下载与安装

相关资源

- SDK 压缩包快速下载地址：见帮助中心

环境依赖

推荐使用 Chrome 或 Firefox 浏览器。

SDK 安装

将下载下来的 sdk，放置在 JavaScript 项目目录下即可。

连接 ZOS

在使用 SDK 之前，首先需要在 ZOS 上注册一个账号（Account），获取 AccessKey 和 SecretKey。其中 AccessKey 和 SecretKey 是您访问 ZOS 的密钥，ZOS 会通过它来验证您的资源请求。

使用方式：
1、在 html 网页中引入 SDK，并进行连接。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script src="aws-sdk-2.936.0.min.js" ></script>
    <script src="connect-ZOS.js"></script>
</body>

</html>
```

2、connect-ZOS.js 中连接后端 ZOS，示例如下：

```
let zos_client = new AWS.S3({
    accessKeyId: "your access key",
    secretAccessKey: "your secret key",
    endpoint: "ZOS endpoint",
    sslEnabled: true,
    s3ForcePathStyle: true
});
```

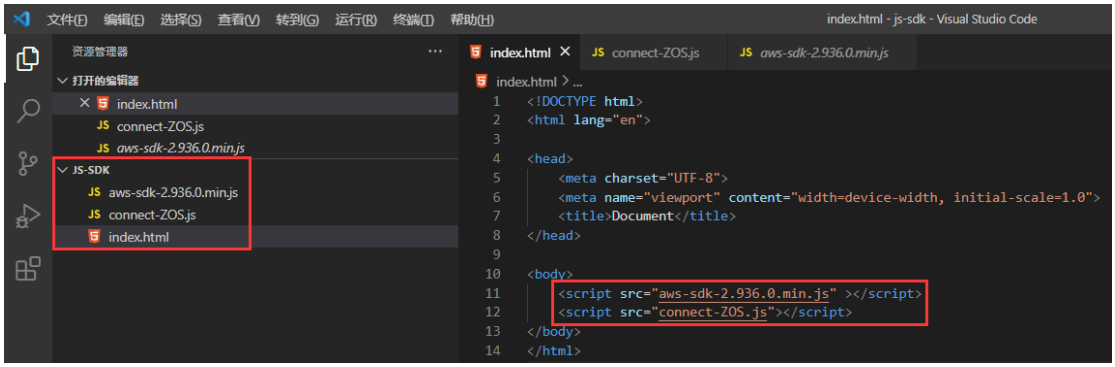
3、在 connect-ZOS.js 中执行具体的操作，示例如下：

```
// 准备操作所需的参数
let params = {
    Bucket: "bucket_name",
    Key: "obj_name"
};
// 执行具体的操作
zos_client.operation(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);           // successful response
});
```

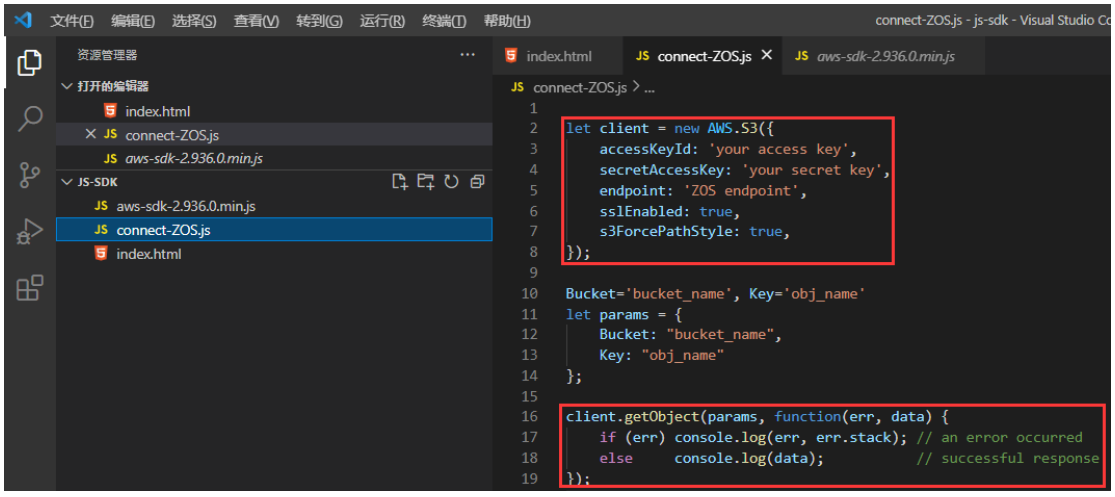
每个具体的 operation 的参数，通过 params 传入。

连接示例

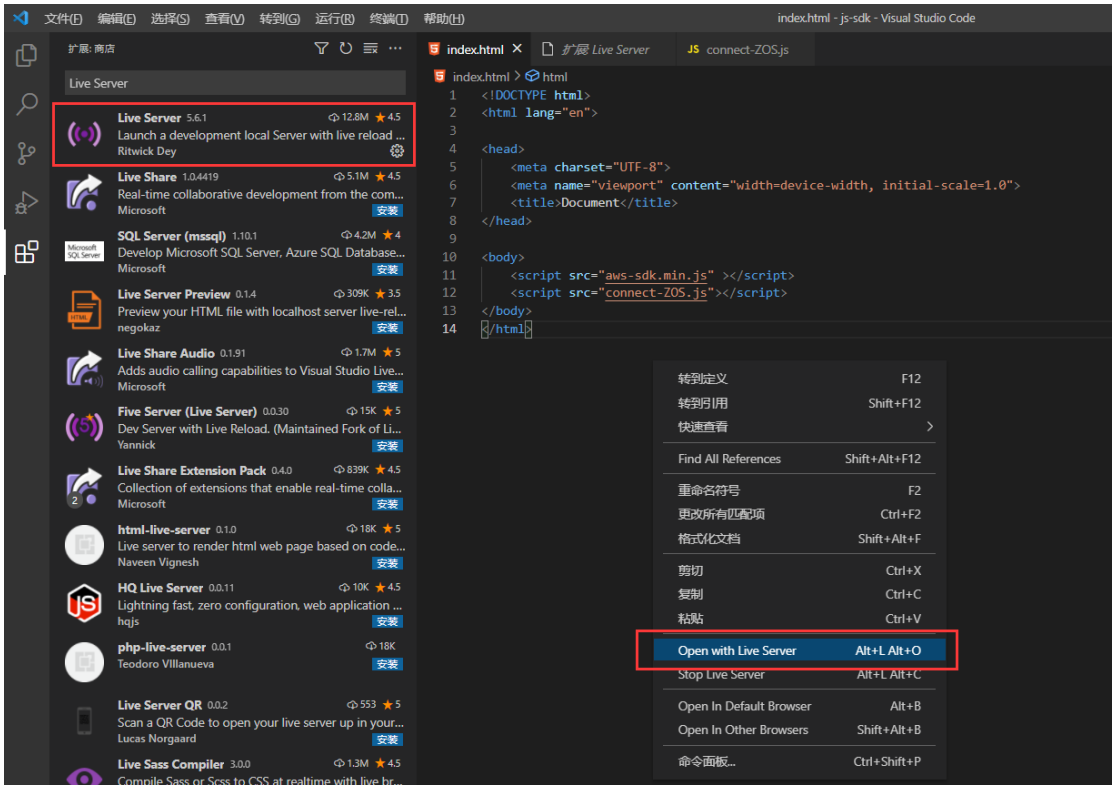
以 vscode 为例，打开本地文件夹，提前将 sdk 置于该文件夹下，然后新建 index.html 和 connect-ZOS.js 两个文件(文件名自行命名)。其中在 index.html 文件中包含 sdk 以及 connect-ZOS.js。



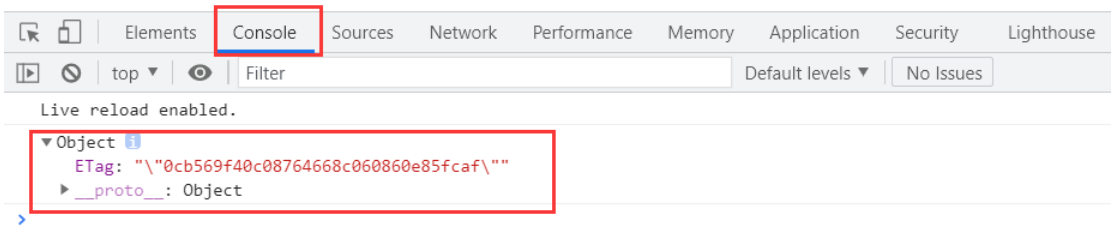
在 connect-ZOS.js 中实现具体的连接和操作。



建议可以在 vscode 中安装本地开发服务器，Live Server 是一款启动具有静态和动态页面的实时重新加载功能的开发本地服务器，在 vscode 扩展商店中搜索即可安装。安装之后即可在 html 文件下右键使用 Live Server 打开，方便开发调试：



右键点击 Live Server，会自动跳转到下面 chrome 界面，按 F12，选择 Console，即可查看 Console 打印结果，如下图所示：



CORS 配置（跨域设置）

由于浏览器的同源策略，在浏览器里调用 JS-SDK 时，和 bucket 相关的功能无法实现。即通过浏览器调用 JS-SDK 前，需要先通过其他语言 SDK 创建 bucket，并将该 bucket 的 CORS 设置好。然后通过 JS-SDK 可以进行 object 的操作。

全局错误码定义

请求可能会返回相关错误，具体错误码编号及信息请参考下表。同一个错误码可能对应不同的错误码描述，具体由接口来决定。

错误码	错误码描述
100	Continue
200	Success
201	Created
202	Accepted
204	NoContent
206	Partial content
304	NotModified
400	InvalidArgument
400	InvalidDigest

400	BadDigest
400	InvalidBucketName
400	InvalidObjectName
400	UnresolvableGrantByEmailAddress
400	InvalidPart
400	InvalidPartOrder
400	RequestTimeout
400	EntityTooLarge
403	AccessDenied
403	UserSuspended
403	RequestTimeTooSkewed
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity
500	InternalError

Object 操作

Get Object

功能说明

Get Object 请求可以将一个文件（Object）下载至本地。该操作需要对目标 Object 具有读权限或目标 Object 对所有人都开放了读权限（公有读）。

方法原型

```
getObject(params = {
                Bucket: String,
                Key: String,
                Range: String,
                VersionId:String
              }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
Range	下载对象的指定范围字节。形式："bytes=0-9"	String	否
VersionId	VersionId 用于引用对象的特定版本。	String	否

返回结果说明

```
{
  Body: Uint8Array(22) []
  ContentLength: 22
  ContentType: "application/octet-stream; charset=UTF-8"
  ETag: "\"0cb569f40c08764668c060860e85fcaf\""
  LastModified: Mon Jul 05 2021 10:21:13 GMT+0800 (中国标准时间)
  Metadata: {}
  StorageClass: "STANDARD"
  VersionId: "Ac9zLoox9n7sdioiV8UWAUuVfcQHiKA"
}
```

返回结果	描述	类型
Body	http 返回中的响应体，即对象的数据部分	Uint8Array
ContentLength	对象的长度	Int
ContentType	对象的类型	String
ETag	对象的 ETag 值	String
LastModified	对象的最后修改时间	Datetime
Metadata	对象在 ZOS 中的元数据	Map
StorageClass	对象在 ZOS 中的存储级别	String
VersionId	对象在 ZO 中的版本信息（前提是 bucket 开启了多版本）	String

返回结果	描述	类型
Location	Bucket 所在位置	*string

示例

```
let client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'ZOS endpoint',
```

```
    sslEnabled: true,
    s3ForcePathStyle: true,
});
let params = {
    Bucket: "bucket1",
    Key: "fire_test.py"
};

client.getObject(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);           // successful response
});
```

Head Object

功能说明

Head Object 请求可以获取对应 Object 的元数据，Head 的权限与 Get 的权限一致。

方法原型

```
headObject(params = {
                  Bucket: String,
                  Key: String,
                  IfMatch: String,
                  IfModifiedSince: new Date || 'Wed Dec 31 1969 16:00:00
 GMT-0800 (PST)' || 123456789,
                  IfNoneMatch: String,
                  IfUnmodifiedSince: new Date || 'Wed Dec 31 1969 16:00:00
 GMT-0800 (PST)' || 123456789,
                 VersionId: String
                  }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
IfMatch	当文件的 Etag 和 IfMatch 中的值一致时返回文件元数据，否则返回 412 错误	String	否
IfModifiedSince	只有指定时间之后有修改记录的文件才会返回元数据，否则返回 304 错误	Date String Number	否
IfNoneMatch	只有文件的 Etag 不满足指定字符串时才会返回元数据，否则返	String	否

	回 304 错误		
IfUnmodifiedSince	只有指定时间之后没有修改记录的文件才会返回元数据，否则返回 412 错误	Date String Number	否
VersionId	文件的某个特定版本的版本号，没有该版本则返回 404 错误	String	否

返回结果说明

```
{
  ContentLength: 748
  ContentType: "text/x-python"
  ETag: "\"2052ea99471bda95d297343d3a977124\""
  LastModified: Tue Jul 06 2021 16:42:15 GMT+0800 (中国标准时间) {}
  Metadata: {s3cmd-attrs:
"atime:1625538907/ctime:1625538904/gid:0/gname:root…7124/mode:33188/mt
 ime:1625538904/uid:0/uname:root"}
  StorageClass: "STANDARD"
  VersionId: "mQZfaCcF-y4rlzY--.242mUOpNUON.t"
}
```

返回结果	描述	类型
ContentLength	Object 的长度	Int
ContentType	Object 的类型	String
ETag	Object 的 ETag 值	String
LastModified	Object 的最后修改时间	Datetime
Metadata	Object 在 ZOS 中的元数据	Map
StorageClass	Object 在 ZOS 中的存储级别	String
VersionId	Object 在 ZO 中的版本信息（前提是 bucket 开启了多版本）	String

示例

```
let zos_client = new AWS.S3({

    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
    Bucket: "bucket-name",
    IfUnmodifiedSince:1625113736,
    Key: "key-name"
};
// 执行具体的操作
zos_client.headObject(params, function(err, data) {
```

```
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response
});
```

Put Object

功能说明

Put Object 请求可以将一个文件（Oject）上传至指定 Bucket。

方法原型

```
putObject(params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  ACL: private | public - read | public - read - write | authenticated -
 read ,
  Body: Buffer.from('...') || 'STRING_VALUE' || streamObject,
  ContentMD5: 'STRING_VALUE',
  GrantFullControl: 'STRING_VALUE',
  GrantRead: 'STRING_VALUE',
  GrantReadACP: 'STRING_VALUE',
  GrantWriteACP: 'STRING_VALUE',
  Metadata: {
      '<MetadataKey>': 'STRING_VALUE',
      /* '<MetadataKey>': ... */
  },
  ObjectLockLegalHoldStatus: ON | OFF,
  ObjectLockMode: GOVERNANCE | COMPLIANCE,
  ObjectLockRetainUntilDate: new Date || 'Wed Dec 31 1969 16:00:00
GMT-0800(PST)' || 123456789,
  Tagging: 'STRING_VALUE',
  Append:  true,
  AppendPosition: 'NUMBER_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	存入 bucket 的名称	String	是
ACL	Bucket 的 ACL 可选类型有 private、public-read、public-read-write 和 authenticated-read	String	是
Body	对象的数据	Buffer, Typed Array, Blob,	否

		String, Readable Stream	
ContentMD5	对象数据的 MD5	String	否
GrantFullControl	被授权用户可以对对象进行 read, write, read ACP, and write ACP 操作 以下 Grant*参数，格式都是”id=xxxx”或”emailAddress=xxxx”或者”uri=xxxx”以及他们的组合（用逗号连接）	String	否
GrantRead	被授权用户可以对对象进行读操作	String	否
GrantReadACP	被授权用户可以读取对象的 ACL	String	否
GrantWriteACP	被授权用户可以修改对象的 ACL	String	否
Metadata	key value 形式的 Object Metadata	map<String>	否
ObjectLockLegalHoldStatus	表示指定对象是否设置依法保留配置。取值为 ON 或 OFF	String	否
ObjectLockMode	表示指定对象的保留期限模式。取值为取值为 GOVERNANCE 或 COMPLIANCE	String	否
ObjectLockRetainUntilDate	对象锁定保留期限过期日期	Date	否
Tagging	标签集	String	否
Append	追加模式上传对象，仅支持设为 true	boolean	否
AppendPosition	追加模式下，指定追加的起始字节位置	Number	否

返回结果说明

未设置 Append

```
{ETag: "\"99c54b46d4f0cb88c04b30f6aea0db9a\""}
```

设置了 Append

```
{
 AppendPosition: "224119",
 ETag: "\"99c54b46d4f0cb88c04b30f6aea0db9a\""
}
```

返回结果	描述	类型
ETag	上传对象的 Etag	String
AppendPosition	下一次请求时的追加位置	Number
VersionId	上传对象的 VersionId	String

示例

```
let client = new AWS.S3({
```

```
  accessKeyId: 'your access key',
  secretAccessKey: 'your secret key',
  endpoint: 'http://192.168.32.4:80',
  sslEnabled: false,
  s3ForcePathStyle: true,
  signatureVersion: 'v2',
});

document.getElementById('fileInput').addEventListener('change',
function selectedFileChanged() {
  if (this.files.length === 0) {
      console.log('No file selected.');
      return;
  }

  const reader = new FileReader();
  reader.onload = function fileReadCompleted() {
      let params = {
          Bucket: "js-sdk-bucket",
          Key: "txt-obj-append",
          Body: reader.result,
          Append: true,
          AppendPosition: 0
      }
      client.putObject(params, function (err, data) {
          if (err) console.log(err, err.stack); // an error occurred
          else console.log(data);       // successful response
      });
  };
  reader.readAsBinaryString(this.files[0])
});
```

Delete Object

功能说明

Delete Object 请求可以将一个文件（Object）删除。

方法原型

```
deleteObject(params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  BypassGovernanceRetention: true || false,
  VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须

Bucket	Bucket 的名称	String	是
Key	要删除的对象的名称	String	是
BypassGovernanceRetention	执行操作时是否绕过 Governance 模式锁的限制	Boolean	否
VersionId	要删除的对象的 VersionId	String	否

返回结果说明

删除普通的 object，返回结果没有需要处理的字段

删除开启 version 的 bucket 中的 object

```
{
  DeleteMarker: true,
  VersionId: "mp9GzAJUOGvT3MFQ-UJXkiUu2.oSxda"
}
```

返回结果	描述	类型
DeleteMarker	指明是否创建了 DeleteMarker	Boolean
VersionId	创建的 DeleteMarker 的 VersionId	String

示例

```
  let client = new AWS.S3({


  accessKeyId: 'your access key',
  secretAccessKey: 'your secret key',
  endpoint: 'http://192.168.32.4:80',
  sslEnabled: false,
  s3ForcePathStyle: true,
  signatureVersion: 'v2',
});

let params = {
  Bucket: "js-sdk-bucket",
  Key: "demo-obj",
}
client.deleteObject(params, function (err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);       // successful response

});
```

Delete Multiple Object

功能说明

Delete Multiple Object 请求实现批量删除文件，最大支持单次删除 1000 个文件。对于返回结果，COS 提供 Verbose 和 Quiet 两种结果模式。Verbose 模式将返回每个 Object 的删除结果；Quiet 模式只返回报错的 Object 信息。

方法原型

```
deleteObjects(params = {
  Bucket: 'STRING_VALUE', /* required */
  Delete: { /* required */
    Objects: [ /* required */
      {
        Key: 'STRING_VALUE', /* required */
        VersionId: 'STRING_VALUE'
      },
      /* more items */
    ],
    Quiet: true || false
  },
  BypassGovernanceRetention: true || false,
} , callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Delete	要删除对象的配置	Map	是
Objects	Delete 集合中描述对象信息的集合	Array<Map>	是
Key	要删除的对象的名称	String	是
VersionId	要删除的对象的 VersionId	String	否
Quiet	设置是否开启 quiet 模式	Boolean	否
BypassGovernanceRetention	执行操作时是否绕过 Governance 模式锁的限制	Boolean	否

返回结果说明

```
{
  "Deleted": [
      {
      "Key": "demo-obj1"
      },
      {
      "Key": "demo-obj2"
      },
      {
      "Key": "demo-obj3"
```

```
      }
  ],
  "Errors": []
}
```

返回结果	描述	类型
Deleted	成功删除的对象集合	Arrary<Map>
Errors	删除失败的对象集合	Array<Map>
DeleteMarkerVersionId	若创建了 DeleteMarker，则包含 DeleteMarker 的 VersionId	String
Key	删除成功的对象的名称	String
VersionId	删除成功的对象的 VersionId	String
Code	删除失败的错误码	String
Key	删除失败的对象名称	String
Message	删除失败的描述信息	String
VersionId	删除失败的对象的 VersionId	String

示例

```
  let client = new AWS.S3({


  accessKeyId: 'your access key',
  secretAccessKey: 'your secret key',
  endpoint: 'http://192.168.32.4:80',
  sslEnabled: false,
  s3ForcePathStyle: true,
  signatureVersion: 'v2',
});

let deleteMap = {
  Objects: [
      {Key: "demo-obj1"},
      {Key: "demo-obj2"},
      {Key: "demo-obj3"},
  ],
  Quiet: false,
}

let params = {
  Bucket: "js-sdk-bucket",
  Delete: deleteMap,
}
client.deleteObjects(params, function (err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);       // successful response
```

```
});
```

Put Object ACL

功能说明

设置 Object 的 ACL，控制对 Object 的访问权限。该操作需要用户具有 WRITE_ACP 权限。

有三种方式设置 ACL，三种方式不可同时使用，每次只能给一种参数赋值。其中，通过 ACL 参数方式进行操作，是设置预定义的固定的 ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另外两种方式实现，该参数使用请求头进行传递；AccessControlPolicy 参数方式和 Grant*参数方式则可以针对特定用户进行授权，AccessControlPolicy 方式通过请求体传递，而 Grant*方式通过请求头传递。三种方式都会覆盖原有 ACL 属性，包括对象所有者自身的权限，如需保留原有 ACL 属性，应将需要保留的原 ACL 添加到本次操作的授权中（ACL 参数方式会默认将对象所有者权限设为 FULL_CONTROL，而另外两种方式则不会保留任何原 ACL 属性）。

方法原型

```
putObjectAcl(params = {
  Bucket: String, /* required */
  Key: String, /* required */
  VersionId: String,
  ACL: String,
  AccessControlPolicy: {
    Grants: [
      {
        Grantee: {
          Type: String, /* required */
          DisplayName: String,
          EmailAddress: String,
          ID: String,
          URI: String
        },
        Permission: String
      },
      /* more items */
    ],
    Owner: {
      DisplayName: String,
      ID: String /* required */
    }
  },
  GrantFullControl: String,
  GrantRead: String,
  GrantReadACP: String,
  GrantWrite: String,
```

```
  GrantWriteACP: String
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
VersionId	VersionId 用于引用对象的特定版本。	String	否
ACL	预定义 ACL，取值范围 "private" "public-read" "public-read-write" "authenticated-read"	String	ACL 参数方式则必须，其他两种方式，则不能使用
AccessControlPolicy	包含授权列表和对象所有者	map	该方式下必须，其他两种方式下则不能使用
Grants	授权列表	Array<map>	该方式下必须
Grantee	被授权用户	map	该方式下必须
Type	被授权用户类型，取值范围 "CanonicalUser" "AmazonCustomerByEmail" "Group"	String	该方式下必须
ID	被授权用户 ID	String	Type 为 'CanonicalUser'，则该字段必须
DisplayName	被授权用户 display name	String	否
EmailAddress	被授权用户邮箱	String	如果 Type 为 'AmazonCustomerByEmail'，则该字段必须
URI	被授权组 URI， 取值范围为 所有用户：http://acs.amazonaws.com/groups/global/AllUsers 所有认证用户： http://acs.amazonaws.com/groups/global/AuthenticatedUsers	String	如果 Type 为'Group'，则该字段必须

Permission	向被授权用户授予的权限， 取值范围 'FULL_CONTROL' 'WRITE' 'WRITE_ACP' 'READ' 'READ_ACP'	String	该方式下必须
Owner	对象所有者	map	该方式下必须
ID	对象所有者 ID	String	该方式下必须
DisplayName	对象所有者 display name	String	否
GrantFullControl	被授权用户可以对对象进行 read, write, read ACP, and write ACP 操作 以下 Grant*参数，格式都是"id=xxxx"或"emailAddress=xxxx"或者"uri=xxxx"以及他们的组合（用逗号连接）	String	否
GrantRead	被授权用户可以对对象进行读操作	String	否
GrantReadACP	被授权用户可以对对象进行写操作，删除或覆盖写该对象	String	否
GrantWrite	被授权用户可以读取对象的 ACL	String	否
GrantWriteACP	被授权用户可以修改对象的 ACL	String	否

返回结果说明

无

示例

```
let zos_client = new AWS.S3({

    accessKeyId: "test-1",
    secretAccessKey: "test-1",
    endpoint: "http://192.168.220.100:7480",
    sslEnabled: false,
    s3ForcePathStyle: true
});

var params = {
    Bucket: "bucket-1",
    Key: "test-1",
    // ACL: "public-read-write",
    AccessControlPolicy: {
    Grants: [
      {
        Grantee: {
          Type: "CanonicalUser", /* required */
          ID: "test-2",
        },
        Permission: "FULL_CONTROL"
```

```
    },
    {
      Grantee: {
        Type: "AmazonCustomerByEmail", /* required */
        EmailAddress: "abc@abc.com",
      },
      Permission: "WRITE"
    },
    {
      Grantee: {
        Type: "Group", /* required */
        URI: "http://acs.amazonaws.com/groups/global/AllUsers"
      },
      Permission: "READ"
    },
    /* more items */
  ],
  Owner: {
    ID: "test-1"
  }
  },
  // GrantFullControl:
"id=test-1,emailaddress=abc@abc.com,emailaddress=def@def.com",
  // GrantRead:
"uri=http://acs.amazonaws.com/groups/global/AllUsers",
};
zos_client.putObjectAcl(params, function (err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);           // successful response


});
```

Get Object ACL

功能说明

获取指定 Object 的 ACL。该操作需要 READ_ACP 权限。该功能返回的结果与 Put Object ACL 参数一致，但是需要注意的是，如果以邮箱类型授权，返回结果中将会以对应被授权用户 ID 形式出现，即 Type 不会是 AmazonCustomerByEmail，而是 CanonicalUser。

方法原型

```
getObjectAcl(params = {
  Bucket: String, /* required */
  Key: String, /* required */
  VersionId: String
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
VersionId	VersionId 用于引用对象的特定版本。	String	否

返回结果说明

```
Grants: [
    {
        Grantee: {
            DisplayName: "test-2",
            ID: "test-2",
            EmailAddress:"",
            Type: "CanonicalUser"
        },
        Permission: "WRITE"
    },
    {
        Grantee: {
            URI: "http://acs.amazonaws.com/groups/global/AllUsers"
            Type: "CanonicalUser"
        },
        Permission: "READ"
    }
],
Owner: {
    DisplayName: "test-1",
    ID: "test-1"
}
```

返回结果	描述	类型
Grants	授权列表	map
Grantee	被授权用户信息	map
Type	被授权用户类型	String
ID	被授权用户 ID	String
DisplayName	被授权用户 display name	String
EmailAddress	被授权用户邮箱	String
URI	被授权组 uri	String
Permission	被授权权限	String
Owner	对象所有者	map
ID(Owner)	对象所有者 ID	String
DisplayName (Owner)	对象所有者 display name	String

示例

```
let zos_client = new AWS.S3({

    accessKeyId: "test-1",
    secretAccessKey: "test-1",
    endpoint: "http://192.168.220.100:7480",
    sslEnabled: false,
    s3ForcePathStyle: true
});

var params = {
    Bucket: "bucket-1",
    Key: "test-1"
};
zos_client.getObjectAcl(params, function (err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else console.log(data);           // successful response

});
```

Put Object Tagging

功能说明

将提供的标签集设置为存储桶中已存在的对象。标签是一个键值对。请注意，Amazon S3 将标签的最大数量限制为每个对象 10 个标签。要使用此操作，您必须具有执行 s3:PutObjectTagging 操作的权限。 默认情况下，Bucket 拥有者拥有此权限，并且可以将此权限授予其他人。要放置任何其他版本的标签，请使用 versionId 查询参数。您还需要 s3:PutObjectVersionTagging 操作的权限。

方法原型

```
putObjectTagging(params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  Tagging: { /* required */
    TagSet: [ /* required */
      {
        Key: 'STRING_VALUE', /* required */
        Value: 'STRING_VALUE' /* required */
      },
      /* more items */
    ]
  },
  VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	对象名称	String	是
Tagging	标签	structure	是
TagSet	标签集合	list	是
VersionId	对象的版本 ID	String	否

返回结果说明

无

示例

```
let zos_client = new AWS.S3({

    accessKeyId: "test",
    secretAccessKey: "test",
    endpoint: "http://192.168.198.110:7480",
    sslEnabled: false,
    s3ForcePathStyle: true
});

let params = {
    Bucket: "java-zos", /* required */
    Key: "sns.log", /* required */
    Tagging: { /* required */
        TagSet: [ /* required */
            {
                Key: 'key1', /* required */
                Value: 'val1' /* required */
            },
            /* more items */
        ]
    },
    VersionId: "3Fi5CejIPj4p1YXAGK8ChQa91zARdWZ"
};

zos_client.putObjectTagging(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response
});
```

Get Object Tagging

功能说明

返回对象的标签集。要使用此操作，您必须具有执行 s3:GetObjectTagging 操作的权限。默认情况下，操作返回有关对象当前版本的信息。对于多版本的存储桶，您的存储桶中可以有一个对象的多个版本。此时，要检索任何其他版本的标签，请使用 versionId 查询参数。同时，您还需要 s3:GetObjectVersionTagging 操作的权限。

默认情况下，存储桶拥有者具有此权限，并且可以将此权限授予其他人。

方法原型

```
getObjectTagging(params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	对象名称	String	是
VersionId	对象的版本 ID	String	否

返回结果说明

```
TagSet: [
  {
     Key: 'STRING_VALUE',
     Value: 'STRING_VALUE'
  }
]
```

返回结果	描述	类型
TagSet	标签集合	list
Key	标签键	String
Value	标签值	String

示例

```
let zos_client = new AWS.S3({

    accessKeyId: "test",
    secretAccessKey: "test",
```

```
    endpoint: "http://192.168.198.110:7480",
    sslEnabled: false,
    s3ForcePathStyle: true
});

let params = {
    Bucket: "java-zos", /* required */
    Key: "sns.log", /* required */
    VersionId: "3Fi5CejIPj4p1YXAGK8ChQa91zARdWZ"
};

zos_client.getObjectTagging(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response
});
```

Delete Object Tagging

功能说明

从指定的对象中删除整个标记集。要使用此操作，您必须具有执行 s3:DeleteObjectTagging 操作的权限。要删除特定对象版本的标签，请在请求中添加 versionId 查询参数。您将需要 s3:DeleteObjectVersionTagging 操作的权限。

方法原型

```
deleteObjectTagging(params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	对象名称	String	是
VersionId	对象的版本 ID	String	否

返回结果说明

无

示例

```
let zos_client = new AWS.S3({

    accessKeyId: "test",
    secretAccessKey: "test",
    endpoint: "http://192.168.198.110:7480",
    sslEnabled: false,
    s3ForcePathStyle: true
});

let params = {
    Bucket: "java-zos", /* required */
    Key: "sns.log", /* required */
    VersionId: "3Fi5CejIPj4p1YXAGK8ChQa91zARdWZ"
};

zos_client.deleteObjectTagging(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);        // successful response
});
```

Put Object Legal Hold

功能说明

put object legal hold 请求在指定对象上使用依法保留配置

方法原型

```


putObjectLegalHold(params = {
    Bucket: 'STRING_VALUE', /* required */
    Key: 'STRING_VALUE', /* required */
    LegalHold: {
        Status: ON | OFF
    },
    VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request


```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	进行依法保留设置的对象名称	String	是
LegalHold	指定对象依法保留配置	map	否
Status	表示指定对象是否设置依法保留配置	String	否

VersionId	配置依法保留的对象的版本 ID	String	否

返回结果说明

无

示例

```
let client = new AWS.S3({

    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'http://192.168.56.103:7480',
    signatureVersion: 'v2',
    sslEnabled: false,
    s3ForcePathStyle: true,
});

let params = {
    Bucket: 'bucket1', /* required */
    Key: 'time', /* required */
    LegalHold: {
        Status: "ON"
    },
    VersionId: 'l7je1kKY7u7AKmdG1P6i05XApIP2TbU'
};

client.putObjectLegalHold(params, function (err, data) {
       if (err) console.log(err, err.stack);
       else console.log(data.Contents);
});
```

Get Object Legal Hold

功能说明

get object legal hold 请求获取指定对象的当前依法保留状态

方法原型

```


getObjectLegalHold(params = {
    Bucket: 'STRING_VALUE', /* required */
    Key: 'STRING_VALUE', /* required */
    VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	进行依法保留设置的对象名称	String	是
VersionId	配置依法保留的对象的版本 ID	String	否

返回结果说明

```
'LegalHold': {
    'Status': 'OFF'
}
```

返回结果	描述	类型
LegalHold	指定对象依法保留配置	map
Status	表示指定对象是否设置依法保留配置	String

示例

```
let client = new AWS.S3({

    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'http://192.168.56.103:7480',
    signatureVersion: 'v2',
    sslEnabled: false,
    s3ForcePathStyle: true,
});

let params = {
    Bucket: 'bucket1', /* required */
    Key: 'time', /* required */
    VersionId: 'l7je1kKY7u7AKmdG1P6i05XApIP2TbU'
};

client.getObjectLegalHold(params, function (err, data) {
       if (err) console.log(err, err.stack);
       else console.log(data.Contents);
});
```

Put Object Retention

功能说明

put object retention 请求设置对象保留期限配置。

方法原型

```
putObjectRetention(params = {
    Bucket: 'STRING_VALUE', /* required */
    Key: 'STRING_VALUE', /* required */
    BypassGovernanceRetention: true || false,
    Retention: {
        Mode: GOVERNANCE | COMPLIANCE,
        RetainUntilDate: new Date || 'Wed Dec 31 1969 16:00:00 GMT-0800
(PST)' || 123456789
    },
    VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	进行依法保留设置的对象名称	String	是
Retention	对象保留期限配置元素	map	否
Mode	表示指定对象的保留期限模式	String	否
RetainUntilDate	对象锁定保留期限过期日期	datetime	否
VersionId	配置依法保留的对象的版本 ID	String	否
BypassGovernance Retention	表示是否这个操作应该绕过监管模式	Boolean	否

返回结果说明

无

示例

```
let client = new AWS.S3({

    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'http://192.168.56.103:7480',
    signatureVersion: 'v2',
    sslEnabled: false,
    s3ForcePathStyle: true,
});

let params = {
    Bucket: 'bucket1', /* required */
    Key: 'time', /* required */
    BypassGovernanceRetention: true,
    Retention: {
      Mode: 'GOVERNANCE',
      RetainUntilDate: 1622269444
```

```
    },
    VersionId: 'l7je1kKY7u7AKmdG1P6i05XApIP2TbU'
};

client.putObjectRetention(params, function (err, data) {
       if (err) console.log(err, err.stack);
       else console.log(data.Contents);
});
```

Get Object Retention

功能说明

get object retention 获取对象的保留期限设置

方法原型

```
getObjectRetention(params = {
    Bucket: 'STRING_VALUE', /* required */
    Key: 'STRING_VALUE', /* required */
    VersionId: 'STRING_VALUE'
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	进行依法保留设置的对象名称	String	是
VersionId	配置依法保留的对象的版本 ID	String	否

返回结果说明

```
Retention': {
    'Mode': 'GOVERNANCE',
    'RetainUntilDate': 2021-07-06T04:41:32.000000000Z

}
```

返回结果	描述	类型
Retention	对象保留期限配置元素	map
Mode	表示指定对象的保留期限模式	String
RetainUntilDate	对象锁定保留期限过期日期	datetime

示例

```
let client = new AWS.S3({

    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'http://192.168.56.103:7480',
    signatureVersion: 'v2',
    sslEnabled: false,
    s3ForcePathStyle: true,
});

let params = {
    Bucket: 'bucket1', /* required */
    Key: 'time', /* required */
    VersionId: 'l7je1kKY7u7AKmdG1P6i05XApIP2TbU'
};

client.getObjectRetention(params, function (err, data) {
       if (err) console.log(err, err.stack);
       else console.log(data.Contents);
});
```

Create Presigned Post

功能说明

Create Presigned Post 请求生成一个临时的预签名的 Url，没有权限访问集群的用户可以通过该 Url 上传文件到集群(通过 http 的 post 方式)。

方法原型

```
createPresignedPost(params = {
                           Bucket: String
                           Fields: {
                               key: String
                           },
                           Expires:Number
                           }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	预签名要上传的文件所在的 Bucket 名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是

Expires	预签名的有效时间	Number	否

返回结果说明

```
{

   fields: {}
   url: "http://192.168.218.130:7480/bucket-1"
}
```

返回结果	描述	类型
fields	服务端返回的数据，包含签名、policy 等数据，需要在 post 文件上传的时候发送到服务端	Json
url	集群生成的临时的 Url	String

示例

```
  let zos_client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
  Bucket: "bucket-1",
  Fields: {
      key: "key-name"
  },
  Expires:7200
};
// 执行具体的操作
zos_client.createPresignedPost(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response
});
```

Copy Object

功能说明

Put Object Copy 请求实现将一个文件从源路径复制到目标路径。

方法原型

```

```

```
copyObject(params = {
  Bucket: 'STRING_VALUE', /* required */
  CopySource: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  ACL: private | public-read | public-read-write | authenticated-read,
  CopySourceIfMatch: 'STRING_VALUE',
  CopySourceIfModifiedSince: new Date || 'Wed Dec 31 1969 16:00:00 GMT-0800
(PST)' || 123456789,
  CopySourceIfNoneMatch: 'STRING_VALUE',
  CopySourceIfUnmodifiedSince: new Date || 'Wed Dec 31 1969 16:00:00
GMT-0800 (PST)' || 123456789,
  GrantFullControl: 'STRING_VALUE',
  GrantRead: 'STRING_VALUE',
  GrantReadACP: 'STRING_VALUE',
  GrantWriteACP: 'STRING_VALUE',
  Metadata: {
    '<MetadataKey>': 'STRING_VALUE',
    /* '<MetadataKey>': ... */
  },
  MetadataDirective: COPY | REPLACE,
  ObjectLockLegalHoldStatus: ON | OFF,
  ObjectLockMode: GOVERNANCE | COMPLIANCE,
  ObjectLockRetainUntilDate: new Date || 'Wed Dec 31 1969 16:00:00 GMT-0800
 (PST)' || 123456789,
  Tagging: 'STRING_VALUE',
  TaggingDirective: COPY | REPLACE,
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	复制目的 Bucket	String	是
CopySource	复制源，格式为 "Bucket/Key"	String	是
Key	复制目的 Object 的名字	String	是
ACL	复制目的对象的 ACL	String	否
CopySourceIfMatch	仅当指定的 Etag 和 Copy Source 指定的 object 的 Etag 匹配时才复制	String	否
CopySourceIfModifiedSince	仅当 CopySource 在指定时间后更新过才复制	Date	否
CopySourceIfNoneMatch	仅当指定的 Etag 和 Copy Source 指定的 object 的 Etag 不匹配时才复制	String	否
CopySourceIfUnmodifiedSince	仅当 CopySource 在指定时间后未更新过才复制	Date	否
GrantFullControl	被授权用户可以对对象进行 read, write, read ACP, and write ACP 操作 以下 Grant*参数，格式都是"id=xxxx"或"emailAddress=xxxx"或者"uri=xxxx"以及他们的组合（用逗号连接）	String	否
GrantRead	被授权用户可以对对象进行读操作	String	否
GrantReadACP	被授权用户可以读取对象的 ACL	String	否
GrantWriteACP	被授权用户可以修改对象的 ACL	String	否

Metadata	复制目的对象的元数据	Map	否
MetadataDirective	Metadata 复制选项，'COPY' 复制源的 Metadata 或 'REPLACE'使用新指定的 Metadata 覆盖	String	否
ObjectLockLegalHoldStatus	设置复制的 Object 的 LegalHold 开关	String	否
ObjectLockMode	设置复制的 Object 的 Retention 模式	String	否
ObjectLockRetainUntilDate	设置复制的 Object 的 Retention 保留过期时间	Date	否
Tagging	设置复制对象的 Tag	String	否
TaggingDirective	Tagging 复制选项 'COPY' 复制源的 Tag 或 'REPLACE'使用新指定的 Tag 覆盖	String	否

返回结果说明

```
{
    "CopyObjectResult": {
        "ETag": "8be53e4ae8d7662e19b972d411d91039-6",
        "LastModified": "2021-07-06T12:55:58.996Z"
    }
}
```

返回结果	描述	类型
CopyObjectResult	复制结果	Map
CopySourceVersionId	复制源对象的 VersionId	String
VersionId	复制目的对象的 VersionId	String
ETag	复制目的对象的 Etag	String
LastModified	复制目的对象的修改时间	Date

示例

```
  let client = new AWS.S3({
  accessKeyId: 'your access key',
  secretAccessKey: 'your secret key',
  endpoint: 'http://192.168.32.4:80',
  sslEnabled: false,
  s3ForcePathStyle: true,
  signatureVersion: 'v2',
});

let params = {
  Bucket: "js-sdk-bucket",
  CopySource: "java-sdk-bucket/zos-sdk-cpp.tar.gz",
  Key:    "js-copy-obj"
}
client.copyObject(params, function (err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);       // successful response
```

```
});
```

UploadPartCopy

功能说明

Copy 请求实现将一个文件从源路径复制到目标路径，支持拷贝大于 5GB 的文件。

方法原型

```
uploadPartCopy(params = {
  Bucket: 'STRING_VALUE', /* required */
  CopySource: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  PartNumber: 'NUMBER_VALUE', /* required */
  UploadId: 'STRING_VALUE', /* required */
  CopySourceIfMatch: 'STRING_VALUE',
  CopySourceIfModifiedSince: new Date || 'Wed Dec 31 1969 16:00:00 GMT-0800
 (PST)' || 123456789,
  CopySourceIfNoneMatch: 'STRING_VALUE',
  CopySourceIfUnmodifiedSince: new Date || 'Wed Dec 31 1969 16:00:00
GMT-0800 (PST)' || 123456789,
  CopySourceRange: 'STRING_VALUE',
}, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	复制目的 Bucket	String	是
Key	复制源，格式为 "Bucket/Key"	String	是
Key	复制目的 Object 的名字	String	是
PartNumber	设置本次复制的分段编号	String	是
UploadId	设置分段上传 ID，由 3.1 返回	String	是
CopySourceIfMatch	仅当指定的 Etag 和 Copy Source 指定的 object 的 Etag 匹配时才复制	String	否
CopySourceIfModifiedSince	仅当 CopySource 在指定时间后更新过才复制	Date	否
CopySourceIfNoneMatch	仅当指定的 Etag 和 Copy Source 指定的 object 的 Etag 不匹配时才复制	String	否
CopySourceIfUnmodifiedSince	仅当 CopySource 在指定时间后未更新过才复制	Date	否
CopySourceRange	复制对象的字节范围，格式为 "bytes={start}-{end}"	String	否

返回结果说明

```
{
    CopyPartResult: {
        ETag: "\"65d16d19e65a7508a51f043180edcc36\"",
        LastModified: "2021-07-06T12:55:58.996Z"
    }
}
```

返回结果	描述	类型
CopyPartResult	复制结果	Map
CopySourceVersionId	复制源对象的 VersionId	String
ETag	复制目的对象的 Etag	String
LastModified	复制目的对象的修改时间	String

示例

```
  var params = {

  Bucket: "js-sdk-bucket",
  CopySource: "java-sdk-bucket/demo-obj",
  CopySourceRange: "bytes=1-100000",
  Key: "partCopyObj",
  PartNumber: 2,
  UploadId: "2~YT2JbF7LxbIWSb1vZUDQoyOr_ueZxzG"
};
s3.uploadPartCopy(params, function (err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);           // successful response

}
```

分块上传操作

Create Multipart Upload

功能说明

Create Multipart Upload 请求实现初始化分片上传，成功执行此请求以后会返回 Upload ID 用于后续的 Upload Part 请求。

方法原型

```
createMultipartUpload(params = {
                            Bucket: String,
                            Key: String
                            }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是

返回结果说明

```
{
    Bucket: "bucket-name"
    Key: "Key-name"
    UploadId: "2~IBxsCuaK6i1nAIse4mu8y6fu5J_Mm3i"
}
```

返回结果	描述	类型
Bucket	Bucketd 的名称	String
Key	对象保持在 Bucket 内的名称	String
UploadId	本次分段上传 UploadId	String

示例

```
let zos_client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
    Bucket: "bucket-name",
    Key: "Key-name"
};
// 执行具体的操作
zos_client.createMultipartUpload(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response

});
```

Upload Part

功能说明

Upload Part 请求实现在初始化以后的分块上传，支持的块的数量为 1 到 10000，块的大小为 1 MB 到 5 GB。在每次请求 Upload Part 时，需要携带 partNumber 和 uploadID，partNumber 为块的编号，支持乱序上传。

方法原型

```
uploadPart(params = {
                  Body: <Binary String>,
                  Bucket: String,
                  Key: String,
                  PartNumber: Int,
                  UploadId: String
                  }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
Body	文件数据	Buffer, Typed Array, Blob, String, Readable Stream	是
PartNumber	分段号	Number	是
UploadId	分段上传的 UploadId，在 2.1 中返回的	String	是

返回结果说明

```
{
    ETag: "\"64150dbad8e4524b90b7c3c609ed2285\""
}
```

返回结果	描述	类型
Etag	当前分段的 Etag	String

示例

```
let zos_client = new AWS.S3({
```

```
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
  Bucket: "bucket-name",
  Key: "Key-name",
  Body: "body-content",
  PartNumber:1,
  UploadId:"2~IBxsCuaK6i1nAIse4mu8y6fu5J_Mm3i"
};
// 执行具体的操作
zos_client.uploadPart(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response

});
```

Complete Multipart Upload

功能说明

Complete Multipart Upload 用来实现完成整个分块上传。当您已经使用 Upload Parts 上传所有块以后,你可以用该 API 完成上传。在使用该 API 时,您必须在 Body 中给出每一个块的 PartNumber 和 ETag，用来校验块的准确性。

方法原型

```
uploadPart(params = {
                    Bucket: String,
                    Key: String,
                    MultipartUpload: {
                        Parts: [
                            {
                                ETag: String,
                                PartNumber: Number
                            },
                            {
                                ETag: String,
                                PartNumber: Number
                             }
                                ]
                  },
                  UploadId: String
                  }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
MultipartUpload	具体的分段信息	Json	是
Parts	具体的分段信息	Json	是
Etag	该分段对应的 Etag	String	是
PartNumber	分段号	String	是

返回结果说明

```
{
   Bucket: "bucket-name"
   ETag: "ffde2a340f15d98af1a04a6ed3cbaf96-1"
   Key: "Key-name"
   Location: "http://192.168.218.130:7480/bucket-1/Key-name"
   VersionId: "au-4.jzHvdEw71mkgCv28aoSTm0fzv6"
}
```

返回结果	描述	类型
Bucket	Bucketd 的名称	String
Etag	文件整体的 Etag	String
Key	对象保持在 Bucket 内的名称	String
Location	文件保存位置	String
VersionId	文件版本号	String

示例

```
  let zos_client = new AWS.S3({

    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
  Bucket: "bucket-1",
  Key: "Key-name",
  MultipartUpload: {
      Parts: [
         {
```

```
       ETag: "\"64150dbad8e4524b90b7c3c609ed2285\"",
       PartNumber: 1
      }
     ]
    },
  UploadId:"2~h7HwIuNvtJhT_HjUfgj_AFGNCVsV1C3"
};
// 执行具体的操作
zos_client.completeMultipartUpload(params, function(err, data) {
   if (err) console.log(err, err.stack); // an error occurred
   else     console.log(data);       // successful response

});
```

List Parts

功能说明

List Parts 用来查询特定分块上传中的已上传的块。

方法原型

```
listParts(params = {
                 Bucket: String,
                 Key: String,
                 UploadId:String,
                 MaxParts:Number,
                 PartNumberMarker:Number
                 }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
UploadId	分段上传的 UploadId，在 2.1 中返回的	String	是
MaxParts	最多返回的分段信息	Number	否
PartNumberMarker	返回的分段信息的分段号起始编号，只有分段号大于该值得分段信息才会返回	Number	否

返回结果说明

```
{
```

```
    Bucket: "bucket-1"
    IsTruncated: false
    Key: "30M"
    MaxParts: 1000
    NextPartNumberMarker: 2
    Owner: {DisplayName: "", ID: ""}
    PartNumberMarker: 0
    Parts: (2) [{…}, {…}]
    StorageClass: "STANDARD"
    UploadId: "2~IBxsCuaK6i1nAIse4mu8y6fu5J_Mm3i"
}
```

返回结果	描述	类型
Bucket	Bucket 名称	String
IsTruncated	是否是截断返回	Boolean
MaxParts	本次最多可返回的分段信息	Number
NextPartNumberMarker	下次 list 的分段起始编号，只要针对分段过多的情况，一次不能全部返回	Number
Owner	分段信息所属用户	Json
PartNumberMarker	当前 List 的分段起始编号	Number
Parts	具体的分段信息	Json
StorageClass	分段数据的存储级别	String
UploadId	分段上传的 UploadId	String

示例

```
let zos_client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
  Bucket: "bucket-1",
  Key: "30M",
  UploadId:"2~IBxsCuaK6i1nAIse4mu8y6fu5J_Mm3i"
};
// 执行具体的操作
zos_client.listParts(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response

});
```

Abort Multipart Upload

功能说明

Abort Multipart Upload 用来实现舍弃一个分块上传并删除已上传的块。当您调用 Abort Multipart Upload 时，如果有正在使用这个 Upload Parts 上传块的请求，则 Upload Parts 会返回失败。

方法原型

```
abortMultipartUpload(params = {
                            Bucket: String,
                            Key: String
                            UploadId: String
                            }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
UploadId	分段上传的 UploadId，在 2.1 中返回的	String	是

返回结果说明

无

示例

```
let zos_client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
  Bucket: "bucket-name",
  Key:"key-name",
  UploadId:"2~-PlqKzMVWQkalw3Xi11ndHP4in9CNQH"
};
// 执行具体的操作
zos_client.abortMultipartUpload(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);       // successful response
```

```
});
```

List Multipart Uploads

功能说明

List Multiparts Uploads 用来查询正在进行中的分块上传。单次最多列出 1000 个正在进行中的分块上传。

方法原型

```
listMultipartUploads(params = {
                            Bucket: String,
                            KeyMarker: String,
                            MaxUploads: Number,
                            }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
KeyMarker	只有 Key 大于 KeyMarker 的分段上传数据才会返回	String	否
MaxUploads	单次最多返回的分段上传数据，大小是 1-1000，超过 1000 的数据会被视为 1000	Number	否

返回结果说明

```
{
    Bucket: "bucket-1"
    IsTruncated: false
    MaxUploads: 1000
    NextKeyMarker: "30M"
    Uploads: (3)
}
```

返回结果	描述	类型
Bucket	Bucket 名称	String
IsTruncated	是否是截断返回	Boolean
MaxUploads	本次最多可返回的分段上传信息	Number
NextKeyMarker	下次 list 的起始 Key，只要针对分段上传过多的情况，一次不能全部返回	String
Uploads	分段上传信息	Json

示例

```
let zos_client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: "http://192.168.218.130:7480",
    sslEnabled: true,
    s3ForcePathStyle: true
});
// 准备操作所需的参数
let params = {
  Bucket: "bucket-1"
};
// 执行具体的操作
zos_client.listMultipartUploads(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);        // successful response

});
```

图片处理

Post 请求处理图片

功能说明

该功能是对存储桶中的图片进行处理，并将处理后的图片持久化到指定的存储桶中，支持处理图片格式 JPG、PNG、WEBP、BMP、TIFF。

方法原型

```
processObject(params = {
                Bucket: String,
                Key: String,
                VersionId: String,
                ZosProcess: String
              }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	指定处理后的图片要存放的存储桶	String	是

ProcessSource	待处理的图片路径，格式{Bucket}/{Object}[?Versionid=]	String	是
Key	处理后的图片要保存的名称	String	否
ZosProcess	图片处理的方式，包含缩放，裁剪，旋转，水印，格式转换，获取信息功能	String	否

ZosProcess 的定义方法为:

- 图片缩放(e.g. image/resize,w_300,h_200,m_fixed)

参数名称	参数用途	取值	是否必须
w	指定目标缩放图宽度	[1,4096]	使用按百分比缩放可不指定宽高
h	指定目标缩放图高度	[1,4096]	使用按百分比缩放可不指定宽高
m	指定缩放模式	lfit（默认值）：等比缩放，目标缩放图为指定 w 和 h 矩形框内的最大图形。 mfit：等比缩放，目标缩放图为延伸出指定 w 和 h 矩形框外的最小图形。 fill：将原图等比缩放为延伸出指定 w 与 h 的矩形框外的最小图片，之后将超出的部分进行居中裁剪。 pad：将原图等比缩放为指定 w 和 h 矩形框内最大的图形，然后使用 color 指定的颜色将矩形框内剩余部分进行填充。 fixed：固定宽高，强制缩放。	否
color	缩放模式为 pad 时，指定填充颜色	RGB 颜色值，默认 FFFFFF(白色)	否（仅当 m 为 pad 模式时使用）
p	按百分比进行缩放	[1,1000] 小于 100 缩小，大于 100 放大	否
limit	指定目标缩放图大于原图时是否缩放	1(默认)：目标缩放图大于原图时返回原图 0：按指定参数缩放	否

- 图片裁剪(e.g. image/crop,w_100,h_100,x_10,y_10,g_se)

参数名称	参数用途	取值	是否必须
w	指定裁剪宽度。	[0,图片宽度] 默认为最大值。	否
h	指定裁剪高度。	[0,图片高度]	否

		默认为最大值。	
x	指定裁剪起点横坐标（默认左上角为原点）。	[0,图片边界]	否
y	指定裁剪起点纵坐标（默认左上角为原点）。	[0,图片边界]	否
g	设置裁剪的原点位置。原点按照九宫格的形式分布，一共有九个位置可以设置，为每个九宫格的左上角顶点。	nw：左上(默认) north：中上 ne：右上 west：左中 center：中部 east：右中 sw：左下 south：中下 se：右下	否

- 图片旋转(e.g. image/rotate,45)

参数名称	参数用途	取值	是否必须
[value]	图片按顺时针旋转的角度。	[0,360] 默认值：0，表示不旋转。	是

- 水印
  - 图片水印(e.g. image/watermark,image_aHVkaWUuanBnP3gtem9zLXByb2N1c3M9aW1hZ2UvcmVzaXplLHBfMzAvcm90YXRlLDE4MA==,g_north,t_40)
  - 文字水印(e.g. image/watermark,text_Q2hpbmFOZWxlY29t,type_heiti,color_FF0000,size_40,g_se,t_80)

参数名称	参数用途	取值	是否必须
t	图片水印或文字水印的透明度	[0, 100]	否
x	文字水印距离图片边界的水平距离	[0, 4096] 默认值：10	否
y	文字水印距离图片边界的垂直距离	[0, 4096] 默认值：10	否
text	指定文字水印内容	Base64 编码后的字符串，编码结果字符串中‘/’要替换为‘_’	否
color	指定文字水印的颜色	RGB 颜色值。 默认：FFFFFF（白色）	否
size	指定文字水印的字体大小	默认值：40	否
type	指定文字水印的字体类型	如 Airal, Helvetica, 支持中文字体包括 yahei(微软雅黑)，heiti(黑体)，kaishu（楷书）,youyuan(幼圆)	否
rotate	指定文字水印顺时针旋转角度	[0, 360] 默认值：0	否
image	指定图片水印名称，水印图片需要和原图存放在相同存储空间	水印图片可以进行预处理（e.g. 水印图片缩放为 30%并旋转 180 度，hudie.jpg?x-zos-process=image/resize,p_30/rotate,180 ），需要转换成 base64 编码，编码结果字符串中‘/’要替换为‘_’	否
g	指定水印在图片中的位置	nw：左上(默认) north：中上 ne：右上 west：左中 center：中部 east：右中 sw：左下 south：中下 se：右下	否

- 格式转化(e.g. image/format,png)

参数名称	参数用途	取值	是否必须
[value]	将原图转换成指定格式	Jpg、png、webp、bmp、tiff	是

- 获取图片信息(e.g. image/info)

返回结果说明

空

示例

processObject.js 示例如下:

```
let client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'ZOS endpoint',
    sslEnabled: true,
    s3ForcePathStyle: true,
});
let params = {
    Bucket: "bucket01",
    ProcessSource: "bucket01/hudie.jpg",
    Key: "hudie_js.jpg",
    ZosProcess: "image/resize,w_100"
};

client.processObject(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);        // successful response
});
```

Get 请求获取图片

功能说明

图片处理是在 get_object 基础进行的扩展，用于对存储桶中的图片文件在线处理，支持处理图片格式 JPG、PNG、WEBP、BMP、TIFF。

方法原型

```
getObject(params = {
                Bucket: String,
                Key: String,
                VersionId: String,
                ZosProcess: String
```

```
            }, callback) ⇒ AWS.Request
```

参数说明

参数名称	参数描述	类型	是否必须
Bucket	Bucket 的名称	String	是
Key	用户存储在 ZOS 中的对象的名字	String	是
VersionId	VersionId 用于引用对象的特定版本。	String	否
ZosProcess	图片处理的方式，包含缩放，裁剪，旋转，水印，格式转换，获取信息功能,定义方法同 4.1 参数说明	String	否

返回结果说明

```
{
  Body: Uint8Array(22) []
  ContentLength: 22
  ContentType: "application/octet-stream; charset=UTF-8"
  ETag: "\"0cb569f40c08764668c060860e85fcaf\""
  LastModified: Mon Jul 05 2021 10:21:13 GMT+0800 (中国标准时间)
  Metadata: {}
  StorageClass: "STANDARD"
  VersionId: "Ac9zLoox9n7sdioiV8UWAUuVfcQHiKA"
}
```

返回结果	描述	类型
Body	http 返回中的响应体，即对象的数据部分	Uint8Array
ContentLength	对象的长度	Int
ContentType	对象的类型	String
ETag	对象的 ETag 值	String
LastModified	对象的最后修改时间	Datetime
Metadata	对象在 ZOS 中的元数据	Map
StorageClass	对象在 ZOS 中的存储级别	String
VersionId	对象在 ZO 中的版本信息（前提是 bucket 开启了多版本）	String

示例

```
let client = new AWS.S3({
    accessKeyId: 'your access key',
    secretAccessKey: 'your secret key',
    endpoint: 'ZOS endpoint',
    sslEnabled: true,
```

```
    s3ForcePathStyle: true,
});
let params = {
    Bucket: "bucket1",
    Key: "fire_test.py",
    ZosProcess: "image/resize,w_200"
};

client.getObject(params, function(err, data) {
    if (err) console.log(err, err.stack); // an error occurred
    else     console.log(data);        // successful response
});
```