阿里云
小程序云

开发指南

文档版本：20200630

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **注意：** 权重设置为0，该服务器不会再接受新请求。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	单击**设置** > **网络** > **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令。	执行cd /d C:/window命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid Instance_ID
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig [-all|-t]
{}或者{a|b}	表示必选项，至多选择一个。	switch {active|stand}

目录

1 SDK 文档

1.1 变更历史

记录各版本SDK的新增特性。

版本	版本号	说明
客户端SDK 2.3	@alicloud/mpserverless-sdk@2.3.2	支付宝小程序修改为静默授权。更多信息请参见安装客户端SDK2.3版本。
客户端SDK 2.2	@alicloud/mpserverless-sdk@2.2.0	增加统计分析功能。更多信息请参见安装客户端SDK2.2版本。
客户端SDK 2.1	@alicloud/mpserverless-sdk@2.1.0	增加匿名用户授权方式。 更多信息，请参见#unique_7。
客户端SDK 2.0	@alicloud/mpserverless-sdk@2.0.0	支持在微信小程序客户端中使用小程序云Serverless服务。 更多信息，请参见在微信小程序中使用SDK。
客户端SDK 1.0	@alicloud/mpserverless-sdk@1.0.3	通过小程序SDK可以直接在支付宝IDE中调用小程序云的云函数、云存储和数据库等服务。

1.2 安装客户端SDK2.3版本

通过小程序云SDK，您可以在小程序中直接访问小程序云Serverless服务。2.3版本支付宝小程序修改为静默授权。

前提条件

在首次使用小程序云服务前，您需要开通小程序云服务。详细信息，请参见#unique_9。

在支付宝小程序中使用SDK

完成以下操作，在支付宝小程序中使用SDK：

1. 在支付宝小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/mpserverless-sdk@2.3.2 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerless from '@alicloud/mpserverless-sdk';

const mpServerless = new MPServerless({
```

```
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode,
  getFileInfo: my.getFileInfo,
  getImageInfo: my.getImageInfo
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在支付宝小程序开放平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'alipay_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

其中：

说明：

- 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
- 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
- 如果没有设置**authType**，则必须设置**authProvider**。
- 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

在微信小程序中使用SDK

完成以下操作，在微信小程序中使用小程序云SDK：

1. 单击这里下载SDK文件。
2. 将下载后的mpserverless.js文件添加到小程序项目目录下，然后在app.js或其他需要的页面中引入SDK文件。

```
//app.js
const MPServerless = require('/sdk/mpserverless.js');

const mpServerless = new MPServerless({
  uploadFile: wx.uploadFile,
  request: wx.request,
```

```
  getAuthCode: wx.login,
  getFileInfo: wx.getFileInfo,
  getImageInfo: wx.getImageInfo,
}, {
  appId: 'appId', // 小程序应用标识
  spaceId: '', // 服务空间标识
  clientSecret: '', // 服务空间 secret key
  endpoint: '', // 服务空间地址，从小程序 serverless 控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在微信公众平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权

```
const res = await mpServerless.user.authorize({
  authProvider: 'wechat_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

其中：

说明：

- 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
- 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
- 如果没有设置**authType**，则必须设置**authProvider**。
- 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

在钉钉小程序中使用SDK

完成以下操作，在第三方个人应用的钉钉小程序中使用小程序云SDK：

1. 在钉钉小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/dt-mpserverless-sdk@0.0.2 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerlessSDK from '@alicloud/dt-mpserverless-sdk';
const mpServerless = new MPServerlessSDK({
  uploadFile: dd.uploadFile,
  request: dd.httpRequest,
  getAuthCode: dd.getAuthCode
}, {
  appId: '1234456789', // 小程序应用标识
```

```
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
})
```

其中：

- **appId**是小程序的ID。您可以在钉钉开放平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'dingtalk_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

其中：

说明：

- 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
- 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
- 如果没有设置**authType**，则必须设置**authProvider**。
- 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

1.3 安装客户端SDK2.2版本

通过小程序云SDK，您可以在小程序中直接访问小程序云Serverless服务。2.2版本开始支持统计分析功能。

前提条件

在首次使用小程序云服务前，您需要开通小程序云服务。详细信息，请参见#unique_9。

在支付宝小程序中使用SDK

完成以下操作，在支付宝小程序中使用SDK：

1. 在支付宝小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/mpserverless-sdk@2.2 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerless from '@alicloud/mpserverless-sdk';

const mpServerless = new MPServerless({
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode,
  getFileInfo: my.getFileInfo,
  getImageInfo: my.getImageInfo
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在支付宝小程序开放平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'alipay_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

其中：

- **authProvider**：指定授权方，取值alipay_openapi和wechat_openapi。
- **authType**：指定授权方式。取值anonymous表示通过匿名方式授权。

> **说明：**
>
> - 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
> - 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
> - 如果没有设置**authType**，则必须设置**authProvider**。
> - 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

在微信小程序中使用SDK

完成以下操作，在微信小程序中使用小程序云SDK：

1. 单击这里下载SDK文件。
2. 将下载后的mpserverless.js文件添加到小程序项目目录下，然后在app.js或其他需要的页面中引入SDK文件。

```
//app.js
const MPServerless = require('/sdk/mpserverless.js');

const mpServerless = new MPServerless({
  uploadFile: wx.uploadFile,
  request: wx.request,
  getAuthCode: wx.login,
  getFileInfo: wx.getFileInfo,
  getImageInfo: wx.getImageInfo,
}, {
  appId: 'appId', // 小程序应用标识
  spaceId: '', // 服务空间标识
  clientSecret: '', // 服务空间 secret key
  endpoint: '', // 服务空间地址，从小程序 serverless 控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在微信公众平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权

```
const res = await mpServerless.user.authorize({
  authProvider: 'wechat_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

其中：

- **authProvider**：指定授权方，取值alipay_openapi和wechat_openapi。
- **authType**：指定授权方式。取值anonymous表示通过匿名方式授权。

> **说明：**
>
> - 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
> - 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
> - 如果没有设置**authType**，则必须设置**authProvider**。

- 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

在钉钉小程序中使用SDK

完成以下操作，在第三方个人应用的钉钉小程序中使用小程序云SDK：

1. 在钉钉小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/dt-mpserverless-sdk@0.0.2 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerlessSDK from '@alicloud/dt-mpserverless-sdk';
const mpServerless = new MPServerlessSDK({
  uploadFile: dd.uploadFile,
  request: dd.httpRequest,
  getAuthCode: dd.getAuthCode
}, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
})
```

其中：

- **appId**是小程序的ID。您可以在钉钉开放平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'dingtalk_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

其中：

- **authProvider**：指定授权方，钉钉SDK仅支持取值dingtalk_openapi。
- **authType**：指定授权方式。取值anonymous表示通过匿名方式授权。

说明：

- 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
- 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
- 如果没有设置**authType**，则必须设置**authProvider**。

- 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

1.4 安装客户端SDK2.2版本

通过小程序云SDK，您可以在小程序中直接访问小程序云Serverless服务。2.2版本开始支持统计分析功能。

前提条件

在首次使用小程序云服务前，您需要开通小程序云服务。详细信息，请参见#unique_9。

在支付宝小程序中使用SDK

完成以下操作，在支付宝小程序中使用SDK：

1. 在支付宝小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/mpserverless-sdk@2.2 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerless from '@alicloud/mpserverless-sdk';

const mpServerless = new MPServerless({
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode,
  getFileInfo: my.getFileInfo,
  getImageInfo: my.getImageInfo
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在支付宝小程序开放平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'alipay_openapi'
});
if (res.success) {
  console.log('授权成功');
```

```
}
```

其中：

- **authProvider**：指定授权方，取值alipay_openapi和wechat_openapi。
- **authType**：指定授权方式。取值anonymous表示通过匿名方式授权。

> **说明：**
> - 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
> - 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
> - 如果没有设置**authType**，则必须设置**authProvider**。
> - 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

在微信小程序中使用SDK

完成以下操作，在微信小程序中使用小程序云SDK：

1. 单击这里下载SDK文件。
2. 将下载后的mpserverless.js文件添加到小程序项目目录下，然后在app.js或其他需要的页面中引入SDK文件。

```
//app.js
const MPServerless = require('/sdk/mpserverless.js');

const mpServerless = new MPServerless({
  uploadFile: wx.uploadFile,
  request: wx.request,
  getAuthCode: wx.login,
  getFileInfo: wx.getFileInfo,
  getImageInfo: wx.getImageInfo,
}, {
  appId: 'appId', // 小程序应用标识
  spaceId: '', // 服务空间标识
  clientSecret: '', // 服务空间 secret key
  endpoint: '', // 服务空间地址，从小程序 serverless 控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在微信公众平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权

```
const res = await mpServerless.user.authorize({
  authProvider: 'wechat_openapi'
});
```

```
if (res.success) {
  console.log('授权成功');
}
```

其中：

- **authProvider**：指定授权方，取值alipay_openapi和wechat_openapi。
- **authType**：指定授权方式。取值anonymous表示通过匿名方式授权。

> **说明：**
>
> - 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
> - 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
> - 如果没有设置**authType**，则必须设置**authProvider**。
> - 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

在钉钉小程序中使用SDK

完成以下操作，在第三方个人应用的钉钉小程序中使用小程序云SDK：

1. 在钉钉小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/dt-mpserverless-sdk@0.0.2 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerlessSDK from '@alicloud/dt-mpserverless-sdk';
const mpServerless = new MPServerlessSDK({
  uploadFile: dd.uploadFile,
  request: dd.httpRequest,
  getAuthCode: dd.getAuthCode
}, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
})
```

其中：

- **appId**是小程序的ID。您可以在钉钉开放平台获取小程序的App ID。
- **spaceId**、**clientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'dingtalk_openapi'
});
if (res.success) {
```

```
  console.log('授权成功');
}
```

其中：

- **authProvider**：指定授权方，钉钉SDK仅支持取值dingtalk_openapi。
- **authType**：指定授权方式。取值anonymous表示通过匿名方式授权。

> **说明：**
>
> - 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
> - 如果将**authType**设置为anonymous，则不需要设置**authProvider**。
> - 如果没有设置**authType**，则必须设置**authProvider**。
> - 匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

1.5 安装客户端SDK2.0版本

通过小程序云SDK，您可以在小程序中直接访问小程序云Serverless服务。2.0版本支持支付宝和微信小程序，在安装SDK后，您需要调用user.authorize进行授权。

前提条件

在首次使用小程序云服务前，您需要开通小程序云服务。详细信息，请参见#unique_9。

在支付宝小程序中使用SDK

完成以下操作，在支付宝小程序中使用SDK：

1. 在支付宝小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/mpserverless-sdk@2.0.0 --save
```

2. 在小程序项目中添加以下代码，初始化SDK。

```
import MPServerless from '@alicloud/mpserverless-sdk';

const mpServerless = new MPServerless({
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
```

```
});
```

其中：

- **appId**是小程序的ID。您可以在支付宝小程序开放平台获取小程序的App ID。
- **spaceIdclientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权。

```
const res = await mpServerless.user.authorize({
  authProvider: 'alipay_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

在微信小程序中使用SDK

完成以下操作，在微信小程序中使用小程序云SDK：

1. 单击这里下载SDK文件。
2. 将下载后的mpserverless.js文件添加到小程序项目目录下，然后在app.js或其他需要的页面中引入SDK文件。

```
//app.js
const MPServerless = require('/sdk/mpserverless.js');

const mpServerless = new MPServerless({
  uploadFile: wx.uploadFile,
  request: wx.request,
  getAuthCode: wx.login,
  getFileInfo: wx.getFileInfo,
  getImageInfo: wx.getImageInfo,
}, {
  appId: 'appId', // 小程序应用标识
  spaceId: '', // 服务空间标识
  clientSecret: '', // 服务空间 secret key
  endpoint: '', // 服务空间地址，从小程序 serverless 控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在微信公众平台获取小程序的App ID。
- **spaceIdclientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

3. 调用user.authorize方法完成授权

```
const res = await mpServerless.user.authorize({
  authProvider: 'wechat_openapi'
});
if (res.success) {
  console.log('授权成功');
```

```
}
```

1.6 安装客户端SDK1.0版本

通过小程序云SDK，您可以在小程序中直接访问小程序云Serverless服务。

前提条件

在首次使用小程序云服务前，您需要开通小程序云服务。详细信息，请参见#unique_9。

操作步骤

1. 在小程序项目的根目录执行以下命令安装SDK。

```
npm install @alicloud/mpserverless-sdk@1.0.3 --save
```

2. 执行以下代码完成初始化。

```
import MPServerless from '@alicloud/mpserverless-sdk';
const mpServerless = new MPServerless({
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
});
```

其中：

- **appId**是小程序的ID。您可以在支付宝小程序控制台获得。
- **spaceIdclientSecret**和**endpoint**在小程序Serverless控制台创建服务空间后可以获得。详情参见#unique_10。

2 客户端API参考

2.1 云函数

2.1.1 invoke

调用云服务。

方法定义

该方法的定义如下：

```
mpserverless.function.invoke(functionName: string, options?: object): Promise<Result>
```

返回参数

字段名	类型	说明
success	Boolean	执行状态。
requestId	String	请求ID。
result	Any	接口返回内容，由开发者代码和请求参数 header 的 content-type 决定，默认 content-type 为 application/json。

示例

```
mpserverless.function.invoke('dataAnalytics', {
    range: 30,
}).then((res) => {
  console.log(res);
}).catch(console.error);
```

2.2 数据存储

2.2.1 aggregate

对数据库执行聚合查询。

方法定义

该方法的定义如下：

```
aggregate(pipeline: object | object[], options?: object): Promise<MongoResult>
```

请求参数

该方法接收11个参数，其定义如下：

字段名	类型	必填	说明
pipeline	Array	是	聚合查询对象。
options	Object	否	控制项。
options.explain	Boolean	否	是否返回执行计划。
options.allowDiskUse	Boolean	否	是否在聚合查询执行的过程中使用磁盘存储临时数据。
options.maxTimeMS	Number	否	执行查询的超时时间。
options.bypassDocumentValidation	Boolean	否	是否允许绕过文档验证。
options.raw	Boolean	否	是否将结果作为 BSON Buffer 返回。
options.promoteLongs	Boolean	否	是否将 long 数据类型提升为 Number。
options.promoteValues	Boolean	否	是否将 BSON 值的类型提升为本地类型。
options.promoteBuffers	Boolean	否	是否将 BSON Buffer 转换为 Node Buffer。
options.collation	Object	否	指定更新的排序顺序。

示例

```
mpserverless.db.collection('users').aggregate([
  {
   $match: {
    _id: {
     $in: insertRet.insertedIds
    }
   },
  },
  {
   $project: {
    name: 1,
    address: 1
   }
  }, {
   $group: {
    _id: '$name',
    count: {$sum: 1}
   }
  },{
   $match: {
    count: {
     $gt: 1
    }
   }
  }
 ]);
```

2.2.2 count

获取集合中符合条件的记录数量。

方法定义

该方法的定义如下：

```
count(query: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 5 个参数，其定义如下：

字段名	类型	必填	说明
query	Object	否	数据库操作时的过滤条件。
options	Object	否	控制项。
options.limit	Boolean	否	是否限制 count 的文档数量。
options.skip	Number	否	count 前跳过的文档数量。
options.maxTimeMS	Number	否	超时时间。

示例

```
mpserverless.db.collection('users').count({age: {$gt: 18}}).then((res) => {
   console.log(res);
}).catch(console.error);
```

2.2.3 deleteOne

删除集合中的一条记录。如果没有查询条件，则默认删除第一行。

方法定义

该方法的定义如下：

```
deleteOne(filter: object): Promise<MongoResult>
```

请求参数

该方法接收一个参数，其定义如下：

字段名	类型	必填	说明
filter	Object	否	数据库操作时的过滤条件

示例

```
mpserverless.db.collection('users').deleteOne({age: {$lt: 18}).then((res) => {
  const hasDeleted = res.affectedDocs > 0;
}).catch(console.error);
```

2.2.4 deleteMany

删除集合中的一批记录。

方法定义

该方法的定义如下：

```
deleteMany(filter: object): Promise<MongoResult>;
```

请求参数

字段名	类型	必填	说明
filter	Object	是	数据库操作时的过滤条件。

示例

```
mpserverless.db.collection('users').deleteMany({age: {$lt: 18}}).then((res) => {
  const hasDeleted = res.affectedDocs > 0;
```

```
}).catch(console.error);
```

2.2.5 distinct

获取某个属性去重后的所有记录。

方法定义

该方法的定义如下：

```
distinct(key: string, query: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 3 个参数，其定义如下：

字段名	类型	必填	说明
key	String	是	待获取的属性名。
query	Object	是	数据库操作时的查询条件。
options	Object	否	控制项。

示例

```
mpserverless.db.collection('users').distinct('name', {age: {$gt: 18})
.then((res) => {})
.catch(console.error);
```

2.2.6 find

查找集合中符合条件的所有记录。

方法定义

该方法的定义如下：

```
find(query?: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 8 个参数，其定义如下：

字段名	类型	必填	说明
query	Object	否	数据库操作时的查询条件。
options	Object	否	控制项。
options.limit	Number	否	查询的文档数量限制。

字段名	类型	必填	说明
options.skip	Number	否	跳过的文档数量。
options.maxTimeMS	Number	否	超时时间。
options.sort	Object	否	指定排序的字段，并使用 1 和 -1 来指定排序的方式。其中： • 1：表示升序排列 • -1：表示降序排列
options.projection	Object	否	使用投影操作符指定返回的键，值设置为1的字段返回，值为0的字段隐藏。
options.hint	Object	否	指定查询时使用的索引。

示例

```
mpserverless.db.collection('users').find({
    age: {$gt: 18}
}, {
    projection: {name: 1},
    limit: 10,
    skip: 10,
    sort: {name: 1},
    hint: {name: 1, _id: 0}
})
.then(res => {})
.catch(console.error);
```

2.2.7 findOne

查询单条记录。

方法定义

该方法的定义如下：

```
findOne(query?: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 8 个参数，其定义如下：

字段名	类型	必填	说明
query	Object	否	数据库操作时的查询条件。

字段名	类型	必填	说明
options	Object	否	控制项。
options.limit	Number	否	查询的文档数量限制。
options.skip	Number	否	跳过的文档数量。
options.maxTimeMS	Number	否	超时时间。
options.sort	Object	否	指定排序的字段，并使用 1 和 -1 来指定排序的方式。其中： • 1：表示升序排列 • -1：表示降序排列
options.projection	Object	否	使用投影操作符指定返回的键，值设置为1的字段返回，值为0的字段隐藏。
options.hint	Object	否	指定查询时使用的索引。

示例

```
mpserverless.db.collection('users')
.findOne({
    age: {$gt: 18}
}, {
    projection: {name: 1},
    limit: 10,
    skip: 10,
    sort: {age:1}
})
.then(res => {})
.catch(console.error)
```

2.2.8 findOneAndDelete

查询并删除一条记录。

方法定义

该方法的定义如下：

```
findOneAndDelete(query?: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 5 个参数，其定义如下：

字段	类型	必填	说明
query	Object	是	数据库操作时的查询条件。
options	Object	否	控制项。
options.maxTimeMS	Number	否	超时时间。
options.sort	Object	否	排序规则。
options.projection	Object	否	查询后过滤的字段。

示例

```
mpserverless.db.collection('users')
.findOneAndDelete({
   age: 18
})
.then(res => {})
.catch(console.error)
```

2.2.9 findOneAndReplace

查询并整体替换一条记录。

方法定义

该方法的定义如下：

```
findOneAndReplace(filter: object, replacement: object, options?: object): Promise<
MongoResult>
```

请求参数

该方法接收 6 个参数，其定义如下：

字段	类型	必填	描述
filter	Object	是	数据库操作时的查询条件。
replacement	Object	是	数据库操作时的替换对象。
options	Object	否	控制项。
options.maxTimeMS	Number	否	超时时间。
options.sort	Object	否	排序规则
options.upsert	Boolean	否	如果查找不到对应文档，是否插入。默认值：false。

字段	类型	必填	描述
options.projection	Object	否	查询后过滤的字段。

示例

```
mpserverless.db.collection(database)
  .findOneAndReplace({
    "score": { $gt: 20000 }
  },
  {
    "team": "Therapeutic Hamsters",
    "score": 22250
  })
  .then(res => {})
  .catch(console.error);
```

2.2.10 findOneAndUpdate

查询并更新记录。

方法定义

该方法的定义如下：

```
findOneAndUpdate(filter: object, update: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 7个参数，其定义如下：

字段	类型	必填	说明
filter	Object	是	数据库操作时的过滤条件。
update	Object	是	数据库操作时的更新对象。
options	String	否	控制项。
options.maxTimeMS	Number	否	超时时间。
options.sort	Object	否	排序规则。
options.upsert	Boolean	否	如果查找不到对应文档，是否插入。默认值：false。
options.projection	Object	否	查询后过滤的字段。

示例

```
mpserverless.db.collection('users')
   .findOneAndUpdate({
      username: "zhangsan"
   },
   {
   $set: {
      age: 18
      }
   })
   .then(res => {})
   .catch(console.error)
```

2.2.11 replaceOne

查询并整体替换这条记录。

方法定义

该方法的定义如下：

```
replaceOne(filter: object, doc: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 4 个参数，其定义如下：

字段名	类型	必填	说明
filter	Object	是	数据库操作时的过滤条件。
doc	Object	是	替换的目标文档。
options	Object	否	控制项。
options.upsert	Boolean	否	如果匹配不到，是否插入。默认值：false。

示例

```
mpserverless.db.collection('users')
.replaceOne({
   name: 'tom'
   age: 18
},{
   name: 'jerry'
})
.then(res => {})
```

```
.catch(console.error);
```

2.2.12 insertMany

在集合中添加一批记录。

方法定义

该方法的定义如下：

```
insertMany(docs: object[]): Promise<MongoResult>
```

请求参数

该方法接收 1 个参数，其定义如下：

字段名	类型	必填	说明
docs	Array	是	待插入的数据，只能为数组。

示例

```
mpserverless.db.collection('users').insertMany([{
    name: 'tom',
    age: 1
},{
    name: 'jerry',
    age: 2
}])
.then(res => {})
.catch(console.error)
```

2.2.13 insertOne

在集合中添加一条记录。

方法定义

该方法的定义如下：

```
insertOne(doc: object): Promise<MongoResult>
```

请求参数

该方法接收 1 个参数，其定义如下：

字段名	类型	必填	说明
doc	Object	是	待插入的数据

示例

```
mpserverless.db.collection('users').insertOne({
    name: 'tom',
    age: 1
})
.then(res => {})
.catch(console.error)
```

2.2.14 updateMany

更新集合中的一批记录。

方法定义

该方法的定义如下：

```
updateMany(filter: object, update: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 4 个参数，其定义如下：

字段名	类型	必填	说明
filter	Object	是	过滤条件。
update	Object	是	更新的文档
options	Object	否	控制项
options.upsert	Boolean	否	不匹配时是否直接插入文档。默认值：false。

示例

```
mpserverless.db.collection('users').updateMany({
    name: 'jerry'
}, {
    $set: {
        age: 10
    }
})
.then(res => {})
```

```
.catch(console.error)
```

2.2.15 updateOne

更新集合中的一条记录。

方法定义

该方法的定义如下：

```
updateOne(filter: object, update: object, options?: object): Promise<MongoResult>
```

请求参数

该方法接收 4 个参数，其定义如下：

字段名	类型	必填	说明
filter	Object	是	过滤条件。
update	Object	是	更新的文档。
options	Object	否	控制项。
options.upsert	Boolean	否	不匹配时是否直接插入文档。默认值：false。

示例

```
mpserverless.db.collection('users').updateOne({
    name: 'jerry'
}, {
    $set: {
        age: 10
    }
})
.then(res => {})
.catch(console.error)
```

2.3 文件存储

2.3.1 uploadFile

上传文件。

方法定义

该方法的定义如下：

```
mpserverless.file.uploadFile(options: object): Promise<Result>
```

请求参数

该方法接收 6 个参数，其定义如下：

字段名	类型	必填	说明
options	Object	是	上传控制项。
options.fileSize	String	否	上传的文件大小。
options.extension	String	否	上传的文件的扩展名。 目前支持上传以下格式的文件： • .jpg • .jpeg • .png • .gif • .image
options.filePath	String	是	本地文件路径，通常可以从图片文件描述中获取文件路径。可上传文件大小限制在 100 MB 以内。
options.env	String	否	文件的获取方式。 唯一可选值 public： 可公开访问的文件。
options.timeout	Number	否	超时时间，以毫秒为单位。默认值：60000。
options.headers	Object	否	文件响应头键值对，可定义如下内容： • cacheControl： 缓存策略。 • contentDisposition： 文件形式。 • contentEncoding： 文件内容编码。 • expires： 缓存有效时长。
options.meta	Object	否	自定义文件响应头键值对。例如，自定义 userId: halo 获得响应头 x-meta-user-id: halo 。

返回参数

字段名	类型	说明
filePath	String	本地文件路径。
fileUrl	String	上传文件后获得的文件链接地址。

示例

小程序选取文件上传示例。

```
my.chooseImage({
  chooseImage: 1,
  success: res => {
    const path = res.apFilePaths[0];
    const options = {
      filePath: path,
      headers: {
        contentDisposition: 'attachment',
      },
    };

    mpserverless.file.uploadFile(options).then(console.log).catch(console.error);
  },
});
```

2.3.2 deleteFile

删除之前上传的文件。

方法定义

该方法的定义如下：

```
mpserverless.file.deleteFile(url: string): Promise<Result>
```

请求参数

该方法接收 1 个参数，其定义如下：

字段名	类型	必填	说明
url	String	是	线上完整路径

示例

```
mpserverless.file.deleteFile('https://resource.bspapp.com/xxx-xx/4b82ded0-0118-4de4
-9f50-ab13110a1ffb.jpg').then(res => {
  console.log(res);
}).catch(err => {
  console.error(err);
```

```
});
```

2.4 用户管理

2.4.1 getInfo

获取用户信息。

方法定义

该方法的定义如下：

```
mpserverless.user.getInfo(options: object): Promise<Result>
```

请求参数

该方法接收以下请求参数。

字段名	类型	必填	说明
options	Object	否	**authProvider{String}**指定客户端类型，取值： • alipay_openapi：支付宝（默认值） • wechat_openapi：微信 • dingtalk_openapi：钉钉

返回参数

字段名	类型	说明
success	Boolean	操作是否成功。
result	Object	用户信息： • **spaceId{String}**：服务空间ID。 • **userId{String}**：用户ID。 • **oAuthUserId{String}**：三方授权用户标识。只有指定了**authProvider**时才会返回该参数。

示例

- 匿名授权方式获取用户信息。

```
const res = await mpServerless.user.authorize({
  authType:'anonymous'
});
if (res.success) {
  console.log('授权成功');
}
mpServerless.user.getInfo().then(user => {
  this.setData(userInfo: user);
```

```
}).catch(console.error);
```

- 非匿名授权方式获取用户信息。
  - 支付宝授权：

```
const res = await app.mpServerless.user.authorize({
  authProvider: 'alipay_openapi'
});
if (res.success) {
  console.log('授权成功');
}
mpServerless.user.getInfo().then(user => {
  this.setData(userInfo: user);
}).catch(console.error);
```

  - 微信授权：

```
const res = await app.mpServerless.user.authorize({
  authProvider: 'wechat_openapi'
});
if (res.success) {
  console.log('授权成功');
}
mpServerless.user.getInfo().then(user => {
  this.setData(userInfo: user);
}).catch(console.error);
```

  - 钉钉授权：

```
const res = await app.mpServerless.user.authorize({
  authProvider: 'dingtalk_openapi'
});
if (res.success) {
  console.log('授权成功');
}
mpServerless.user.getInfo().then(user => {
  this.setData(userInfo: user);
}).catch(console.error);
```

说明：

应用获取授权后方能获取用户信息，获取授权接口信息参见authorize。

2.4.2 authorize

获取用户授权。

方法定义

该方法的定义如下：

```
authorize(options: AuthorizeOptions): Promise<{success: boolean;}>
```

请求参数

字段名	类型	必填	说明
options	Object	否	• **authProvider{String}**：客户端类型。取值： - alipay_openapi（默认值） - wechat_openapi • **authType{String}**：授权方式。取值： - 空字符串（默认值）。 - anonymous：匿名方式

说明：

- 如果同时指定了**authProvider**和**authType**，**authType**配置优先。
- 如果将**authType**设置为**anonymous**，则不需要设置**authProvider**。
- 如果没有设置**authType**，则必须设置**authProvider**。

返回参数

字段名	类型	说明
success	Boolean	操作是否成功。

示例

- 非匿名授权：

```
const res = await mpServerless.user.authorize({
```

```
  authProvider: 'alipay_openapi'
});
if (res.success) {
  console.log('授权成功');
}
```

- 匿名授权：

```
const res = await mpServerless.user.authorize({
  authType:'anonymous'
});
if (res.success) {
  console.log('授权成功');
}
```

说明：

匿名授权获取的用户身份是一个临时身份，每次匿名授权得到的用户标识是随机不固定的，若需要获取固定用户标识，则需要通过三方授权获取。

3 服务端SDK使用参考

3.1 上传文件到文件存储

本文介绍如何使用阿里云小程序云Serverless服务端SDK上传文件到文件存储。

向文件存储中上传文件分为三步。

1. 获取上传文件签名。
2. 使用第一步得到的签名，直传文件到OSS。
3. 标记文件已上传完成。

以下为NodeJS调用服务端SDK上传文件示例代码：

```
const Core = require('@alicloud/pop-core');
const runscript = require('runscript');

const client = new Core({
  accessKeyId: 'uFBT0jSx2S5AtLHy',
  accessKeySecret: 'xxxxxxxxxxxxxxxxxxxxxxx',
  endpoint: 'https://mpserverless.aliyuncs.com',
  apiVersion: '2019-06-15'
});

const ContentType = 'text/html';
const filePath = 'index.html';
const SpaceId = '81dd5357-7d48-4616-8cbf-b4308dd*****';

const params1 = {
  "RegionId": "cn-hangzhou",
  "Filename": filePath,
  "Size": 240, //size可以随意填写，无影响
  SpaceId,
  ContentType,
};

const params2 = {
  "RegionId": "cn-hangzhou",
  SpaceId,
};

const requestOption = {
  method: 'POST'
};

async function main() {
  const result1 = await client.request('DescribeFileUploadSignedUrl', params1, requestOpt
ion);
  const {Id, SignUrl} = result1;
  console.log(Id);
  console.log(SignUrl);
  console.log(result1);

  const script = `curl -X PUT -H 'Content-Type:${ContentType}' -T 'index.html' '${SignUrl}'`;
  console.log(script);
```

```
  const scriptResult = await runscript(script);
  console.log(scriptResult);

  params2.Id = Id;
  const result2 = await client.request('RegisterFile', params2, requestOption);
  console.log(result2);
};

main().then();
```

相关文档

DescribeFileUploadSignedUrl

RegisterFile