简要描述：

- 获取RAIL_EXPIRATION和RAIL_DEVICEID的值

请求URL：

- `https://kyfw.12306.cn/otn/HttpZF/logdevice`

请求方式：

- GET

参数：

algID	eN9MhjRjlm
hashCode	PYB6tp4uDcxvqEHECbPgSsBFoEEtW_3eJFLpZj34Xy8
FMQw	0
q4f3	zh-CN
VySQ	FGE_zbWgiHXDonvHSCVjlB0yh59roB2Z
VPIf	1
custID	133
VEek	unknown
dzuS	0
yD16	0
EOQP	c227b88b01f5c513710d4b9f16a5ce52
jp76	52d67b2a5aa5e031084733d5006cc664
hAqN	MacIntel
platform	WEB
ks0Q	d22ca0b81584fbea62237b14bd04c866
TeRS	969x1680
tOHY	24xx1050x1680
Fvje	i1l1o1s1
q5aJ	-8
wNLf	99115dfb07133750ba677d055874de87
0aew	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36
E3gR	141495a03ea3e03df9e4da5db09a3ab5
timestamp	1584918667461

返回示例

```
callbackFunction('{"exp":"1585183379011","dfp":"hnT7l0nGG32N_huT3nndmlEvqMjiiLfIykKW_4bzFH4d0LPEfUbtjZQ3wN9eWQa4YjZQ2pIkJeagLdC8kE44dmi1VWHnMLHLXsSAqAqaMGe7QPJmM_BQ_QdVlAEl8T7DE9yT2DOxQqPLrmw7Lj2Mn6aDEMa9reZc"}')
```

返回参数说明

参数名	类型	说明
exp	string	RAIL_EXPIRATION
dfp	string	RAIL_DEVICEID

备注

- 请求的参数中，algID和hashCode是核心。
- algID是在getJS文件中直接就有的，可以以下通过正则表达式进行匹配。

```
algID\\x3d(.*?)\\x26j
```

- hashCode通过hashAlg函数动态生成，该函数如下所示。

```
hashAlg: function(a, b, c) {
            a.sort(function(a, b) {
                var c, d;
                if ("object" === typeof a && "object" === typeof b && a && b)
return c = a.key, d = b.key, c === d ? 0 : typeof c === typeof d ? c < d ? -1
: 1 : typeof c < typeof d ? -1 : 1;
                throw "error";
            });
            for (var d = 0; d < a.length; d++) {
                var e = a[d].key.replace(RegExp("%", "gm"), ""),
                    f = "",
                    f = "string" == typeof a[d].value ?
a[d].value.replace(RegExp("%", "gm"), "") : a[d].value;
                "" !== f && (c += e + f, b += "\x26" + (void 0 == fb[e] ? e :
fb[e]) + "\x3d" + f)
            }
            a = R.SHA256(c).toString(R.enc.Base64);
            a = xa(a);
            c = a.length;
            d = a.split("");
            for (e = 0; e < parseInt(c / 2); e++) 0 == e % 2 && (f =
a.charAt(e), d[e] = d[c - 1 - e], d[c - 1 - e] = f);
            a = d.join("");
            a = xa(a);
            c = xa(a);
            c = R.SHA256(c).toString(R.enc.Base64);
            return new p(b, c)
        },
```

输入的参数a是根据浏览器设备信息生成的，并且只需要修改cookieCode的值，12306就会认为用户使用的是不同的设备：

```
0: n {key: "adblock", value: "0"}
1: n {key: "browserLanguage", value: "zh-CN"}
2: n {key: "cookieCode", value: "FGE_zbWgiHXDonvHSCVjlB0yh59roB2Z"}
3: n {key: "cookieEnabled", value: "1"}
4: n {key: "custID", value: "133"}
5: n {key: "doNotTrack", value: "unknown"}
6: n {key: "flashVersion", value: 0}
7: n {key: "javaEnabled", value: "0"}
8: n {key: "jsFonts", value: "c227b88b01f5c513710d4b9f16a5ce52"}
9: n {key: "mimeTypes", value: "52d67b2a5aa5e031084733d5006cc664"}
10: n {key: "os", value: "MacIntel"}
11: n {key: "platform", value: "WEB"}
```

```
12: n {key: "plugins", value: "d22ca0b81584fbea62237b14bd04c866"}
13: n {key: "timeZone", value: -8}
14: n {key: "touchSupport", value: "99115dfb07133750ba677d055874de87"}
15: n {key: "userAgent", value: "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_3) Ap…L, like Gecko) Chrome/80.0.3987.149 Safari/537.36"}
16: n {key: "webSmartID", value: "141495a03ea3e03df9e4da5db09a3ab5"}
17: n {key: "storeDb", value: "i1l1o1s1"}
18: n {key: "srcScreenSize", value: "24xx1050x1680"}
19: n {key: "scrAvailSize", value: "969x1680"}
```

hashAlg函数首先会对a中的key按字典顺序进行排序，然后将a中的key映射为logdevice请求参数的模样

```
{
        hasLiedResolution: "3neK",
        systemLanguage: "e6OK",
        cookieEnabled: "VPIf",
        timeZone: "q5aJ",
        srcScreenSize: "tOHY",
        scrAvailHeight: "88tV",
        indexedDb: "3sw-",
        hasLiedLanguages: "j5po",
        scrColorDepth: "qmyu",
        localStorage: "XM7l",
        hasLiedOs: "ci5c",
        localCode: "lEnu",
        hasLiedBrowser: "2xC5",
        historyList: "kU5z",
        webSmartID: "E3gR",
        plugins: "ks0Q",
        userLanguage: "hLzX",
        browserName: "-UVA",
        openDatabase: "V8vl",
        userAgent: "0aew",
        appMinorVersion: "qBVW",
        jsFonts: "EOQP",
        browserLanguage: "q4f3",
        scrAvailWidth: "E-lJ",
        adblock: "FMQw",
        appcodeName: "qT7b",
        cookieCode: "VySQ",
        touchSupport: "wNLf",
        scrDeviceXDPI: "3jCe",
        scrWidth: "ssI5",
        online: "9vyE",
        os: "hAqN",
        doNotTrack: "VEek",
        storeDb: "Fvje",
        mimeTypes: "jp76",
        scrAvailSize: "TeRS",
        sessionStorage: "HVia",
        javaEnabled: "yD16",
        browserVersion: "d435",
        flashVersion: "dzuS",
        cpuClass: "Md7A",
        scrHeight: "5Jwy"
    }
```

hashAlg函数返回的是一个dict，其中b是字符串，由a映射后的key和对应的value进行拼接得到：

```
"&FMQw=0&q4f3=zh-
CN&VySQ=FGE_zbWgiHXDonvHSCVjlB0yh59roB2Z&VPIf=1&custID=133&VEek=unknown&dzuS=0
&yD16=0&EOQP=c227b88b01f5c513710d4b9f16a5ce52&jp76=52d67b2a5aa5e031084733d5006
cc664&hAqN=MacIntel&platform=WEB&ks0Q=d22ca0b81584fbea62237b14bd04c866&TeRS=96
9x1680&tOHY=24xx1050x1680&Fvje=i1l1o1s1&q5aJ=-8&wNLf=99115dfb07133750ba677d055
874de87&0aew=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149
Safari/537.36&E3gR=141495a03ea3e03df9e4da5db09a3ab5"
```

而c其实就是最重要的hashCode的值：

```
"PYB6tp4uDcxvqEHECbPgSsBFoEEtW_3eJFLpZj34Xy8"
```

在获取这些参数之后，前端将请求logdevice接口，并根据返回值设置RAIL_EXPIRATION和RAIL_DEVICEID的值。

```
e = c.hashAlg(k, a, e);
a = e.key;
e = e.value;
a += "\x26timestamp\x3d" + (new Date).getTime();
Za.getJSON("https://kyfw.12306.cn/otn/HttpZF/logdevice" + ("?
algID\x3deN9MhjRjlm\x26hashCode\x3d" + e + a), null, function(a) {
var b = JSON.parse(a);
void 0 != lb && lb.postMessage(a, r.parent);
for (var d in b)
"dfp" == d ? F("RAIL_DEVICEID") != b[d] && (W("RAIL_DEVICEID", b[d], 1E3),
c.deviceEc.set("RAIL_DEVICEID", b[d])) : "exp" == d ? W("RAIL_EXPIRATION",
b[d], 1E3) : "cookieCode" == d && (c.ec.set("RAIL_OkLJUJ", b[d]),
W("RAIL_OkLJUJ", "", 0))
})
```

最后要注意R.SHA256(c).toString(R.enc.Base64)函数，它首先对输入的字符串进行sha256编码，然后再对编码结果进行base64编码，最后对结果中的/和=等字符进行处理，该函数逻辑用java实现如下：

```
private static String sha256(String c) {
  String result = null;
  try {
    MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
    messageDigest.update(c.getBytes(StandardCharsets.UTF_8));
    result = Base64.getEncoder().encodeToString(messageDigest.digest());
  } catch (NoSuchAlgorithmException ex) {
    LOGGER.error(ex.getMessage());
  }
  if (result != null) {
    result = result.replaceAll("\\+", "-").replaceAll("/",
"_").replaceAll("=", "");
  }
  return result;
}
```