2024

eRetail 3.2 System Integration Manual

V1.1

DALIAN SERTAG TECHNOLOGY CO., LTD

Version History

Version	Date	Summarize	Author	Approval
1.0	2022-09-06	Document Initialization		
1.1	2024-07-22	Price Tag Refresh Interface and Price Tag Image Refresh Interface Parameter Changes		
1.1	2024-08-06	Translation update		

Catalog

1 Summary

This document applies to projects based on .NET 6.0 or later.

This document applies to projects based on ESL Gen3.

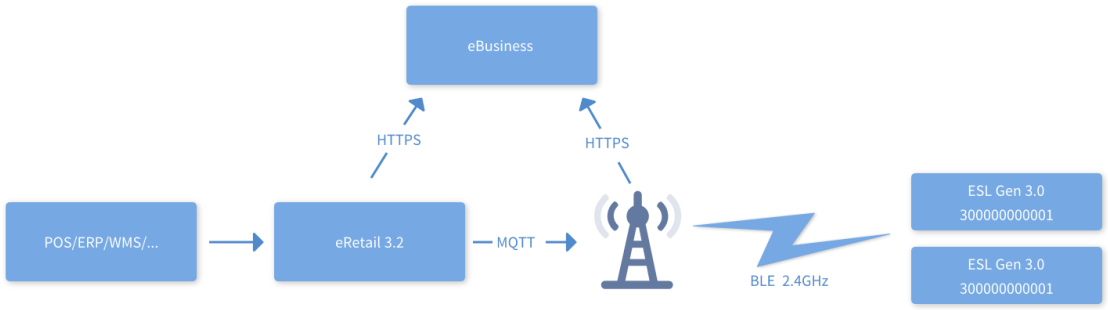
This document applies to projects based on eRetail 3.2 or later.

This document applies to projects based on AP05.

1.1 System Architecture

This document is for the eRetail3.2 system docking method description, eRetail3.2 system supports three docking methods: API interface, file synchronization, and database synchronization. This document focuses on using the API interface, which is a Restful style WebAPI responsible for providing an interface for customer systems to push product data actively.

The overall structure of the eRetail 3.2 e-labeling solution is divided into the following views according to the above:



1.2 Comparison of Systems Integration Options

In general, system integration consists of three aspects: management of data, management of templates, and management of devices. The following table is used to compare the characteristics of the various interfacing solutions so that project managers can further evaluate

the solution options for system integration:

Docking Programme	Docking Direction	Need to Pay Attention	Don't Need to Pay Attention
Web API: Push product data	Customer System->eRetail	Product data content	Template content Equipment management
Web API: Push tag images	Customer System->eRetail	Product data content Template content	Equipment management
1. Data Sync. Provide commodity data	eRetail->Customer System	Merchandise data content Data source: File/DB/FTP	Equipment management

from this, it can be seen that:

1. This solution is applicable when the customer has human resources for development, but the customer only has product data;

2. This solution is applicable when the customer has developed human resources and the customer's system can manage the display of commodity labels. This solution is applied when the customer does not develop human resources, but the customer has commodity data;

2 Electronic Price Tag Interface

Where the text relates to API addresses, it is assumed that eRetail 3.2 is deployed at 192.168.1.92 on port 4000.

Note: HTTPS is recommended over HTTP.

See also Enforce HTTPS in ASP.NET Core | Microsoft Docs Enforce HTTPS in ASP.NET Core | Microsoft Docs.

2.1 Test Interface

Application: To test if the eRetail 3.2 API is accessible.

HTTP GET

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

Return: "OK"

Note: This interface is used for connectivity testing and informal scenarios.

2.2 Handshake Interface

Application: This interface is used for authentication. All subsequent accesses to the interface depend on the data obtained from this interface.

HTTP POST

URL: http:// 192.168.1.92:5000/api/login

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
username	String	User ID
password	String	Password

Return Format:

Parameter	Types	Instruction
Code	Int	0: Success 4004: Username or password is wrong 6001: Role not exists 4003: User not exists
Message	String	Success or error message
Body	String	Token for subsequent sessions

Note: Once get the Token, the header of subsequent HTTP requests needs to add this.

Such as: "Authorisation: Bearer {token}", valid for six hours.

Example:

Request


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


Response


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


2.3 Label Entry Interface

Application: Label batch entry uses this interface. This is a system integration-specific interface that integrates eRetail 3.2 into the customer's system. If not necessary, it can directly use eRetail 3.2's backend management system or APP.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
shopCode	String	System Store Number
shopCodeCst	String	The customer shop number, default is empty. Note: If the system shop number and customer shop number exist at the same time, the customer shop number will match first.
tagIdList	StringArray	List of IDs for labels

Return Format:

Parameter	Typology	Instruction
code	Int	0: Success

		parameter calibration 400: Parameter error 3001: Invalid Tag ID
message	String	Success or error message
body	String	Default is empty

Example: shopCode or shopCodeCst must be passed a

Request

```
{
   "shopCode": "0001".
   "tagIdList": [
      "440000010478".
"440000010479"
   ]
}
```

Response

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

2.4 Price Tag Out Interface

Application: This interface is used for labeling out

HTTP POST

URL: http:// 192.168.1.92:5000/api/esl/tag/delete

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
	String Array	List of IDs for labels

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error

		3001: Invalid Tag ID
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
["440000010478", "440000010479"]
```

Response

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

2.5 Label Binding Interface

Application: This is the interface that binds the label ID to the item number. This is a system integration-specific interface that integrates eRetail 3.2 into the customer's system. If not necessary, it can directly use eRetail 3.2's backend management system or APP.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
shopCode	String	System Store Number
shopCodeCst	String	Customer shop number, default is empty. Note: If the system shop number and customer shop number exist at the same time, the customer shop number will be matched first.
binds	Array	Binding arrays
tagID	Array String	Label ID
goodsCode	Array String	Product Number

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 1001: Tag is null (tag is empty) 3001: Invalid Tag ID 3004: Tag is not exist 7001: Goods do not exist
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
    "shopCode": "0001".
    "ap": "",
    "binds": [{
        "tagID": "40000000EA40",
        "goodsCode": "0001123456"
    }]
}
```

Response

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

2.6 Label Unbinding Interface

Application: Label ID and item number are unbound to use this interface. T This is a system integration-specific interface that integrates eRetail 3.2 into the customer's system. If not necessary, can directly use eRetail 3.2's backend management system or APP.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
shopCode	String	System Store Number
shopCodeCst	String	Customer shop number, default is empty. Note: If the system shop number and customer shop number exist at the same time, the customer shop number will be matched first.
unbindType	Int	Default pass 1
idList	String Array	Label ID

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 1001: The price tag or product ID is empty. unbundle price tag 3001: Tag format error 3004: Tag does not exist Unbundled goods 7001: Commodity does not exist incorrect 1001: Failure to unbind
message	String	Success or error message
body	String	Default is empty

Example:

Request

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

Response

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

2.7 Tag Combination Binding Interface

Application: This interface is used to bind a label ID to multiple item numbers in combination, i.e. multiple item information can be displayed on a single label. This is a system integration-specific interface for integrating eRetail 3.2 into customer systems. If not necessary, you can directly use the backend management system or app of eRetail 3.2.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
shopCode	String	The system shop number can be empty
shopCodeCst	String	Customer shop number, default is empty. Note: If the system shop number and customer shop number exist at the same time, the customer shop number will be matched first.
ap	String	Specify the base station ID
template	String	Template type (layout scheme), e.g. 2X2 2X3, etc.
tagID	String	Label ID
goodsCode	Array String	Shop Number + Product Number

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 3001: Invalid Tag ID 3004: Tag is not exist 7001: Goods do not exist 8001: Template not exists 1001: GroupBind Fail
message	String	Success or error message
body	String	Default is empty

Example:

Request


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


Response


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


2.8 Tag Flasher Interface

Application: This interface is used to notify flashing lights of labels. This is a system integration-specific interface for integrating eRetail 3.2 within a customer's system. If not necessary, the eRetail 3.2 backend management system or app can be used directly.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
shopCode	String	System Store Number
shopCodeCst	String	Customer shop number, default is empty. Note: If the system shop number and customer shop number exist at the same time, the customer shop number will be matched first.
rgb	string	Strobe Colour(R-Red,G-green,B-Blue)

		String format: as if "R" for red light. "RG" stands for Red Green Light. "RGB" stands for red, green, and blue all lit up
times	int	Flashing time (sec)
idList	Array	Price tags or goods arrays
ledType	Int	Strobe Type(1:Specified Price Tag,0:Specified Product)

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 1001: List data is null (list data is empty) 1001: LED Fail
message	String	Success or error message
body	String	Default is empty

Example:

Request

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

Response

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

2.9Tagged Data Push Refresh

Application: Used for customers to push product data directly and refresh to the specified label.

The system background will not guarantee user data.

HTTP POST

URL: http:// 192.168.1.92:5000/api/esl/tag/push

Content-Type: application/json

Request Parameters.

Parameter	Types	Instruction
shopCode	String	System shop number can be empty
shopCodeCst	String	Customer shop number can be empty
tagID	string	Price tag ID
ap	string	Specify the base station to send the Default null
item	Object	Data entities (corresponding to the fields below)
goodsCode	string	Product Unique Code
goodsName	string	Product Name
Template	string	Template name
items	Array(Character Array)	An array of data details (corresponding fields are used to bind to the template display)

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 3001: Invalid Tag ID 3004: Tag is not exists(价签不存在) 2006: is not exists (commodity code does not exist) 8001: Template not exists 1001: Push tag data fail
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
   "shopCode": "",
"shopCodeCst": "0002",
   "tagID": "4F000001320A",
   "ap": "",
   "item": {
      "GoodsCode" : "123456".
      "GoodsName" : "Château d'Aubatrice dry red".
```

```
        "Template" : "SAL".
        "Items" : [
                "0002".
                "9999".
                "Shanghai".
                "12315".
                "Zhang San".
                "39880".
                "Orbat Ligborg dry red".
                " Orbat Ligborg dry red ".
                "",
                "240209".
                "750ml".
                "Bottle".
                "France".
                "828".
                "88888".
                "0".
                "10".
                "Qualified"
        ]
}
}
```

Response

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

2.10 Tagged Data Batch Push Refresh

Application: Used for customers to push product data directly and refresh to the specified label.

The system background will not guarantee user data.

HTTP POST

URL: http:// 192.168.1.92:5000/api/esl/tag/pushList

Content-Type: application/json

Request Parameters: Push as an Array

Parameter	Types	Instruction
shopCode	String	System shop number can be empty
shopCodeCst	String	Customer shop number can be empty
tagID	string	Price tag ID
goodsCode	string	Product ID - goodsId changed to goodsCode
goodsName	string	Product Name
template	string	Template name
items	StringArray	An array of data details (corresponding fields are used to bind to the template display)

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 3001: Invalid Tag ID {0} (invalid price tag id) 2006: {0} is not exists (data does not exist) 2007: Data {0} is empty 3004: Tag {0} is not exists 8001: Template not exists {0},{1},{2} (template does not exist) incorrect 1001: Push tag data fail
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
[
{
   "shopCode": "",
"shopCodeCst": "A050".
   "tagID": "4F000001320A",
"goodsCode": "123456",
"goodsName": "GoodsName",
"template": "Template name",
"items" : [
            "123456".
            "Trade names".
            "Shanghai".
            "12315".
```

```
            "Zhang San".
            "39880".
            " Orbat Ligborg dry red ".
            " Orbat Ligborg dry red ".
            "",
            "240209".
            "750ml".
            "Bottle".
            "France".
            "828".
            "88888".
            "0".
            "10".
            "Qualified"
        ]
}
.... Multiple price tags and product data
]
```

Response

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

2.11 Price Tag Image Data Push

Application: Push image data directly to the system to send to the price tag (note that the image size and the price tag should ideally be the same)

HTTP POST

URL: http:// 192.168.1.92:5000/api/esl/tag/Image

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
shopCode	String	System shop number can be empty
shopCodeCst	String	Customer shop number can be empty
tagID	String	Price tag ID
base64Data	string	Image base64 string Support image format: BMP, jpeg,jpg, png
isDither	Bool	Does the image need to be dithered Jitter algorithm used by default: Floyd-Steinberg diffusion jitter algorithm
batchID	String	Nullable, task unique ID
backUrl	String	Nullable

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 3001: Invalid Tag ID {0} (invalid price tag id) 3004: Tag {0} is not exists incorrect 1001: Preview Fail 1001: PushImage Error
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
   "shopCode": "",
"shopCodeCst": "",
   "tagID": "400001234567",
   "base64Data": "image base64 string",
   "batchID": "",
   "isDither": true
}
```

Response

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

HTTP POST

URL: http:// 192.168.1.92:5000/api/esl/tag/ImageList

Content-Type: application/json

Request Parameters.

Parameter	Types	Description
	ObjectArray	push target array
shopCode	String	System shop number can be empty
shopCodeCst	String	Customer shop number can be empty
tagID	String	Price tag ID
base64Data	string	Image base64 string
isDither	Bool	Does the image need to be dithered
batchID	String	Nullable, task unique ID
backUrl	String	Nullable

Return Format:

Parameter	Types	Instruction
code	Int	0: Success parameter calibration 400: Parameter error 3001: Invalid Tag ID {0} (invalid price tag id) 3004: Tag {0} is not exists incorrect 1001: Preview Fail 1001: PushImage Error
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
[
{
   "shopCode": "",
"shopCodeCst": "",
   "tagID": "400001234567",
   "base64Data": "image base64 string",
   "batchID": "",
   "isDither": false
},
{
   "shopCode": "",
"shopCodeCst": "",
   "tagID": "400001234568",
   "base64Data": "image base64 string",
   "batchID": "",
   "isDither": false
}
]
```

Response


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


2.12 Tag Search

Application: Query the label information based on the label ID.

HTTP GET

URL: http:// 192.168.1.92:5000/api/esl/tag/query/{id}

Content-Type: application/json

Request Parameters.

Parameter	Types	Instruction
id	string	Price tag ID

Return Format:

Parameter	Types	Instruction
code	Int	0: Success
message	String	Success or error message
body	Object	label entity
id	string	Label ID
defaultAP	string	Final transmission base station
shopCode	string	shop number
shopCodeCst	string	Customer Store Number
tagType	string	Label type
tagStatus	int	Label status (0-unused, 1-normal, 2-low battery, 3-communicating, 5-suspected lost, 6-final failure)
powerVal	int	Electricity value (return value is an integer, e.g.: 31 for 3.1V) Commonly used proportional conversion rules for power ranges: Power >= 30 -> 100% 28<=Electricity <30 -> 90 per cent 27<=power <29-> 80 per cent 26<=Electricity <28 -> 60 per cent 25<=Electricity <27 -> 60 per cent

		24<=Power <26 -> 20 per cent 23<=Power <25 -> 10 per cent Electricity <23 -> 0 per cent
temperature	int	temp
rssi	int	signal strength
token	int	Send TOKEN
goodsCode	Array	Binding goods ID
totalSend	int	Total number of transmissions
errorCount	int	Total number of failed transmissions
lastSendTime	string	Last time to send
lastRecvTime	string	Last return time

Example:

Request

```
http://192.168.1.92:5000/api/esl/Tag/query/3600000148E3
```

Response

```
{
   "code": 0,
   "message": "success",
   "body": {
      "id": "40000000E4A2".
      "shopCode": "0001".
"shopCodeCst": "A050".
      "goodsCode": ["100000"],
      "tagType": "40".
      "tagStatus": 1,
      "powerVal": 29.
      "temperature": 24.
      "rssi": -61.
      "totalSend": 8,
"errorCount": 0,
      "lastSendTime": "2024-06-07 16:35:50",
      "lastRecvTime": "2024-06-07 16:38:10",
      "defaultAP": "01",
      "token": 0
   }
}
```

2.13 Base Station Feedback Callback Interface

Application: The base station is online and offline messages are proactively pushed to the customer's system.

Note: This interface needs to be provided by the customer. This interface is called back when the base station status changes after the feedback is turned on by the system configuration.

For Url format, please use the following format:http://xxx.xxx.xxx.xxx/apheart

HTTP POST

URL: http://customer interface address /apheart

Content-Type: application/json

Request Parameters.

Parameter	Types	Instruction
	ObjectArray	Returns the message array object
ShopCode	string	System shop number
MAC	string	BTS MAC
APID	string	Base Station ID
ApStatus	Int	State 0 initialisation 1 Online 2 Heartbeat 3 Offline
LastOnlineTime	DateTime	Last updated

Return Format:

Parameter	Types	Instruction
code	Int	0: Success
message	String	Success or error message
body	Object	Return entity

Example: http://127.0.0.1/api/apheart

Request

```
[
{
   "ShopCode": "9998".
   "MAC": "1111111111".
   "APID": "0001".
   "ApStatus":1
   "LastOnlineTime": "2020-05-01 11:11:59"
},
{
   "ShopCode": "9998".
   "MAC": "111111112".
   "APID": "0002",
   "ApStatus":1
   "LastOnlineTime": "2020-05-01 11:11:59"
}

]
```

Response


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


2.14 Price Tag Feedback Callback Interface

Application: Push the price tag to the customer's system when the price tag is refreshed successfully or failed.

Note: This interface needs to be provided by the customer. After the system is configured to enable feedback, the system will call back this interface when it receives feedback from the price tag

For Url format, please use the following format:http://xxx.xxx.xxx.xxx/callback

HTTP POST

URL: http:// customer interface address/callback

Content-Type: application/json

Request Parameters.

Parameter	Types	Instruction
	ObjectArray	Returns the message array object
ShopCode	string	System shop number
TagID	string	Price tag ID
TagType	string	Type of price tag
BatchID	string	Send batch number
APID	string	Send Base Station ID
GoodsCode	string	Product ID
GoodsName	string	Product Name
TempType	string	Template name
TaskStatus	Int	Task Status 0 initialisation 1 Communication in progress 2 Success 4 Failure 5 Timeout 8 Override
TaskType	Int	Type of mission
Rssi	Int	signal strength
PowerVal	Double	quantity of electric charge or current
Temperature	Double	temp
LastRecvTime	DateTime	Final feedback time

Return Format:

Parameter	Types	Instruction

code	Int	0: success, other: error
message	String	Success or error message
body	Object	Return entity

Request


```
[
{
   "ShopCode": "9998".
   "TagID": "401234567890",
   "TagType": "40".
   "BatchID": "1323dwafs11343".
   "APID": "0001".
   "GoodsCode": "123456".
   "GoodsName": "test",
   "TempType": "REG".
   "TaskStatus": 1,
   "TaskType": 1,
   "Rssi": -56.
   "PowerVal": 32.
   "Temperature": 32.
   "LastRecvTime": "2020-05-01 11:11:59"
},
{
   "ShopCode": "9998".
   "TagID": "401234567891",
   "TagType": "40".
   "BatchID": "1323dwafs11343".
   "APID": "0001".
   "GoodsCode": "123456".
   "GoodsName": "test",
   "TempType": "REG".
   "TaskStatus": 1,
   "TaskType": 1,
   "Rssi": -56.
   "PowerVal": 32.
   "Temperature": 32.
   "LastRecvTime": "2020-05-01 11:11:59"
}

]
```


Response


```
{
```

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

2.15 Price Tag Refreshes (Designated price tags or store-wide)

Application: This interface is used to refresh the screen with labels, without changing the display content.

HTTP POST

URL: http:// 192.168.1.92:5000/api/esl/tag/Refresh

Content-Type: application/json

Request Parameters.

Parameter	Types	Required	Description
shopCode	String	Yes	System shop number
refreshType	Int	Yes	Refresh Type 1:Template, 2:Price Tag Type, 3:Specified Store, 4:Price Tag ID List
refreshName	String	No	Template name when refreshType=1, price tag type when refreshType=2, otherwise null
Tags	String Array	No	Array specifying the price tag ID when refreshType=4. Other is null

Return Format:

Parameter	Types	Instruction
code	Int	0: Success Others are failures
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
    "shopCode": "0001".
    "refreshType": 4.
    "refreshName": "",
    "tags": [
```

```
"401234567890".
"401234567891"
   ]
}
```

Response

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

3 Store Management

3.1 Store Creation

Application: Creating shops

HTTP POST

URL: http:// 192.168.1.92:5000/api/shop/add

Content-Type: application/json

Request Parameters.

Parameter	Types	Instructions	Description
shopCode	String	be	System shop number
shopCodeCst	Int	be	Customer Store Number
shopName	String	be	Shop Name
shopAddress	String Array	clogged	Shop Address
sysOrgId	Int	be	Default 1-Root Node

Return Format:

Parameter	Types	Instruction
code	Int	0: Success Others are failures
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
   "shopCode": "0001".
```

```
    "shopCodeCst": "A001".
    "shopName": "Test shop",
    "shopAddress": "Suzhou"
}
```

Response

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

3.2 Store Deletion

Application: Deleting a shop will result in losing the original data, so please use it cautiously.

HTTP GET

URL: http:// 192.168.1.92:5000/api/shop/delete/{id}

Content-Type: application/json

Request Parameters.

Parameter	Types	Mandatory Field	Descriptions
id	int	be	Shop Primary Key ID Refer to 3.3, which can be obtained by shop query

Return Format:

Parameter	Types	Instructions
code	Int	0: Success Others are failures
message	String	Success or error message
body	String	Default is empty

Example: http:// 192.168.1.92:5000/api/shop/delete/1

Request

```

```

Response

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

3.3 Store Enquiry

Application: To query the relationship between shops under the current user.

HTTP GET

URL: http:// 192.168.1.92:5000/api/shop/queryShopListByUser

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions

Return Format:

Parameter	Types	Instructions
code	Int	0: Success
message	String	Success or error message
body	ObjectArray	Shop List Entity
shopCode	String	System shop number
shopCodeCst	String	Customer Store Number
shopName	String	Shop Name
shopAddress	String	address
shopStatus	String	Status: 0 Normal, 1 Off
Id	Int	Shop Master Key ID

Note: Once the Token has been obtained, this needs to be added to the header of subsequent HTTP requests. E.g. "Authorisation: Bearer {token}", valid for six hours.

Example:

Request

```
```

Response

```
{
   "code": 0,
   "message": "success",
   "body": [
      {
         "shopCode": "0001".
         "shopCodeCst": "A050".
         "shopName": "23 Test Stores".
         "shopAddress": "sz".
         "shopStatus": 0,
         "id": 22
      },
      {
```

```
            "shopCode": "0002",
            "shopCodeCst": "0002",
            "shopName": "0002",
            "shopAddress": "0002",
            "shopStatus": 0,
            "id": 29
        }
    ]
}
```


4 Digital Signage Interface

4.1 Signage list query interface

Application: This interface is used for signage list queries. This system integration-specific interface integrates eRetail 3.2 within the customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP POST

URL: http:// 192.168.1.92:5000/ api/tft/tft/queryList

Content-Type: application/json

Request Parameters.

Parameter	Typology	Descriptions
TFTId	String	Device Id
shopCode	String	Shop Number
shopCodeCst	String	Customer Store Number
pageIndex	Int	Page Index
pageSize	Int	Page Size

Return Format:

Parameter	Types	Instructions
code	Int	0: Success parameter calibration 400: Parameter error
message	String	Success or error message
Body	String	Default is empty
totalCount	Int	Number of queries
items	Array	Query Result List
tftId	String	Device Id

custId	String	Customer Id
instId	String	Entity Id
tftType	String	Equipment type
tftVersion	String	Device version number
lastHeartbeatTime	DateTime	final heart rate
lastOfflineTime	DateTime	Last offline time
lastOnlineTime	DateTime	Last online time
lastUpdatedTime	DateTime	Last updated
createdTime	DateTime	Creation time
tftToken	String	communication token
videoTaskId	String	Task Id
shopCode	String	Shop Number
shopCodeCst	String	Customer Store Number
goods	Array	Product Id (array)
templateName	String	Template name
tftBindRes	String	Device Binding Resources
areaId	Int	Region Id
ssid	String	Wifi SSID
ssidpwd	String	Wifi SSID Password
serverUrl	String	service address
certUrl	String	certificate address
certMD5	String	Certificate MD5 verification
status	Int	Device status identifier: 0 Offline 1 Online 2 Downloading 3 Not activated
tftName	String	Equipment name
firmware	String	Firmware version
lastVideoTaskId	String	Final Task Id
firstDownloadTime	DateTime	First download time
lastDownloadTime	DateTime	First download time
lastConfirmTime	DateTime	Final confirmation time
downLoadCount	Int	Number of downloads
confirmCount	Int	Number of confirmations
heartBeatCount	Int	Heartbeat statistics
taskCount	Int	Total number of mandates
lastSpanTime	DateTime	Time spent on last task (seconds)
areaName	String	Region Name
statusStr	String	Device Status Name
goodsStr	String	Trade names (array)
lastHeartbeatTimeStr	String	Last heartbeat time (MM-dd HH:mm)
lastDownloadTimeStr	String	Last download time (MM-dd HH:mm)
totalCount	Int	Number of queries

Example:

Request


```
{
    "TFTId": "",
    "shopCode": "0001".
    "pageIndex": 1,
    "pageSize": 30
}
```


Response


```
{
    "code": 0,
    "message": "success",
    "body": {
        "totalCount": 149.
        "items": [
            {
                "tftId": "C49C14844658".
                "custId": "ETAG".
                "instId": "0001".
                "tftType": "5C".
                "tftVersion": "61",
                "lastHeartbeatTime": "2022-09-06 13:17:09",
                "lastOfflineTime": "1900-01-01 00:00:00",
                "lastOnlineTime": "1900-01-01 00:00:00",
                "lastUpdatedTime": "2022-09-06 13:17:09",
                "createdTime": "2022-09-02 19:24:25",
                "tftToken": "",
                "videoTaskId": "b88d7987-4ad3-462a-9aa7-f917f7267d82",
                "shopCode": "0001".
                "goods": [
                    "1"
                ],
                "templateName": "Show regional video",
                "tftBindRes": [],
                "areaId": 7.
                "ssid": "",
                "ssidpwd": "",
                "serverUrl": "",
                "certUrl": "",
                "certMD5": "",
                "status": 1.
                "tftName": "",
                "firmware": "",
                "lastVideoTaskId": "b88d7987-4ad3-462a-9aa7-f917f7267d82",
                "firstDownloadTime": "2022-09-06 13:16:10",
```

```
                    "lastDownloadTime": "2022-09-06 13:16:10",
                    "lastConfirmTime": "2022-09-06 13:17:02",
                    "downLoadCount": 152,
                    "confirmCount": 152,
                    "heartBeatCount": 257.
                    "taskCount": 152,
                    "lastSpanTime": 52,
                    "areaName": "video",
                    "statusStr": "",
                    "goodsStr": "",
                    "lastHeartbeatTimeStr": "",
                    "lastDownloadTimeStr": ""
                }
            ]
        }
}
```


4.2 Signage Query Interface

Application: This interface is used for signage queries. It is a system integration-specific interface for integrating eRetail 3.2 within the customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP GET

URL: http:// 192.168.1.92:5000/ api/tft/tft?tftId=C49C14844658

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions
TFTId	String	Device Id

Return Format:

Parameter	Typology	Instructions
code	Int	0: Success 400: Parameter error
message	String	Success or error message
body	String	Default is empty

Example:

Request

TFTId =C49C14844658

Response


```
{
    "code": 0,
    "message": "success",
    "body": {
        "tftId": "C49C14844658".
        "custId": "ETAG".
        "instId": "0001".
        "tftType": "5C".
        "tftVersion": "61",
        "lastHeartbeatTime": "2022-09-06 13:17:09",
        "lastOfflineTime": "1900-01-01 00:00:00",
        "lastOnlineTime": "1900-01-01 00:00:00",
        "lastUpdatedTime": "2022-09-06 13:17:09",
        "createdTime": "2022-09-02 19:24:25",
        "tftToken": "",
        "videoTaskId": "b88d7987-4ad3-462a-9aa7-f917f7267d82",
        "shopCode": "0001".
        "shopCodeCst": "A050".
        "goods": [
            "00011"
        ],
        "templateName": "Show regional video",
        "tftBindRes": [],
        "areaId": 7.
        "ssid": "",
        "ssidpwd": "",
        "serverUrl": "",
        "certUrl": "",
        "certMD5": "",
        "status": 1.
        "tftName": "",
        "firmware": "",
        "lastVideoTaskId": "b88d7987-4ad3-462a-9aa7-f917f7267d82",
        "firstDownloadTime": "2022-09-06 13:16:10",
        "lastDownloadTime": "2022-09-06 13:16:10",
        "lastConfirmTime": "2022-09-06 13:17:02",
        "downLoadCount": 152,
        "confirmCount": 152,
        "heartBeatCount": 257.
        "taskCount": 152,
        "lastSpanTime": 52,
        "areaName": "video",
```

```
            "statusStr": "",
            "goodsStr": "",
            "lastHeartbeatTimeStr": "",
            "lastDownloadTimeStr": ""
        }
}
```

4.3 Signage Preview Interface

Application: This interface is used to preview the signage. This system integration-specific interface integrates eRetail 3.2 within the customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP GET

URL: http:// 192.168.1.92:5000/ api/tft/tft/PreviewTft/C49C14844658

Content-Type: application/json

Request Parameters.

Parameter	Typology	Descriptions
TFTId	String	Device Id

Return Format:

Parameter	Typology	Instructions
code	Int	0: Success 404: Resource does not exist 1001: The device has not been bound to a template (设备尚未绑定到模板)
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
TFTId =C49C14844658
```

Response

```
{
"code": 0,
"message": "success",
"body": [
{
"id": 0,
```

```
"createdBy": null,
"createdTime": "0001-01-01 00:00:00",
"mac": null,
"displayId": 0,
}
]
}
```


4.4 Signage template query name interface

Application: This interface is used for signage preview. This system integration-specific interface integrates eRetail 3.2 within the customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP POST

URL: http:// 192.168.1.92:5000/ api/template/queryTemplateByName

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions
groupTemp		Whether to combine templates 0 All, 1 Individually, 2 Combined
shopCode	String	Shop Number
tagType	String	Equipment Model
templateName	String	Template name

Return Format:

Parameter	Types	Instructions
code	Int	0: Success 400: Parameter error
message	String	Success or error message
body	String	Default is empty

Example:

Request


```
{
    "groupTemp": 0,
    "shopCode": "0001".
    "tagType": "5C".
    "templateName": "Show Region Video"
}
```

Response


```
{
    "code": 0,
    "message": "success",
    "body": [
        {
            "templateName": "Show regional video",
            "tagType": "5C".
            "shopCode": "0001".
            "templateType": 1,
            "templateStatus": 0,
            "blocks": 2,
            "preview":
"api/template/preview_template?shopCode=0001&tagType=5C&name=%e6%98%be%e7%a4
%ba%e5%8c%ba%e5%9f%9f%e8%a7%86%e9% a2%91&timeStamp=1661614464253",
            "templateData": null,
            "id": 60.
            "isDeleted": false,
            "createdBy": "System",
            "createdTime": "2022-08-16 16:58:12",
            "lastUpdatedBy": "System",
            "lastUpdatedTime": "2022-08-27 23:34:24"
        }
    ]
}
```


4.5 Signage template query ID interface

Application: This interface is used for signage templates to look up IDs. this is a system integration-specific interface for integrating eRetail 3.2 within the customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP GET

URL: http:// 192.168.1.92:5000/ api/template/query/60

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions
ID	Int	Template Id

Return Format:

Parameter	Types	Instructions
code	Int	0: Success
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
```

Response

```
{
"code": 0,
"message": "success",
"body": {
"templateName": "Show regional video",
"tagType": "5C".
"shopCode": "0001".
"templateType": 1,
"templateStatus": 0,
"blocks": 2,
"preview":
"api/template/preview_template?shopCode=0001&tagType=5C&name=%e6%98%be%e7%a4
%ba%e5%8c%ba%e5%9f%9f%e8%a7%86%e9% a2%91&timeStamp=1661614464253",
}
}
```


4.6 Label Binding Interface

Application: This interface is used for signage binding. This system integration-specific interface integrates eRetail 3.2 within a customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions
areaId	Int	Region Id

bindRes	Entity	List of entities for the location of template elements
displayIndex	Int	Screen index (default value 0)
goods	Array	product array
shopCode	String	Shop number can be empty Note: Store number and customer shop number must be filled in one.
templateName	String	Template name
tftId		Device Id

Return Format:

Parameter	Types	Instructions
code	Int	0: Success parameter calibration 400: Parameter error 1100: tag not fond (价签未找找到) incorrect 3201： goods {item} is not exist （商品不存在） 1001: Binding error
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
    "areaId": 7.
    "bindRes": [],
    "displayIndex": 0,
    "goods": [
        "00011"
    ],
"shopCode": "0001".
"shopCodeCst": "A050".
    "templateName": "Show regional video",
    "tftId": "C49C14844658"
}
```

Response

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

4.7 Label Unbinding Interface

Application: This interface is used for signage unbinding. This system integration-specific interface integrates eRetail 3.2 within a customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP POST

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

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions
idList	Array	device array
shopCode	String	Shop number can be empty Note: Store number and customer shop number must be filled in one.

Return Format:

Parameter	Types	Instructions
code	Int	0: Success parameter calibration 400: Parameter error 3102: tft is null (tft does not exist)
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
    "idList": [
        "F49520998244"
    ],
"shopCode": "0001"
}
```

Response

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

4.8 Signage Rest/Light Interface

Application: This interface is used for signage rest/light. This system integration-specific interface integrates eRetail 3.2 within the customer's system. The eRetail 3.2 backend management system or app can be used directly if not necessary.

HTTP POST

URL: http:// 192.168.1.92:5000/ api/tft/task/switchTask

Content-Type: application/json

Request Parameters.

Parameter	Types	Descriptions
tftIds	Array	device array
taskCode	Int	Task code: 11:Turn off backlight, 12:Light up backlight

Return Format:

Parameter	Typology	Instructions
code	Int	0: Success parameter calibration 400: Parameter error incorrect 1001: TFTIds is not null (colour screen id is not null) 1001: Switch Task Settings HOST is null (Lighting and hibernation task settings HOST is null) 1001: Swtich Task TFTIds is error: (Brightening and hibernating task device MAC error message)
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
    "tftIds": [
        "C49C14844658"
    ],
    "taskCode": "11"
}
```

Response

```
{
    "code": 0,
```

```
    "message": "success",
    "body": ""
}
```

4.9 Signage Refresh Interface

Application: This interface is used for signage refresh. This system integration-specific interface integrates eRetail 3.2 within a customer's system. If not necessary, the eRetail 3.2 backend management system or app can be used directly.

HTTP POST

URL: http:// 192.168.1.92:5000/ api/tft/tft/refresh

Content-Type: application/json

Request Parameters.

Parameter	Types	Instructions
tftId	String	Device Id
shopCode	String	Shop Number

Return Format:

Parameter	Types	Instructions
code	Int	0: Success parameter calibration 400: Parameter error incorrect 1001: No items selected 1001: refresh error
message	String	Success or error message
body	String	Default is empty

Example:

Request

```
{
    "tftId": [
        "F4952099968E"
    ],
    "shopCode": "0001"
}
```

Response

```
{
    "code": 0,
```

```
    "message": "success",
    "body": ""
}
```

5 Product Management

5.1 Product Data Interface

Purpose: This interface is used to add and update multiple product data.

HTTP POST

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

Content-Type: application/json

Request Parameters: --array

Parameter	Types	Descriptions
shopCode	String	Customer/system shop number, you need to create the corresponding shop in the system shop list first.
template	String	Template name, you need to first create the corresponding template in the template list
items	Array String[27]	Commodity data attributes, temporarily fixed at 27 (expandable)

Note: For shop and template creation, please contact an eTaylor salesperson or pre-sales/sales support engineer.

Return Format:

Parameter	Types	Instructions
code	Int	0: Success parameter calibration 400: Parameter error 1001: Shop is not exists(shop does not exist) 9801: is null ShopCode(shop number is null) 9801: upc index too long(upc index too long) 9801: Cannot find goods code index(找不到商品代码索引) 9801: Goods name index too long(商品名称索引过长)
message	String	Success or error message
body	Json Node	message body

Example:

http:// 192.168.1.92:5000/api/goods/saveList?NR=false

Request

```
[
  {
    "shopCode": "0001", // customer shop number
    "template": "REG", // Price tag type: REG - General Sale, SAL - Sale, NOR - Out of Stock,
MER - Member ……
    "items": [
      "A050", // Customer shop number
      "123456", // commodity unique code: can be commodity number or commodity
barcode
      "Commodity 1", // Commodity name
      "123456789012", //UPC1: Commodity barcode 1 or nominal code
      "123456789013", // UPC2: commodity barcode 2 or designation code
      "123456789014", // UPC3: commodity barcode 3 or designation code
      "8.98", //price 1: retail or original price
      "8.95", //price 2: promotional price
      "8.96", //price 3: member price
      "Shanghai", // Origin
      "300ml", //Specifications
      "Bottle", //unit
      "Pass", //Grade
      "2021/12/20", //promotion start date
      "2021/12/25", //promotion end date
      "http://www.baidu.com", // QR code
      "Zhangsan", //pricekeeper
      "5.1", //inventory
      "Extension 1", // extension field 1 not passed empty ""
      "Extension 2", //extension field 2
      "Extension 3", //extension field 3
      "Extension 4".
      "Extension 5".
      "Extension 6".
      "Extension 7".
      "Extension 8".
      "Extension 9".
      "Extension 10"
    ]
  }
]
```

Response

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

```
}
```

Note: By default, the field attributes in this example are fixed. In fact, the content and template of the product data field can be set freely.

5.2 Product Inquiry

Application: Used to query product information according to customer shop number.

HTTP POST

URL: http:// 192.168.1.92:5000/api/Goods/getList

Content-Type: application/json

Request Parameters.

Parameter	Types	Instructions
pageIndex	Int	pagination
pageSize	Int	Number of page breaks
sort	string	voidable
desc	string	Nullable asc/desc
shopCodeCst	string	Empty Customer shop number
shopCode	string	Empty System shop number
goodsCode	string	Empty Unique Code

Return Format:

Parameter	Types	Instructions
code	Int	0: Success
message	String	Success or error message
body	Object	Return entity
totalCount	int	aggregate
itemList	ObjectArray	List of commodity entities
id	string	Commodity Master Key
shopCodeCst	string	Customer Store Number
shopCode	string	System shop number
goodsCode	string	Product Unique Code
goodsName	string	Product Name
template	string	Template name
upc	String array	retrieval code
items	String Array	Array of product details, subject to actual data

Example:

Request

```
{
   "pageIndex": 1,
   "pageSize": 100,
   "sort": "",
   "desc": "",
   "shopCodeCst": "012412".
   "shopCode": "",
   "goodsCode": ""
}
```


Response


```
{
   "code": 0,
   "message": "success",
   "body": {
"totalCount": "999",
"itemList": [
{
      "shopCodeCst": "012412".
      "id": "00011000786815",
      "shopCode": "0001".
      "goodsCode": "1000786815",
      "goodsName": "A",
      "template": "REG",
      "upc": [
         "1000786815"
      ],
      "items": [
         "14297".
         "1000786815".
         "A".
         "",
         "6903148083109".
         "0.00",
         "34.90".
         "0".
         "0.00",
         "2.8KG".
         "",
         "",
         "Qualified".
         "",
         "",
         "Bag".
         "Bag".
```

```
            "0518".
            "60081373".
            "",
            "Guangzhou City"
]}.
......
    ]
}
}
```

5.3 Product Deletion

HTTP GET

URL: http:// 192.168.1.92:5000/api/Goods/delete/{id}

Content-Type: application/json

Request Parameters.

Parameter	Types	Instructions
id	string	Product unique ID, generally: shop number + product code

Return Format:

Parameter	Types	Instructions
code	Int	0: Success
message	String	Success or error message
body	Object	Return entity
totalCount	int	aggregate
itemList	ObjectArray	List of commodity entities
id	string	Commodity Master Key
shopCodeCst	string	Customer Store Number
shopCode	string	System shop number
goodsCode	string	Product Unique Code
goodsName	string	Product Name
template	string	Template name
upc	String array	retrieval code
items	String Array	Array of product details, subject to actual data

Example:

Request

```
http:// 192.168.1.92:5000/api/Goods/delete/0001123456
```

Response

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

6 Data synchronization instructions

Data synchronization includes common synchronization methods: query statements or stored procedures based on databases/intermediate tables/views; files based on defined formats (e.g. Excel, CSV/TXT, XML, JSON, etc.); file systems based on FTP, etc.

6.1 D2M Dynamic Data Modelling

For simple data structures, data synchronisation can be performed using the Dynamic Model (D2M) approach. It is usually the responsibility of the implementer to make the selection based on 1) the data structure definition 2) the template switching logic definition. See the eRetail D2M Configuration page for details.

6.2 Excel Product Import

After setting up the dynamic model of commodity data, you can export the corresponding data template in "Data Management - Commodity Management". Then you can add or edit the commodity data, and then import it into the system for price change.

Note: The Excel file format is xlsx, the file size is less than 10M, and the number of commodity data items is around 100,000 items.

The template field is fixed and cannot be modified.

6.3 Customize

If cannot effectively solve the customer's system integration needs with the above-customized development docking solution, please contact the eTailer project department for further information on the specific solution.