天翼云媒体存储

PHP SDK 使用指导书

2024-10-18

天翼云科技有限公司

目 录

1. SDK 安装

1.1. 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。

1.2. 安装方式

1.3. 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');
}
```

1.4. 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
```

2. 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	是

参数	说明	是否必要
	指定 http 或 https 前缀	

3. 桶相关接口

3.1. 创建桶

3.1.1. 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。

3.1.2. 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";
    }
}
```

3.1.3. 请求参数

参数	类型	说明	是否必要
Bucket	String	桶名称	是

3.2. 获取桶列表

3.2.1. 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。

3.2.2. 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";
    }
}
```

3.2.3. 返回结果

参数	类型	说明
Buckets	Bucket array	桶信息列表

3.3. 判断桶是否存在

3.3.1. 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。

3.3.2. 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";
    }
}
```

3.3.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.3.4. 返回结果

参数	类型	说明
exist	bool	桶是否存在

3.4. 删除桶

3.4.1. 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）。

3.4.2. 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";
    }
}
```

3.4.3. 请求参数

参数	类型	说明	是否必要
Bucket	String	桶名称	是

3.5. 设置桶访问权限

3.5.1. 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。

3.5.2. 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-re
ad-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/g
lobal/AllUsers',
                        ],
                        'Permission' => 'READ',
                    ],
                    // ...
                ],
            ],
        ]);
        echo "Succeed in setting bucket ACL.\n";
    } catch (AwsException $e) {
```

```
        echo $e->getMessage();
        echo "\n";
    }
}
```

3.5.3. 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

3.6. 获取桶访问权限

3.6.1. 功能说明

您可以使用 getBucketAcl 接口获取桶的访问权限。

3.6.2. 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";
    }
}
```

3.6.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.6.4. 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

3.7. 设置桶策略

3.7.1. 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
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	可选，NotResource 与 Resource 选其一

元素	描述	是否必要
	值同 Resource	
Condition	可选关键字，本条 statement 生效的条件	可选

3.7.2. 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";
}
```

3.7.3. 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

3.8. 获取桶策略

3.8.1. 功能说明

您可以使用 getBucketPolicy 接口获取桶策略。

3.8.2. 代码示例

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

3.8.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.8.4. 返回结果

参数	类型	说明

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 。

3.9. 删除桶策略

3.9.1. 功能说明

您可以使用 deleteBucketPolicy 接口删除桶策略。

3.9.2. 代码示例

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

3.9.3. 请求参数

参数	类型	说明	是否必要
Bucket	String	桶名称	是

3.10. 设置桶生命周期配置

3.10.1. 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。

3.10.2. 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";
}
```

3.10.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是
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

参数	类型	说明	是否必要
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

3.11. 获取桶生命周期配置

3.11.1. 功能说明

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

3.11.2. 代码示例

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

3.11.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.11.4. 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）

参数	类型	说明
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

3.12. 删除桶生命周期配置

3.12.1. 功能说明

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

3.12.2. 代码示例

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

3.12.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.13. 设置桶跨域访问配置

3.13.1. 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。

3.13.2. 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";
}
```

3.13.3. 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

3.14. 获取桶跨域访问配置

3.14.1. 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。

3.14.2. 代码示例

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

3.14.3. 请求参数

参数	类型	说明	是否必要

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.14.4. 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

3.15. 删除桶跨域访问配置

3.15.1. 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。

3.15.2. 示例代码

```
public function DeleteBucketCors()
{
    $this->s3Client->deleteBucketCors([
```

```
        'Bucket' => '<your-bucket-name>',
    ]);
    echo "deleteBucketCors success\n";
}
```

3.15.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.16. 设置桶版本控制状态

3.16.1. 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。

3.16.2. 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";
}
```

3.16.3. 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

3.17. 获取桶版本控制状态

3.17.1. 功能说明

您可以使用 getBucketVersioning 接口获取版本控制状态。

3.17.2. 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";
    }
}
```

3.17.3. 请求参数

参数	类型	说明	是否必要
Bucket	string	桶名称	是

3.17.4. 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

4. 对象相关接口

4.1. 获取对象列表

4.1.1. 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。

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

4.1.3. 请求参数

参数	类型	说明	是否必须
Bucket	string	设置桶名称	是

4.1.4. 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

4.2. 上传对象

4.2.1. 功能说明

您可以使用 putObject 接口上传对象。

4.2.2. 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:

```
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";
    }
}
```

4.2.3. 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

4.2.4. 返回结果

参数	类型	说明
ETag	string	对象的唯一标签

4.3. 下载对象

4.3.1. 功能说明

您可以使用 getObject 接口下载对象。

4.3.2. 代码示例

```
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";
    }
}
```

4.3.3. 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	否

参数	类型	说明	是否必要
		304 代码	
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。

4.3.4. 返回结果

参数	类型	说明
Body	string	对象数据内容

4.4. 复制对象

4.4.1. 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。

4.4.2. 代码示例

复制一个对象

```php
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。Aws3，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。

4.4.3. 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	是

参数	类型	说明	是否必要
		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	

4.4.4. 返回结果

参数	类型	说明
ETag	string	对象的唯一标签

4.5. 删除对象

4.5.1. 功能说明

您可以使用 deleteObject 接口删除单个对象。

4.5.2. 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";
```

```
    }
}
```

4.5.3. 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

4.6. 批量删除对象

4.6.1. 功能说明

您可以使用 deleteObjects 接口批量删除多个对象。

4.6.2. 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";
    }
}
```

4.6.3. 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

4.7. 获取对象元数据

4.7.1. 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。

4.7.2. 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";
    }
}
```

4.7.3. 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

4.7.4. 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

4.8. 设置对象访问权限

4.8.1. 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。

4.8.2. 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";
}
}
```

4.8.3. 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

参数	类型	说明	是否必要
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

4.9. 获取对象访问权限

4.9.1. 功能说明

您可以使用 getObjectAcl 接口获取对象访问的权限。

4.9.2. 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";
    }
}
```

4.9.3. 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

4.9.4. 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

4.10. 获取对象标签

4.10.1. 功能说明

您可以使用 GetObjectTagging 接口获取对象标签。

4.10.2. 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";
}
```

4.10.3. 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

4.10.4. 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

4.11. 删除对象标签

4.11.1. 功能说明

您可以使用 deleteObjectTagging 接口删除对象标签。

4.11.2. 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";
}
```

4.11.3. 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

4.12. 设置对象标签

4.12.1. 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。

4.12.2. 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";
}
```

4.12.3. 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
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

4.13. 生成预签名 URL

4.13.1. 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。

4.13.2. 代码示例

```
public function generateGetobjectPresignedUrl()
{
    $bucket = '<your-bucket-name>';
    $objectName = '<your-object-key>';
    $cmd = $this->s3Client->getCommand('GetObject', [
        'Bucket' => $bucket,
        'Key' => $objectName,
    ]);
    $request = $this->s3Client->createPresignedRequest($cmd, '+5 minu
tes'); // 5 分钟之后过期
    $presignedUrl = (string)$request->getUri();
    echo "generateGetobjectPresignedUrl success " . $presignedUrl . "
```

```
\n";
}
```

4.13.3. 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

4.14. 上传对象-Post 上传

4.14.1. 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。

4.14.2. 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);
}
```

4.14.3. 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

4.14.4. 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>
```

4.15. 获取多版本对象列表

4.15.1. 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。

4.15.2. 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";
    }
}
```

4.15.3. 请求参数

参数	类型	说明	是否必须
Bucket	string	设置桶名称	是

4.15.4. 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

5. 分片上传接口

5.1. 融合接口

5.1.1. 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。Aws3 接口提供了简洁的分片上传方式，Aws3 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。

5.1.2. 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 。

5.1.3. 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

5.2. 分片上传-初始化分片上传任务

5.2.1. 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。

5.2.2. 代码示例

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

```
]);
$uploadId = $result['UploadId'];
```

5.2.3. 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

5.2.4. 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

5.3. 分片上传-上传分片

5.3.1. 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。

5.3.2. 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 crea
teMultipartUpload 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);
```

5.3.3. 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
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

5.3.4. 返回结果

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

5.4. 分片上传-合并分片

5.4.1. 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。

5.4.2. 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";
```

5.4.3. 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

5.4.4. 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

参数	类型	说明
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

5.5. 分片上传-列举分片上传任务

5.5.1. 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。

5.5.2. 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";
    }
}
```

5.5.3. 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

5.5.4. 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

参数	类型	说明
		者、对象 key、对象拥有者、存储类型和 uploadId 等息

5.6. 分片上传-列举已上传的分片

5.6.1. 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。

5.6.2. 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:

```php
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";
    }
}
```

5.6.3. 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

参数	类型	说明	是否必要
Key	string	对象的 key	是
UploadId	string	需要查询分片信息的 uploadid	是

5.6.4. 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

5.7. 分片上传-复制分片

5.7.1. 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]。

5.7.2. 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
$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createMu
ltipartUpload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";
```

5.7.3. 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	是

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

5.7.4. 返回结果

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

5.8. 分片上传-取消分片上传任务

5.8.1. 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。

5.8.2. 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;
}
```

5.8.3. 请求参数

abortMultipartUpload 可设置的参数如下：

参数	类型	说明	是否必要

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

6. 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。

6.1. 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,
]);
```

6.2. 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

参数	类型	描述	是否必要
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

6.3. 使用临时 token

```
public function StsClientTest($credentials, $endpoint, $bucket)
{
    $stsCredentials = new Credentials($credentials['AccessKeyId'], $c
redentials['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";
```

```
    }
}
```