// 192.168.1.92:5000/api/tag/groupBind

Content-Type: application/json

Request parameter:

parameter	type	description
shopCode	String	As chapter 2.3 named shop code
ap	String	Designated base station ID

template	String	Template type（layout solution）, such as 2X2 2X3 etc.
tagID	String	Tag ID
goodsCode	Array String	Commodity ID

Return format:

parameter	type	description
code	Int	0: success, other: error
message	String	Success or error message
body	String	Default is null

example:

Request

```
{
   "shopCode": "0001",
   "ap": "",
   "template": "1x2",
   "tagID": "40000000EA40",
   "goodsCode": [
      "123456",
      "123457"
   ]
}
```

Response

```
{
   "code": 0,
   "message": "success",
   "body": ""
}
```

2.8 Tag LED flash interface

This interface is used to notify the Tag to flash. This is a special interface for system integration, which is used to integrate eRetail 3.0 into the customer system. If not necessary, you can directly use the background management system or APP of eRetail 3.0.

HTTP POST

URL: http:// 192.168.1.92:5000/api/tag/led

Content-Type: application/json

Request parameter:

parameter	type	description

shopCode	String	As chapter 2.3 named shop code
rgb	string	LED Color (R-Red,G-green,B-Blue)
times	int	Flash time(S)
idList	Array	Tag or commodity array
ledType	Int	Flash type (1: Designated tag,0: Designated commodity)

Return format:

parameter	type	description
code	Int	0: success, other: error
message	String	Success or error message
body	String	Default is null

example:

Request

```
{
   "shopCode": "0001",
   "rgb": "RGB", //R-Red,G-Green,B-Blue
   "times": 10,
   "idList": [
      "40000000EA40" //ledType 1-tag ID,    0->commodity ID
   ],
   "ledType": 1
}
```

Response

```
{
   "code": 0,
   "message": "success",
   "body": ""
}
```