标拓云打印服务开放API v1.04

作为系统服务提供商，我们为第三方应用（如移动应用、小程序和H5网页）提供一套完整的API接口，使这些客户端能够安全、灵活地管理用户和设备，并获取访问打印服务所需的认证令牌（TOKEN）。本文件详细说明了各个API端点的使用方法，包括用户与设备管理、获取TOKEN、两种打印操作模式（基于TOKEN的打印和匿名打印）、以及实时监控打印任务和设备状态的WebSocket（WSS）通道，以优化用户体验并降低第三方集成的复杂性。

目录

1. 匿名打印（Anonymous Print）

描述:
匿名打印允许用户无需进行身份验证或用户管理，直接通过打印机设备标识和验证码提交打印任务。这种方式简化了用户操作，提升了用户体验，适用于无需个性化用户管理的场景。

请求方式:

```
POST
```

服务地址:

```
https://www2.biaotop.com/btc/api/print
```

Content-Type:

```
multipart/form-data
```

Header

Header 参数	是否必须	描述
x-Cust-prntId	是	机身序列号，至少7位字符，用于标识具体的打印设备。
x-Cust-key	是	一组随机数，最短13个字节，用于认证设备。

Header 参数	是否必须	描述
`x-Cust-code`	是	使用SHA-256生成的认证码： `x-Cust-key + x-Cust-prntId + 机身验证码` 。
`x-Callback-url`	否	消息回调url,base64url
`x-Page-size`	否	宽度（点）x高度（点）

Body

- **类型**： `multipart/form-data`
- **内容**：选择以下两种模式中的一种提交：
  i. **文件模式**：上传多个PDF或JPG文件。
  - 可以直接提交单个文件 image/jpeg image/png
  ii. **打印内容模式**：提交一个包含打印内容布局的JSON文件,可以作为单独文件也可以插入form队列提交。

注意： `printJson` 与文件上传不可同时使用。请选择其中一种模式提交打印任务。

打印内容格式（打印内容模式）

为了确保打印内容的准确性和顺序，客户需要按照以下格式定义打印内容：

```
{
  "command": "PrintJson",
  "sver": "",
  "printJson":{
    "0":{
      "width": 1440,
      "height" : 990,
      "elements": {
        "0":{
          "cont": "地址臣田工业区",
          "type": "text",
          "font": "yh",
          "size": "12",
          "angle": 0,
          "reverse": 0,
          "x": 210,
          "y": 20
        },
        "1":{
          "cont": "电话：1862099999",
          "type": "text",
          "font": "hei",
          "size": "7",
          "x": 210,
          "y": 170
        },
        "2":{
          "cont": "件数",
          "type": "text",
          "font": "hei",
          "size": "9",
          "x": 260,
          "y": 270
        }
      }
    }
  }
}
```

- `cont`：内容文本或图片文件名。
- `type`： `text` 或 `image` 。
- `font`：song 宋体 hei 黑体 yh雅黑
- `size`：仅适用于文本类型。
- `angle`：旋转角度。
- `reverse`：反白。
- `mirror`：镜像。90/180
- `url`：图片的URL地址，适用于图片类型。
- `x`，`y`：打印起始坐标（单位：毫米）。
- `width`，`height`：图片尺寸（单位：毫米），适用于图片类型。

打印任务取消

```
{
  "command": "PrintCancel",
  "sver": "",
  "printCancel": {
    "printJobId": "job id",
    "printFileId": "file id",
  }
}
```

示例

curl 请求示例

以下是使用 curl 进行匿名打印的示例，包含两种模式的提交方法。

1. **文件模式**
   假设：
   - x-Cust-key：randomkey12345
   - x-Cust-prntId：12345678901
   - 验证码：verifyCode123

   ```
   # 生成 `x-Cust-code` 的示例（使用 OpenSSL）
   echo -n "randomkey12345612345678901verifyCode123" | openssl dgst -sha256
   # 输出示例：e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
   ```

   发送打印请求：

   ```
   curl -X POST "https://www2.biaotop.com/btc/api/print" \
        -F "files=@/path/to/document1.pdf;filename=document1.pdf" \
        -F "files=@/path/to/image1.jpg;filename=image1.jpg" \
        -F "files=@/path/to/document2.pdf;filename=document2.pdf" \
        -H "x-Cust-prntId: 12345678901" \
        -H "x-Cust-key: randomkey12345" \
        -H "x-Cust-code: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
   ```

2. **打印内容模式**
   假设：
   - x-Cust-key：randomkey12345
   - x-Cust-prntId：12345678901
   - 验证码：verifyCode123

   生成 x-Cust-code 的示例（使用 OpenSSL）：

   ```
   echo -n "randomkey12345612345678901verifyCode123" | openssl dgst -sha256
   # 输出示例：e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
   ```

   创建 printContent.json 文件：

```
{
 "command":"PrintJson",
  "printJson": [
    {"cont": "地址工业区", "type": "text", "font": "hei", "size": "12", "x": 10, "y": 20},
    {"cont": "电话：18699999", "type": "text", "font": "hei", "size": "7", "x": 10, "y": 70},
    {"cont": "件数", "type": "text", "font": "hei", "size": "9", "x": 60, "y": 70}
  ]
}
```

发送打印请求：

```
curl -X POST "https://www2.biaotop.com/btc/api/print" \
     -H "x-Cust-prntId: 12345678901" \
     -H "x-Cust-key: randomkey12345" \
     -H "x-Cust-code: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855" \
     --header 'Content-Type: application/json' \
     --data '@/path/to/printContent.json'
```

> **注意**：
>
> - 在文件模式中，上传的文件将按顺序打印。
> - 在打印内容模式中， `printJson` 定义了打印内容的布局，不与文件上传同时使用。
> - 确保 `printJson` 中的图片 `url` 可访问，并且图片分辨率为360dpi，纸张尺寸与图片尺寸匹配。

成功响应示例

```
{
  "status": "success",
  "message": "打印任务已提交。",
}
```

失败响应示例

```
{
  "status": "error",
  "message": "打印任务提交失败。",
  "error": {
    "code": "INVALID_DEVICE",
    "message": "设备认证信息无效或已过期。"
  }
}
```

返回参数

Content-Type:

`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"`。
`message`	String	对请求结果的简要描述，如 `"打印任务已提交。"`。
`data`	Object	包含具体的响应数据对象。
`data.printJobId`	String	打印任务的唯一标识符。
`data.deviceId`	String	与请求中的 `x-Cust-prntId` 相对应的设备标识。
`data.submittedAt`	String	打印任务提交时间，格式为ISO 8601。
`data.printJson`	Object	打印内容的具体定义，见打印内容格式部分。（仅适用于打印内容模式）

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"`。
`message`	String	对请求结果的简要描述，如 `"打印任务提交失败。"`。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`INVALID_DEVICE`	设备认证信息无效或已过期。
`MISSING_PARAM`	缺少必要的Header参数。
`SERVER_ERROR`	服务器内部错误，请稍后再试。
`INVALID_FORMAT`	提供的数据格式不正确。
`FILE_NOT_SUPPORTED`	提交的文件类型不支持。

回调数据结构

```
{
  "PrinterState":{
    "prntId" : "",
    "prntState" : "Printer Printing",
    "jobState" : {
      "jobId0" : "Job completed",
      "jobId1" : "Job printing"
    }

  }
}
```

回调信息说明

msg	描述
Job pending	任务挂起
Job incoming	打印任务传入
Job held	打印任务已完成
Job created	已创建打印任务
Job canceling	取消任务
Job printing	正在打印
Job stopped	任务已停止
Job canceled	任务已取消
Job aborted	任务已中止
Job completed	任务已完成
Printer Idle	打印机空闲
Printer Printing	打印机正在打印
Printer Error	打印机故障
Printer offline	打印机离线

2. 添加用户（AddUser）

描述:
此接口用于为特定应用（ `app-id` ）添加新用户。第三方应用通过此接口为每个用户分配一个唯一的

`userId` （UUID）和对应的 `userKey` ，并可选择性地关联用户的详细信息或Open-ID。第三方负责管理和维护这些 `userId` 和 `userKey` ，确保其与用户的Open-ID正确关联。

请求方式:

`POST`

服务地址:

`https://www2.biaotop.com/btc/api/AddUser`

Content-Type:

`application/json`

Body

```
{
  "appId": "your_app_id",
  "secureKey": "your_secure_key",
  "userId": "unique_user_uuid",
  "userDetails": {
    "name": "User Name",
    "email": "user@example.com",
    "openId": "user_open_id_optional"
  }
}
```

参数说明

参数名称	类型	是否必须	描述
`appId`	String	是	预先分配的应用标识（`app-id`），用于标识客户端应用。
`secureKey`	String	是	应用的安全密钥（`secure-key`），必须妥善保管。
`userId`	String	是	用户唯一标识符（`user-id`），由第三方生成并管理的UUID。
`userDetails`	Object	否	用户的详细信息，可选字段，如姓名、邮箱、关联的Open-ID等。

用户详情说明

参数名称	类型	描述
`name`	String	用户的姓名，可选。
`email`	String	用户的邮箱地址，可选。
`openId`	String	用户的Open-ID，用于跨平台身份关联，可选。

示例

请求示例

```
curl -X POST "https://www2.biaotop.com/btc/api/AddUser" \
     -H "Content-Type: application/json" \
     -d '{
           "appId": "app1234567890",
           "secureKey": "secureKeyXYZ12345",
           "userId": "userUUID1234567890",
           "userDetails": {
             "name": "John Doe",
             "email": "john.doe@example.com",
             "openId": "openid1234567890"
           }
         }'
```

成功响应示例

```
{
  "status": "success",
  "message": "用户添加成功。",
  "data": {
    "userId": "userUUID1234567890",
    "appId": "app1234567890",
    "userKey": "userKeyABC1234567890",
    "userDetails": {
      "name": "John Doe",
      "email": "john.doe@example.com",
      "openId": "openid1234567890"
    }
  }
}
```

失败响应示例

```
{
  "status": "error",
  "message": "用户添加失败。",
  "error": {
    "code": "USER_EXISTS",
    "message": "该用户ID已存在。"
  }
}
```

返回参数

Content-Type:

`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"`。
`message`	String	对请求结果的简要描述，如 `"用户添加成功。"`。
`data`	Object	包含具体的响应数据对象。
`data.userId`	String	添加的用户唯一标识符。
`data.appId`	String	与请求中的 `appId` 相对应的应用标识。
`data.userKey`	String	返回的用户密钥，用于在 `GetToken` 接口中校验用户。
`data.userDetails`	Object	用户的详细信息对象。
`data.userDetails.name`	String	用户的姓名。
`data.userDetails.email`	String	用户的邮箱地址。
`data.userDetails.openId`	String	用户的Open-ID（如果关联）。

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"`。
`message`	String	对请求结果的简要描述，如 `"用户添加失败。"`。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`USER_EXISTS`	该用户ID已存在。
`INVALID_AUTH`	认证信息无效或已过期。
`MISSING_PARAM`	缺少必要的请求参数，例如 `appId`、`secureKey`、`userId` 等。
`INVALID_FORMAT`	提供的数据格式不正确。
`SERVER_ERROR`	服务器内部错误，请稍后再试。

3. 删除用户（DeleteUser）

描述:
此接口用于移除特定应用（ app-id ）下的用户。删除用户后，相关联的TOKEN将失效，无法再用于访问打印服务。第三方应用通过此接口管理和维护用户的生命周期。

请求方式:
`DELETE`

服务地址:
`https://www2.biaotop.com/btc/api/DeleteUser`

Content-Type:
`application/json`

Body

```
{
  "appId": "your_app_id",
  "secureKey": "your_secure_key",
  "userId": "existing_user_uuid"
}
```

参数说明

参数名称	类型	是否必须	描述
appId	String	是	预先分配的应用标识（ app-id ）。
secureKey	String	是	应用的安全密钥（ secure-key ）。
userId	String	是	要删除的用户唯一标识符（ user-id ）。

示例

请求示例

```
curl -X DELETE "https://www2.biaotop.com/btc/api/DeleteUser" \
     -H "Content-Type: application/json" \
     -d '{
           "appId": "app1234567890",
           "secureKey": "secureKeyXYZ12345",
           "userId": "userUUID1234567890"
         }'
```

成功响应示例

```
{
  "status": "success",
  "message": "用户删除成功。"
}
```

失败响应示例

```
{
  "status": "error",
  "message": "用户删除失败。",
  "error": {
    "code": "USER_NOT_FOUND",
    "message": "未找到指定的用户ID。"
  }
}
```

返回参数

Content-Type:

`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"`。
`message`	String	对请求结果的简要描述，如 `"用户删除成功。"`。

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"`。
`message`	String	对请求结果的简要描述，如 `"用户删除失败。"`。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`USER_NOT_FOUND`	未找到指定的用户ID。

错误代码	描述
`INVALID_AUTH`	认证信息无效或已过期。
`MISSING_PARAM`	缺少必要的请求参数，例如 `appId` 、 `secureKey` 、 `userId` 等。
`SERVER_ERROR`	服务器内部错误，请稍后再试。
`INVALID_FORMAT`	提供的数据格式不正确。

4. 获取TOKEN（GetToken）

描述:
此接口用于获取访问打印服务所需的认证令牌（TOKEN）。系统将验证提供的 `appId` 、 `userId` 和 `userKey` 的唯一性，以确保用户的权限和合法性。成功获取TOKEN后，客户端可在后续的API请求中使用该TOKEN进行认证，确保只有授权的应用和用户可以访问打印服务。TOKEN默认有效期为1天。

请求方式:
`POST`

服务地址:
`https://www2.biaotop.com/btc/api/GetToken`

Content-Type:
`application/json`

Body

```
{
  "appId": "your_app_id",
  "userId": "your_user_id", // 第三方提供的Open-ID或自定义UUID
  "nonce": "random_string",
  "signature": "generated_signature"
}
```

参数说明

参数名称	类型	是否必须	描述
`appId`	String	是	预先分配的应用标识（`app-id`），用于标识客户端应用。
`userId`	String	是	用户唯一标识符（`user-id`）。为第三方提供的Open-ID（如微信的 `openId`）或自定义生成的UUID。
`nonce`	String	是	客户端生成的随机字符串（随机数），用于防止重放攻击。 建议使用高熵的随机数或UUID。
`signature`	String	是	使用 `secureKey` 、 `appId` 、 `userId` 和 `nonce` 生成的签名， 用于身份验证。

参数名称	类型	是否必须	描述
			签名生成方式为： `HMAC-SHA256(appId + userId + userKey + nonce)` 。

签名生成说明

1. **拼接字符串**：

```
concatenatedString = appId + userId + userKey + nonce
```

2. **生成签名**：使用 `secureKey` 作为密钥，采用 HMAC-SHA256 算法对拼接后的字符串进行哈希运算，生成签名（`signature`）。

```
const crypto = require('crypto');

function generateSignature(appId, secureKey, userId, userKey, nonce) {
    const concatenatedString = appId + userId + userKey + nonce;
    return crypto.createHmac('sha256', secureKey)
                 .update(concatenatedString)
                 .digest('hex');
}

const appId = 'app1234567890';
const secureKey = 'secureKeyXYZ12345'; // 保密，勿泄露
const userId = 'openid1234567890';     // 或自定义UUID
const userKey = 'userKeyABC1234567890';
const nonce = 'randomString123';

const signature = generateSignature(appId, secureKey, userId, userKey, nonce);
console.log(signature);
```

示例

请求示例

1. **生成 `signature` 的示例（使用 OpenSSL）**
   假设：
   - `appId`：`app1234567890`
   - `userId`：`openid1234567890`
   - `userKey`：`userKeyABC1234567890`
   - `nonce`：`randomString123`

   拼接字符串：

```
app1234567890openid1234567890userKeyABC1234567890randomString123
```

   生成签名：

```
echo -n "app1234567890openid1234567890userKeyABC1234567890randomString123" | openssl dgst -sha25
# 输出示例：f4d5c8e12345e6789abcd...（请使用实际生成的签名）
```

2. **发送请求**

```
curl -X POST "https://www2.biaotop.com/btc/api/GetToken" \
     -H "Content-Type: application/json" \
     -d '{
           "appId": "app1234567890",
           "userId": "openid1234567890",
           "nonce": "randomString123",
           "signature": "f4d5c8e12345e6789abcd..."
         }'
```

成功响应示例

```
{
  "status": "success",
  "message": "TOKEN获取成功。",
  "data": {
    "token": "tokenABC1234567890",
    "expiresAt": "2024-04-28T12:34:56Z"
  }
}
```

失败响应示例

```
{
  "status": "error",
  "message": "TOKEN获取失败。",
  "error": {
    "code": "INVALID_SIGNATURE",
    "message": "签名验证失败。"
  }
}
```

返回参数

Content-Type:
`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"` 。
`message`	String	对请求结果的简要描述，如 `"TOKEN获取成功。"` 。
`data`	Object	包含具体的响应数据对象。
`data.token`	String	获取到的认证TOKEN，用于后续API请求的身份验证。
`data.expiresAt`	String	TOKEN的过期时间，格式为ISO 8601。

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"` 。
`message`	String	对请求结果的简要描述，如 `"TOKEN获取失败。"` 。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`INVALID_SIGNATURE`	签名验证失败。
`INVALID_AUTH`	认证信息无效或已过期。
`MISSING_PARAM`	缺少必要的请求参数，例如 `appId` 、 `userId` 、 `nonce` 等。
`USER_NOT_FOUND`	未找到指定的用户。
`SERVER_ERROR`	服务器内部错误，请稍后再试。
`INVALID_FORMAT`	提供的数据格式不正确。

5. 绑定设备（BindDevice）

描述:
此接口用于将设备（打印机）绑定到特定用户。绑定设备后，相关用户可以通过TOKEN管理和控制该设备。绑定过程中仅需提供设备的唯一标识符（ `deviceId` ）和设备密钥（ `deviceKey` ），无需其他设备详细信息。 `deviceKey` 为设备出厂时打印在机身的验证码，用于验证设备的合法性。

请求方式:

`POST`

服务地址:

`https://www2.biaotop.com/btc/api/BindDevice`

Content-Type:

`application/json`

Header

Header 参数	是否必须	描述
`Authorization`	是	Bearer TOKEN，用于身份验证。

Body

```
{
  "deviceId": "device1234567890",
  "deviceKey": "deviceKeyXYZ1234567890"
}
```

参数说明

参数名称	类型	是否必须	描述
`deviceId`	String	是	设备唯一标识符（`device-id`），由设备供应商提供。
`deviceKey`	String	是	设备密钥（`device-key`），设备出厂时打印在机身的验证码。

示例

请求示例

```
curl -X POST "https://www2.biaotop.com/btc/api/BindDevice" \
     -H "Content-Type: application/json" \
     -H "Authorization: Bearer tokenABC1234567890" \
     -d '{
           "deviceId": "device1234567890",
           "deviceKey": "deviceKeyXYZ1234567890"
         }'
```

成功响应示例

```
{
  "status": "success",
  "message": "设备绑定成功。",
  "data": {
    "deviceId": "device1234567890",
    "userId": "userUUID1234567890",
    "boundAt": "2024-04-27T13:00:00Z"
  }
}
```

失败响应示例

```
{
  "status": "error",
  "message": "设备绑定失败。",
  "error": {
    "code": "INVALID_DEVICE_KEY",
    "message": "设备密钥无效。"
  }
}
```

返回参数

Content-Type:
`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"`。
`message`	String	对请求结果的简要描述，如 `"设备绑定成功。"`。
`data`	Object	包含具体的响应数据对象。
`data.deviceId`	String	绑定的设备唯一标识符。
`data.userId`	String	绑定到的用户唯一标识符。
`data.boundAt`	String	设备绑定时间，格式为ISO 8601。

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"`。
`message`	String	对请求结果的简要描述，如 `"设备绑定失败。"`。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`INVALID_DEVICE_KEY`	设备密钥无效。
`DEVICE_ALREADY_BOUND`	设备已绑定到其他用户。

错误代码	描述
DEVICE_NOT_FOUND	未找到指定的设备ID。
MISSING_PARAM	缺少必要的请求参数，例如 deviceId 、 deviceKey 等。
INVALID_TOKEN	提供的TOKEN无效或已过期。
SERVER_ERROR	服务器内部错误，请稍后再试。
INVALID_FORMAT	提供的数据格式不正确。

6. 解绑设备（UnbindDevice）

描述:
此接口用于将已绑定的设备（打印机）从特定用户解绑。解绑设备后，相关用户将无法再通过TOKEN管理和控制该设备。解绑过程中仅需提供设备的唯一标识符（ deviceId ），无需其他设备详细信息。

请求方式:
POST

服务地址:
https://www2.biaotop.com/btc/api/UnbindDevice

Content-Type:
application/json

Header

Header 参数	是否必须	描述
Authorization	是	Bearer TOKEN，用于身份验证。

Body

```
{
  "deviceId": "device1234567890"
}
```

参数说明

参数名称	类型	是否必须	描述
deviceId	String	是	设备唯一标识符（device-id），由设备供应商提供。

示例

请求示例

```
curl -X POST "https://www2.biaotop.com/btc/api/UnbindDevice" \
     -H "Content-Type: application/json" \
     -H "Authorization: Bearer tokenABC1234567890" \
     -d '{
           "deviceId": "device1234567890"
         }'
```

成功响应示例

```
{
  "status": "success",
  "message": "设备解绑成功。",
  "data": {
    "deviceId": "device1234567890",
    "userId": "userUUID1234567890",
    "unboundAt": "2024-04-28T10:00:00Z"
  }
}
```

失败响应示例

```
{
  "status": "error",
  "message": "设备解绑失败。",
  "error": {
    "code": "DEVICE_NOT_BOUND",
    "message": "设备未绑定或已解绑。"
  }
}
```

返回参数

Content-Type:

`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"`。
`message`	String	对请求结果的简要描述，如 `"设备解绑成功。"`。
`data`	Object	包含具体的响应数据对象。
`data.deviceId`	String	解绑的设备唯一标识符。

参数名称	类型	描述
`data.userId`	String	解绑自的用户唯一标识符。
`data.unboundAt`	String	设备解绑时间，格式为ISO 8601。

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"` 。
`message`	String	对请求结果的简要描述，如 `"设备解绑失败。"` 。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`DEVICE_NOT_BOUND`	设备未绑定或已解绑。
`DEVICE_NOT_FOUND`	未找到指定的设备ID。
`MISSING_PARAM`	缺少必要的请求参数，例如 `deviceId` 。
`INVALID_TOKEN`	提供的TOKEN无效或已过期。
`SERVER_ERROR`	服务器内部错误，请稍后再试。
`INVALID_FORMAT`	提供的数据格式不正确。

7. 获取设备详情（GetDeviceDetail）

描述:
此接口用于获取特定打印设备的详细信息和状态。支持两种访问方式：

- **匿名访问**：通过提交 `deviceId` 和 `deviceKey` 进行设备认证。
- **TOKEN访问**：使用获取到的 `TOKEN` 直接访问用户下所有设备的详情。

绑定设备时由于设备与 `userId` 相关联，使用TOKEN进行绑定和解绑操作更为合适。

请求方式:

`POST`

服务地址:

`https://www2.biaotop.com/btc/api/GetDeviceDetail`

Content-Type:
`application/json`

Header

Header 参数	是否必须	描述
`Authorization`	否	`Bearer` 风格的认证令牌，用于TOKEN访问。

Body

根据访问方式的不同，提交的信息有所不同。

8. 基于TOKEN的打印（Print with Token）

描述:
基于TOKEN的打印模式允许已认证的用户通过TOKEN提交打印任务。用户可以选择指定特定设备进行打印，或在未指定设备的情况下，将打印任务发送至用户下的所有设备。使用TOKEN授权的打印请求可以简化设备管理，并确保打印任务的安全。

请求方式:
`POST`

服务地址:
`https://www2.biaotop.com/btc/api/printWithToken`

Content-Type:
`multipart/form-data`

Header

Header 参数	是否必须	描述
`Authorization`	是	Bearer TOKEN，用于身份验证。
`x-DeviceIds`	否	以逗号分隔的设备ID列表，指定要打印的设备。如果未提供， 打印任务将发送至用户下的所有设备。

Body

- **类型**： `multipart/form-data`
- **内容**：上传多个PDF或JPG文件。

示例

`curl` 请求示例

以下是使用 `curl` 进行基于TOKEN的打印的示例，包含指定设备和所有设备两种情况。

1. **发送打印任务至指定设备**
   假设：
   - TOKEN：tokenABC1234567890

   发送打印请求：

```
curl -X POST "https://www2.biaotop.com/btc/api/printWithToken" \
     -F "file=@/path/to/document1.pdf;filename=document1.pdf" \
     -F "file=@/path/to/image1.jpg;filename=image1.jpg" \
     -F "file=@/path/to/document2.pdf;filename=document2.pdf" \
     -H "Authorization: Bearer tokenABC1234567890" \
     -H "x-DeviceIds: device1234567890,device0987654321"
```

2. **发送打印任务至所有绑定设备**
   假设：
   - TOKEN：tokenABC1234567890

   发送打印请求：

```
curl -X POST "https://www2.biaotop.com/btc/api/printWithToken" \
     -F "file=@/path/to/document1.pdf;filename=document1.pdf" \
     -F "file=@/path/to/image1.jpg;filename=image1.jpg" \
     -F "file=@/path/to/document2.pdf;filename=document2.pdf" \
     -H "Authorization: Bearer tokenABC1234567890" \
```

成功响应示例

```
{
  "status": "success",
  "message": "打印任务已提交。",
  "data": {
    "printJobId": "printJob0987654321",
    "deviceIds": ["device1234567890", "device0987654321"],
    "submittedAt": "2024-04-28T09:30:00Z"
  }
}
```

失败响应示例

```
{
  "status": "error",
  "message": "打印任务提交失败。",
  "error": {
    "code": "INVALID_TOKEN",
    "message": "提供的TOKEN无效或已过期。"
  }
}
```

返回参数

Content-Type:

`application/json`

成功响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"success"`。
`message`	String	对请求结果的简要描述，如 `"打印任务已提交。"`。
`data`	Object	包含具体的响应数据对象。
`data.printJobId`	String	打印任务的唯一标识符。
`data.deviceIds`	Array	已提交打印任务的设备ID列表。
`data.submittedAt`	String	打印任务提交时间，格式为ISO 8601。

失败响应参数

参数名称	类型	描述
`status`	String	请求状态，固定为 `"error"`。
`message`	String	对请求结果的简要描述，如 `"打印任务提交失败。"`。
`error`	Object	包含具体的错误信息对象。
`error.code`	String	错误代码，用于程序化地识别错误类型。
`error.message`	String	对错误的详细描述，帮助用户理解问题所在。

错误代码说明

错误代码	描述
`INVALID_TOKEN`	提供的TOKEN无效或已过期。
`DEVICE_NOT_FOUND`	未找到指定的设备ID。
`MISSING_PARAM`	缺少必要的请求参数。
`SERVER_ERROR`	服务器内部错误，请稍后再试。
`INVALID_FORMAT`	提供的数据格式不正确。
`FILE_NOT_SUPPORTED`	提交的文件类型不支持。

9. 实时监控（WebSocket - WSS）

描述:
实时监控接口通过WebSocket（WSS）通道，允许客户端实时获取打印任务和设备状态。不同的认证方式决定了客户端可以获取的监控信息：

- **匿名访问**：通过提交 `printJobId` ，获取特定打印任务的状态。
- **基于TOKEN的访问**：通过提交TOKEN，获取用户下所有设备及其相关打印任务的状态。

WebSocket地址:

```
wss://www2.biaotop.com/btc/api/Monitor
```

连接方式

1. **建立连接**
   客户端通过WebSocket协议连接到监控服务器：

   ```
   const socket = new WebSocket('wss://www2.biaotop.com/btc/api/Monitor');

   socket.onopen = () => {
       console.log('WebSocket连接已建立。');
       // 根据认证方式发送认证信息
   };

   socket.onerror = (error) => {
       console.error('WebSocket连接错误：', error);
   };

   socket.onmessage = (event) => {
       const data = JSON.parse(event.data);
       // 处理监控数据
       console.log('接收到数据：', data);
   };

   socket.onclose = () => {
       console.log('WebSocket连接已关闭。');
   };
   ```

2. **认证方式**
   - **匿名访问**:
     连接建立后，发送包含 `printJobId` 的消息以获取特定打印任务的状态。

     ```
     const printJobId = 'printJob1234567890'; // 替换为实际的printJobId

     const authMessage = {
         type: 'anonymous',
         printJobId: printJobId
     };

     socket.send(JSON.stringify(authMessage));
     ```

- **基于TOKEN的访问**:
  连接建立后，发送包含TOKEN的消息以获取用户下所有设备及其打印任务的状态。

```
const token = 'tokenABC1234567890'; // 替换为实际的TOKEN

const authMessage = {
    type: 'token',
    token: token
};

socket.send(JSON.stringify(authMessage));
```

接收消息格式

- **匿名访问**:

```
{
  "printJobId": "printJob1234567890",
  "status": "completed",
  "completedAt": "2024-04-27T13:00:00Z"
}
```

- **基于TOKEN的访问**:

```
{
  "devices": [
    {
      "deviceId": "device1234567890",
      "status": "online",
      "tasks": [
        {
          "printJobId": "printJob1234567890",
          "status": "completed",
          "completedAt": "2024-04-27T13:00:00Z"
        },
        {
          "printJobId": "printJob0987654321",
          "status": "printing",
          "startedAt": "2024-04-27T14:00:00Z"
        }
      ]
    },
    {
      "deviceId": "device0987654321",
      "status": "offline",
      "tasks": []
    }
  ]
}
```

发送和接收消息示例

1. **匿名访问示例**

```
const socket = new WebSocket('wss://www2.biaotop.com/btc/api/Monitor');

socket.onopen = () => {
    const printJobId = 'printJob1234567890';
    const authMessage = {
        type: 'anonymous',
        printJobId: printJobId
    };
    socket.send(JSON.stringify(authMessage));
};

socket.onmessage = (event) => {
    const data = JSON.parse(event.data);
    console.log(`打印任务 ${data.printJobId} 的状态：`, data.status);
};
```

2. **基于TOKEN的访问示例**

```
const socket = new WebSocket('wss://www2.biaotop.com/btc/api/Monitor');

socket.onopen = () => {
    const token = 'tokenABC1234567890';
    const authMessage = {
        type: 'token',
        token: token
    };
    socket.send(JSON.stringify(authMessage));
};

socket.onmessage = (event) => {
    const data = JSON.parse(event.data);
    data.devices.forEach(device => {
        console.log(`设备 ${device.deviceId} 的状态：`, device.status);
        device.tasks.forEach(task => {
            console.log(`打印任务 ${task.printJobId} 的状态：`, task.status);
        });
    });
};
```

消息协议

- 所有消息均采用JSON格式。
- 消息包括认证信息和数据更新。
- 认证信息的 `type` 字段用于区分认证模式（ `anonymous` 或 `token` ）。

示例响应消息

1. **匿名访问成功响应**

```
{
  "printJobId": "printJob1234567890",
  "status": "completed",
  "completedAt": "2024-04-27T13:00:00Z"
}
```

2. **基于TOKEN的访问成功响应**

```
{
  "devices": [
    {
      "deviceId": "device1234567890",
      "status": "online",
      "tasks": [
        {
          "printJobId": "printJob1234567890",
          "status": "completed",
          "completedAt": "2024-04-27T13:00:00Z"
        },
        {
          "printJobId": "printJob0987654321",
          "status": "printing",
          "startedAt": "2024-04-27T14:00:00Z"
        }
      ]
    },
    {
      "deviceId": "device0987654321",
      "status": "offline",
      "tasks": []
    }
  ]
}
```

错误响应示例

```
{
  "error": {
    "code": "INVALID_TOKEN",
    "message": "提供的TOKEN无效或已过期。"
  }
}
```

注意事项

- **连接安全**：确保使用 `wss://` 协议，以保障数据传输的安全性。

- **认证信息**：根据使用的认证方式，正确发送认证信息。错误的认证信息将导致连接失败或权限不足。
- **数据格式**：确保发送和接收的JSON数据格式正确，避免解析错误。
- **实时性**：监控数据为实时更新，客户端应及时处理接收到的消息。

10. 错误代码说明

详细列出了各个API接口可能返回的错误代码及其描述，帮助开发者快速定位和解决问题。

错误代码	描述
`INVALID_DEVICE`	设备认证信息无效或已过期。
`INVALID_DEVICE_KEY`	设备密钥无效。
`DEVICE_ALREADY_BOUND`	设备已绑定到其他用户。
`DEVICE_NOT_FOUND`	未找到指定的设备ID。
`USER_EXISTS`	该用户ID已存在。
`USER_NOT_FOUND`	未找到指定的用户ID。
`INVALID_SIGNATURE`	签名验证失败。
`INVALID_AUTH`	认证信息无效或已过期。
`MISSING_PARAM`	缺少必要的请求参数。
`INVALID_FORMAT`	提供的数据格式不正确。
`SERVER_ERROR`	服务器内部错误，请稍后再试。
`FILE_NOT_SUPPORTED`	提交的文件类型不支持。

11. 注意事项

1. **用户标识唯一性**：
   - 确保每个 `userId` 在同一 `appId` 下是唯一的，避免数据冲突和安全问题。
2. **设备标识唯一性**：
   - 确保每个 `deviceId` 在系统中是唯一的，避免设备混淆和权限问题。
3. **安全存储**：
   - `secureKey` 和 `userKey` 必须妥善保管，避免泄露。建议通过安全的环境变量或安全存储机制管理，不要在客户端代码中硬编码。
4. **关联Open-ID**：
   - 如果需要跨平台用户识别，可以在 `userDetails` 中关联用户的Open-ID。
5. **数据验证**：
   - 服务器应对输入的数据进行严格验证，确保字段格式和内容的合法性。
6. **图片要求**：

- 上传的JPG图片必须为360dpi。
- 纸张高度和宽度必须与图片的实际高度和宽度准确对应，避免打印比例失衡。

7. **TOKEN管理**：
   - TOKEN默认有效期为1天。开发者应定期刷新TOKEN，确保打印服务的连续性。
8. **实时监控**：
   - 在使用WebSocket进行实时监控时，确保客户端能够处理和更新接收到的数据，保持监控界面的实时性和准确性。
9. **打印内容格式**：
   - `printJson` 模式下，不可同时上传文件和打印内容布局。
   - 确保 `printJson` 中的元素类型一致，不混合文本和图片的定义，以免导致打印错误。
10. **网络稳定性**：
    - 保持客户端与服务器的网络连接稳定，特别是在进行实时监控和大文件打印时，避免因网络波动导致的打印任务失败或延迟。