天翼云媒体存储

PHP SDK 使用指导书

天翼云科技有限公司

2026年3月27日

天翼云媒体存储PHP SDK使用指导书

SDK安装

PHP版本要求

天翼云媒体存储PHP SDK 要求使用 PHP 5.5 或更高版本。可以从 https://www.php.net/downloads 下载最新版本PHP。

安装方式

方式一：官网下载PHP-SDK

在天翼云官网下载 xos-php-sdk，下载地址： xos-php-sdk.zip

解压压缩包至任意目录，并在您的 php 脚本中包含解压目录中自动加载工具：

```
<?php
   require '/path/to/autoload.php';
?>
```

注意：使用PHP 5.5版本建议使用官网下载的PHP-SDK，需要设置默认timezone

```
// 使用php5.5需要设置默认timezone
if(!ini_get('date.timezone')){
    date_default_timezone_set('UTC');
}
```

方式二：添加AWS s3依赖

天翼云媒体存储兼容AWS s3接口，您可以通过AWS s3接口使用天翼云媒体存储。若您需要使用AWS s3接口，建议通过composer方式安装AWS s3 php sdk。您可以从 https://getcomposer.org/ 获取 composer 的最新版本下载地址、安装方式与用户文档。

在确保 composer 已经安装完成后，在工程根目录下执行以下命令安装 AWS s3 php sdk：

```
composer require aws/aws-sdk-php
```

初始化

使用SDK功能前，需要新建s3Client，代码如下：

```
require '/path/to/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\Credentials\Credentials;

const endpoint = '<your-endpoint>'; // e.g. http://endpoint or https://endpoint
```

```
const access_key = '<your-access-key>';
const secret_key = '<your-secret-key>';

$credentials = new Credentials(access_key, secret_key);

$s3Client = new S3Client([
    'region' => 'ctyun',                // region固定填ctyun
    'version' => '2006-03-01',          // s3接口版本号，固定填2006-03-01
    'credentials' => $credentials,
    'endpoint' => endpoint,
]);
```

参数	说明	是否必要
access_key	用户账号 access key	是
secret_key	用户账号 secret key	是
endpoint	天翼云资源池的地址，必须指定http或https前缀	是

桶相关接口

创建桶

功能说明

桶（Bucket）是用于存储对象（Object）的容器，所有的对象都必须隶属于某个桶。您可以使用 createBucket 接口创建桶。

代码示例

```
public function CreateBucket()
{
    try {
        $res = $this->s3Client->createBucket([
            'Bucket' => '<your-bucket-name>',
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	String	桶名称	是

获取桶列表

功能说明

桶（Bucket）是用于存储对象（Object）的容器，所有的对象都必须隶属于某个桶。您可以通过listBuckets接口获取桶列表信息。

代码示例

```
public function ListBucket()
{
    try {
        $res = $this->s3Client->listBuckets();
        var_dump($res->get('Buckets'));
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

返回结果

参数	类型	说明
Buckets	Bucket array	桶信息列表

判断桶是否存在

功能说明

桶（Bucket）是用于存储对象（Object）的容器，所有的对象都必须隶属于某个桶。您可以使用doesBucketExist 接口判断桶是否存在。

代码示例

```
public function DoesBucketExist()
{
    $bucket = '<your-bucket-name>';
    try {
        $exist = $this->s3Client->doesBucketExist($bucket);
        echo $exist;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

返回结果

参数	类型	说明
exist	bool	桶是否存在

删除桶

功能说明

桶（Bucket）是用于存储对象（Object）的容器，所有的对象都必须隶属于某个桶。您可以通过 deleteBucket 接口删除桶。删除一个桶前，需要先删除该桶中的全部对象（包括对象版本）。

代码示例

```
public function DeleteBucket()
{
    try {
        $res = $this->s3Client->deleteBucket([
            'Bucket' => '<your-bucket-name>',
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	String	桶名称	是

设置桶访问权限

功能说明

您可以使用putBucketAcl接口进行桶访问权限的修改。用户在设置 bucket 的 ACL 之前需要具备 WRITE_ACP 权限。桶访问权限包含了 AccessControlList 与CannedAccesssControlList 两种格式。

代码示例

- CannedAccesssControlList

使用 CannedAccesssControlList 设置桶的访问权限示例代码如下:

```
public function PutBucketAcl()
{
    //设置桶为公共读写
    try {
        $this->s3Client->putBucketAcl([
            'Bucket' => '<your-bucket-name>',
            'ACL' => 'public-read', // private、public-read、public-read-write
        ]);
        echo "Succeed in setting bucket ACL.\n";
```

```
    } catch (AwsException $e) {
        // Display error message
        echo $e->getMessage();
        echo "\n";
    }
    echo "putBucketAcl success\n";
}
```

- AccessControlList

使用 AccessControlList 设置桶访问权限时，可以设置特定用户对桶的访问权限。

```
public function PutBucketAcl2()
{
    try {
        $this->s3Client->putBucketAcl([
            'Bucket' => '<your-bucket-name>',
            'AccessControlPolicy' => [
                // 可以从 getBucketAcl 接口获取 Owner 信息
                'Owner' => [
                    'ID' => 'exampleuser',
                    'DisplayName' => 'Example DisplayName',
                ],
                'Grants' => [
                    [
                        //开启用户 <your-user-id> 的完全控制权限
                        'Grantee' => [
                            'ID' => '<your-user-id>',
                            'Type' => 'CanonicalUser',
                        ],
                        'Permission' => 'FULL_CONTROL', // FULL_CONTROL、WRITE、
WRITE_ACP、READ、READ_ACP
                    ],
                    [
                        //开启所有用户的读权限
                        'Grantee' => [
                            'Type' => 'Group',
                            'URI' =>
'http://acs.amazonaws.com/groups/global/AllUsers',
                        ],
                        'Permission' => 'READ',
                    ],
                    // ...
                ],
            ],
        ]);
        echo "Succeed in setting bucket ACL.\n";
    } catch (AwsException $e) {
        echo $e->getMessage();
        echo "\n";
    }
}
```

请求参数

- CannedAccesssControlList

使用 CannedAccesssControlList 方式设置桶权限参数如下：

参数	类型	说明	是否必要
Bucket	string	桶名称	是
ACL	string	CannedAccesssControlList 值	是

CannedAccesssControlList 是一系列的预定义访问权限，通过 ACL 参数设置，ACL可设置为以下值：

ACL值	权限
private	私有读写
public-read	公共读私有写
public-read-write	公共读写

- AccessControlList

使用 AccessControlList 方式设置桶权限参数如下：

参数	类型	说明	是否必要
Bucket	string	桶名称	是
AccessControlPolicy	AccessControlPolicy	acl详细配置	是

在 AccessControlList 中可通过 Grants 设置权限，Grants 中关于Permission说明如下：

Permission值	权限
READ	允许列出桶中的对象
WRITE	允许创建、覆盖、删除桶中的对象
READ_ACP	允许获取桶的ACL信息
WRITE_ACP	允许修改桶的ACL信息
FULL_CONTROL	获得READ、WRITE、READ_ACP、WRITE_ACP权限

获取桶访问权限

功能说明

您可以使用 getBucketAcl 接口获取桶的访问权限。

代码示例

```
public function GetBucketAcl()
{
    try {
        $resp = $this->s3Client->getBucketAcl([
            'Bucket' => '<your-bucket-name>'
```

```
        ]);
        //打印获取的桶 owner displayname，ID 以及访问权限信息
        echo "Succeed in retrieving bucket ACL as follows: \n";
        echo 'Owner DisplayName: ' . $resp['Owner']['DisplayName'] . "\n";
        echo 'Owner ID: '          . $resp['Owner']['ID']          . "\n";
        foreach ($resp['Grants'] as $grant) {
            echo "Grant: \n";
            foreach($grant['Grantee'] as $k=>$val)
            {
                echo $k . ":  " . $val . "\n";
            }
            echo 'Permission: ' . $grant['Permission'] . "\n";
        }
    } catch (AwsException $e) {
        echo 'error:' . $e->getMessage();
        echo "\n";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

返回结果

参数	类型	说明
Owner	Owner	桶的owner信息
Grants	Grants	桶的访问权限信息

设置桶策略

功能说明

您可以使用 putBucketPolicy 接口设定桶策略，桶策略可以灵活地配置用户各种操作和访问资源的权限。访问控制列表只能对单一对象设置权限，而桶策略可以基于各种条件对一个桶内的全部或者一组对象配置权限。桶的拥有者拥有 PutBucketPolicy 操作的权限，如果桶已经设置了 policy ，则新设置的 policy 会覆盖原有的policy。

policy的示例如下：

```
{
    "Id":"PolicyId",
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ExampleStatementID1",
            "Principal":{
                "AWS":[
                    "arn:aws:iam:::user/userId",
                    "arn:aws:iam:::user/userName"
```

```
                ]
            },
            "Effect":"Allow",
            "Action":[
                "s3:ListBucket",
                "s3:CreateBucket"
            ],
            "Resource":[
                "arn:aws:iam:::exampleBucket"
            ],
            "Condition":"some conditions"
        },
        ......
    ]
}
```

Statement的内容说明如下：

元素	描述	是否必要
Sid	statement Id，可选关键字，描述statement的字符串	否
Principal	可选关键字，被授权人，指定本条statement权限针对的Domain以及User，支持通配符“*”，表示所有用户（匿名用户）。当对Domain下所有用户授权时，Principal格式为arn:aws:iam:::user/*。当对某个User进行授权时，Principal格式为arn:aws:iam:::user/<your-user-name>	可选，Principal与NotPrincipal选其一
NotPrincipal	可选关键字，不被授权人，statement匹配除此之外的其他人。取值同Principal	可选，NotPrincipal与Principal选其一

元素	描述	是否必要
Action	可选关键字，指定本条statement作用的操作，Action字段为对象存储支持的所有操作集合，以字符串形式表示，不区分大小写。支持通配符“*”，表示该资源能进行的所有操作。例如：“Action”:[“s3:List*”, “s3:Get*”]	可选，Action与NotAction选其一
NotAction	可选关键字，指定一组操作，statement匹配除该组操作之外的其他操作。 取值同Action	可选，NotAction与Action选其一
Effect	必选关键字，指定本条statement的权限是允许还是拒绝，Effect的值必须为Allow或者Deny	必选
Resource	可选关键字，指定statement起作用的一组资源，支持通配符“*”，表示所有资源	可选，Resource与NotResource选其一
NotResource	可选关键字，指定一组资源，statement匹配除该组资源之外的其他资源。 取值同Resource	可选，NotResource与Resource选其一
Condition	可选关键字，本条statement生效的条件	可选

代码示例

在上传对象时如果不设置对象访问权限，默认下只有对象的拥有者才能访问该对象。如果需要使桶内对象可公共读，可以通过设置桶策略的方式允许桶内对象公共读，以下代码展示如何设置桶内对象可公共读的策略：

```
public function PutBucketPolicy()
{
    $bucket = '<your-bucket-name>';
    $this->s3Client->putBucketPolicy([
        'Bucket' => $bucket,
        'Policy' => '{
                "Version": "2012-10-17",
                "Statement": [{
                    "Sid": "1",
                    "Effect": "Allow",
                    "Principal": {"AWS": "*"},
                    "Action": ["s3:GetObject"],
                    "Resource": ["arn:aws:s3:::' . $bucket . '/*"]
                }]
            }',
    ]);
    echo "putBucketPolicy success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Policy	string	策略内容，json字符串	是

获取桶策略

功能说明

您可以使用 getBucketPolicy 接口获取桶策略。

代码示例

```
public function GetBucketPolicy()
{
    $res = $this->s3Client->getBucketPolicy([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "getBucketPolicy success " . $res->get('Policy') . "\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

返回结果

参数	类型	说明
Policy	string	json格式的桶策略

关于桶策略的说明请参考 桶策略说明 。

删除桶策略

功能说明

您可以使用 deleteBucketPolicy 接口删除桶策略。

代码示例

```
public function DeleteBucketPolicy()
{
    $this->s3Client->deleteBucketPolicy([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "deleteBucketPolicy success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	String	桶名称	是

设置桶生命周期配置

功能说明

生命周期管理可以通过设置规则实现自动清理过期的对象，优化存储空间。本文介绍如何设置桶（Bucket）生命周期配置。
您可以使用putBucketLifecycleConfiguration接口设置桶的生命周期配置，配置规则可以通过匹配对象key前缀、标签的方法设置当前版本或者历史版本对象的过期时间，对象过期后会被自动删除。

代码示例

```
public function PutBucketLifecycleConfiguration()
{
    $this->s3Client->putBucketLifecycleConfiguration([
        'Bucket' => '<your-bucket-name>',
        'LifecycleConfiguration' => [
            'Rules' => [
                [
                    'ID' => 'TestOnly', // unique id
                    'Expiration' => [
                        'Days' => 365,
                    ],
                    'Status' => 'Enabled', // required
                    'Filter' => [       // required
                        'Prefix' => '',
                    ],
                    'Transitions' => [
                        [
                            'Days' => 365,
                            'StorageClass' => 'GLACIER',
                        ],
                    ],
                    'AbortIncompleteMultipartUpload' => [
                        'DaysAfterInitiation' => 365,
                    ],
                ],
            ],
        ],
    ]);
    echo "putBucketLifecycleConfiguration success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

参数	类型	说明	是否必要
LifecycleConfiguration	LifecycleConfiguration	封装了生命周期规则的数组，最多包含1000条规则	是

关于生命周期规则Rule一些说明

参数	类型	说明	是否必要
ID	string	规则ID	否
Status	string	是否启用规则（Enabled|Disabled）	是
Expiration	Expiration	文件过期时间	否
AbortIncompleteMultipartUpload	AbortIncompleteMultipartUpload	未完成上传的分片过期时间	否
Transitions	Transition数组	文件转换到低频存储规则（距离修改时间）	否
Filter	Filter	应用范围，可以指定前缀或对象标签	否

关于Expiration的说明：

参数	类型	说明
Days	int	过期天数

关于AbortIncompleteMultipartUpload的说明：

参数	类型	说明
DaysAfterInitiation	int	过期天数

关于Transition的说明：

参数	类型	说明
Days	int	转换过期天数
StorageClass	StorageClass	要转换的存储类型

关于Filter的说明：

参数	类型	说明
Prefix	string	需要过滤的前缀

获取桶生命周期配置

功能说明

您可以使用 GetBucketLifecycleConfiguration 接口获取桶的生命周期配置。

代码示例

```php
public function GetBucketLifecycleConfiguration()
{
    $res = $this->s3Client->getBucketLifecycleConfiguration([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "getBucketLifecycleConfiguration success " . $res . "\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

返回结果

参数	类型	说明
Rules	Rules	一个描述生命周期管理的规则数组，一条规则包含了规则ID、匹配的对象key前缀、匹配的对象标签信息、当前版本对象过期时间、历史版本对象过期时间和是否生效标识等信息

关于生命周期规则Rule一些说明

参数	类型	说明
ID	string	规则ID
Status	string	是否启用规则（Enabled|Disabled）
Expiration	Expiration	文件过期时间
AbortIncompleteMultipartUpload	AbortIncompleteMultipartUpload	未完成上传的分片过期时间
Transitions	Transition数组	文件转换到低频存储规则（距离修改时间）
Filter	Filter	应用范围，可以指定前缀或对象标签

关于Expiration的说明：

参数	类型	说明
Days	int	过期天数

关于AbortIncompleteMultipartUpload的说明：

参数	类型	说明
DaysAfterInitiation	int	过期天数

关于Transition的说明：

参数	类型	说明
Days	int	转换过期天数
StorageClass	StorageClass	要转换的存储类型

关于Filter的说明：

参数	类型	说明
Prefix	string	需要过滤的前缀

删除桶生命周期配置

功能说明

您可以使用deleteBucketLifecycle接口删除桶的生命周期配置。

代码示例

```
public function DeleteBucketLifecycleConfiguration()
{
    $this->s3Client->deleteBucketLifecycle([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "deleteBucketLifecycle success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

设置桶跨域访问配置

功能说明

跨域资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。利用 CORS 支持，您可以构建丰富的客户端 Web 应用程序，同时可以选择性地允许跨源访问您的资源。

您可以通过putBucketCors接口设置桶的跨域访问配置。

代码示例

```
public function PutBucketCors()
{
    $this->s3Client->putBucketCors([
        'Bucket' => '<your-bucket-name>',
        'CORSConfiguration' => [
            'CORSRules' => [
```

```
                [
                    'AllowedHeaders' => ["*"],
                    'AllowedMethods' => ["POST", "GET", "PUT", "DELETE",
"HEAD"],
                    'AllowedOrigins' => ["*"], // 可以使用http://domain:port
                    'ExposeHeaders' => ["ETag"],
                    'MaxAgeSeconds' => 3600,
                ],
            ],
        ]
    ]);
    echo "putBucketCors success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
CORSConfiguration	CORSConfiguration	跨域访问规则	是

关于CORSRules一些说明

参数	说明
AllowedMethods	允许的请求方法
AllowedOrigins	允许的请求源
AllowedHeaders	允许的请求头
ExposedHeaders	允许返回的Response Header
MaxAgeSeconds	跨域请求结果的缓存时间

获取桶跨域访问配置

功能说明

跨域资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。利用 CORS 支持，您可以构建丰富的客户端 Web 应用程序，同时可以选择性地允许跨源访问您的资源。

您可以通过getBucketCors接口获取桶跨域访问配置。

代码示例

```
public function GetBucketCors()
{
    $res = $this->s3Client->getBucketCors([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "getBucketCors success " . $res . "\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

返回结果

参数	类型	说明
CORSRules	CORSRules	跨域访问规则

关于CORSRules一些说明

参数	说明
AllowedMethods	允许的请求方法
AllowedOrigins	允许的请求源
AllowedHeaders	允许的请求头
ExposedHeaders	允许返回的Response Header
MaxAgeSeconds	跨域请求结果的缓存时间

删除桶跨域访问配置

功能说明

跨域资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。利用 CORS 支持，您可以构建丰富的客户端 Web 应用程序，同时可以选择性地允许跨源访问您的资源。

您可以通过deleteBucketCors接口删除桶跨域访问配置。

示例代码

```
public function DeleteBucketCors()
{
    $this->s3Client->deleteBucketCors([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "deleteBucketCors success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

设置桶版本控制状态

功能说明

通过媒体存储提供的版本控制，您可以在一个桶中保留多个对象版本。例如，image.jpg(版本1)和image.jpg(版本2)。如果您希望防止自己意外覆盖和删除版本，或存档对象，以便您可以检索早期版本的对象，您可以开启版本控制功能。

- 开启版本控制

  对桶中的所有对象启用版本控制，之后每个添加到桶中的对象都会被设置一个唯一的version id。

- 暂停版本控制

  对桶中的所有对象暂停版本控制，之后每个添加到桶中的对象的version ID会被设置为null。桶开启版本控制功能之后，无法再关闭该功能，只能暂停。

您可以使用 putBucketVersioning 接口开启或暂停版本控制。

代码示例

```
public function PutBucketVersioning()
{
    //启用版本控制
    $this->s3Client->putBucketVersioning([
        'Bucket' => '<your-bucket-name>',
        'VersioningConfiguration' => [
            'Status' => 'Enabled', //启用版本控制：Enabled，暂停版本控制：Suspended
        ],
    ]);
    echo "putBucketVersioning success\n";
}
```

请求参数

PutBucketVersioning 的参数说明：

参数	类型	说明	是否必要
Bucket	String	桶名称	是
VersioningConfiguration	VersioningConfiguration	版本控制设置	是

关于 VersioningConfiguration 中 Status 的一些说明

参数	说明
Enabled	开启版本控制
Suspended	暂停版本控制

获取桶版本控制状态

功能说明

您可以使用 getBucketVersioning接口获取版本控制状态。

代码示例

```
public function GetBucketVersioning()
{
    try {
        $res = $this->s3Client->getBucketVersioning([
            'Bucket' => '<your-bucket-name>',
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

返回结果

参数	类型	说明
Status	string	桶是否开启版本控制。若无Status返回，则未开启；如Status为Enabled，则已开启；如Status为Suspended，则已暂停

对象相关接口

获取对象列表

功能说明

您可以使用 listObjects 接口列举对象，每次最多返回1000个对象。

代码示例

以下代码展示如何简单列举对象：

```
public function ListObjects()
{
    try {
        $res = $this->s3Client->listObjects([
            'Bucket' => '<your-bucket-name>',
        ]);
        foreach ($res['Contents']  as $object) {
            echo $object['Key'] . "\n";
        }
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

如果 list 大于1000，则可以使用 getPaginator 接口列举所有对象。列举所有对象示例代码如下：

```
public function ListObjects2()
{
    try {
        $results = $this->s3Client->getPaginator('ListObjectsV2', [
            'Bucket' => '<your-bucket-name>',
        ]);
        foreach ($results as $result) {
            foreach ($result['Contents'] as $object) {
                echo $object['Key'] . "\n";
            }
        }
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo $e->getMessage() . "\n";
    }
}
```

请求参数

参数	类型	说明	是否必须
Bucket	string	设置桶名称	是

返回结果

属性名	类型	说明
Contents	object array	返回的对象数组，包含对象名，最后修改时间，ETag等信息

上传对象

功能说明

您可以使用 putObject 接口上传对象。

代码示例

```
public function PutObject()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $res = $this->s3Client->putObject([
            'Bucket' => $bucket,
            'Key' => $objectName,
            'Body' => "1234",
            'ACL' => 'public-read',          //设置 ACL 为公共读
            'ContentType' => "text/json",    // 设置content-type
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
```

```
    }
}
```

上传本地文件到对象存储：

```php
public function PutObjectLocalFile()
{
    $file_Path = '<your-file-path>';
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $res = $this->s3Client->putObject([
            'Bucket' => $bucket,
            'Key' => $objectName,
            'SourceFile' => $file_Path,
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名	是
Key	string	对象名	是
Body	string	对象内容	否
SourceFile	string	要上传的文件路径	否
ACL	string	对象访问权限，取值private | public-read | public-read-write	否

返回结果

参数	类型	说明
ETag	string	对象的唯一标签

下载对象

功能说明

您可以使用 getObject 接口下载对象。

代码示例

```php
public function GetObject()
{
```

```
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        //            $dateTime = new DateTime('<your-datetime>');
        $res = $this->s3Client->getObject([
            'Bucket' => $bucket,
            'Key' => $objectName,
            //'IfModifiedSince' => $dateTime,       //指定晚于修改时间
            //'IfMatch' => '<your-ETag>',           //指定匹配的ETag
        ]);
        echo $res->get('Body')->getContents();
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	执行本操作的桶名称	是
Key	string	对象的key	是
IfModifiedSince	DateTime	如果指定的时间早于实际修改时间，则正常传送。否则返回304代码	否
IfUnmodifiedSince	DateTime	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件；否则返回304代码	否
IfMatch	string	如果传入的ETag和Object的 ETag匹配，则正常传输；否则返回412代码	否
IfNoneMatch	string	如果传入的ETag值和Object的ETag不匹配，则正常传输；否则返回412代码	否

下载文件时，可以指定一个或者多个限定条件，满足条件时才下载对象，不满足时则返回错误代码，不下载对象。

返回结果

参数	类型	说明
Body	string	对象数据内容

复制对象

功能说明

您可以使用 copyObject 接口复制对象，您需要设置复制的对象名，所在的桶以及目标桶和对象名。

代码示例

复制一个对象

```
public function CopyObject()
{
    $desBucket = '<your-bucket-name>';   //目标桶
    $desKeyName = '<your-object-key>';   //目标对象名
    $srcBucket = '<source-bucket-name>'; //从此桶复制
    $srcKeyName = '<source-object-key>'; //复制的对象名
    try {
        $result = $this->s3Client->copyObject(array(
            'Bucket' => $desBucket,
            'Key' => $desKeyName,
            'CopySource' => '/' . $srcBucket . '/' . $srcKeyName,
        ));
        echo $result;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo $e->getMessage() . "\n";
    }
}
```

文件比较大（超过1GB）的情况下，直接使用copyObject 可能会出现超时，需要使用分片复制的方式进行文件复制。Aws\S3\MultiPartCopy封装了分片复制的接口，可以用于分片复制文件，具体示例请参考 分片上传融合接口 中的使用 MultiPartCopy 进行分片复制部分。

请求参数

参数	类型	说明	是否必要
Bucket	string	目标桶名称	是
Key	string	目标对象key	是
CopySource	string	URL格式的复制对象数据来源，包含了桶名称和对象key的信息，二者之间使用正斜杆（/）分割。例如，"/foo/boo"表示复制foo桶中的boo对象	是

返回结果

参数	类型	说明
ETag	string	对象的唯一标签

删除对象

功能说明

您可以使用 deleteObject 接口删除单个对象。

代码示例

```
public function DeleteObject()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $res = $this->s3Client->deleteObject([
            'Bucket' => $bucket,
            'Key' => $objectName,
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名	是
Key	string	对象名	是

批量删除对象

功能说明

您可以使用 deleteObjects 接口批量删除多个对象。

代码示例

```
public function DeleteObjects()
{
    try {
        $res = $this->s3Client->deleteObjects([
            'Bucket' => '<your-bucket-name>',
            'Delete' => [
                'Objects' => [
                    [
                        'Key' => '<your-object-key1>',
                    ],
                    [
                        'Key' => '<your-object-key2>',
                    ],
                ]
            ],
        ]);
        echo $res;
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Delete	Delete	要删除的对象key列表	是

获取对象元数据

功能说明

您可以使用 headObject 接口获取对象元数据。headObject 操作的请求参数与 getObject 类似，但是 headObject 返回的http响应中没有对象数据。

代码示例

```
public function HeadObject()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $res = $this->s3Client->headObject([
            'Bucket' => $bucket,
            'Key' => $objectName,
        ]);
        echo 'ETag: ' . $res->get('ETag') . "\n";                    //打印对象ETag
        echo 'ContentLength: ' . $res->get('ContentLength') . "\n";  //打印对象大小
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象key	是

返回结果

返回的属性如下：

参数	类型	说明
ContentLength	Integer	本次请求返回对象数据的大小（单位：字节）
ContentType	String	对象文件格式的标准MIME类型
ETag	String	对象的Entity Ttag
LastModified	Date	最近一次修改对象的时间
VersionId	String	对象最新的版本ID

设置对象访问权限

功能说明

与桶访问权限类似，对象访问权限同样具有 AccessControlList 与 CannedAccessControlList 两种。需要注意的是，对象的访问优先级要高于桶访问权限。比如桶访问权限是 private，但是对象访问权限是 public read，则所有用户都可以访问该对象。默认情况下，只有对象的拥有者才能访问该对象，即对象的访问权限默认是 private。设置对象ACL操作需要具有对象的WRITE_ACP权限。

代码示例

- CannedAccesssControlList

使用 CannedAccesssControlList 设置桶的访问权限示例代码如下:

```
public function PutObjectAcl(){
    //设置对象为公共读写
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $this->s3Client->putObjectAcl([
            'ACL' => 'public-read',
            'Bucket' => $bucket,
            'Key' => $objectName,
        ]);
        echo "Succeed in setting object ACL.\n";
    } catch (AwsException $e) {
        echo $e->getMessage();
        echo "\n";
    }
}
```

- AccessControlList

使用 AccessControlList 设置对象访问权限时，可以设置特定用户对象的访问权限。使用 AccesssControlList设置对象的权限示例代码如下:

```
public function PutObjectAcl2(){
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $this->s3Client->putObjectAcl([
            'AccessControlPolicy' => [
                'Grants' => [
```

```
                    [
                        //开启用户 exampleuser 的完全控制权限
                        'Grantee' => [
                            'ID' => 'exampleuser',
                            'Type' => 'CanonicalUser',
                        ],
                        'Permission' => 'FULL_CONTROL',
                    ],
                    [
                        //开启所有用户的读权限
                        'Grantee' => [
                            'Type' => 'Group',
                            'URI' =>
'http://acs.amazonaws.com/groups/global/AllUsers',
                        ],
                        'Permission' => 'READ',
                    ],
                    // ...
                ],
                // 可以从 getObjectAcl 接口获取 Owner 信息
                'Owner' => [
                    'DisplayName' => 'exampleuser',
                    'ID' => 'Example DisplayName',
                ],
            ],
            'Bucket' => $bucket,
            'Key' => $objectName,
        ]);
        echo "Succeed in setting object ACL.\n";
    } catch (AwsException $e) {
        // Display error message
        echo $e->getMessage();
        echo "\n";
    }
}
```

请求参数

- CannedAccesssControlList

使用 CannedAccesssControlList 方式设置桶权限参数如下：

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象名	是
ACL	string	CannedACL值	是

CannedAccesssControlList 是一系列的预定义访问权限。

ACL值	权限
private	私有读写
public-read	公共读私有写
public-read-write	公共读写

- AccessControlList

使用 AccessControlList 方式设置桶权限参数如下：

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象名	是
AccessControlPolicy	AccessControlPolicy	acl详细配置	是

在 AccessControlList 中可通过 Grants 设置权限，Grants 中关于Permission说明如下：

Permission值	权限
READ	允许读取对象数据和元数据
WRITE	不可作用于对象
READ_ACP	允许获取对象的ACL信息
WRITE_ACP	允许修改对象的ACL信息
FULL_CONTROL	获得READ、READ_ACP、WRITE_ACP权限

获取对象访问权限

功能说明

您可以使用 getObjectAcl 接口获取对象访问的权限。

代码示例

```
public function GetObjectAcl(){
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    try {
        $resp = $this->s3Client->getObjectAcl([
            'Bucket' => $bucket,
            'Key' => $objectName,
        ]);
        echo "Succeed in retrieving object ACL as follows: \n";
        //打印获取的对象 owner displayname，ID 以及访问权限信息
        echo 'Owner DisplayName: ' . $resp['Owner']['DisplayName'] . "\n";
        echo 'Owner ID: '          . $resp['Owner']['ID']          . "\n";
        foreach ($resp['Grants'] as $grant) {
            echo "Grant: \n";
```

```
            foreach($grant['Grantee'] as $k=>$val)
            {
                echo $k . ":  " . $val . "\n";
            }
            echo 'Permission: ' . $grant['Permission'] . "\n";
        }
    } catch (AwsException $e) {
        echo $e->getMessage();
        echo "\n";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象key	是
VersionId	string	设置标签信息的对象的版本Id	否

返回结果

参数	类型	说明
Owner	Owner	对象的owner信息
Grants	Grants	对象的访问权限信息

获取对象标签

功能说明

您可以使用GetObjectTagging接口获取对象标签。

代码示例

```
public function GetObjectTagging()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $res = $this->s3Client->getObjectTagging([
        'Bucket' => $bucket,
        'Key' => $objectName,
    ]);
    echo "getObjectTagging success " . $res . "\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象key	是
VersionId	string	设置标签信息的对象的版本Id	否

返回参数

参数	类型	说明
TagSet	TagSet	设置的标签信息，包含了一个Tag结构体的数组，每个Tag以Key-Value的形式说明了标签的内容

删除对象标签

功能说明

您可以使用deleteObjectTagging接口删除对象标签。

代码示例

```
public function DeleteObjectTagging()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $this->s3Client->deleteObjectTagging([
        'Bucket' => $bucket,
        'Key' => $objectName,
    ]);
    echo "deleteObjectTagging success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	执行本操作的桶名称	是
Key	string	设置标签信息的对象key	是
VersionId	string	设置标签信息的对象的版本Id	否

设置对象标签

功能说明

您可以使用putObjectTagging接口为对象设置标签。标签是一个键值对，每个对象最多可以有10个标签。bucket的拥有者默认拥有给bucket中的对象设置标签的权限，并且可以将权限授予其他用户。每次执行PutObjectTagging操作会覆盖对象已有的标签信息。每个对象最多可以设置10个标签，标签Key和Value区分大小写，并且Key不可重复。每个标签的Key长度不超过128字节，Value长度不超过255字

节。SDK通过HTTP header的方式设置标签且标签中包含任意字符时，需要对标签的Key和Value做URL编码。设置对象标签信息不会更新对象的最新更改时间。

代码示例

```
public function PutObjectTagging()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $this->s3Client->putObjectTagging([
        'Bucket' => $bucket,
        'Key' => $objectName,
        'Tagging' => [ // required
            'TagSet' => [
                [
                    'Key' => 'key1',
                    'Value' => 'value1',
                ],
            ]
        ],
    ]);
    echo "putObjectTagging success\n";
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象key	是
Tagging	Tagging	设置的标签信息，包含了一个Tag结构体的数组，每个Tag以Key-Value的形式说明了标签的内容	是
VersionId	string	设置标签信息的对象的版本Id	否

生成下载预签名URL

功能说明

您可以通过 createPresignedRequest 接口为一个指定对象生成一个预签名的下载链接。

代码示例

生成下载预签名URL：

```php
public function generateGetobjectPresignedUrl()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $expires = "+5 minutes"; // 表示5分钟后过期
    $cmd = $this->s3Client->getCommand('GetObject', [
        'Bucket' => $bucket,
        'Key' => $objectName,
    ]);
    $request = $this->s3Client->createPresignedRequest($cmd, $expires);
    $presignedUrl = (string)$request->getUri();
    echo "generateGetobjectPresignedUrl success " . $presignedUrl . "\n";
}
```

生成下载对象的预签名URL后，可以通过该URL下载文件：

```php
public function getObjUsingPresignedUrl($presignedUrl, $localFilePath)
{
    try {
        $objectContent = file_get_contents($presignedUrl);

        if ($objectContent === false) {
            $error = error_get_last();
            echo "Download failed: " . $error['message'] . "\n";
            return;
        }

        file_put_contents($localFilePath, $objectContent);

        echo "Download successful. File saved to: " . $localFilePath . "\n";
    } catch (Exception $e) {
        echo "Download failed: " . $e->getMessage() . "\n";
    }
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象key	是
expires	int|string	过期时间（Unix时间戳或者能用strtotime解析的字符串）	是

返回结果

生成对应的预签名下载 URL，该链接允许用户在指定的时间内直接从媒体存储下载对象。

生成上传预签名URL

功能说明

您可以通过 createPresignedRequest 接口为一个指定对象生成一个预签名的上传链接。

代码示例

生成上传预签名URL：

```
public function generatePutobjectPresignedUrl()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $expires = time() + 300; // 当前时间戳加上300秒，表示5分钟后过期
    $cmd = $this->s3Client->getCommand('PutObject', [
        'Bucket' => $bucket,
        'Key' => $objectName,
    ]);
    $request = $this->s3Client->createPresignedRequest($cmd, $expires);
    $presignedUrl = (string)$request->getUri();
    echo "generatePutobjectPresignedUrl success " . $presignedUrl . "\n";
}
```

通过该预签名URL，可以直接将文件上传到指定的桶：

```
public function putObjUsingPresignedUrl($presignedUrl, $localFilePath)
{
    if (!file_exists($localFilePath)) {
        echo "File does not exist: " . $localFilePath . "\n";
        return;
    }

    $file = fopen($localFilePath, 'r');

    $ch = curl_init($presignedUrl);
    curl_setopt($ch, CURLOPT_PUT, true);
    curl_setopt($ch, CURLOPT_INFILE, $file);
    curl_setopt($ch, CURLOPT_INFILESIZE, filesize($localFilePath));

    $result = curl_exec($ch);

    if ($result === false) {
        echo "Upload failed: " . curl_error($ch) . "\n";
    } else {
        echo "Upload successful. File uploaded from: " . $localFilePath . "\n";
    }

    // 关闭资源
    curl_close($ch);
    fclose($file);
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
Key	string	对象key	是
expires	int|string	过期时间（Unix时间戳或者能用strtotime解析的字符串）	是

返回结果

生成对应的预签名上传 URL，该链接允许用户在指定的时间内直接将对象上传到媒体存储存储桶。

Post上传

功能说明

PostObjectV4接口为一个指定对象生成一个支持post方式上传文件的参数集合，可以在前端使用post form-data的方式上传文件。

代码示例

```
public function postPresign() {
    $objectName = '<your-object-key>';
    $formInputs = [
        'key' => $objectName
    ];
    $options = [
        ['starts-with', '$bucket', ''],
        ['starts-with', '$key', ''],
        ['starts-with', '$acl', ''],
        ['starts-with', '$Content-Type', ''],
    ];

    $expires = '+2 hours';
    $postObject = new Aws\S3\PostObjectV4(
        $this->s3Client,
        '<your-bucket-name>',
        $formInputs,
        $options,
        $expires
    );

    $formAttributes = $postObject->getFormAttributes();
    $formInputs = $postObject->getFormInputs();
    var_dump($formAttributes);
    var_dump($formInputs);
}
```

请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
formInputs	array	前端输入参数，用于配置acl，ContentType	是，可以为空
options	array	参数策略，可以限制输入的参数，至少需要指定bucket	是
expires	string	过期时间，能用strtotime解析的字符串	否

返回结果

参数	类型	说明
FormAttributes	FormAttributes	请求上传的url，http方法等
FormInputs	FormInputs	前端输入参数，包括v4签名和policy

关于FormAttributes的说明：

参数	类型	说明
action	string	请求上传的url

关于FormInputs的说明：

参数	类型	说明
Policy	string	服务端用于校验的policy
X-Amz-Algorithm	string	v4签名，哈希算法
X-Amz-Signature	string	v4签名，请求的参数签名
X-Amz-Date	string	v4签名，日期信息
X-Amz-Credential	string	v4签名，ak信息

前端使用方式如下：

```
<form action="<action>" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="Policy" value="<data.fields['Policy']>" />
    <input type="hidden" name="X-Amz-Algorithm" value="<data.fields['X-Amz-
Algorithm']>" />
    <input type="hidden" name="X-Amz-Credential" value="<data.fields['X-Amz-
Credential']>" />
    <input type="hidden" name="X-Amz-Date" value="<data.fields['X-Amz-Date']>"
/>
    <input type="hidden" name="X-Amz-Signature" value="<data.fields['X-Amz-
Signature']>" />
    <input type="hidden" name="bucket" value="<data.fields['bucket']>" />
    <input type="hidden" name="key" value="<data.fields['key']>" />

    <input type="file" name="file" value="" />
    <input type="submit" value="Submit" />
</form>
```

获取多版本对象列表

功能说明

如果桶开启了版本控制，您可以使用 listObjectVersions 接口列举对象的版本，每次list操作最多返回1000个对象版本。

代码示例

以下代码展示如何简单列举对象版本：

```
public function ListObjectVersions(){
    $versions = $this->s3Client->listObjectVersions([
        'Bucket' => '<your-bucket-name>'
    ])->search('Versions');
    foreach ($versions as $version){
        echo "Key: " . $version['Key'] . "\n";
        echo "VersionId: " . $version['VersionId'] . "\n";
    }
}
```

如果 list 大于1000，则可以使用 Paginator 接口列举所有对象版本。列举所有对象版本示例代码如下：

```
public function ListObjectVersions2(){
    $versions = $this->s3Client->getPaginator('ListObjectVersions', [
        'Bucket' => '<your-bucket-name>'
    ])->search('Versions');
    foreach ($versions as $version){
        echo "Key: " . $version['Key'] . "\n";
        echo "VersionId: " . $version['VersionId'] . "\n";
    }
}
```

请求参数

参数	类型	说明	是否必须
Bucket	string	设置桶名称	是

返回结果

属性名	类型	说明
Versions	version array	返回的对象版本数组，包含对象名，版本id，最后修改时间，ETag等信息

分片上传

融合接口

功能说明

分片上传步骤较多，包括初始化、文件切片、各个分片上传、完成上传。分片复制包括了初始化、源对象信息获取、各个分片复制、完成复制。为了简化分片上传和复制，PHP SDK 提供了对分片上传和分片复制的封装。Aws\S3\MultipartUploader 接口提供了简洁的分片上传方式，Aws\S3\MultiPartCopy 接口提供了简洁的分片复制方式。在使用这些封装接口的同时，您同样可以配置一些参数，控制分片的大小、并发数。

代码示例

- 使用 MultipartUploader 进行分片上传：

```
public function MultiPartUpload()
{
    $file_Path = '<your-file-path>';
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';

    $uploader = new Aws\S3\MultipartUploader($this->s3Client, $file_Path, [
        'bucket' => $bucket,
        'key'    => $objectName,
        'concurrency' => 5,         // 设置上传分片 UploadPart 操作的最大并行数量，默认为5.
        'part_size' => 5242880,     // 设置分片大小，默认为5M.
        'acl' => 'public-read',     // 设置ACL，参考值private | public-read
        'before_initiate' => function(\Aws\Command $command)
        {
            $command['ContentType'] = 'text/json';      // 设置content-type
        },
    ]);

    try {
        $result = $uploader->upload();
        echo "Upload complete: {$result['ObjectURL']}" . "\n";
    } catch (Aws\Exception\MultipartUploadException $e) {
        echo $e->getMessage() . "\n";
    }
}
```

- 使用 MultiPartCopy 进行分片复制：

```
public function MultiPartCopy()
{
    $src_bucket = '<source-bucket-name>'; //从此桶复制
    $src_key = '<source-object-key>'; //复制的对象名

    $dst_bucket = '<your-bucket-name>';   //目标桶
    $dst_key = '<your-object-key>';   //目标对象名

    $source = '/'.$src_bucket.'/'.$src_key;
    $uploader = new Aws\S3\MultiPartCopy($this->s3Client, $source, [
        'bucket' => $dst_bucket,
        'key'    => $dst_key,
        'concurrency' => 5,          // 设置上传分片 UploadPart 操作的最大并行数量，默认为5.
        'part_size' => 5242880,      // 设置分片大小，默认为5M.
        'acl' => 'public-read',      // 设置ACL，参考值private | public-read
        'before_initiate' => function(\Aws\Command $command)
        {
            $command['ContentType'] = 'text/json';      // 设置content-type
        },
    ]);

    try {
        $result = $uploader->upload();
        echo "Upload complete: {$result['ObjectURL']}" . "\n";
    } catch (Aws\Exception\MultipartUploadException $e) {
        echo $e->getMessage() . "\n";
    }
}
```

- 关于Content-Type的配置

Content-Type用于标识文件的资源类型，比如`image/png`， `image/jpg`是图片类型，`video/mpeg`， `video/mp4`是视频类型，`text/plain`， `text/html`是文本类型， 浏览器针对不同的Content-Type会有不同的操作，比如图片类型可以预览，视频类型可以播放，文本类型可以直接打开。`application/octet-stream`类型会直接打开下载窗口。

有些用户反馈图片和视频无法预览的问题，主要就是Content-Type没有正确设置导致的；Content-Type参数需要用户主动设置，默认是`application/octet-stream`。在php sdk中，可以根据对象key值后缀扩展名来决定文件的Content-Type，可参考 mime.php 。

请求参数

参数	类型	说明
bucket	string	桶名
key	string	对象名
acl	string	private，public-read，public-read-write
concurrency	int	并发数
part_size	int	分片大小，默认5MB
before_initiate	函数	用于设置content-type

初始化分片上传任务

功能说明

分片上传操作可以将超过5GB的大文件分割后上传，分片上传对象首先需要发起分片上传请求获取一个upload id。

代码示例

```
// createMultipartUpload
$result = $this->s3Client->createMultipartUpload([
    'Bucket' => '<your-bucket-name>',
    'Key'    => '<your-object-key>',
    //'ACL'    => 'public-read',
]);
$uploadId = $result['UploadId'];
```

请求参数

createMultipartUpload 可设置的参数如下：

参数	类型	说明	是否必要
bucket	String	桶名称	是
key	String	对象的key	是
ACL	String	配置上传对象的预定义的标准ACL信息，详细说明见 设置对象访问权限 一节	否

返回结果

返回的属性如下：

参数	类型	说明
Bucket	string	执行分片上传的桶的名称
Key	string	本次分片上传对象的名称
UploadId	string	本次生成分片上传任务的id

上传分片

功能说明

初始化分片上传任务后，指定分片上传任务的id可以上传分片数据，可以将大文件分割成分片后上传，除了最后一个分片，每个分片的数据大小为5MB~5GB，每个分片上传任务最多上传10000个分片。

代码示例

```
$file_Path = '<your-file-path>';
$bucket = '<your-bucket-name>';
$objectName = '<your-object-key>';
// uploadPart
$file = fopen($file_Path, 'r');
$partNumber = 1;
while (!feof($file)) {
    // 创建分片复制请求
    $result = $this->s3Client->uploadPart([
        'Bucket'     => $bucket,
        'Key'        => $objectName,
        'UploadId'   => $uploadId,                    //uploadId从
createMultipartUpload返回值获取
        'PartNumber' => $partNumber,                  //设置分片号
        'Body'       => fread($file, 5 * 1024 * 1024), //读取文件分片，分片大小为5M
    ]);
    $parts['Parts'][$partNumber] = [
        // 记录ETag
        'PartNumber' => $partNumber,
        'ETag' => $result['ETag'],
    ];
    echo "Uploading part {$partNumber}" . "\n";
    $partNumber++;
}
fclose($file);
```

请求参数

uploadPart 可设置的参数如下：

参数	类型	说明	是否必要
Bucket	string	执行分片上传的桶的名称	是
Key	string	对象的key	是
Body	string	分片的数据	是

参数	类型	说明	是否必要
PartNumber	int	说明当前数据在文件中所属的分片，大于等于1，小于等于10000	是
UploadId	string	通过 CreateMultipartUpload 操作获取的UploadId，与一个分片上传的对象对应	是

返回结果

参数	类型	说明
ETag	string	本次上传分片对应的Entity Tag

合并分片

功能说明

合并指定分片上传任务id对应任务中已上传的对象分片，使之成为一个完整的文件。

代码示例

```
// completeMultipartUpload
$result = $this->s3Client->completeMultipartUpload([
    'Bucket'   => '<your-bucket-name>',
    'Key'      => '<your-object-key>',
    'UploadId' => '<your-upload-id>',
    'MultipartUpload' => $parts,
]);
echo $result;
echo "Upload success";
```

请求参数

completeMultipartUpload可设置的参数如下：

参数	类型	说明	是否必要
Bucket	string	执行分片上传的桶的名称	是
Key	string	对象的key	是
MultipartUpload	string array	每个已上传的分片的PartNumber和对应的ETag，生成方式可查看 分片上传-上传分片 一节的代码示例	是
UploadId	string	通过CreateMultipartUpload操作获取的UploadId，与一个对象的分片上传对应	是

返回结果

参数	类型	说明
Bucket	String	执行分片上传的桶的名称
Key	String	对象的key
Etag	String	本次上传对象后对应的Entity Tag
Location	String	合并生成对象的URI信息
VersionId	String	上传对象后相应的版本ID

列举分片上传任务

功能说明

列举分片上传操作可以列出一个桶中正在进行的分片上传，这些分片上传的请求已经发起，但是还没完成或者被中止。listMultipartUploads 操作可以通过指定maxUploads参数来设置返回分片上传信息的数量，maxUploads参数的最大值和默认值均为1000。如果返回结果中的isTruncated字段为true，表示还有符合条件的分片上传信息没有列出，可以通过设置请求中的keyMarker和uploadIdMarker参数，来列出符合筛选条件的正在上传的分片信息。

代码示例

```
public function ListMultipartUploads()
{
    $result = $this->s3Client->listMultipartUploads([
        'Bucket' => '<your-bucket-name>',
    ]);
    echo $result;
}
```

如果list大于1000，则可以使用 getPaginator 接口列举所有分片上传任务。列举所有分片上传任务示例代码如下：

```
public function ListMultipartUploads2()
{
    try {
        $results = $this->s3Client->getPaginator('ListMultipartUploads', [
            'Bucket' => '<your-bucket-name>',
        ]);
        foreach ($results as $result) {
            foreach ($result['Uploads'] as $upload) {
                echo 'object key: ' . $upload['Key'] . "\n";
                echo 'uploadId: ' . $upload['UploadId'] . "\n";
            }
        }
    } catch (S3Exception $e) {
        echo $e->getMessage() . "\n";
    }
}
```

请求参数

listMultipartUploads 可设置的参数如下：

参数	类型	说明	是否必要
Bucket	string	执行本操作的桶名称	是

返回结果

参数	类型	说明
Bucket	string	执行本操作的桶名称。
Uploads	upload array	包含了零个或多个已初始化的上传分片信息的数组。数组中的每一项包含了分片初始化时间、分片上传操作发起者、对象key、对象拥有者、存储类型和uploadId等息

列举已上传的分片

功能说明

列举已上传分片操作可以列出一个分片上传操作中已经上传完毕但是还未合并的分片信息。请求中需要提供object key和 upload id，返回的结果最多包含1000个已上传的分片信息，默认返回1000个，可以通过设置maxParts参数的值指定返回结果中分片信息的数量。如果已上传的分片信息的数量多于1000个，则返回结果中的isTruncated字段为true，可用通过设置partNumberMarker参数获取partNumber大于该参数的分片信息。

代码示例

```
public function ListParts()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $uploadId = '<your-upload-id>';
    $result = $this->s3Client->listParts([
        'Bucket' => $bucket,
        'Key' => $objectName,
        'UploadId' => $uploadId,
    ]);
    echo $result;
}
```

如果 list 大于1000，则可以使用 getPaginator 接口列举所有分片。列举所有分片示例代码如下：

```
public function ListParts2()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $uploadId = '<your-upload-id>';
    try {
        $results = $this->s3Client->getPaginator('ListParts', [
            'Bucket' => $bucket,
            'Key' => $objectName,
            'UploadId' => $uploadId,
```

```
        ]);
        foreach ($results as $result) {
            foreach ($result['Parts'] as $part) {
                echo 'part number:' . $part['PartNumber'] . "\n";
                echo 'ETag:' . $part['ETag'] . "\n";
                echo 'size' . $part['Size'] . "\n";
            }
        }
    } catch (S3Exception $e) {
        echo $e->getMessage() . "\n";
    }
}
```

请求参数

ListPartsRequest 可设置的参数如下：

参数	类型	说明	是否必要
Bucket	string	执行本操作的桶名称	是
Key	string	对象的key	是
UploadId	string	需要查询分片信息的uploadid	是

返回结果

参数	类型	说明
Bucket	string	执行本操作的桶名称
Key	string	本次分片上传对象的名称
Parts	Part array	包含了已上传分片信息的数组，数组中的每一项包含了该分片的Entity tag、最后修改时间、PartNumber和大小等信息
UploadId	string	需要查询的分片上传操作Id

复制分片

功能说明

复制分片操作可以从一个已存在的对象中复制指定分片的数据。您可以使用 uploadPartCopy 复制分片。在复制分片前，需要使用 createMultipartUpload 接口获取一个upload id，在完成复制和上传分片操作之后，需要使用 completeMultipartUpload 操作组装分片成为一个对象。当复制的对象大小超过5GB，必须使用复制分片操作完成对象的复制。除了最后一个分片外，每个复制分片的大小范围是[5MB，5GB]。

代码示例

```
// copyPart
$desBucket = '<your-bucket-name>';   //目标桶
$desKeyName = '<your-object-key>';   //目标对象名
$srcBucket = '<source-bucket-name>'; //从此桶复制
```

```php
$srcKeyName = '<source-object-key>'; //复制的对象名
$result = $this->s3Client->uploadPartCopy([
    'Bucket'     => $desBucket,
    'Key'        => $desKeyName,
    'CopySource' => '/' . $srcBucket . '/' . $srcKeyName,
    'UploadId'   => $uploadId,                    //uploadId从
createMultipartUpload返回值获取
    'PartNumber' => $partNumber,                  //设置分片号
    'CopySourceRange' => 'bytes=' . $firstByte . '-' . $lastByte,          //复制
文件分片的数据范围
]);
echo $result;
$parts['Parts'][$partNumber] = [
    // 记录ETag
    'PartNumber' => $partNumber,
    'ETag' => $result['CopyPartResult']['ETag'],
];
echo "Uploading part {$partNumber}" . "\n";
```

请求参数

uploadPartCopy 可设置的参数如下：

参数	类型	说明	是否必要
Bucket	string	目标桶名称	是
Key	string	目标对象key	是
CopySource	string	URL格式的复制对象数据来源，包含了桶名称和对象key的信息，二者之间使用正斜杆（/）分割，versionId可选参数用于指定原对象的版本。例如，"/foo/boo?versionId=11111"表示复制foo桶中的boo对象，其版本id为11111。如果不指定versionId参数，则默认复制当前版本的对象数据	是
CopySourceRange	string	指定本次分片复制的数据范围，必须是"bytes=first-last"的格式，例如"bytes=0-9"表示复制原对象中前10字节的数据，只有当复制的分片大小大于5MB的时候有效	是
PartNumber	int	说明本次分片复制的数据在原对象中所属的部分	是

参数	类型	说明	是否必要
UploadId	string	与本次复制操作相应的分片上传Id	是

返回结果

参数	类型	说明
ETag	string	包含复制分片的Entity Tag

取消分片上传任务

功能说明

取消分片上传任务操作用于终止一个分片上传。当一个分片上传被中止后，不会再有数据通过与之相应的upload id上传，同时已经被上传的分片所占用的空间会被释放。执行取消分片上传任务操作后，正在上传的分片可能会上传成功也可能会被中止，所以必要的情况下需要执行多次取消分片上传任务操作去释放全部上传成功的分片所占用的空间。可以通过执行列举已上传分片操作来确认所有中止分片上传后所有已上传分片的空间是否被被释放。

代码示例

```
public function abortMultipartUpload() {
    //UploadId从createMultipartUpload中获取
    $bucket = '<your-bucket-name>';
    $keyname = '<your-object-key>';
    $uploadId = '<your-upload-id';

    $result = $this->s3Client->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId,
    ]);
    echo $result;
}
```

请求参数

abortMultipartUpload 可设置的参数如下：

参数	类型	说明	是否必要
Bucket	string	执行本操作的桶名称	是
Key	string	分片上传的对象的key	是
UploadId	string	指定需要终止的分片上传的id	是

安全凭证服务(STS)

STS即Secure Token Service 是一种安全凭证服务，可以使用STS来完成对于临时用户的访问授权。对于跨用户短期访问对象存储资源时，可以使用STS服务。这样就不需要透露主账号AK/SK，只需要生成一个短期访问凭证给需要的用户使用即可，避免主账号AK/SK泄露带来的安全风险。

初始化STS服务

```
require '/path/to/autoload.php';
use Aws\Sts\StsClient;
use Aws\Exception\AwsException;
use Aws\Credentials\Credentials;

const endpoint = '<your-endpoint>'; // e.g. http://endpoint or https://endpoint
const access_key = '<your-access-key>';
const secret_key = '<your-secret-key>';

$credentials = new Credentials(access_key, secret_key);

$this->stsClient = new StsClient([
    'region' => 'ctyun',       // region固定填ctyun
    'version' => '2011-06-15', // sts接口版本号，固定填2011-06-15
    'credentials' => $credentials,
    'endpoint' => endpoint,
]);
```

获取临时token

```
public function AssumeRole()
{
    $bucket = '<your-bucket-name>';
    $arn = '<your-role-arn>';

    $roleSessionName = '<your-role-session-name>';
    $roleArn = "arn:aws:iam:::role/$arn";
    $policy = "{\"Version\":\"2012-10-17\",\"Statement\":
{\"Effect\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource\":
[\"arn:aws:s3:::$bucket\",\"arn:aws:s3:::$bucket/*\"]}}";

    try {
        $res = $this->stsClient->assumeRole([
            'Policy' => $policy,
            'RoleArn' => $roleArn,
            'RoleSessionName' => $roleSessionName,
        ]);
        var_dump($res->get('Credentials'));
    } catch (Aws\Sts\Exception\StsException $e) {
        echo "Exception: $e";
    }
}
```

参数	类型	描述	是否必要
RoleArn	String	角色的ARN，在控制台创建角色后可以查看	是
Policy	String	角色的policy，需要是json格式，限制长度1~2048	是
RoleSessionName	String	角色会话名称，此字段为用户自定义，限制长度2~64	是
DurationSeconds	Integer	会话有效期时间，默认为3600s	否

使用临时token

```
public function StsClientTest($credentials, $endpoint, $bucket)
{
    $stsCredentials = new Credentials($credentials['AccessKeyId'],
$credentials['SecretAccessKey'], $credentials['SessionToken']);
    $s3Client = new S3Client([
        'region' => 'ctyun',        // region固定填ctyun
        'version' => '2006-03-01',  // s3接口版本号，固定填2006-03-01
        'credentials' => $stsCredentials,
        'endpoint' => $endpoint,
    ]);
    try {
        $res = $s3Client->listObjects([
            'Bucket' => $bucket,
        ]);
        var_dump($res->get('Contents'));
    } catch (Aws\S3\Exception\S3Exception $e) {
        echo "Exception: $e";
    }
}
```