Alibaba Cloud

#象存#

SDK リファレンス

Document Version20200423

目次

1 はじめに.....1
2 Java.....2
2.1 はじめに..... 2
2.2 インストール..... 3
2.3 クイックスタート.....5
2.4 初期化..... 8
2.5 バケット (Bucket)..... 13
2.5.1 バケットのタグ..... 13
2.5.2 ライフサイクルルールの管理.....15
2.5.3 Anti-leech.....18
2.5.4 ログの設定..... 20
2.5.5 静的 Web サイトホスティング.....21
2.5.6 クロスリージョンレプリケーション..... 23
2.5.7 CORS.....26
2.6 オブジェクトのアップロード.....28
2.6.1 概要.....28
2.6.2 簡易アップロード.....29
2.6.3 フォームアップロード..... 31
2.6.4 追加アップロード.....34
2.6.5 再開可能アップロード..... 36
2.6.6 マルチパートアップロード.....37
2.6.7 アップロードの進捗プログレスバー..... 46
2.6.8 アップロードコールバック.....47
2.7 オブジェクトのダウンロード.....48
2.7.1 概要..... 48
2.7.2 ストリーミングダウンロード.....49
2.7.3 ローカルファイルへのダウンロード.....50
2.7.4 部分ダウンロード..... 50
2.7.5 再開可能ダウンロード.....51
2.7.6 条件付きダウンロード.....54
2.7.7 ダウンロードの進捗プログレスバー.....55
2.8 オブジェクトの管理.....57
2.8.1 概要.....57
2.8.2 指定したオブジェクトの存在有無の確認.....57
2.8.3 オブジェクトの ACL の管理..... 58
2.8.4 オブジェクトメタの管理.....59
2.8.5 オブジェクトの一覧表示.....63
2.8.6 オブジェクトの削除..... 73
2.8.7 アーカイブオブジェクトの復元..... 75
2.9 サーバー側暗号化.....76

2.10 データ暗号化.....79
2.11 データのセキュリティ.....79
2.12 画像処理.....81
2.13 例外処理.....88
2.14 よくある質問.....91
3 Python.....100
3.1 インストール.....100
3.2 クイックスタート.....103
3.3 初期化.....105
3.4 オブジェクトのアップロード.....107
3.4.1 概要.....107
3.4.2 簡易アップロード.....108
3.4.3 追加アップロード.....110
3.4.4 再開可能アップロード.....111
3.4.5 マルチパートアップロード.....113
3.4.6 アップロードの進捗プログレスバー.....114
3.4.7 アップロードコールバック.....114
3.5 オブジェクトのダウンロード.....115
3.5.1 概要.....115
3.5.2 ストリーミングダウンロード.....116
3.5.3 ローカルファイルへのダウンロード.....117
3.5.4 部分ダウンロード.....117
3.5.5 再開可能ダウンロード.....118
3.5.6 ダウンロードの進捗プログレスバー.....120
3.6 オブジェクトの管理.....121
3.6.1 オブジェクトメタの管理.....121
3.7 許可されたアクセス.....123
3.8 Python SDK のログ機能の有効化.....125
3.9 漢字と時刻.....126
4 Go.....130
4.1 はじめに.....130
4.2 インストール.....131
4.3 クイックスタート.....132
4.4 初期化.....136
4.5 オブジェクトのアップロード.....138
4.5.1 概要.....138
4.5.2 簡易アップロード.....139
4.5.3 追加アップロード.....141
4.5.4 再開可能アップロード.....143
4.5.5 マルチパートアップロード.....145
4.5.6 プログレスバー.....151
4.6 オブジェクトのダウンロード.....152
4.6.1 概要.....152

4.6.2 ストリーミングダウンロード..... 152
4.6.3 ローカルファイルへのダウンロード..... 155
4.6.4 再開可能なダウンロード.....155
4.6.5 条件付きダウンロード..... 158
4.6.6 圧縮オブジェクトのダウンロード.....160
4.6.7 プログレスバー..... 160
4.7 オブジェクトの管理.....161
4.7.1 概要.....161
4.7.2 指定したオブジェクトの存在有無の確認..... 162
4.7.3 オブジェクトの ACL の管理.....162
4.7.4 オブジェクトメタの管理..... 164
4.7.5 オブジェクトの一覧表示..... 167
4.7.6 オブジェクトの削除.....173
4.7.7 オブジェクトのコピー.....175
4.7.8 アーカイブオブジェクトの復元.....179
4.7.9 シンボリックリンクの管理..... 181
4.8 画像処理..... 183
4.9 エラー処理.....191
5 C++..... 196
6 PHP..... 197
6.1 はじめに.....197
6.2 インストール.....197
6.3 クイックスタート..... 200
6.4 初期化..... 203
6.5 オブジェクトのアップロード.....206
6.5.1 概要.....207
6.5.2 簡易アップロード.....207
6.5.3 追加アップロード.....208
6.5.4 マルチパートアップロード.....210
6.5.5 コールバックのアップロード..... 217
6.6 オブジェクトのダウンロード.....220
6.6.1 概要.....220
6.6.2 ローカルファイルへのダウンロード..... 220
6.6.3 ローカルメモリへのダウンロード..... 221
6.6.4 部分ダウンロード.....222
6.6.5 条件付きダウンロード.....222
6.7 オブジェクトの管理.....224
6.7.1 概要.....224
6.7.2 指定したオブジェクトの存在有無の確認..... 225
6.7.3 オブジェクの ACL の管理..... 225
6.7.4 オブジェクトメタの管理..... 227
6.7.5 オブジェクトの一覧表示..... 230
6.7.6 オブジェクトの削除..... 232

6.7.7 オブジェクトのコピー....233
6.7.8 アーカイブオブジェクトの復元....235
6.7.9 シンボリックリンクの管理....236
6.7.10 MD5 を有効にする....238
6.8 画像処理....239
6.9 例外処理....245
7 C....248
7.1 はじめに....248
7.2 インストール....249
7.3 クイックスタート....253
7.4 初期化....260
7.5 オブジェクトのアップロード....263
7.5.1 概要....263
7.5.2 簡易アップロード....263
7.5.3 追加アップロード....266
7.5.4 再開可能アップロード....269
7.5.5 マルチパートアップロード....270
7.5.6 アップロードの進捗プログレスバー....275
7.5.7 アップロードコールバック....276
7.6 オブジェクトのダウンロード....278
7.6.1 概要....278
7.6.2 ローカルファイルへのオブジェクトのダウンロード....278
7.6.3 ローカルメモリへのダウンロード....280
7.6.4 部分ダウンロード....281
7.6.5 再開可能ダウンロード....283
7.6.6 ダウンロードの進捗プログレスバー....285
7.7 オブジェクトの管理....286
7.7.1 概要....286
7.7.2 オブジェクトのACLの管理....287
7.7.3 オブジェクトメタの管理....289
7.7.4 オブジェクトの一覧表示....290
7.7.5 オブジェクトの削除....297
7.7.6 オブジェクトのコピー....301
7.7.7 アーカイブオブジェクトの復元....305
7.7.8 シンボリックリンクの管理....307
7.8 許可されたアクセス....309
7.9 画像処理....314
7.10 エラー処理....323
7.11 FAQ....325
8 . NET....326
8.1 インストール....326
8.2 クイックスタート....328
8.3 初期化....331

8.4 オブジェクトのアップロード.....334
8.4.1 概要.....334
8.4.2 簡易アップロード.....335
8.4.3 追加アップロード.....338
8.4.4 再開可能アップロード.....339
8.4.5 マルチパートアップロード.....340
8.4.6 アップロードの進捗プログレスバー.....347
8.4.7 アップロードコールバック.....348
8.5 オブジェクトのダウンロード.....350
8.5.1 概要.....350
8.5.2 ストリーミングダウンロード.....350
8.5.3 部分ダウンロード.....351
8.5.4 再開可能ダウンロード.....352
8.5.5 ダウンロード進捗プログレスバー.....353
8.6 オブジェクトの管理.....354
8.6.1 概要.....354
8.6.2 指定したオブジェクトの存在有無の確認.....355
8.6.3 オブジェクトの ACL の管理.....355
8.6.4 オブジェクトメタの管理.....357
8.6.5 オブジェクトの一覧表示.....358
8.6.6 オブジェクトの削除.....363
8.6.7 オブジェクトのコピー.....365
8.6.8 アーカイブオブジェクトの復元.....368
8.6.9 シンボリックリンクの管理.....369
8.7 画像処理.....370
8.8 例外処理.....377

1 はじめに

OSS SDK を使用する前に、次の条件を満たしている必要があります。

- Alibaba Cloud OSS サービスの内容を理解し、Alibaba Cloud OSS を有効化する
- AccessKey の作成

本ページでは、以下の言語における OSS SDKのインストール方法、開発方法、およびトラブルシューティング方法について説明しているページを一覧表示し、パラメーターおよび例を提供します。

言語	リファレンス
Java	Java SDK references
Python	Python SDK references
Android	Android SDK references
iOS	iOS SDK references
.NET	.NET SDK references
Node.js	Node.js SDK references
Browser.js	Browser.js SDK references
PHP	PHP SDK references
C	C SDK references
Ruby	Ruby SDK references
Go	Go SDK references
Media-C	Media-C SDK references

2 Java

2.1 はじめに

本ドキュメントは、OSS Java SDK 2.8.3 に基づき作成されています。

互換性

- SDK バージョン 2.x.x
  - API: 互換性があります。
  - 名前空間: 互換性があります。
- SDK バージョン 1.0.x
  - API: 互換性があります。
  - 名前空間: 互換性があります。 バージョン 1.0.x の Table Store のコードは、バージョン 2.0.0 から削除されています。 `com.aliyun.openservices.*` および `com.aliyun.openservices.oss.*` というパッケージ名は、`com.aliyun.oss.*` へ置き換えられています。

SDK ソースコードと API ドキュメント

OSS Java SDK ソースコードについては、『GitHub』をご参照ください。 詳細については、『OSS Java SDK API ドキュメント』をご参照ください。

サンプルコード

OSS SDK for Java は、参考用または使用目的でさまざまなサンプルコードを提供しています。 次の表にサンプルコードの内容を示します。

サンプルファイル	コンテンツ
GetStartedSample.java	クイックスタート
CreateFolderSample.java	簡易アップロードにフォルダを作成
AppendObjectSample.java	アップロードの追加
UploadSample.java	再開可能アップロード
MultipartUploadSample.java	マルチパーツアップロード
GetProgressSample.java	アップロードプログレスバー
CallbackSample.java	アップロードコールバック
CRCSample.java	ファイルアップロード用 CRC

サンプルファイル	コンテンツ
SimpleGetObjectSample.java	オブジェクトのダウンロード
DownloadSample.java	再開可能ダウンロード
ObjectMetaSample.java	オブジェクトメタの管理
ListObjectsSample.java	オブジェクトの一覧
UploadPartCopySample.java	オブジェクトのコピー
DeleteObjectsSample.java	オブジェクトの削除
BucketOperationsSample.java	バケットの管理: 許可されたアクセス、ライフサイクル管理、ログファイルアクセス、アンチリーチ、CORS
ImageSample.java	画像処理
PostObjectSample.java	PostObject は、Java SDK から独立しています。
ConcurrentGetObjectSample.java	複数のオブジェクトの同時ダウンロード。 再開可能ダウンロードを推奨します。

2.2 インストール

このトピックでは、OSS Java SDK のインストール方法について説明します。

準備

- 環境要件

  Java 1.6 以降のバージョンの使用が必要です。

- Java バージョンの確認

  `java -version` を実行し、Java バージョンを確認します。

SDK のダウンロード

- 直接ダウンロード
- GitHub からダウンロード
- 以前のバージョンのダウンロード

SDK のインストール

- メソッド 1: Maven に依存関係を追加 (推奨)

  Maven で OSS Java SDK を使用するには、pom.xml ファイルに対応する依存関係を追加するだけで済みます。 この例では、OSS Java SDK 2.8.3 を使用しています。 以下の内容を <dependencies> に追加します。

```
<dependency>
   <groupId>com.aliyun.oss</groupId>
   <artifactId>log4j-core</artifactId>
   <version>5.1.6</version>
</dependency>
```

- メソッド 2: Eclipse に jar ファイルをインポート

  例として、OSS Java SDK 2.8.3 を取り上げます。 jar ファイルを Eclipse にインポートするには、次の手順に従います。

  1. Java SDK をダウンロードします。
  2. Java SDK を解凍します。
  3. aliyun-sdk-oss-2.8.3.jar および lib フォルダーからプロジェクトを含むすべてのファイルをコピーします。
  4. Eclipse でプロジェクト名を右クリックします。 プロパティ > **Java** ビルドパス > **jar** を追加の順で選択します。
  5. 手順 3 でコピーした jar ファイルをすべて選択し、それらをライブラリにインポートします。

- メソッド 3: jar ファイルを Intellij IDEA へインポート

  例として、OSS Java SDK 2.8.3 を使用します。 jar ファイルを Intellij IDEA にインポートするには、次の手順に従います。

  1. Java SDK をダウンロードします。
  2. Java SDK を解凍します。
  3. aliyun-sdk-oss-2.8.3.jar および lib フォルダーからプロジェクトを含むすべてのファイルをコピーします。
  4. Intellij IDEA で、プロジェクト名を右クリックします。 ファイル > プロジェクト構造 > モジュール > 依存関係 > **+** > **jar** または ディレクトリの順で選択します。
  5. 手順 3 でコピーしたすべての jar ファイルを選択し、それらを外部ライブラリにインポートします。

2.3 クイックスタート

このトピックでは、OSS Java SDK を使用してバケットの作成、オブジェクトのアップロード、オブジェクトのダウンロードなどのルーチン操作を実行する方法について説明します。

サンプルプロジェクト

OSS Java SDK は、 Maven ベースと Ant ベースのサンプルプロジェクトを提供しています。 サンプルプロジェクトをローカルデバイスでコンパイルして実行したり、サンプルプロジェクトに基づいて独自のアプリケーションを開発することが可能です。 プロジェクトのコンパイルおよび実行方法については、プロジェクトディレクトリの「README.md」をご参照ください。

- Maven サンプルプロジェクト: aliyun-oss-java-sdk-demo-mvn.zip
- Ant サンプルプロジェクト: aliyun-oss-java-sdk-demo-ant.zip

バケットの作成

バケットは OSS のグローバルネームスペースです。 ファイルを格納するデータコンテナーのようなものです。 次のコードを実行してバケットを作成します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Create a bucket.
ossClient.createBucket(bucketName);

// Close your OSSClient instance.
ossClient.shutdown();
```

バケットの命名ルールの詳細については、「基本概念」の命名ルールをご参照ください。 バケットの作成の詳細については、「 バケットの管理」をご参照ください。

エンドポイントの詳細については、「リージョンおよびエンドポイント」をご参照ください。

オブジェクトのアップロード

次のコードを実行してファイルを OSS にオブジェクトをアップロードします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload content to the specified bucket and save it as the specified object name.
String content = "Hello OSS";
ossClient.putObject(bucketName, objectName, new ByteArrayInputStream(content.getBytes()));

// Close your OSSClient instance.
ossClient.shutdown();
```

詳しくは、「オブジェクトのアップロード」をご参照ください。

オブジェクトのダウンロード

次のコードを実行して、オブジェクトのコンテンツを取り込みます。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Call ossClient.getObject to return an OSSObject instance. The OSSObject instance contains the object content and Object Meta.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
// Call ossObject.getObjectContent to obtain object InputStream. Read the InputStream to obtain object content.
InputStream content = ossObject.getObjectContent();
if (content ! = null) {
    BufferedReader reader = new BufferedReader(new InputStreamReader(content));
    while (true) {
        String line = reader.readLine();
        if (line == null) break;
        System.out.println("\n" + line);
    }
    // If you do not close the reader after the data is read, connection leaks may occur. Consequently, no available connections are left and an exception occurs.
    content.close();
}

// Close your OSSClient instance.
```

```
ossClient.shutdown();
```

詳しくは、「オブジェクトのダウンロード」をご参照ください。

オブジェクトの一覧表示

次のコードを実行して、バケット内のオブジェクトを一覧で表示します。 デフォルトでは最大100 個のオブジェクトをリストできます。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Use ossClient.listObjects to return an ObjectListing instance. The ObjectListing instance contains the response from listObject.
ObjectListing objectListing = ossClient.listObjects(bucketName);
// Use objectListing.getObjectSummaries to obtain the descriptions of all objects.
for (OSSObjectSummary objectSummary : objectListing.getObjectSummaries()) {
    System.out.println(" - " + objectSummary.getKey() + "  " +
            "(size = " + objectSummary.getSize() + ")");
}

// Close your OSSClient instance.
ossClient.shutdown();
```

詳しくは、「オブジェクトの管理」の「オブジェクトの一覧表示」をご参照ください。

オブジェクトの削除

オブジェクトを削除するための完全なコードは、「GitHub」をご参照ください。

指定したオブジェクトを削除するには、次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Delete objects.
```

```
ossClient.deleteObject(bucketName, objectName);

// Close your OSSClient instance.
ossClient.shutdown();
```

詳しくは、「オブジェクトの管理」の「オブジェクトの削除」をご参照ください。

2.4 初期化

OSSClient は、バケットやオブジェクトなどの OSS リソースを管理するための OSS Java クライアントとして機能します。 Java SDK で OSS リクエストを開始するには、まず OSSClient インスタンスを開始してからClientConfiguration のデフォルト設定項目を変更する必要があります。

OSSClientインスタンスの作成

OSSClient インスタンスを作成するには、エンドポイントを指定する必要があります。 エンドポイントの詳細については、「リージョンおよびエンドポイント」および「カスタムドメイン名のバインド方法」をご参照ください。

- OSS ドメインを使用して OSSClient インスタンスを作成

  次のコードを実行して、OSS によって割り当てられたドメインを使用して OSSClient インスタンスを作成します

  ```
  // This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
  String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
  // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
  String accessKeyId = "<yourAccessKeyId>";
  String accessKeySecret = "<yourAccessKeySecret>";

  // Create an OSSClient instance.
  OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

  // Close your OSSClient instance.
  ossClient.shutdown();
  ```

  プロジェクトには1つ以上の OSS クライアントを使用できます。 つまり、複数の OS クライアントを同時に使用することができます。

- カスタムドメイン (CNAME) を使用して OSSClient インスタンスを作成

  次のコードを実行して、CNAME で OSSClient インスタンスを作成します。

  ```
  String endpoint = "<yourEndpoint>";
  // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
  ```

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create a ClientConfiguration instance. Modify parameters as required.
ClientConfiguration conf = new ClientConfiguration();
// Enable CNAME. CNAME indicates a custom domain bound to a bucket.
conf.setSupportCname(true);

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

注：

CNAME が使用されている場合、ossClient.listBuckets メソッドは使用できません。

- Apsara Stack の OSSClient インスタンスを作成

  次のコードを実行して、Apsara Stack の OSSClient インスタンスを作成します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create a ClientConfiguration instance. Modify parameters as required.
ClientConfiguration conf = new ClientConfiguration();
// Disable CNAME.
conf.setSupportCname(false);

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

- IP アドレスを使用して OSSClient インスタンスを作成

  次のコードを実行して、IP アドレスを指定して OSSClient インスタンスを作成します。

```
// In some special cases, use the IP address as an endpoint. Specify the actual IP
address based on your requirements.
String endpoint = "http://10.10.10.10";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create ClientConfiguration.
ClientConfiguration conf = new ClientConfiguration();
```

```
// Enable OSS access with the level-2 domain. Access with the level-2 domain is disabled by default. You need to configure this function for OSS Java SDK versions 2.1.2 or earlier because OSS Java SDK versions later than 2.1.2 automatically detect IP addresses.
conf.setSLDEnabled(true);

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

- STS を使用して OSSClient インスタンスを作成

  以下のコードを実行して、STS (Security Token Service) を使用して OSSClient インスタンスを作成します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret, securityToken);

// Close your OSSClient instance.
ossClient.shutdown();
```

詳しくは、「RAM および STS の概要」および「許可されたアクセス」をご参照ください。

OSSClient インスタンスの設定

ClientConfiguration は OSSClient の設定クラスです。 ClientConfiguration は、ユーザーエージェント、ホストプロキシ、接続タイムアウト、最大接続数などのパラメーターを設定するために使用されます。 以下のパラメーターを設定できます。

パラメータ	説明	設定方法
MaxConnections	有効にできる HTTP 接続の最大数を指定します。 デフォルト値は 1024 です。	ClientConfiguration.setMaxConnections
SocketTimeout	ソケット層でのデータ転送のタイムアウト時間をミリ秒単位で指定します。 デフォルト値は 50,000 です。	ClientConfiguration.setSocketTimeout

パラメータ	説明	設定方法
ConnectionTimeout	接続が確立されるときのタイムアウト時間をミリ秒単位で指定します。 <p data-spm-anchor-id="a2762.11472859.0.i61.7eca203bijNJDs">デフォルト値は 50,000 秒です。	ClientConfiguration.setConnectionTimeout
ConnectionRequestTimeout	接続プールから接続を取得するためのタイムアウト時間をミリ秒単位で指定します。 デフォルトではタイムアウトは設定されていません。 デフォルトではタイムアウトは設定されていません。	ClientConfiguration.setConnectionRequestTimeout
IdleConnectionTime	アイドル接続タイムアウト時間を指定します。 ミリ秒単位のアイドル接続時間が指定された値を超えると、接続は閉じられます。 デフォルト値は 60,000 です。	ClientConfiguration.setIdleConnectionTime
MaxErrorRetry	リクエストエラーが発生した場合の最大再試行回数を指定します。 デフォルト値は 3 です。	ClientConfiguration.setMaxErrorRetry
SupportCname	CNAME をエンドポイントとして使用できるかどうかを指定します。 デフォルトで CNAME をエンドポイントとして使用できます。	ClientConfiguration.setSupportCname
SLDEnabled	レベル 2 ドメインとのアクセスを有効にするかどうかを指定します。 レベル 2 ドメインでのアクセスは、デフォルトで無効になっています。	ClientConfiguration.setSLDEnabled
Protocol	OSS への接続に使用されるプロトコル（HTTP または HTTPS）を指定します。 デフォルト値は HTTP です。	ClientConfiguration.setProtocol

パラメータ	説明	設定方法
UserAgent	HTTP ヘッダー内のユーザエージェントを指定します。 デフォルト値は “aliyun-sdk-java” です。	ClientConfiguration.setUserAgent
ProxyHost	プロキシホストにアクセスするための IP アドレスを指定します。	ClientConfiguration.setProxyHost
ProxyPort	プロキシホストのポートを指定します。	ClientConfiguration.setProxyPort
ProxyUsername	プロキシホストによって確認されたユーザ名を指定します。	ClientConfiguration.setProxyUsername
ProxyPassword	プロキシホストによって確認されたパスワードを指定します。	ClientConfiguration.setProxyPassword

次のコードを実行して、ClientConfiguration を使用して OSSClient インスタンスのパラメーターを構成します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create ClientConfiguration. ClientConfiguration is a configuration class of OSSClient. You can use ClientConfiguration to configure parameters such as user agents, host proxies, connection timeout, and the maximum number of connections.
ClientConfiguration conf = new ClientConfiguration();

// Configure the maximum HTTP connections allowed by an OSSClient instance. The default value is 1024.
conf.setMaxConnections(200);
// Configure the timeout time in milliseconds for data transmission at SSL. The default value is 50,000.
conf.setSocketTimeout(10000);
conf.setSocketTimeout(10000);
// Configure the timeout time in milliseconds when connections are established. The default value is 50,000.
conf.setConnectionTimeout(10000);
// Configure the timeout time in milliseconds for retrieving connections from the connection pool. This function is disabled by default.
conf.setConnectionRequestTimeout(1000);
// Configure idle connection timeout. If the idle connection time in milliseconds exceeds the specified value, the connection is closed. The default value is 60,000.
conf.setIdleConnectionTime(10000);
```

```
// Configure the maximum number of retry attempts in the case of a request error. The
default value is 3.
conf.setMaxErrorRetry(5);
// Check whether CNAME can be used as an endpoint. You can use CNAME as an
endpoint by default.
conf.setSupportCname(true);
// Check whether access with the level-2 domain is enabled. Access with the level-2
domain is disabled by default.
conf.setSLDEnabled(true);
// Configure the protocol (HTTP or HTTPS) used to connect to OSS. HTTP is used by default
.
conf.setProtocol(Protocol.HTTP);
// Configure the user agent (the user agent in the HTTP header). The default value is "
aliyun-sdk-java."
conf.setUserAgent("aliyun-sdk-java");
// Configure the port for the proxy host.
conf.setProxyHost("<yourProxyHost>");
// Configure the username verified by the proxy host.
conf.setProxyUsername("<yourProxyUserName>");
// Configure the password verified by the proxy host.
conf.setProxyPassword("<yourProxyPassword>");

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

2.5 バケット (Bucket)

2.5.1 バケットのタグ

バケットのタグ機能を使用すると、バケットを分類し、管理できます。 たとえば、特定のタグを持つバケットのみが ListBucket 操作で返されるように指定できます。

- バケットにタグを追加できるのは、バケット所有者と許可された RAM ユーザーのみです。 それ以外の場合、403 Forbidden エラーがエラーコード AccessDenied と共に返されます。
- バケットには最大 20 個のタグ (キーと値のペア) を追加できます。
- キーは、最大 64 バイトです。 タグのキーは null にできません。また、プレフィックスを http://、https://、または Aliyun にすることはできません。
- タグの値は、最大 128 バイトです。 タグの値は null にできます。
- タグのキーと値は UTF-8 でエンコードされている必要があります。
- PutBucketTagging を使用してバケットにタグを追加すると、バケットに追加されている元のタグは完全に上書きされます。

バケットへのタグの追加

次のコードを実行して、タグをバケットに追加できます。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM user account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Adds tags to the bucket.
SetBucketTaggingRequest request = new SetBucketTaggingRequest("<yourBucketName>");
request.setTag("<yourTagkey1>", "<yourTagValue1>");
request.setTag（"<yourTagkey2>"、 "<yourTagValue2>"）;
ossClient.setBucketTagging(request);

// Closes the OSSClient.
ossClient.shutdown();
```

バケットに追加されているタグの取得

次のコードを実行して、バケットに追加されているタグを取得できます。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM user account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Obtains the tags added to the bucket.
TagSet tagSet = ossClient.getBucketTagging(new GenericRequest("<yourBucketName>"));
Map<String, String> tags = tagSet.getAllTags();

// Closes the OSSClient instance.
ossClient.shutdown();
```

特定のタグを持つバケットの一覧表示

次のコードを実行して、特定のタグを持つバケットを一覧表示します。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
```

```
user to access APIs or perform routine operations and maintenance. To create a RAM user
 account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Lists buckets with the specified tag.
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
listBucketsRequest.setTag("<yourTagKey>", "<yourTagValue>");
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket o : bucketList.getBucketList()) {
  System.out.println("list result bucket: " + o.getName());
}

// Closes the OSSClient instance.
ossClient.shutdown();
```

バケットに追加されているタグの削除

次のコードを実行して、バケットに追加されているタグを削除できます。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify an actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM user
 account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Deletes the tags added to the bucket.
ossClient.deleteBucketTagging(new GenericRequest("<yourBucketName>"));

// Closes the OSSClient instance.
ossClient.shutdown();
```

2.5.2 ライフサイクルルールの管理

このトピックでは、ライフサイクルルールを管理する方法について説明します。

OSS を使用して、ストレージコストを削減するためのライフサイクルルールを設定します。 ライフサイクル管理ルールは、期限切れのオブジェクトやフラグメントの自動削除、あるいは期限切れになるオブジェクトのストレージクラスから低頻度アクセスへの変換に使用できます。 各ルールは次のとおりです。

- ルール ID: ルールを識別します。 バケット内の各ルール ID が一意であることを確認します。

- ポリシー: 以下の設定方法を使用してポリシーを設定します。 バケットに設定できるメソッドは 1 つだけです。
  - プレフィックスで設定: この方法により複数のルールを作成できます。 各プレフィックスが一意であることを確認します。
  - バケット全体を設定: この方法で設定できるルールは 1 つだけです。
- 期限切れ: 次の設定方法で設定できます。
  - 日数「N」の設定: オブジェクトの最終変更時刻からの「N」日後、オブジェクトは削除されます。
  - 日付による設定: オブジェクトが作成される前の日付を指定します。 オブジェクトが指定した日付より前に作成された場合、オブジェクトは削除されます。
- ステータス: ライフサイクルルールが有効もしくは無効

uploadPart メソッドによってアップロードされたパーツもライフサイクルルールをサポートします。 オブジェクトの最終変更時刻は、マルチパートアップロードイベントが開始された時刻です。

ライフサイクル管理の詳細については、「オブジェクトライフサイクルの管理」をご参照ください。

ライフサイクルルールを設定

次のコードを実行してライフサイクルルールを設定します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

SetBucketLifecycleRequest request = new SetBucketLifecycleRequest(bucketName);

// Configure the rule ID and object name prefix.
String ruleId0 = "rule0";
String matchPrefix0 = "A0/";
String ruleId1 = "rule1";
String matchPrefix1 = "A1/";
String ruleId2 = "rule2";
String matchPrefix2 = "A2/";
String ruleId3 = "rule3";
String matchPrefix3 = "A3/";
```

```
// Set the expiration duration to 3 days after last modified date.
request.AddLifecycleRule(new LifecycleRule(ruleId0, matchPrefix0, RuleStatus.Enabled, 3
));

// Configure expiration date to delete objects that are created before the date.
LifecycleRule rule = new LifecycleRule(ruleId1, matchPrefix1, RuleStatus.Enabled);
rule.setCreatedBeforeDate(DateUtil.parseISO8601Date("2022-10-12T00:00:00.000Z"));
request.AddLifecycleRule(rule);

// Set the expiration duration to 3 days for parts in an object.
rule = new LifecycleRule(ruleId2, matchPrefix2, RuleStatus.Enabled);
LifecycleRule.AbortMultipartUpload abortMultipartUpload = new LifecycleRule.AbortMulti
partUpload();
abortMultipartUpload.setExpirationDays(3);
rule.setAbortMultipartUpload(abortMultipartUpload);
request.AddLifecycleRule(rule);

// Configure expiration date to delete parts that are created before the date.
rule = new LifecycleRule(ruleId3, matchPrefix3, RuleStatus.Enabled);
abortMultipartUpload = new LifecycleRule.AbortMultipartUpload();
abortMultipartUpload.setCreatedBeforeDate(DateUtil.parseISO8601Date("2022-10-12T00
:00:00.000Z"));
rule.setAbortMultipartUpload(abortMultipartUpload);
request.AddLifecycleRule(rule);

ossClient.setBucketLifecycle(request);

// Close your OSSClient.
ossClient.shutdown();
```

ライフサイクルルールの表示

ライフサイクルルールを表示するには、次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

List<LifecycleRule> rules = ossClient.getBucketLifecycle(bucketName);
for (LifecycleRule rule : rules) {
    System.out.println（image.getImageId（））。
    System.out.println(rule.getPrefix());
    System.out.println(rule.getExpirationDays());
    System.out.println(rule.getCreatedBeforeDate());
    if(rule.hasAbortMultipartUpload()) {
  System.out.println(rule.getAbortMultipartUpload().getExpirationDays());
  System.out.println(rule.getAbortMultipartUpload().getCreatedBeforeDate());
    }
}

// Close your OSSClient.
```

```
ossClient.shutdown();
```

ライフサイクルルールの削除

ライフサイクルルールを削除するには、次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

ossClient.deleteBucketLifecycle(bucketName);

// Close your OSSClient.
ossClient.shutdown();
```

2.5.3 Anti-leech

このトピックでは、anti-leech の使い方について説明します。

OSS 上のデータ盗難を防ぐために、OSS は、HTTP ヘッダー内の referer フィールド設定による Anti-leech 機能をサポートしています。設定には次のパラメーターが含まれます。

- リファラーホワイトリスト: 指定されたドメインに対してのみ、OSS データへのアクセスを許可するために使用されます。
- 空のリファラー: リファラーを空にできるかどうかを決定します。 許可されていない場合は、HTTP ヘッダーまたは HTTPS ヘッダーにファイルされているリファラー要求のみが、OSS データにアクセスできます。

アンチリーチ機能の詳細については、「アンチリーチ機能の設定」をご参照ください。

リファラーホワイトリストの設定

次のコードを実行して、参照元のホワイトリストを設定します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

List<String> refererList = new ArrayList<String>();
// Add the referer field. The referer field allows question marks (?) and asterisks (*) for
wildcard use.
refererList.add("http://www.aliyun.com");
refererList.add("http://www. *.com");
refererList.add("http://www.?.aliyuncs.com");
// Configure the referer list for a bucket. Set the parameter to true, which allows the
referer field to be empty.
BucketReferer br = new BucketReferer(true, refererList);
ossClient.setBucketReferer(bucketName, br);

// Close your OSSClient.
ossClient.shutdown();
```

リファラーホワイトリストの取得

次のコードを実行して、リファラーホワイトリストを取得します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Obtain the referer list for a bucket.
BucketReferer br = ossClient.getBucketReferer(bucketName);
List<String> refererList = br.getRefererList();
for (String referer : refererList) {
 System.out.println(referer);
}

// Close your OSSClient.
ossClient.shutdown();
```

リファラーホワイトリストの解除

次のコードを実行して、リファラーホワイトリストを解除します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
```

```
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// You cannot clear a referer whitelist directly. To clear a referer whitelist, you need to
create the rule that allows an empty referer field and replace the original rule with the
new rule.
BucketReferer br = new BucketReferer();
ossClient.setBucketReferer(bucketName, br);

// Close your OSSClient.
ossClient.shutdown();
```

2.5.4 ログの設定

アクセスログを有効にし、指定したバケットに保存されているログファイルへのバケットアクセスを記録します。

ログ ファイルの形式は次のとおりです。

```
<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString
```

アクセスログファイルの詳細については、「アクセスログの設定」をご参照ください。

アクセスログの有効化

次のコードを実行してバケットアクセスログを有効にします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

SetBucketLoggingRequest request = new SetBucketLoggingRequest("<yourSource
BucketName>");
// Configure the bucket that stores log files.
request.setTargetBucket("<yourTargetBucketName>");
// Configure the log file storage directory.
request.setTargetPrefix("<yourTargetPrefix>");
ossClient.setBucketLogging(request);

// Close your OSSClient.
ossClient.shutdown();
```

アクセスログ設定の表示

次のコードを実行して、バケットのアクセスログ設定を表示します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

BucketLoggingResult result = ossClient.getBucketLogging("<yourSourceBucketName>");
System.out.println(result.getTargetBucket());
System.out.println(result.getTargetPrefix());

// Close your OSSClient.
ossClient.shutdown();
```

アクセスログの無効化

次のコードを実行して、バケットのアクセスログを無効にします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

SetBucketLoggingRequest request = new SetBucketLoggingRequest("<yourSourceBucketName>");
request.setTargetBucket(null);
request.setTargetPrefix(null);
ossClient.setBucketLogging(request);

// Close your OSSClient.
ossClient.shutdown();
```

2.5.5 静的 Web サイトホスティング

バケット設定を静的 Web サイトホスティングモードに設定します。 設定が有効になった後、バケットドメインを使用してこの静的 Web サイトにアクセスし、指定したインデックスページまたはエラーページにリダイレクトします

静的 Web サイトホスティングの詳細については、「静的ウェブサイトホスティング」をご参照ください。

静的 Web サイトホスティングの設定

静的 Web サイトホスティングを設定するには、次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

SetBucketWebsiteRequest request = new SetBucketWebsiteRequest("<yourBucketName>");
request.setIndexDocument("index.html");
request.setErrorDocument("error.html");
ossClient.setBucketWebsite(request);

// Close your OSSClient.
ossClient.shutdown();
```

静的 Web サイトのホスティング設定の表示

静的 Web サイトホスティング設定を表示するには、次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

BucketWebsiteResult result = ossClient.getBucketWebsite("<yourBucketName>");
System.out.println(result.getIndexDocument());
System.out.println(result.getErrorDocument());

// Close your OSSClient.
ossClient.shutdown();
```

静的 Web サイトホスティング設定の削除

静的 Web サイトホスティング設定を削除するには、次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
```

```
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

ossClient.deleteBucketWebsite("<yourBucketName>");

// Close your OSSClient.
ossClient.shutdown();
```

2.5.6 クロスリージョンレプリケーション

クロスリージョンレプリケーションは、異なるリージョンのバケット間でオブジェクトを自動的かつ非同期的にコピーするため、使用されます。 ソースバケット内のオブジェクトに対する変更操作 (作成、置換、および削除) は、対象のバケットと同期されます。 この機能は、クロスリージョンディザスターリカバリーとデータレプリケーションの要件を満たすため、使用されます。

クロスリージョンレプリケーションの詳細については、「クロスリージョンレプリケーションの管理」をご参照ください。

クロスリージョンレプリケーションの有効化

クロスリージョンレプリケーションを有効にするには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

AddBucketReplicationRequest request = new AddBucketReplicationRequest(bucketName
);
request.setReplicationRuleID("<yourRuleId>");
request.setTargetBucketName("<yourTargetBucketName>");
// This example uses the target endpoint China (Beijing).
request.setTargetBucketLocation("oss-cn-beijing");
// Disable historical data replication. Historical data replication is enabled by default.
request.setEnableHistoricalObjectReplication(false);
ossClient.addBucketReplication(request);

// Close your OSSClient.
```

```
ossClient.shutdown();
```

クロスリージョンレプリケーション構成の表示

次のコードを使用して、バケットを対象としたクロスリージョンレプリケーション構成を表示します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

List<ReplicationRule> rules = ossClient.getBucketReplication(bucketName);
for (ReplicationRule rule : rules) {
    System.out.println(rule.getReplicationRuleID());
    System.out.println(rule.getTargetBucketLocation());
    System.out.println(rule.getTargetBucketName());
}

// Close your OSSClient.
ossClient.shutdown();
```

クロスリージョンレプリケーション進行状況の表示

クロスリージョンレプリケーションの進行状況には、履歴データの同期とリアルタイムデータの同期の進行状況が含まれます。

- パーセント記号 (%) は、履歴データの同期進行状況を示します。 この機能は、履歴データの同期が有効になっているバケットに対してのみ機能します。
- "時点" (データが最後に書き込まれた時間) は、リアルタイムデータの同期進行状況を示します。 ある時点が表示されている場合、その時点より前に作成されたデータは同期されています。

クロスリージョンレプリケーションの進行状況を表示するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

BucketReplicationProgress process = ossClient.getBucketReplicationProgress(
bucketName, "<yourRuleId>");
System.out.println(process.getReplicationRuleID());
// Check whether historical data synchronization is enabled.
System.out.println(process.isEnableHistoricalObjectReplication());
// Print historical data synchronization progress.
System.out.println(process.getHistoricalObjectProgress());
// Print real-time data synchronization progress.
System.out.println(process.getNewObjectProgress());

// Close your OSSClient.
ossClient.shutdown();
```

同期を行うため、対象のリージョンを表示する

次のコードを実行して、バケットを同期する対象のリージョンを取得します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

List<String> locations = ossClient.getBucketReplicationLocation(bucketName);
for (String loc : locations) {
    System.out.println(loc);
}

// Close your OSSClient.
ossClient.shutdown();
```

クロスリージョンレプリケーションの無効化

クロスリージョンレプリケーションを無効にするには、次のコードを使用します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
```

```
// Disable cross-region replication. After this function is disabled, objects in the target bucket exist and all changes in the source objects cannot be synchronized.
ossClient.deleteBucketReplication(bucketName, "<yourRuleId>");

// Close your OSSClient.
ossClient.shutdown();
```

2.5.7 CORS

このトピックでは、CORS の実行方法について説明します。

クロスオリジンリソース共有 (CORS) を使用すると、Web アプリケーションは別のリージョンに属するリソースにアクセスできます。 OSS はクロスオリジンアクセスを制御するための CORS API を提供します。

詳細は、OSS 開発者ガイドの「クロスオリジンリソース共有」と「PutBucketcors」をご参照ください。

CORS ルールの設定

次のコードを実行して、指定したバケットの CORS ルールを設定します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

SetBucketCORSRequest request = new SetBucketCORSRequest(bucketName);

// Create a CORS rule. A maximum of 10 rules can be configured for each bucket.
ArrayList<CORSRule> putCorsRules = new ArrayList<CORSRule>();

CORSRule corRule = new CORSRule();

ArrayList<String> allowedOrigin = new ArrayList<String>();
// Specify the source of the cross-origin access request.
allowedOrigin.add( "http://www.b.com");

ArrayList<String> allowedMethod = new ArrayList<String>();
// Specify the cross-region request methods (GET, PUT, DELETE, POST, and HEAD) that are allowed.
allowedMethod.add("GET");

ArrayList<String> allowedHeader = new ArrayList<String>();
// Specify whether the header specified in Access-Control-Request-Headers of pre-flight request (OPTIONS) is allowed.
allowedHeader.add("x-oss-test");

ArrayList<String> exposedHeader = new ArrayList<String>();
```

```
// Specify the response header that allows user access from applications.
exposedHeader.add("x-oss-test1");
// AllowedOrigins and AllowedMethods allow only one wildcard asterisk (*). Wildcard
 asterisks (*) indicate that all sources of the cross-origin requests and operations are
allowed.
corRule.setAllowedMethods(allowedMethod);
corRule.setAllowedOrigins(allowedOrigin);
// AllowedHeaders and ExposeHeaders do not allow wildcard asterisks (*).
corRule.setAllowedHeaders(allowedHeader);
corRule.setExposeHeaders(exposedHeader);
// Specify the cache time (seconds) for the response of browser pre-flight (OPTIONS)
requests to a specific resource.
corRule.setMaxAgeSeconds(10);

// A maximum of 10 rules is allowed.
putCorsRules.add(corRule);
// The existing rules will be replaced.
request.setCorsRules(putCorsRules);
ossClient.setBucketCORS(request);

// Close your OSSClient.
ossClient.shutdown();
```

CORS ルールの取得

次のコードを実行して、CORS ルールを取得します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

ArrayList<CORSRule> corsRules;
// Obtain the CORS rule list.
corsRules =  (ArrayList<CORSRule>) ossClient.getBucketCORSRules(bucketName);
for (CORSRule rule : corsRules) {
    for (String allowedOrigin1 : rule.getAllowedOrigins()) {
        // Obtain allowed sources of cross-origin requests.
        System.out.println(allowedOrigin1);
    }

    for (String allowedMethod1 : rule.getAllowedMethods()) {
        // Obtain the allowed cross-origin request method.
        System.out.println(allowedMethod1);
    }

    if (rule.getAllowedHeaders().size() > 0){
        for (String allowedHeader1 : rule.getAllowedHeaders()) {
            // Obtain the header list for allowed cross-origin requests.
            System.out.println(allowedHeader1);
        }
    }
```

```
    if (rule.getExposeHeaders().size() > 0) {
        for (String exposeHeader : rule.getExposeHeaders()) {
            // Obtain the header for an allowed cross-origin request.
            System.out.println(exposeHeader);
        }
    }

    if ( null ! = rule.getMaxAgeSeconds()) {
        System.out.println(rule.getMaxAgeSeconds());
    }
}

// Close your OSSClient.
ossClient.shutdown();
```

CORS ルールの削除

次のコードを実行して、指定したバケットのすべての CORS ルールを削除します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Delete all CORS rules for a specified bucket.
ossClient.deleteBucketCORSRules(bucketName);

// Close your OSSClient.
ossClient.shutdown();
```

2.6 オブジェクトのアップロード

2.6.1 概要

このトピックでは、OSS Java SDK のオブジェクトアップロードの概要を説明します。

OSS では、オブジェクトはデータ操作の基本単位です。 OSS Java SDK は次のファイルアップロード方法を提供します。

- 簡易アップロード: 最大 5 GB までのファイルのアップロードをサポートします。 ストリーミングアップロードとファイルアップロードが含まれます。
- フォームアップロード：最大 5 GBまでのファイルのアップロードをサポートします。

- 追加アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 再開可能アップロード: 48.8 TB までのファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。 同時アップロードをサポートしており、各パーツのサイズを定義できます。
- マルチパートアップロード: 最大 48.8 TB のファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。

注:

各アップロード方法の使用シナリオの詳細については、「OSS 開発者ガイド」の「ファイルのアップロード」をご参照ください。

ファイルのアップロード中、 オブジェクトメタを設定することで、アップロードの進行状況をアップロード進行状況バーに表示します。 ファイルのアップロードが完了したら、 アップロードコールバックを実行できます。

2.6.2 簡易アップロード

簡易アップロードには、ストリーミングアップロードとファイルアップロードが含まれます。 ストリーミングアップロードは、ソースオブジェクトとして InputStream を使用します。 ファイルアップロードは、ソースファイルとしてローカルファイルを使用します。

簡易アップロードのための完全なコードについては、『GitHub』をご参照ください。

ストリーミングアップロード

データストリームを OSS へアップロードするには、ossClient.putObject を使用します。

- 文字列のアップロード

  次のコードを実行し、文字列をアップロードします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a string.
String content = "Hello OSS";
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()));
```

```
// Close your OSSClient instance.
ossClient.shutdown();
```

- バイト配列のアップロード

  次のコードを実行し、バイト配列をアップロードします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,accessKeySecret);

// Upload a byte array.
byte[] content = "Hello OSS".getBytes();
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content));

// Close your OSSClient instance.
ossClient.shutdown();
```

- ネットワークストリームのアップロード

  次のコードを実行し、ネットワークストリームをアップロードします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a network stream.
InputStream inputStream = new URL("https://www.aliyun.com/").openStream();
ossClient.putObject("<yourBucketName>", "<yourObjectName>", inputStream);

// Close your OSSClient instance.
ossClient.shutdown();
```

- ファイルストリームのアップロード

  次のコードを実行し、ファイルストリームをアップロードします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. To ensure cloud security, we
```

```
recommend that you follow best practices of Resource Access Management and log
on as a RAM user to access APIs or perform routine operations and maintenance. To
create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a file stream.
InputStream inputStream = new FileInputStream("<yourlocalFile>");
ossClient.putObject("<yourBucketName>", "<yourObjectName>", inputStream);

// Close your OSSClient instance.
ossClient.shutdown();
```

ファイルのアップロード

次のコードを実行し、ローカルファイルをアップロードします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a file. <yourLocalFile> consists of a local file path and a file name that includes
 an extension, for example, /users/local/myfile.txt.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new File("<yourLocalFile
>"));

// Close your OSSClient instance.
ossClient.shutdown();
```

2.6.3 フォームアップロード

フォームアップロードでは、HTML フォームを使用してファイルを指定されたバケットにアップロードします。 フォームアップロードごとのファイルの最大サイズは 5 GB です。

フォームアップロードの適用シナリオの詳細については、OSS 開発者ガイド内にある「フォームアップロード」をご参照ください。

フォームアップロードを行うため、次のコードを実行します。

```
public class PostObjectSample {
    // Upload a file.
    private String localFilePath = "<yourLocalFile>";
    // This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
    private String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
    private String accessKeyId = "<yourAccessKeyId>";
    private String accessKeySecret = "<yourAccessKeySecret>";
    // Specify the bucket name.
    private String bucketName = "<yourBucketName>";
    // Specify the file name.
    private String objectName = "<yourObjectName>";
    /**
     * Start form upload.
     * @throws Exception
     */
    private void PostObject() throws Exception {
        // Add the bucket name to the URL. The URL format is http://yourBucketName.oss-cn
-hangzhou.aliyuncs.com.
        String urlStr = endpoint.replace("http://", "http://" + bucketName+ ".") ;
        // Create Map for the form.
        Map<String, String> formFields = new LinkedHashMap<String, String>();
        // Specify the file name.
        formFields.put("key", this.objectName);
        // Configure Content-Disposition.
        formFields.put("Content-Disposition", "attachment;filename="
                + localFilePath);
        // Configure the callback parameter.
        Callback callback = new Callback();
        // Configure the IP address of the server you want to send the callback request to, for
 example, http://oss-demo.aliyuncs.com:23450 or http://127.0.0.1:9090.
        callback.setCallbackUrl("<yourCallbackServerUrl>");
        // Configure the host field value in the callback request header, such as oss-cn-
hangzhou.aliyuncs.com.
        callback.setCallbackHost("<yourCallbackServerHost>");
        // Configure the body field value for the callback request.
        callback.setCallbackBody("{\\\"mimeType\\\":${mimeType},\\\"size\\\":${size}}");
        // Configure Content-Type for the callback request.
        callback.setCalbackBodyType(CalbackBodyType.JSON);
        // Configure the custom parameters used for sending the callback request. Each
custom parameter consists of a key and a value. The key must be in lowercase and start
with x:.
        callback.addCallbackVar("x:var1", "value1");
        callback.addCallbackVar("x:var2", "value2");
        // Configure the callback parameter for Map in the form.
        setCallBack(formFields, callback);
        // Configure OSSAccessKeyId.
        formFields.put("OSSAccessKeyId", accessKeyId);
        String policy = "{\"expiration\": \"2120-01-01T12:00:00.000Z\",\"conditions\": [[\"
content-length-range\", 0, 104857600]]}";
        String encodePolicy = new String(Base64.encodeBase64(policy.getBytes()));
        // Configure the policy.
        formFields.put("policy", encodePolicy);
        // Generate the URL for signature.
        String signaturecom = com.aliyun.oss.common.auth.ServiceSignature.create().
computeSignature(accessKeySecret, encodePolicy);
        // Configure the signature.
        formFields.put("Signature", signaturecom);
        String ret = formUpload(urlStr, formFields, localFilePath);
        System.out.println("Post Object [" + this.objectName + "] to bucket [" + bucketName +
 "]");
        System.out.println("post reponse:" + ret);
    }
    private static String formUpload(String urlStr, Map<String, String> formFields, String
localFile)
            throws Exception {
```

```
        String res = "";
        HttpURLConnection conn = null;
        String boundary = "9431149156168";
        try {
            URL url = new URL(urlStr);
            conn = (HttpURLConnection) url.openConnection();
            conn.setConnectTimeout(5000);
            conn.setReadTimeout(30000);
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.setRequestMethod("POST");
            conn.setRequestProperty("User-Agent",
                    "Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.2.6)");
            // Configure the MD5 value. The MD5 value is calculated based on the entire body.
            conn.setRequestProperty("Content-MD5", "<yourContentMD5>");
            conn.setRequestProperty("Content-Type",
                    "multipart/form-data; boundary=" + boundary);
            OutputStream out = new DataOutputStream(conn.getOutputStream());
            // Recursively read all Map data in the form and write the data to the OutputStre
am.
            if (formFields ! = null) {
                StringBuffer strBuf = new StringBuffer();
                Iterator<Entry<String, String>> iter = formFields.entrySet().iterator();
                int i = 0;
                while (iter.hasNext()) {
                    Entry<String, String> entry = iter.next();
                    String inputName = entry.getKey();
                    String inputValue = entry.getValue();
                    if (inputValue == null) {
                        continue;
                    }
                    if (i == 0) {
                        strBuf.append("--").append(boundary).append("\r\n");
                        strBuf.append("Content-Disposition: form-data; name=\""
                                + inputName + "\"\r\n\r\n");
                        strBuf.append(inputValue);
                    } else {
                        strBuf.append("\r\n").append("--").append(boundary).append("\r\n");
                        strBuf.append("Content-Disposition: form-data; name=\""
                                + inputName + "\"\r\n\r\n");
                        strBuf.append(inputValue);
                    }
                    i++;
                }
                out.write(strBuf.toString().getBytes());
            }
            // Read the file and write it to the OutputStream.
            File file = new File(localFile);
            String filename = file.getName();
            String contentType = new MimetypesFileTypeMap().getContentType(file);
            if (contentType == null || contentType.equals("")) {
                contentType = "application/octet-stream";
            }
            StringBuffer strBuf = new StringBuffer();
            strBuf.append("\r\n").append("--").append(boundary)
                    .append("\r\n");
            strBuf.append("Content-Disposition: form-data; name=\"file\"; "
                    + "filename=\"" + filename + "\"\r\n");
            strBuf.append("Content-Type: " + contentType + "\r\n\r\n");
            out.write(strBuf.toString().getBytes());
            DataInputStream in = new DataInputStream(new FileInputStream(file));
            int bytes = 0;
            byte[] bufferOut = new byte[1024];
            while ((bytes = in.read(bufferOut)) ! = -1) {
```

```
            out.write(bufferOut, 0, bytes);
        }
        in.close();
        byte[] endData = ("\r\n--" + boundary + "--\r\n").getBytes();
        out.write(endData);
        out.flush();
        out.close();
        // Read the returned data.
        strBuf = new StringBuffer();
        BufferedReader reader = new BufferedReader(new InputStreamReader(conn.
getInputStream()));
        String line = null;
        while ((line = reader.readLine()) ! = null) {
            strBuf.append(line).append("\n");
        }
        res = strBuf.toString();
        reader.close();
        reader = null;
    } catch (Exception e) {
        System.err.println("Send post request exception: " + e);
        throw e;
    } finally {
        if (conn ! = null) {
            conn.disconnect();
            conn = null;
        }
    }
    return res;
  }
   private static void setCallBack(Map<String, String> formFields, Callback callback) {
    if (callback ! = null) {
        String jsonCb = OSSUtils.jsonizeCallback(callback);
        String base64Cb = BinaryUtil.toBase64String(jsonCb.getBytes());
        formFields.put("callback", base64Cb);
        if (callback.hasCallbackVar()) {
            Map<String, String> varMap = callback.getCallbackVar();
            for (Entry<String, String> entry : varMap.entrySet()) {
                formFields.put(entry.getKey(), entry.getValue());
            }
        }
    }
  }
  public static void main(String[] args) throws Exception {
    PostObjectSample ossPostObject = new PostObjectSample();
    ossPostObject.PostObject();
  }
}
```

注：

コールバックの詳細については、「コールバック」をご参照ください。

2.6.4 追加アップロード

このトピックでは、追加アップロードの使用方法について説明します。

追加オブジェクトに対して copyObject を実行することはできません。

追加アップロードでファイルをアップロードするには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

String content1 = "Hello OSS A \n";
String content2 = "Hello OSS B \n";
String content3 = "Hello OSS C \n";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

ObjectMetadata meta = new ObjectMetadata();
// Specify the content type of the object you want to upload.
meta.setContentType("text/plain");

// Configure parameters with AppendObjectRequest.
AppendObjectRequest appendObjectRequest = new AppendObjectRequest("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content1.getBytes()),meta);

// Configure a single parameter with AppendObjectRequest.
// Specify the bucket name.
//appendObjectRequest.setBucketName("<yourBucketName>");
// Specify the object name.
//appendObjectRequest.setKey("<yourObjectName>");
// Configure the type of content you want to append. Two types of content are available: InputStream and File. Select InputStream.
//appendObjectRequest.setInputStream(new ByteArrayInputStream(content1.getBytes()));
// Configure the type of content you want to append. Two types of content are available: InputStream and File. Select File.
//appendObjectRequest.setFile(new File("<yourLocalFile>"));
// Specify object meta, because the specified object meta takes effect from the first append.
//appendObjectRequest.setMetadata(meta);

// Start the first append.
// Configure the position where the object is appended based on the previous object length.
appendObjectRequest.setPosition(0L);
AppendObjectResult appendObjectResult = ossClient.appendObject(appendObjectRequest);
// Calculate the 64-bit CRC value. The value is calculated based on the ECMA-182 standard. This value is calculated based on ECMA-182 criteria.
System.out.println(appendObjectResult.getObjectCRC());

// Start the second append.
// nextPosition indicates the position provided for the next request, namely, the length of the current object.
appendObjectRequest.setPosition(appendObjectResult.getNextPosition());
appendObjectRequest.setInputStream(new ByteArrayInputStream(content2.getBytes()));
appendObjectResult = ossClient.appendObject(appendObjectRequest);

// Start the third append.
appendObjectRequest.setPosition(appendObjectResult.getNextPosition());
appendObjectRequest.setInputStream(new ByteArrayInputStream(content3.getBytes()));
```

```
appendObjectResult = ossClient.appendObject(appendObjectRequest);

// Close your OSSClient.
ossClient.shutdown();
```

追加アップロードについての詳細は、「OSS 開発者ガイド」内の「追加オブジェクト」 をご参照ください。 追加アップロードの完全なコードについては、『GitHub』をご参照ください。

2.6.5 再開可能アップロード

再開可能アップロードを使用すると、アップロードするファイルを複数のパーツに分割して同時にアップロードします。 すべてのパーツをアップロードしたら、各パーツを組み合わせて完全なオブジェクトにすることができます。 このようにして、オブジェクト全体のアップロードを完了させます。

再開可能アップロードについての詳細は、「OSS 開発者ガイド」の 「再開可能アップロード」 をご参照ください。 再開可能アップロードの完全なコードについては、『GitHub』をご参照ください。

再開可能アップロードを行うには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

ObjectMetadata meta = new ObjectMetadata();
// Specify the content type of the object you want to upload.
meta.setContentType("text/plain");

// Configure parameters with UploadFileRequest.
UploadFileRequest uploadFileRequest = new UploadFileRequest("<yourBucketName>","<
yourObjectName>");

// Configure parameters with UploadFileRequest.
// Specify the bucket name.
//uploadFileRequest.setBucketName("<yourBucketName>");
// Specify the object name.
//uploadFileRequest.setKey("<yourObjectName>");
// Specify the local file you want to upload.
uploadFileRequest.setUploadFile("<yourLocalFile>");
// Specify the number of threads for simultaneous upload. The default value is 1.
uploadFileRequest.setTaskNum(5);
// Specify the size of a part you want to upload. The value can be from 100 KB to 5 GB.
The default size of a part is calculated as: file size/10,000.
uploadFileRequest.setPartSize(1 * 1024 * 1024);
// Start resumable upload (that is disabled by default.).
uploadFileRequest.setEnableCheckpoint(true);
```

```
// Configure the file used to record the result of local part uploads. You need to configure
 this parameter when you enable resumable upload. This file stores progress informatio
n. If a part upload fails, it can be continued based on the progress information recorded
in this file. After you have uploaded the file, the file is deleted. By default, this file shares
the same directory (uploadFile.ucp) as the file you want to upload.
uploadFileRequest.setCheckpointFile("<yourCheckpointFile>");
// Configure object meta.
uploadFileRequest.setObjectMetadata(meta);
// Configure callback. Set the yourCallbackEvent parameter to Callback.
uploadFileRequest.setCallback("<yourCallbackEvent>");

// Start resumable upload.
ossClient.uploadFile(uploadFileRequest);

// Close your OSSClient.
ossClient.shutdown();
```

2.6.6 マルチパートアップロード

このトピックでは、マルチパートアップロードの使用方法について説明します。

マルチパートアップロードの完全なコードは、GitHub をご参照ください。

マルチパートアップロードを有効にするには、次の手順を実行します。

1. マルチパートアップロードイベントを開始します。

   ossClient.initiateMultipartUpload を呼び出すと、OSS で作成されたグローバルに一意の uploadId が返されます。

2. パートをアップロードします。

   ossClient.uploadPart を呼び出して、パートデータをアップロードします。

   注：

   - 同じ uploadId を持つパートの場合、パート番号によってパートが順序付けされます。 あるパートをアップロードし、同じパート番号を使用して別のパートをアップロードすると、後のパートが前のパートに置き換わります。
   - OSS は パートデータの MD5 値を ETag に保存し、MD5 値をユーザーに返します。
   - SDK は自動的に Content-MD5 を設定します。 OSS はアップロードされたデータの MD5 値を計算し、SDK で計算された MD5 値と比較します。 2 つの値が異なると、エラーコード InvalidDigest が返されます。

3. マルチパートアップロードを完了します。

   すべてのパートをアップロードした後、partossClient.completeMultipartUpload を呼び出して各パートを 1 つの完全なオブジェクトとして結合します。

次のコード例を使用して、マルチパートアップロードのプロセスを説明します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

/* Step 1: Initiate a multipart upload event.
*/
InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest(bucketName, objectName);
InitiateMultipartUploadResult result = ossClient.initiateMultipartUpload(request);
// An uploadId is returned. It is the unique identifier for a part upload event. You can initiate related operations (such as part upload cancelation and query) based on the uploadId.
String uploadId = result.getUploadId();

/* Step 2: Upload parts.
*/
// partETags is a set of PartETags. A PartETag consists of an ETag and a part number.
List<PartETag> partETags =  new ArrayList<PartETag>();
// Calculate the total number of parts.
final long partSize = 1 * 1024 * 1024L;   // 1MB
final File sampleFile = new File("<localFile>");
long fileLength = sampleFile.length();
int partCount = (int) (fileLength / partSize);
if (fileLength % partSize ! = 0) {
    partCount++;
 }
// Upload each part simultaneously until all parts are uploaded.
for (int i = 0; i < partCount; i++) {
    long startPos = i * partSize;
    long curPartSize = (i + 1 == partCount) ? (fileLength - startPos) : partSize;
    InputStream instream = new FileInputStream(sampleFile);
    // Skip parts that have been uploaded.
    instream.skip(startPos);
    UploadPartRequest uploadPartRequest = new UploadPartRequest();
    uploadPartRequest.setBucketName(bucketName);
    uploadPartRequest.setKey(objectName);
    uploadPartRequest.setUploadId(uploadId);
    uploadPartRequest.setInputStream(instream);
    // Configure the size available for each part. Aside from the last part, all parts are sized 100 KB at least.
    uploadPartRequest.setPartSize(curPartSize);
    // Configure part numbers. Each part is configured with a part number. The value can be from 1 to 10,000. If you configure a number beyond the range, OSS returns an InvalidArgument error code.
    uploadPartRequest.setPartNumber( i + 1);
    // Do not upload each part sequentially. They can be uploaded from different OSSClients. Then they are sequenced and combined into a complete object based on part numbers.
    UploadPartResult uploadPartResult = ossClient.uploadPart(uploadPartRequest);
```

```
    // After you upload a part, OSS returns a result that contains a PartETag. The PartETag
is stored in partETags.
    partETags.add(uploadPartResult.getPartETag());
}

/* Step 3: Complete multipart upload.
*/
// Sequence parts. partETags must be sequenced by part numbers in ascending order.
Collections.sort(partETags, new Comparator<PartETag>() {
    public int compare(PartETag p1, PartETag p2) {
        return p1.getPartNumber() - p2.getPartNumber();
    }
});
// You must provide all valid PartETags when you perform this operation. OSS verifies
the validity of all parts one by one after it receives PartETags. After part verification is
successful, OSS combines these parts into a complete object.
CompleteMultipartUploadRequest completeMultipartUploadRequest =
        new CompleteMultipartUploadRequest(bucketName, objectName, uploadId,
partETags);
ossClient.completeMultipartUpload(completeMultipartUploadRequest);

// Close your OSSClient.
ossClient.shutdown();
```

マルチパートアップロードイベントのキャンセル

ossClient.abortMultipartUpload を呼び出して、マルチパートアップロードイベントをキャンセルします。 マルチパートアップロードイベントをキャンセルすると、この uploadId を使用して他の操作を実行することはできなくなります。 アップロード済みのパートは削除されます。

マルチパートアップロードイベントをキャンセルするには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Cancel multipart upload. The uploadId is from InitiateMultipartUpload.
AbortMultipartUploadRequest abortMultipartUploadRequest =
        new AbortMultipartUploadRequest("<yourBucketName>", "<yourObjectName>", "<
uploadId>");
ossClient.abortMultipartUpload(abortMultipartUploadRequest);

// Close your OSSClient.
ossClient.shutdown();
```

アップロードしたパートの一覧表示

ossClient.listParts を呼び出して、指定した uploadId を持つすべてのアップロード済みパートを一覧表示します。

- 簡易リスト

  簡易リストを表示するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// List uploaded parts. The uploadId is returned from InitiateMultipartUpload.
ListPartsRequest listPartsRequest = new ListPartsRequest("<yourBucketName>", "<yourObjectName>", "<uploadId>");
 // Configure uploadId.
 //listPartsRequest.setUploadId(uploadId);
 // Set the maximum number of parts that can be displayed on each page to 100. The default number is 1,000.
 listPartsRequest.setMaxParts(100);
 // Specify the initial position in the list. Only the parts with part numbers greater than this parameter value are listed.
 listPartsRequest.setPartNumberMarker(2);
PartListing partListing = ossClient.listParts(listPartsRequest);

for (PartSummary part : partListing.getParts()) {
    // Obtain part numbers.
    part.getPartNumber();
    // Obtain the size of part data.
    part.getSize();
    // Obtain ETag.
    part.getETag();
    // Obtain the latest time parts are modified.
    part.getLastModified();
}

// Close your OSSClient.
ossClient.shutdown();
```

- アップロードされたすべてのパートの一覧表示

  デフォルトでは、listParts は一度に最大 1,000 個のパートを一覧表示できます。 1,000 を超えるアップロード済みパートを一覧表示するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// List all uploaded parts.
PartListing partListing;
ListPartsRequest listPartsRequest = new ListPartsRequest("<yourBucketName>", "<yourObjectName>", "<uploadId>");

do {
    partListing = ossClient.listParts(listPartsRequest);

    for (PartSummary part : partListing.getParts()) {
        // Obtain part numbers.
        part.getPartNumber();
        // Obtain the part size.
        part.getSize();
        // Obtain ETag.
        part.getETag();
        // Obtain the latest time parts are modified.
        part.getLastModified();
    }
// Specify the initial position for the list. Only the parts with part numbers greater than the initial value are listed.
listPartsRequest.setPartNumberMarker(partListing.getNextPartNumberMarker());
} while (partListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown()
```

- アップロードされたすべてのパートを 1 ページまたは複数ページに一覧表示

  各ページに表示されるパートの最大数を指定するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
```

```
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// List all uploaded parts on one or more pages.
PartListing partListing;
ListPartsRequest listPartsRequest = new ListPartsRequest("<yourBucketName>", "<
yourObjectName>", "<uploadId>");
// Set the maximum number of parts displayed on each page to 100.
listPartsRequest.setMaxParts(100);

do {
   partListing = ossClient.listParts(listPartsRequest);

   for (PartSummary part : partListing.getParts()) {
      // Obtain part numbers.
      part.getPartNumber();
      // Obtain the part size.
      part.getSize();
      // Gets the etag of the slice.
      // Obtain ETag.
      // Obtain the latest time parts are modified.
      part.getLastModified();
   }

   listPartsRequest.setPartNumberMarker(partListing.getNextPartNumberMarker());

} while (partListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

バケットのすべてのパートアップロードイベントの一覧表示

ossClient.listMultipartUploads を呼び出して、進行中のすべてのパートアップロードイベント (開始されたが、完了していないかキャンセルされたイベント) を一覧表示します。 次のパラメーターを設定できます。

パラメーター	説明	設定方法
prefix	返されるオブジェクト名に含める必要があるプレフィックス。 クエリにプレフィックスを使用した場合、返されるオブジェクト名にはそのプレフィックスが含まれます。	ListMultipartUploadsRequest.setPrefix(String prefix)
delimiter	オブジェクト名のグループ化に使用されるスラッシュ (/)。 指定されたプレフィックスと、最初のスラッシュ (/) の間にあるオブジェクトは commonPrefixes です。	ListMultipartUploadsRequest.setDelimiter(String delimiter)

パラメーター	説明	設定方法
maxUploads	パートアップロードイベントの最大数。 設定できる最大値 (またはデフォルト値) は 1,000 です。	ListMultipartUploadsRequest.setMaxUploads(Integer maxUploads)
keyMarker	アルファベット順で keyMarker 値より後の文字で名前が始まるオブジェクトのマルチパートアップロードイベントをすべてリストします。 このパラメーターを uploadIdMarker パラメーターと一緒に使用して、返される結果内の初期位置を指定できます。	ListMultipartUploadsRequest.setKeyMarker(String keyMarker)
uploadIdMarker	このパラメーターを keyMarker パラメーターと一緒に使用して、返される結果内の初期位置を指定できます。 keyMarker を設定しない場合、uploadIdMarker パラメーターは無効です。 keyMarker を設定した場合、クエリ結果には以下が含まれます。 • keyMarker 値より後の文字で名前が始まるすべてのオブジェクト。 • アルファベット順で keyMarker 値と同じ文字で名前が始まり、uploadId の値が uploadIdMarker よりも大きいオブジェクト。	ListMultipartUploadsRequest.setUploadIdMarker(String uploadIdMarker)

- 簡易リスト

  マルチパートアップロードイベントを一覧表示するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a
```

```
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Lists multipart upload events. You can list 1,000 parts by default.
ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploads
Request(bucketName);
MultipartUploadListing multipartUploadListing = ossClient.listMultipartUploads(
listMultipartUploadsRequest);

for (MultipartUpload multipartUpload : multipartUploadListing.getMultipartUploads())
 {
   // Obtain uploadId.
   multipartUpload.getUploadId();
   // Obtain object names.
   multipartUpload.getKey();
   // Obtain the time part upload events are initiated.
   multipartUpload.getInitiated();
}

// Close your OSSClient.
ossClient.shutdown();
```

返された結果の isTruncated 値が false の場合、nextKeyMarker と nextUploadIdMarker の値が返され、次のオブジェクト読み取りの初期位置として使用されます。 一度にすべてのパートアップロードイベントを取得できない場合、ページ区切りで一覧表示されます。

- すべてのパートアップロードイベントの一覧表示

  デフォルトでは、listMultipartUploads は一度に最大 1,000 個のパートを一覧表示します。 1,000 を超えるパートを一覧表示するには、次のコードを実行してすべてのパートを一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Lists multipart upload events.
MultipartUploadListing multipartUploadListing;
ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploads
Request(bucketName);
```

```
do {
    multipartUploadListing = ossClient.listMultipartUploads(listMultipartUploads
Request);

    for (MultipartUpload multipartUpload : multipartUploadListing.getMultipartUploads
()) {
        // Obtain uploadId.
        multipartUpload.getUploadId();
        // Obtain object names.
        multipartUpload.getKey();
        // Obtain the time part upload events are initiated.
        multipartUpload.getInitiated();
    }

    listMultipartUploadsRequest.setKeyMarker(multipartUploadListing.getNextKey
Marker());

    listMultipartUploadsRequest.setUploadIdMarker(multipartUploadListing.getNextUpl
oadIdMarker());
} while (multipartUploadListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

- パートアップロードイベントを 1 ページまたは複数ページに一覧表示

  パートアップロードイベントを 1 ページまたは複数ページに一覧表示するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Lists multipart upload events.
MultipartUploadListing multipartUploadListing;
ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploads
Request(bucketName);
// Configure the number of part upload events that can be displayed on each page.
listMultipartUploadsRequest.setMaxUploads(50);

do {
    multipartUploadListing = ossClient.listMultipartUploads(listMultipartUploads
Request);

    for (MultipartUpload multipartUpload : multipartUploadListing.getMultipartUploads
()) {
        // Obtain uploadId.
        multipartUpload.getUploadId();
        // Obtain object names.
```

```
        multipartUpload.getKey();
        // Obtain the time part upload events are initiated.
        multipartUpload.getInitiated();
    }

    listMultipartUploadsRequest.setKeyMarker(multipartUploadListing.getNextKey
Marker());
    listMultipartUploadsRequest.setUploadIdMarker(multipartUploadListing.getNextUpl
oadIdMarker());

} while (multipartUploadListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

2.6.7 アップロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を示すことができます。

次のコードは、ossClient.putObject を使用した進行状況を表示する方法の説明例として使用されています。

```
public class PutObjectProgressListener implements ProgressListener {
    private long bytesWritten = 0;
    private long totalBytes = -1;
    private boolean succeed = false;

    @Override
    public void progressChanged(ProgressEvent progressEvent) {
        long bytes = progressEvent.getBytes();
        ProgressEventType eventType = progressEvent.getEventType();
        switch (eventType) {
        case TRANSFER_STARTED_EVENT:
            System.out.println("Start to upload.....");
            break;
        case REQUEST_CONTENT_LENGTH_EVENT:
            this.totalBytes = bytes;
            System.out.println(this.totalBytes + " bytes in total will be uploaded to OSS");
            break;
        case REQUEST_BYTE_TRANSFER_EVENT:
            this.bytesWritten += bytes;
            if (this.totalBytes ! = -1) {
                int percent = (int)(this.bytesWritten * 100.0 / this.totalBytes);
                System.out.println(bytes + " bytes have been written at this time, upload
progress: " + percent + "%(" + this.bytesWritten + "/" + this.totalBytes + ")");
            } else{
                System.out.println(bytes + " bytes have been written at this time, upload ratio:
unknown" + "(" + this.bytesWritten + "/...)");
            }
            break;
        case TRANSFER_COMPLETED_EVENT:
            this.succeed = true;
            System.out.println("Succeed to upload, " + this.bytesWritten + " bytes have been
transferred in total");
            break;
        case TRANSFER_FAILED_EVENT:
            System.out.println("Failed to upload, " + this.bytesWritten + " bytes have been
transferred");
```

```
            break;
        default:
            break;
        }
    }

    public boolean isSucceed() {
        return succeed;
    }

    public static void main(String[] args) {
        // This example uses endpoint China (Hangzhou). Specify the actual endpoint based
 on your requirements.
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";
        String objectName = "<yourObjectName>";

        // Create an OSSClient instance.
        OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

        try {
            // Upload with a progress bar displayed.
            ossClient.putObject(new PutObjectRequest(bucketName, objectName, new File("<
yourLocalFile>")).
                    <PutObjectRequest>withProgressListener(new PutObjectProgressListener()));

        } catch (Exception e) {
            e.printStackTrace();
        }
        // Close your OSSClient instance.
        ossClient.shutdown();
    }
}
```

進行状況は、ossClient.putObject、ossClient.getObject、またはossClient.uploadPartで表示できます。 ただし、ossClient.uploadFileまたはossClient.downloadFileを使用して進行状況を表示することはできません。

アップロードの進捗プログレスバーに関する完全なコードについては、『GitHub』をご参照ください。

2.6.8 アップロードコールバック

このトピックではアップロードコールバックの使い方について説明します。

アップロードコールバックの完全なコードについては、『GitHub』をご参照ください。

アップロードコールバックを行うには次のコードを実行します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Specify the IP address of the server you want to send the callback request to, for example, http://oss-demo.aliyuncs.com:23450 or http://127.0.0.1:9090.
String callbackUrl = "<yourCallbackServerUrl>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

String content = "Hello OSS";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName, new ByteArrayInputStream(content.getBytes()));

// Configure upload callback parameters.
Callback callback = new Callback();
callback.setCallbackUrl(callbackUrl);
// Configure the value of the host field carried in the callback request header, such as oss-cn-hangzhou.aliyuncs.com.
callback.setCallbackHost("oss-cn-hangzhou.aliyuncs.com");
// Configure the value of the body field carried in the callback request.
callback.setCallbackBody("{\\\"mimeType\\\":${mimeType},\\\"size\\\":${size}}");
// Configure Content-Type for the callback request.
callback.setCalbackBodyType(CallbackBodyType.JSON);
// Configure the custom parameters used to initiate a callback request. Each custom parameter consists of a key and a value. The key must start with x:.
callback.addCallbackVar("x:var1", "value1");
callback.addCallbackVar("x:var2", "value2");
putObjectRequest.setCallback(callback);

PutObjectResult putObjectResult = ossClient.putObject(putObjectRequest);

// Read the message returned from upload callback.
byte[] buffer = new byte[1024];
putObjectResult.getCallbackResponseBody().read(buffer);
// If you do not close the reader after the data is read, connection leaks may occur. Consequently, no available connections are left and an exception occurs.
putObjectResult.getCallbackResponseBody().close();

// Close your OSSClient instance.
ossClient.shutdown();
```

詳しくは、「OSS開発者ガイド」の「アップロードコールバック」をご参照ください。

2.7 オブジェクトのダウンロード

2.7.1 概要

このトピックでは、OSS Java SDK でのオブジェクトダウンロードの概要を説明します。

OSS Java SDK は以下のダウンロード方法を提供します。

- ストリーミングダウンロード

- ローカルファイルへのダウンロード
- 部分ダウンロード
- 再開可能なダウンロード
- 条件付きダウンロード

ダウンロードの進行状況は、ダウンロード進行状況バーに表示されます。

2.7.2 ストリーミングダウンロード

大容量のオブジェクトをダウンローズする場合、またはオブジェクト全体を一度にダウンロードするのに時間がかかる場合は、ストリーミングダウンロードを使用します。 ストリーミングダウンロードでは、オブジェクト全体をダウンロードし終わるまで、オブジェクトのパーツをその都度ダウンロードし続けます。

使用後に OssObject を終了しない場合、接続リークが発生する可能性があります。 その結果、利用可能な接続がなくなり、プログラムを正しく実行できません。 次のコードを実行し、OssObject を終了します。

```
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
ossObject.close();
```

以下のコードはストリーミングダウンロードに使用されます。

```
// This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// ossObject includes the bucket name, key (object name), object meta (meta information
), and InputStream.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);

// Read the object content.
System.out.println("Object content:");
BufferedReader reader = new BufferedReader(new InputStreamReader(ossObject.
getObjectContent()));
while (true) {
    String line = reader.readLine();
    if (line == null) break;

    System.out.println("\n" + line);
}
```

```
// If you do not close the reader after the data is read, connection leaks may occur. Consequently, no connection is available and the program cannot run properly.
reader.close();

//Close your OSSClient.
ossClient.shutdown();
```

ストリーミングダウンロードの完全なコードについては、『GitHub』をご参照ください。

2.7.3 ローカルファイルへのダウンロード

このトピックでは、OSS からローカルファイルにオブジェクトをダウンロードする方法について説明します。

次のコードを実行して、指定されたオブジェクトを OSSClient からローカルファイルにダウンロードします。

```
//This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Download an object to your local file. If the name of the object is the same as that of your local file, the object will replace the local file. If the name of the object is different from that of your local file, the object will be downloaded and a new file will be added to your local device.
ossClient.getObject(new GetObjectRequest(bucketName, objectName), new File("<yourLocalFile>"));

//Close your OSSClient.
ossClient.shutdown();
```

2.7.4 部分ダウンロード

オブジェクトのデータの一部だけが必要な場合は、部分ダウンロードを使用して指定したコンテンツをダウンロードします。

部分ダウンロードは次のコードを実行します。

```
//This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
```

```
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, objectName
);
// Obtain data between the starting byte value of 0 and the maximum byte value of 1,
000 inclusive (that is, a total of 1,001 bytes). If the specified value range is invalid, for
example, the specified range includes a negative number, or the specified value is larger
than the object size, the entire object is downloaded.
getObjectRequest.setRange(0, 1000);

// Start range download.
OSSObject ossObject = ossClient.getObject(getObjectRequest);

// Read data.
byte[] buf = new byte[1024];
InputStream in = ossObject.getObjectContent();
for (int n = 0; n ! = -1; ) {
    n = in.read(buf, 0, buf.length);
}

// If you do not close the reader after the data is read, connection leaks may occur.
Consequently, no connections are available and an exception occurs.
rs.close();

//Close your OSSClient.
ossClient.shutdown();
```

ストリーミングされているときにすべてのデータを一度に読み取ることはできません。 64 KB のデータブロックをストリーミングするには、次のコードを使用して、オブジェクト全体がダウンロードされるまでデータを複数回読み取ります。 詳しくは、 「InputStream.read」 をご参照ください。

```
byte[] buf = new byte[1024];
InputStream in = ossObject.getObjectContent();
for (int n = 0; n ! = -1; ) {
    n = in.read(buf, 0, buf.length);
}
rs.close();
```

2.7.5 再開可能ダウンロード

大規模なオブジェクトは、不安定なネットワークまたはその他の例外のためにダウンロードに失敗する可能性があり、オブジェクト全体を再度ダウンロードする必要があります。 ただし、オブジェクトは複数回試行してもダウンロードに失敗することがあります。 そのため、OSS は再開可能ダウンロードを提供します。

再開可能ダウンロードを有効にするには、ダウンロードするオブジェクトを複数のパーツに分割して別々にダウンロードする必要があります。 すべてのパーツをダウンロードした後に、これらのパーツは完全なオブジェクトに結合されます。

再開可能なダウンロードを有効にするには、ossClient.downloadFile を使用します。

DownloadFileRequest には、以下のパラメーターが含まれています。

パラメーター	説明	必須またはオプション	デフォルト値	設定方法
bucket	バケットの名前を指定します。	必須項目	なし	Constructor
key	オブジェクトの名前を指定します。	必須項目	なし	Constructor
downloadFile	ローカルファイルを指定します。オブジェクトをローカルファイルにダウンロードできます。	オプション	オブジェクト名	Constructor or setDownloadFile
partSize	部品のサイズを指定します。 値は 1 バイトから 5 GB までです。	オプション	オブジェクトのサイズ / 10000	setPartSize
taskNum	同時にダウンロードする必要があるパーツの数を指定します。	オプション	1	setTaskNum
enableCheckpoint	再開可能ダウンロードを有効にするかどうかを指定します。	オプション	システム時間の使用	setEnableCheckpoint

パラメーター	説明	必須またはオプション	デフォルト値	設定方法
checkpointFile	パーツダウンロードの結果を記録するファイルを指定します。 再開可能ダウンロードを有効にするには、このパラメーターを設定する必要があります。 このファイルにはダウンロードの進行状況に関する情報が格納されています。 パーツのダウンロードに失敗した場合、ダウンロードは記録された進行状況に基づいて続行されます。 オブジェクトがローカルデバイスにダウンロードされると、このファイルは削除されます。	オプション	downloadFile.ucp (DownloadFile と同じディレクトリを共有します)	setCheckpointFile

再開可能ダウンロードを行うには、次のコードを実行します。

```
//This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
//It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Send a request to download 10 tasks simultaneously and start resumable download.
DownloadFileRequest downloadFileRequest = new DownloadFileRequest(bucketName, objectName);
downloadFileRequest.setDownloadFile("<yourDownloadFile>");
```

```
downloadFileRequest.setPartSize(1 * 1024 * 1024);
downloadFileRequest.setTaskNum(10);
downloadFileRequest.setEnableCheckpoint(true);
downloadFileRequest.setCheckpointFile("<yourCheckpointFile>");

// Download the object to your local file.
DownloadFileResult downloadRes = ossClient.downloadFile(downloadFileRequest);
// After the download succeeds, object meta is returned.
downloadRes.getObjectMetadata();

//Close your OSSClient.
ossClient.shutdown();
```

2.7.6 条件付きダウンロード

ダウンロードに対して 1 つ以上の条件を設定できます。 指定した条件が満たされると、オブジェクトをダウンロードできます。 指定された条件が満たされないと、エラーコードが返され、オブジェクトをダウンロードできません。 以下の条件を設定できます。

パラメーター	説明
If-Modified-Since	指定された時間がオブジェクトが変更された時間より早い場合は、オブジェクトをダウンロードできます。 そうでない場合は、エラー "304 Not modified" が返されます。
If-Unmodified-Since	指定された時間がオブジェクトが変更された時間以降である場合は、オブジェクトをダウンロードできます。 そうでない場合は、エラー "412 Precondition failed" が返されます。
If-Match	指定された ETag がオブジェクトのものと一致すれば、そのオブジェクトをダウンロードすることができます。 そうでない場合は、エラー "412 Precondition failed" が返されます。
If-None-Match	指定された ETag がオブジェクトのものと一致しない場合は、オブジェクトをダウンロードできます。 そうでない場合は、エラー "304 Not modified" が返されます。

"If-Modified-Since" と "If-Unmodified-Since" の両方が、オブジェクトのダウンロード条件と同時に存在することがあります。 "If-Match" と "If-None-Match" の両方が、オブジェクトのダウンロード条件と同時に存在することがあります。

ETag を取得するには、ossClient.getObjectMeta を使用します。

条件付きダウンロード用に次のコードを実行します。

```
//This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
// Configure conditions.
request.setModifiedSinceConstraint(new Date());

// Download an object to a local file.
ossClient.getObject(request, new File("<yourLocalFile>"));

//Close your OSSClient.
ossClient.shutdown();
```

条件付きダウンロードは、ossClient.getObject および ossClient.downloadFile をサポートします。

2.7.7 ダウンロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を表示することができます。

次のコードは、ossClient.getObject を使用した進行状況を表示する方法を説明するための例として、使用されています。

```
static class GetObjectProgressListener implements ProgressListener {
        private long bytesRead = 0;
        private long totalBytes = -1;
        private boolean succeed = false;
        @Override
        public void progressChanged(ProgressEvent progressEvent) {
            long bytes = progressEvent.getBytes();
            ProgressEventType eventType = progressEvent.getEventType();
            switch (eventType) {
            case TRANSFER_STARTED_EVENT:
                System.out.println("Start to download.....");
                break;
            case RESPONSE_CONTENT_LENGTH_EVENT:
                this.totalBytes = bytes;
                System.out.println(this.totalBytes + " bytes in total will be downloaded to a local file");
                break;
            case RESPONSE_BYTE_TRANSFER_EVENT:
                this.bytesRead += bytes;
                if (this.totalBytes ! = -1) {
```

```
                int percent = (int)(this.bytesRead * 100.0 / this.totalBytes);
                System.out.println(bytes + " bytes have been read at this time, download
progress: " +
                        percent + "%(" + this.bytesRead + "/" + this.totalBytes + ")");
            } else {
                System.out.println(bytes + " bytes have been read at this time, download ratio
: unknown" +
                        "(" + this.bytesRead + "/...)");
            }
            break;
        case TRANSFER_COMPLETED_EVENT:
            this.succeed = true;
            System.out.println("Succeed to download, " + this.bytesRead + " bytes have been
 transferred in total");
            break;
        case TRANSFER_FAILED_EVENT:
            System.out.println("Failed to download, " + this.bytesRead + " bytes have been
transferred");
            break;
        default:
            break;
        }
    }
    public boolean isSucceed() {
        return succeed;
    }
  }
public static void main(String[] args) {
    // This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
    String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
    String accessKeyId = "<yourAccessKeyId>";
    String accessKeySecret = "<yourAccessKeySecret>";
    String bucketName = "<yourBucketName>";
    String objectName = "<yourObjectName>";
    OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
    try {
        // Download with a progress bar displayed.
        ossClient.getObject(new GetObjectRequest(bucketName, objectName).
            <GetObjectRequest>withProgressListener(new GetObjectProgressListener()),
            new File("<yourLocalFile>"));
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Close your OSSClient instance.
    ossClient.shutdown();
}
```

進行状況は、ossClient.putObject、ossClient.getObject、またはossClient.uploadPart で表示できます。 ただし、ossClient.uploadFile または ossClient.downloadFile を使用して進行状況を表示することはできません。

ダウンロードの進捗プログレスバーの完全なコードについては、『GitHub』をご参照ください。

2.8 オブジェクトの管理

2.8.1 概要

このトピックでは、OSS Java SDK でのオブジェクト管理の概要を説明します。

API を使ってバケット内のオブジェクトに対して次の操作を実行できます。

- オブジェクトの存在有無の確認
- オブジェクトの ACL を管理
- オブジェクトメタの管理
- オブジェクトの一覧表示
- オブジェクトの削除
- オブジェクトのコピー
- アーカイブオブジェクトの復元
- シンボリックリンクの管理

2.8.2 指定したオブジェクトの存在有無の確認

このトピックでは、指定したオブジェクトの存在有無の確認方法を説明します。

指定したオブジェクトが存在するかを判断するには、次のコードを使用します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Determine whether the specified object exists. The doesObjectExist includes a parameter: isOnlyInOSS. If the value of this parameter is true, 302 Redirect or OSS mirroring is ignored. Otherwise, 302 Redirect or OSS mirroring is considered.

boolean found = ossClient.doesObjectExist("<yourBucketName>", "<yourObjectName>");
System.out.println();

// Close your OSSClient.
```

```
ossClient.shutdown();
```

2.8.3 オブジェクトの ACL の管理

このトピックでは、オブジェクト のACL を管理する方法について説明します。

次の表は、オブジェクトのアクセス制御リスト (ACL) に含まれるアクセス許可について説明しています。

アクセス許可	説明	値
default	オブジェクトの ACL は、そのバケットの ACL と同一です。	CannedAccessControlList.Default
Private	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。	CannedAccessControlList.Private
Public read	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。 他のユーザーはオブジェクトを読むことしかできません。 この許可は慎重に承認してください。	CannedAccessControlList.PublicRead
Public read-write	すべてのユーザーがオブジェクトを読み書きできます。 この許可は慎重に承認してください。	CannedAccessControlList.PublicReadWrite

オブジェクトの ACL はバケットの ACL よりも優先されます。 たとえば、バケットの ACL が "Private" で、オブジェクトの ACL が "Public read-write" である場合、すべてのユーザーがそのオブジェクトを読み書きできます。 オブジェクトが ACL で設定されていない場合、既定ではオブジェクトの ACL はバケットの ACL です。

オブジェクトの ACL の設定

次のコードを実行して、指定したオブジェクトの ACL を設定できます。

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Configure the ACL for the object to public read.
ossClient.setObjectAcl("<yourBucketName>", "<yourObjectName>", CannedAcce
ssControlList.PublicRead);

// Close your OSSClient.
ossClient.shutdown();
```

オブジェクトの ACL の取得

指定したオブジェクトの ACL を取得するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Obtain the ACL for an object.
ObjectAcl objectAcl = ossClient.getObjectAcl("<yourBucketName>", "<yourObjectName
>");
System.out.println(objectAcl.getPermission().toString());

// Close your OSSClient.
ossClient.shutdown();
```

2.8.4 オブジェクトメタの管理

オブジェクトメタには、HTTP ヘッダーとユーザー定義のオブジェクトメタが含まれます。 詳しくは、「OSS 開発者ガイド」の「オブジェクトメタ」をご参照ください。

HTTP ヘッダーの設定

- HTTP ヘッダーの設定

  HTTP ヘッダーを設定するには、以下のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

String content = "Hello OSS";

// Create Object Meta. You can configure HTTP headers with Object Meta.
```

```
ObjectMetadata meta = new ObjectMetadata();

String md5 = BinaryUtil.toBase64String(BinaryUtil.calculateMd5(content.getBytes()));
// Start object content verification with MD5 values. After the verification is started, the
 MD5 value calculated by OSS is compared with that of an object. If the two values are
different, an error occurs.
meta.setContentMD5(md5);
// Specify the type of content you want to upload. The content type determines the
form and format in which your browser reads an object. The object extension is used
 as the content type if the content type is not specified. If no object extension exists,
the default value application/octet-stream is used.
meta.setContentType("text/plain");
// Specify the name of the content when you download an object.
meta.setContentDisposition("attachment; filename=\"DownloadFilename\"");
// Configure the length for an object. If the actual object length is greater than the
 configured length, only the configured length will be uploaded. If the configured
length is greater than the actual object length, the entire object will be uploaded.
meta.setContentLength(content.length());
// Configure the cache action of a webpage when an object is downloaded.
meta.setCacheControl("Download Action");
// Configure the expiration time (GMT is used).
meta.setExpirationTime(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
// Configure the encoding format when an object is downloaded.
meta.setContentEncoding("utf-8");
// Configure the HTTP header.
meta.setHeader("<yourHeader>", "<yourHeaderValue>");
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload the object.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayI
nputStream(content.getBytes()), meta);

// Close your OSSClient.
ossClient.shutdown();
```

HTTP ヘッダーの詳細については、 「RFC2616」をご参照ください。

- ユーザー定義のオブジェクトメタの構成

  オブジェクト記述にはユーザー定義のオブジェクトメタを定義できます。

  ユーザー定義のオブジェクトメタを設定するには、以下のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

String content = "Hello OSS";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Create Object Meta.
ObjectMetadata meta = new ObjectMetadata();
```

```
// Set property to property-value for user-defined Object Meta. We recommend that you use Base64 encoding method.
meta.addUserMetadata("property", "property-value");

// Upload the object.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()), meta);

// Obtain Object Meta.
ossClient.getObjectMetadata("<yourBucketName>", "<yourObjectName>");

// Close your OSSClient.
ossClient.shutdown();
```

オブジェクトがダウンロードされると、そのオブジェクトメタもダウンロードされます。 1つのオブジェクトに複数のオブジェクトメタを含めることができ、合計最大サイズは 8 KB 未満です。

オブジェクトメタの変更

オブジェクトメタを変更するには、以下のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String sourceBucketName = "<yourSourceBucketName>";
String sourceObjectName = "<yourSourceObjectName>";
String destinationBucketName = "<yourDestinationBucketName>";
String destinationObjectName = "<yourDestinationObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,accessKeySecret);

// Replace the source object with the target object. Call ossClient.copyObject to modify Object Meta.
CopyObjectRequest request = new CopyObjectRequest(sourceBucketName, sourceObjectName, destinationBucketName, destinationObjectName);

ObjectMetadata meta = new ObjectMetadata();
// Specify the type of content you want to upload. The content type determines the form and format that your browser uses to read an object. The object extension is used for the content type if the content type is not specified. If no object extension exists, the default value application/octet-stream is used.
meta.setContentType("text/plain");
// Specify the name of the content when you download an object.
meta.setContentDisposition("Download File Name");
// Configure the cache action of a webpage when an object is downloaded.
meta.setCacheControl("Download Action");
// Configure the expiration time (GMT is used).
meta.setExpirationTime(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
// Configure the encoding type when the object is downloaded.
meta.setContentEncoding("utf-8");
// Configure the HTTP header.
meta.setHeader("<yourHeader>", "<yourHeaderValue>");
```

```
// Set property to property-value for user-defined Object Meta.
meta.addUserMetadata("property", "property-value");
request.setNewObjectMetadata(meta);

//Modify Object Meta.
ossClient.copyObject(request);

// Close your OSSClient.
ossClient.shutdown();
```

オブジェクトメタの取得

次の表は、オブジェクトメタを取得するために使用できるメソッドの一覧です。

方法	説明	利点
ossClient.getSimplifiedObjectMeta	オブジェクトの ETag、サイズ、および LastModified（オブジェクトが最後に変更された時刻）を取得します。	より軽量で高速
ossClient.getObjectMetadata	すべてのメタデータを取得します。	なし

次のコードを実行してオブジェクトメタを取得します。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Obtain partial Object Meta.
SimplifiedObjectMeta objectMeta = ossClient.getSimplifiedObjectMeta("<yourBucketName>", "<yourObjectName>");
System.out.println(objectMeta.getSize());
System.out.println(objectMeta.getETag());
System.out.println(objectMeta.getLastModified());

// Obtain all Object Meta.
ObjectMetadata metadata = ossClient.getObjectMetadata("<yourBucketName>", "<yourObjectName>");
System.out.println(metadata.getContentType());
System.out.println(metadata.getLastModified());
System.out.println(metadata.getExpirationTime());

// Close your OSSClient.
```

```
ossClient.shutdown();
```

2.8.5 オブジェクトの一覧表示

このトピックでは、オブジェクトを一覧表示する方法について説明します。

オブジェクトはアルファベット順に一覧表示されます。 ossClient.listObjects を使用すると、バケット内のオブジェクトを一覧表示できます。 ListObjects では、次のパラメーターを使用できます。

- ObjectListing listObjects (String bucketName)：バケット内のオブジェクトを一覧表示します。 最大 100 個のオブジェクトを一覧表示できます。
- ObjectListing listObjects(String bucketName, String prefix)：バケット内で、指定されたプレフィックスを持つオブジェクトを一覧表示します。 最大 100 個のオブジェクトを一覧表示できます。
- ObjectListing listObjects(ListObjectsRequest listObjectsRequest)：オブジェクトを柔軟にクエリするための複数のフィルタリング機能を提供します。

次の表に ObjectListing のパラメーターを示します。

パラメーター	説明	設定方法
objectSummaries	返される Object Meta。	List<OSSObjectSummary> getObjectSummaries()
prefix	必要なオブジェクトが一覧表示されるようにするためのプレフィックス。	String getPrefix()
delimiter	オブジェクトをグループ化するための区切り文字。	String getDelimiter()
marker	リスト内の最初のオブジェクト。	String getMarker()
maxKeys	一覧表示可能なオブジェクトの最大数。	int getMaxKeys()
nextMarker	次のオブジェクトの初期位置。	String getNextMarker()

パラメーター	説明	設定方法
isTruncated	リストされたオブジェクトが切り捨てられているかどうか。 • オブジェクトが切り捨てられずにすべてリストされている場合、false 値が返されます。 • リストされたオブジェクトが切り捨てられた場合、true 値が返されます。	boolean isTruncated()
commonPrefixes	名前がプレフィックスを共有し、スラッシュ (/) で終わるオブジェクトのセット。	List<String> getCommonPrefixes()
encodingType	レスポンスに使用されるエンコードタイプ。	String getEncodingType()

簡易リスト

次のコードを実行して、指定したバケット内のオブジェクトを一覧表示します。 デフォルトでは最大 100 個のオブジェクトを一覧表示できます。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String KeyPrefix = "<yourKeyPrefix>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// List objects. If you do not configure KeyPrifex, all objects in the bucket are listed. If you configure KeyPrifex, objects with a specified prefix in the bucket are listed.
ObjectListing objectListing = ossClient.listObjects(bucketName, KeyPrefix);
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

ListObjectsRequest によるオブジェクトの一覧表示

ListObjectsRequest のパラメーターを設定することで、オブジェクトを柔軟にクエリすることができます。 次の表に ListObjectsRequest のパラメーターを示します。

パラメーター	説明	設定方法
prefix	返されるオブジェクトに含める必要があるプレフィックス。	setPrefix(String prefix)
delimiter	名前に基づいてオブジェクトをグループ化するためのスラッシュ (/) を指定します。 指定したプレフィックスと、最初のスラッシュ (/) の間にあるサブストリングは commonPrefixes です。	setDelimiter(String delimiter)
marker	marker 値の後ろにあるオブジェクトを一覧表示します。	setMarker(String marker)
maxKeys	一覧表示可能なオブジェクトの最大数。 デフォルト値は 100 です。 最大値は 1,000 です。	setMaxKeys(Integer maxKeys)
encodingType	レスポンス本文内のオブジェクト名のエンコードタイプ。 URL エンコードタイプのみがサポートされています。	setEncodingType(String encodingType)

- 指定した数のオブジェクトの一覧表示

  次のコードを実行して、指定した数のオブジェクトを一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
```

```
// Configure the maximum number.
final int maxKeys = 200;
// List objects.
ObjectListing objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName
).withMaxKeys(maxKeys));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

- 指定したプレフィックスのオブジェクトの一覧表示

  次のコードを実行して、指定したプレフィックスを持つオブジェクトを一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Specify a prefix.
final String keyPrefix = "<yourkeyPrefix>";

// List objects with a specified prefix. By default, a maximum of 100 objects can be
listed.
ObjectListing objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName
).withPrefix(keyPrefix));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

- marker によるオブジェクトの一覧表示

  marker パラメーターは、一覧を開始するオブジェクトの名前を示します。 次のコードを実行して、一覧を開始するオブジェクト (maker で指定) を指定します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
```

```
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);
final String marker = "<yourMarker>";

// List objects placed after the object (specified with marker). By default, a maximum
of 100 objects can be listed.
ObjectListing objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName
).withMarker(marker));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

- すべてのオブジェクトを 1 ページまたは複数ページに一覧表示

  次のコードを使用して、すべてのオブジェクトを 1 ページまたは複数ページに一覧表示します。 各ページに表示可能なオブジェクトの最大数は、maxKeys で指定されています。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

final int maxKeys = 200;
String nextMarker = null;
ObjectListing objectListing;

do {
    objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).
withMarker(nextMarker).withMaxKeys(maxKeys));

    List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
    for (OSSObjectSummary s : sums) {
        System.out.println("\t" + s.getKey());
    }

    nextMarker = objectListing.getNextMarker();

} while (objectListing.isTruncated());

// Close your OSSClient.
```

```
ossClient.shutdown();
```

- 指定したプレフィックスのオブジェクトを 1 ページまたは複数ページに一覧表示

  次のコードを実行して、指定したプレフィックスのオブジェクトを 1 ページまたは複数ページに一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// A maximum of 200 objects can be listed on each page.
final int maxKeys = 200;
final String keyPrefix = "<yourkeyPrefix>";
String nextMarker = "<yourNextMarker>";
ObjectListing objectListing;

do {
    objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).
            withPrefix(keyPrefix).withMarker(nextMarker).withMaxKeys(maxKeys));

    List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
    for (OSSObjectSummary s : sums) {
        System.out.println("\t" + s.getKey());
    }

    nextMarker = objectListing.getNextMarker();

} while (objectListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

- エンコードするオブジェクト名の指定

  名前に次の特殊文字が含まれている場合は、オブジェクト名をエンコードする必要があります。 OSS は、URL エンコードをサポートしています。

  - 一重引用符 (')
  - 二重引用符 (“)
  - アンパサンド (&)
  - 山かっこ (< >)
  - 一時停止マーカー (、)
  - 漢字、平仮名、片仮名

  次のコードを実行して、指定したオブジェクト名をエンコードします。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

final int maxKeys = 200;
final String keyPrefix = "<yourkeyPrefix>";
String nextMarker = "<yourNextMarker>";
ObjectListing objectListing;

do {
    ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);
    listObjectsRequest.setPrefix(keyPrefix);
    listObjectsRequest.setMaxKeys(maxKeys);
    listObjectsRequest.setMarker(nextMarker);

    // Specify the name of an object for encoding.
    listObjectsRequest.setEncodingType("url");

    objectListing = ossClient.listObjects(listObjectsRequest);

    // Decode the object.
    for (OSSObjectSummary objectSummary: objectListing.getObjectSummaries()) {
        System.out.println("Key:" + URLDecoder.decode(objectSummary.getKey(), "UTF-8"));
    }

    // Decode the commonPrefixes parameter.
    for (String commonPrefixes: objectListing.getCommonPrefixes()) {
```

```
        System.out.println("CommonPrefixes:" + URLDecoder.decode(commonPrefixes, "
UTF-8"));
    }

    // Decode the nextMarker parameter.
    if (objectListing.getNextMarker() ! = null) {
        nextMarker = URLDecoder.decode(objectListing.getNextMarker(), "UTF-8");
    }
} while (objectListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

フォルダーのシミュレーション

OSS は、フォルダーをサポートしません。 すべての要素は、オブジェクトとして保存されます。 OSS コンソールでフォルダーをシミュレートするには、名前がスラッシュ (/) で終わる 0 MB のオブジェクトを作成します。 このオブジェクトは、アップロードとダウンロードが可能です。 デフォルトでは、名前がスラッシュ (/) で終わるオブジェクトは、フォルダーとして OSS コンソールに表示されます。

delimiter パラメーターと prefix パラメーターは、フォルダー機能をシミュレートするために使用できます。

- Prefix：フォルダーの名前としてプレフィックスを指定します。 フォルダーを使用して、このプレフィックスで始まるファイル (フォルダー内) とサブフォルダー (フォルダー内のディレクトリ) を一覧表示します。 これらのファイルとサブフォルダーは、Objects リストに含まれています。
- Delimiter：区切り文字がスラッシュ (/) として指定されている場合、ファイルとサブフォルダー (ディレクトリー) のみが表示されます。 サブフォルダー (ディレクトリー) は、CommonPrefixes リストに含まれていますが、サブフォルダー内のファイルとフォルダーは表示されません。

フォルダーの詳細は、「フォルダーシミュレーション」をご参照ください。 オブジェクトを作成するコード全体は、GitHub をご参照ください。

オブジェクト oss.jpg、fun/test.jpg、fun/movie/001.avi、および fun/movie/007.avi がバケットに保存されているとします。 フォルダーの区切り文字として、スラッシュ (/) を使用します。 次の例は、フォルダー機能をシミュレートする方法を示しています。

- バケット内のすべてのオブジェクトの一覧表示

  次のコードを実行して、バケット内のすべてのオブジェクトを一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Create ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);

// List objects.
ObjectListing listing = ossClient.listObjects(listObjectsRequest);

// Print all objects.
System.out.println("Objects:");
for (OSSObjectSummary objectSummary : listing.getObjectSummaries()) {
    System.out.println(objectSummary.getKey());
}

// Print all commonPrefixes.
System.out.println("CommonPrefixes:");
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}

// Disable the OSSClient.
// Close your OSSClient.
```

返される結果は、次のとおりです。

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg
oss.jpg

CommonPrefixes:
```

- 指定したフォルダー内のすべてのオブジェクトの一覧表示

  次のコードを実行して、指定したフォルダー内のすべてのオブジェクトを一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
```

```
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Create ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);
// Configure the Prefix parameter to obtain all objects in the fun folder.
listObjectsRequest.setPrefix("fun/");

// Recursively list all objects in the fun folder.
ObjectListing listing = ossClient.listObjects(listObjectsRequest);

// Print all objects.
System.out.println("Objects:");
for (OSSObjectSummary objectSummary : listing.getObjectSummaries()) {
   System.out.println(objectSummary.getKey());
}

// Print all commonPrefixes.
System.out.println("\nCommonPrefixes:");
for (String commonPrefix : listing.getCommonPrefixes()) {
   System.out.println(commonPrefix);
}

// Close your OSSClient.
ossClient.shutdown();
```

返される結果は、次のとおりです。

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg

CommonPrefixes:
```

- フォルダー内のオブジェクトとサブフォルダーの一覧表示

  次のコードを実行して、フォルダー内のオブジェクトとサブフォルダーを一覧表示します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret
);

// Create ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);

// Set forward slashes (/) as the delimiter for folders.
listObjectsRequest.setDelimiter("/");
```

```
// List all objects and folders in the fun folder.
listObjectsRequest.setPrefix("fun/");

ObjectListing listing = ossClient.listObjects(listObjectsRequest);

// Print all objects.
System.out.println("Objects:");
// List objects in the fun folder in a objectSummaries list.
for (OSSObjectSummary objectSummary : listing.getObjectSummaries()) {
    System.out.println(objectSummary.getKey());
}

// Print all commonPrefixes.
System.out.println("\nCommonPrefixes:");
// commonPrefixs lists all subfolders in the fun folder. The fun/movie/001.avi and fun
/movie/007.avi objects are not listed, because they are objects in the movie folder (a
subfolder of the fun folder).
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}

// Close your OSSClient.
ossClient.shutdown();
```

返される結果は、次のとおりです。

```
Objects:
fun/test.jpg

CommonPrefixes:
fun/movie/
```

2.8.6 オブジェクトの削除

警告：

削除されたオブジェクトは回復できないため、慎重に操作を実行する必要があります。

オブジェクトの削除

単一のオブジェクトを削除するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
```

```
// Delete an object.
ossClient.deleteObject(bucketName, objectName);

// Close your OSSClient.
ossClient.shutdown();
```

複数のオブジェクトの削除

一度に最大 1,000 個のオブジェクトを削除できます。 オブジェクトの削除には、詳細 (verbose) と簡易 (quiet) モードの 2 つのモードがあります。

- verbose: 削除に成功したオブジェクトの一覧を返します。 デフォルトのモードは、verbose です。
- quiet: 削除に失敗したオブジェクトのリストを返します。

次の表では、DeleteObjectsRequest パラメーターを解説します。

パラメータ	説明	設定方法
Keys	削除したいオブジェクトを指定します。	setKeys(List<String>)
quiet	削除結果を指定します。 True は、quiet を示します。 False は、verbose を示します。 デフォルトモードでは、verbose が設定されています。	setQuiet(boolean)
encodingType	返却されるオブジェクトの名前のエンコードです。 HTTPS のみがサポートされます。	setEncodingType(String)

以下の表は、DeleteObjectsResult パラメーターを解説します。

パラメータ	説明	設定方法
deletedObjects	削除結果を指定します。 verbose は、削除に成功したオブジェクトの一覧を示します。quiet: 削除に失敗したオブジェクトのリストを示します。	List<String> getDeleted Objects()
encodingType	deletedObjects 内のオブジェクトの名前のエンコード。	getEncodingType()

複数のオブジェクトを一括で削除するには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Delete objects.
List<String> keys = new ArrayList<String>();
keys.add("key0");
keys.add("key1");
keys.add("key2");

DeleteObjectsResult deleteObjectsResult = ossClient.deleteObjects(new DeleteObje
ctsRequest(bucketName).withKeys(keys));
List<String> deletedObjects = deleteObjectsResult.getDeletedObjects();

// Close your OSSClient.
ossClient.shutdown();
```

複数のオブジェクトを一括で削除するための完全なコードは、『GitHub』をご参照ください。

2.8.7 アーカイブオブジェクトの復元

このトピックでは、アーカイブオブジェクトを復元する方法について説明します。

アーカイブオブジェクトを復元するための完全なコードについては、『GitHub』をご参照ください

アーカイブオブジェクトを読み込む前に復元する必要があります。 非アーカイブオブジェクトに対して、"restoreObject "を呼び出さないでください。

アーカイブオブジェクトの状態変換プロセスは次のとおりです。

1. アーカイブオブジェクトはフリーズ状態です。
2. 復元のために送信した後、サーバーはオブジェクトを復元します。 オブジェクトは復元中の状態です。
3. 復元後にオブジェクトを読むことができます。 オブジェクトが復元された状態は、既定で 1 日続きます。 この期間は最大 7 日間まで延長できます。 この期間が終了すると、オブジェクトはフリーズ状態に戻ります。

次のコードを実行してオブジェクトを復元します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

ObjectMetadata objectMetadata = ossClient.getObjectMetadata(bucketName, key);

// Verify whether the object is an archive object.// Verify that the file is an archive file.
StorageClass storageClass = objectMetadata.getObjectStorageClass();
if (storageClass == StorageClass.Archive) {
    // Restore the object.
    ossClient.restoreObject(bucketName, objectName);

    // Wait until the object is restored.
    do {
        Thread.sleep(1000);
        objectMetadata = ossClient.getObjectMetadata(bucketName, objectName);
    } while (! objectMetadata.isRestoreCompleted());
}

// Obtain the restored object.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
ossObject.getObjectContent().close();

// Close your OSSClient.
ossClient.shutdown();
```

アーカイブストレージクラスの詳細については、 「ストレージクラスの概要」をご参照ください。

2.9 サーバー側暗号化

OSS では、アップロードされたデータのサーバー側暗号化をサポートしています。 したがって、ユーザーデータがアップロードされると、OSS でデータを暗号化し、永続的に保存します。 ユーザーがデータをダウンロードする際、データは自動的に復号化され、元のデータが返されます。 レスポンスヘッダーには、サーバー側でデータが暗号化されたことが記載されます。

さまざまな適用シナリオで、次のサーバー側暗号化方式を利用できます。

- KMS が管理する CMK を使用したサーバー側暗号化 (SSE-KMS) による暗号化と復号化

  オブジェクトをアップロードする際、CMK ID、または KMS で管理されるデフォルト CMK を使用して、大量のデータを暗号化および復号化できます。 この方式は、暗号化や復号化の際、ネットワーク経由でユーザーデータを KMS サーバーに送信する必要がないので、コスト効率に優れています。

  :

  CMK を使用してデータを暗号化または復号化する場合、API 呼び出し料金が発生します。

- OSS によるフルマネージドのサーバー側暗号化 (SSE-OSS)

  オブジェクトがアップロードされると、OSS によるフルマネージドの AES256 を使用して、サーバー側で暗号化されます。 この方式では、AES256 を使用して、オブジェクトごとに異なるキーで暗号化します。 さらに個々のキーは、定期的に更新される CMK で暗号化されるので、より安全です。 この方式は、大量のデータの暗号化や復号化に適用されます。

:

- 1 つのオブジェクトに対して同時に使用できるサーバー側暗号化方式は 1 つのみです。
- バケットにサーバー側暗号化を設定する場合、バケットをアップロードまたはコピーする際、単一のオブジェクトに暗号化方式を設定できます。 この場合、オブジェクトに設定された暗号化方式が優先されます。 詳細は、「PutObject」をご参照ください。
- サーバー側暗号化の詳細は、「サーバー側暗号化」をご参照ください。

バケットのサーバー側暗号化の設定

次のコードを実行して、バケットにデフォルトの暗号化方式を設定できます。 方式が設定された後、暗号化方法が設定されていないバケットにアップロードされたオブジェクトは、設定済みのデフォルトの暗号化方式で暗号化されます。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM user account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Configures server-side encryption for the bucket.
ServerSideEncryptionByDefault applyServerSideEncryptionByDefault = new ServerSideEncryptionByDefault(SSEAlgorithm.KMS);
```

```
applyServerSideEncryptionByDefault.setKMSMasterKeyID("<yourTestKmsId>");
ServerSideEncryptionConfiguration sseConfig = new ServerSideEncryptionConfiguration
();
sseConfig.setApplyServerSideEncryptionByDefault(applyServerSideEncryptionByDefault);
SetBucketEncryptionRequest request = new SetBucketEncryptionRequest("<yourBucket
Name>", sseConfig);
ossClient.setBucketEncryption(request);

// Closes the OSSClient instance.
ossClient.shutdown();
```

バケットのサーバー側暗号化設定の取得

次のコードを実行して、バケットのサーバー側暗号化設定を取得できます。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM user
 account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Obtains the server-side encryption settings of the bucket.
ServerSideEncryptionConfiguration sseConfig = ossClient.getBucketEncryption("<
yourBucketName>");
System.out.println("get Algorithm: " + sseConfig.getApplyServerSideEncryptionByDefault
().getSSEAlgorithm());
System.out.println("get kmsid: " + sseConfig.getApplyServerSideEncryptionByDefault().
getKMSMasterKeyID());

// Closes the OSSClient instance.
ossClient.shutdown();
```

バケットのサーバー側暗号化設定の削除

次のコードを実行して、バケットのサーバー側暗号化設定を削除できます。

```
// This example uses the China East 1 (Hangzhou) endpoint. Specify the actual endpoint
based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM user
 account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Creates an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Deletes the server-side encryption settings of the bucket.
ossClient.deleteBucketEncryption("<yourBucketName>");

// Closes the OSSClient instance.
```

```
ossClient.shutdown();
```

2.10 データ暗号化

2.11 データのセキュリティ

OSS Java SDK は、ファイル (オブジェクト) のアップロード中、ダウンロード中、およびレプリケーション中にデータのセキュリティを保証するため、サーバーとクライアントでデータの整合性チェックとファイルの暗号化を提供します。

データの整合性チェック

OSS Java SDK は、オブジェクト (ファイル) のアップロード、ダウンロード、およびレプリケーションの整合性を保証するため、MD5 認証と巡回冗長検査 (CRC) を提供します。

- MD5 認証

  ファイルをアップロードするため、Content-MD5 を設定した場合、OSS は受信したコンテンツに基づき、MD5 値を計算します。 MD5 値がアップロードされたファイルの値と一致しない場合は、InvalidDigest が返されます。 これにより、データの完全性が保証されます。

  ファイルのアップロード中に MD5 認証を行うには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a string.
String content = "Hello OSS";

ObjectMetadata meta = new ObjectMetadata();
// Configure MD5 authentication.
String md5 = BinaryUtil.toBase64String(BinaryUtil.calculateMd5(content.getBytes()));
meta.setContentMD5(md5);

ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()), meta);

// Close your OSSClient instance.
```

```
ossClient.shutdown();
```

注：

MD5 認証 は、putObject、getObject、appendObject、postObject、および uploadPart を対象に行います。

- CRC64

CRC は、データの整合性を保証するため、既定ではオブジェクト (ファイル) のアップロード中、ダウンロード中、およびレプリケーション中に有効になります。

次のコードを実行し、ファイルのアップロード中に CRC が有効になっているかどうかを確認します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a string.
String content = "Hello OSS";

PutObjectResult result = ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()));

// Check whether CRC is enabled. True indicates it is enabled while false indicates it is disabled.
if (ossClient.getClientConfiguration().isCrcCheckEnabled()) {
 // Check whether CRC takes effect based on the response.
   OSSUtils.checkChecksum(result.getClientCRC(), result.getServerCRC(), result.getRequestId());
 }
// Close your OSSClient instance.
ossClient.shutdown();
```

注：

- CRC64 認証は、putObject、getObject、appendObject、および uploadPart を対象に行います。
- CRC64 認証では CPU スペースを使用するため、アップロードとダウンロードの進行状況に影響を与えます。

サーバー上のファイルの暗号化

ファイルのアップロード時にファイルを暗号化してサーバーに保存するため、オブジェクトメタの serverSideEncryption パラメーターを AES256 または KMS に設定します。 詳しくは、OSS 開発者ガイド内の、「サーバー側の暗号化」をご参照ください。

ファイルのアップロード中にサーバー上でファイルの暗号化を行うには、次のコードを実行します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Upload a string.
String content = "Hello OSS";

ObjectMetadata meta = new ObjectMetadata();
// Set the server-side encryption code to AES256 for the file.
meta.setServerSideEncryption(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);

ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()), meta);

// Close your OSSClient instance.
ossClient.shutdown();
```

クライアント側の暗号化

クライアント上のファイルの暗号化は、ユーザーのローカルデバイス上で行うことができます。復号化は、OSS にアップロードされたファイルをローカルデバイスにダウンロードして行います。 詳しくは、OSS 開発者ガイド内の、「クライアント側の暗号化 SDKのイントロダクション」をご参照ください。

2.12 画像処理

Image Processing (IMG) は、大規模で安全、低コストで、信頼性の高い画像処理サービスです。ソース画像 が OSS にアップロードされると、簡易な RESTful API を通して、いつでも、どこからでも、どのインターネットデバイス上の画像でも処理できます。

IMG の詳細については、「Image Processing」をご参照ください。

画像処理の完全なコードについては、『xrefGitHubxref』をご参照ください。

基本機能

OSS は以下の IMG 機能を提供します。

- ドミナントイメージトーンの取得
- 画像フォーマット変換
- 画像サイズの変更
- 内接円の調整
- 画像の向きの調整
- 画像輝度
- 透かしの追加: 画像、テキスト、または画像テキストの透かしを別の画像に追加します。
- 画像処理
- 画像処理アクセスルール: 複数の IMG 関数を呼び出します。

使用方法

IMG"は標準の HTTP GET を使用します。 URL のクエリ文字列に IMG パラメータ を設定します。

イメージオブジェクトの ACL が "非公開読み書き" の場合、承認されたユーザーだけがアクセスを許可されます。

- 匿名アクセス

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
  ```

Parameter	説明
bucket	バケットの名前を指定します。
endpoint	リージョンへのアクセスに使用されるエンドポイントを指定します。
object	画像オブジェクトの名前を指定します。
image	IMG の予約済み識別子を指定します。
action	拡大縮小、トリミング、回転など、画像に対する操作を指定します。

Parameter	説明
param	画像に対する操作を示すパラメーターを指定します。

- 基本操作

  たとえば、画像を 100 PX の幅に拡大縮小します。 比率に基づいて高さを調整します。

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- カスタマイズされた画像スタイル

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  ■ style: カスタマイズされた画像スタイルの予約済み識別子を指定します。

  ■ yourStyleName：カスタム画像スタイルの名前を指定します。 OSS コンソールで作成されたルールで指定された名前です。

  例：

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- カスケード処理

  カスケード処理では、画像に対して複数の操作を順番に実行します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  例：

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- HTTPS アクセス

  IMG は HTTPS を介したアクセスをサポートしています。 例：

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

• 許可されたアクセス

  アクセス許可により、イメージスタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

次のコードを使用して IMG の署名付き URL を生成します。

```
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Customize the image style.
String style = "image/resize,m_fixed,w_100,h_100/rotate,90";
// Set the expiration time to 10 minutes.
Date expiration = new Date(new Date().getTime() + 1000 * 60 * 10 );
GeneratePresignedUrlRequest req = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.GET);
req.setExpiration(expiration);
req.setProcess(style);
URL signedUrl = ossClient.generatePresignedUrl(req);
System.out.println(signedUrl);
// Close your OSSClient.
ossClient.shutdown();sClient.shutdown();
```

- SDK を使用したアクセス

  SDK を使用して、画像オブジェクトにアクセスして処理します。

  SDK を使用すると、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

  - 基本操作

    次のコードを実行して、画像に対して基本操作を実行します。

    ```
    // This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
    String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
    String accessKeyId = "<yourAccessKeyId>";
    String accessKeySecret = "<yourAccessKeySecret>";
    String bucketName = "<yourBucketName>";
    String objectName = "<yourObjectName>";
    // Create an OSSClient instance.
    OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
    // Scale the image.
    String style = "image/resize,m_fixed,w_100,h_100";
    GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
    request.setProcess(style);
    ossClient.getObject(request, new File("example-resize.jpg"));
    // Crop the image.
    style = "image/crop,w_100,h_100,x_100,y_100,r_1";
    ```

```
request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
ossClient.getObject(request, new File("example-crop.jpg"));
// Rotate the image.
style = "image/rotate,90";
request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
ossClient.getObject(request, new File("example-rotate.jpg"));
// Sharpen the image.
style = "image/sharpen,100";
request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
ossClient.getObject(request, new File("example-sharpen.jpg"));
// Add watermarks.
style = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ";
request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
ossClient.getObject(request, new File("example-watermark.jpg"));
// Convert the image format.
style = "image/format,png";
request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
ossClient.getObject(request, new File("example-format.png"));
// Obtain image information.
style = "image/info";
request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
ossClient.getObject(request, new File("example-info.txt"));
// Close your OSSClient.
ossClient.shutdown();
```

- イメージスタイルのカスタマイズ

  次のコードを実行して画像スタイルをカスタマイズします。

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Customize the image style.
String style = "style/<yourCustomStyleName>";
GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);

ossClient.getObject(request, new File("example-new.jpg"));

// Close your OSSClient.
```

```
ossClient.shutdown();
```

- カスケード処理

  次のコードを実行して、画像に対してカスケード処理を実行します:

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// Perform cascade operations.
String style = "image/resize,m_fixed,w_100,h_100/rotate,90";
GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);

ossClient.getObject(request, new File("example-new.jpg"));

// Disable the OSSClient.
ossClient.shutdown();
```

画像処理の持続性

画像処理の持続性のために次のコードを実行します:

```
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance.
To create a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String sourceImage = "<yourSourceImageName>";

try {
 // Create an OSSClient instance.
 OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

 // Image processing persistence: Scale the image.
 StringBuilder sbStyle = new StringBuilder();
 Formatter styleFormatter = new Formatter(sbStyle);
 String styleType = "image/resize,m_fixed,w_100,h_100";
 String targetImage = "example-resize.png";
 styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
   BinaryUtil.toBase64String(targetImage.getBytes()),
   BinaryUtil.toBase64String(bucketName.getBytes()));
 System.out.println(sbStyle.toString());
```

```
 ProcessObjectRequest request = new ProcessObjectRequest(bucketName, sourceImage,
sbStyle.toString());
 GenericResult processResult = ossClient.processObject(request);
 String json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF
-8");
 processResult.getResponse().getContent().close();
 System.out.println(json);

 // Image processing persistence: Clip the image.
 sbStyle.delete(0, sbStyle.length());
 styleType = "image/crop,w_100,h_100,x_100,y_100,r_1";
 targetImage = "example-crop.png";
 styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
   BinaryUtil.toBase64String(targetImage.getBytes()),
   BinaryUtil.toBase64String(bucketName.getBytes()));
 System.out.println(sbStyle.toString());
 request = new ProcessObjectRequest(bucketName, sourceImage, sbStyle.toString());
 processResult = ossClient.processObject(request);
 json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF-8");
 processResult.getResponse().getContent().close();
 System.out.println(json);

 // Image processing persistence: Rotate the object.
 sbStyle.delete(0, sbStyle.length());
 styleType = "image/rotate,90";
 targetImage = "example-rotate.png";
 styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
   BinaryUtil.toBase64String(targetImage.getBytes()),
   BinaryUtil.toBase64String(bucketName.getBytes()));
 request = new ProcessObjectRequest(bucketName, sourceImage, sbStyle.toString());
 processResult = ossClient.processObject(request);
 json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF-8");
 processResult.getResponse().getContent().close();
 System.out.println(json);

 // Image processing persistence: Sharpen the image.
 sbStyle.delete(0, sbStyle.length());
 styleType = "image/sharpen,100";
 targetImage = "example-sharpen.png";
 styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
   BinaryUtil.toBase64String(targetImage.getBytes()),
   BinaryUtil.toBase64String(bucketName.getBytes()));
 request = new ProcessObjectRequest(bucketName, sourceImage, sbStyle.toString());
 processResult = ossClient.processObject(request);
 json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF-8");
 processResult.getResponse().getContent().close();
 System.out.println(json);

 // Image processing persistence: Add watermarks.
 sbStyle.delete(0, sbStyle.length());
 styleType = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ";
 targetImage = "example-watermark.png";
 styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
   BinaryUtil.toBase64String(targetImage.getBytes()),
   BinaryUtil.toBase64String(bucketName.getBytes()));
 request = new ProcessObjectRequest(bucketName, sourceImage, sbStyle.toString());
 processResult = ossClient.processObject(request);
 json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF-8");
 processResult.getResponse().getContent().close();
 System.out.println(json);

 // Image processing persistence: Convert the image format.
 sbStyle.delete(0, sbStyle.length());
 styleType = "image/format,jpg";
```

```
 targetImage = "example-formatconvert.jpg";
 styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
   BinaryUtil.toBase64String(targetImage.getBytes()),
   BinaryUtil.toBase64String(bucketName.getBytes()));
 request = new ProcessObjectRequest(bucketName, sourceImage, sbStyle.toString());
 processResult = ossClient.processObject(request);
 json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF-8");
 processResult.getResponse().getContent().close();
 System.out.println(json);

} catch (Exception e) {
 e.printStackTrace();
}
```

IMG ツール

IMG ビューアー (ImageStyleViever) を使用すると、IMG の結果を直接表示できます。

2.13 例外処理

OSS Java SDK には、ClientException と OSSException の 2 種類の例外があります。 ともに RuntimeException のプロパティを共有します。

例外処理の例

次のコードを実行し、例外を処理します。

```
try {
    // Perform operations on OSS, for example, object uploads.
    ossClient.putObject(...);
} catch (Exception e) {
    System.out.println("Caught an OSSException, which means your request made it to
OSS, "
            + "but was rejected with an error response for some reason.") ;
    System.out.println("Error Message: " + oe.getErrorCode());
    System.out.println("Error Code:       " + oe.getErrorCode());
    System.out.println("Request ID:      " + oe.getRequestId());
    System.out.println("Host ID:           " + oe.getHostId());
} catch (ClientException e) {
    System.out.println("Caught an ClientException, which means the client encountered "
            + "a serious internal problem while trying to communicate with OSS, "
            + "such as not being able to access the network.") ;
    System.out.println("Error Message: " + ce.getMessage());
} finally {
    if (ossClient ! = null) {
        ossClient.shutdown();
    }
}
```

ClientException

ClientException 例外は、クライアントがリクエストを送信しようとしたとき、およびデータが OSS に送信されたときに発生します。 たとえば、リクエストが送信されると、利用できないネットワーク接続が原因で ClientException が発生することがあります。 I/O 例外のため、ファイルのアップロード中に ClientException が発生することがあります。

OSSException

OSSException は、サーバー例外を示します。 サーバーエラーメッセージが解析されることが原因で、例外が発生します。 OSSException には、OSS から返されたエラーコードと情報が含まれます。 OSSException は問題を見つけて処理するために使用されます。

次の表に、一般的な OSSException の情報を示します。

パラメーター	説明
Code	OSS から返されたエラーコードを特定します。
Message	OSS から返される詳細なエラー情報を特定します。
RequestId	UUID を特定します。 UUID は固有のものであり、リクエストを識別するために使用されます。 問題が解決しない場合は、OSS 開発者に "RequestId" を送り、問い合わせます。
HostId	OSS クラスターを識別します。 "HostId" の値が、リクエストで使用されているホスト (バケットにアクセスするためのエンドポイント) と一致していることを確認してください。

OSS エラーコード

エラーコード	説明	**HTTP** ステータスコード
AccessDenied	アクセスが拒否されました。	403
BucketAlreadyExists	バケットが既に存在します。	409
BucketNotEmpty	バケットが空ではありません。	409
EntityTooLarge	エンティティサイズが上限を超えています。	400
EntityTooSmall	エンティティサイズが下限を下回っています。	400
FileGroupTooLarge	ファイルグループの合計サイズが上限を超えています。	400
FilePartNotExist	そのようなオブジェクトパーツは存在しません。	400
FilePartStale	オブジェクトパーツは有効期限切れです。	400

エラーコード	説明	**HTTP** ステータスコード
InvalidArgument	パラメーターの形式が無効です。	400
InvalidAccessKeyId	そのような AccessKeyId は存在しません。	403
InvalidBucketName	バケット名が無効です。	400
InvalidDigest	ダイジェストが無効です。	400
InvalidObjectName	オブジェクト名が無効です。	400
InvalidPart	パーツが無効です。	400
InvalidPartOrder	パーツの順序が無効です。	400
InvalidTargetBucketForLogging	ログファイルを格納するバケットが無効です。	400
InternalError	内部 OSS エラーです。	500
MalformedXML	XML 形式が無効です。	400
MethodNotAllowed	メソッドがサポートされていません。	405
MissingArgument	パラメーターが設定されていません。	411
MissingContentLength	コンテンツの長さが設定されていません。	411
NoSuchBucket	そのようなバケットは存在しません。	404
NoSuchKey	そのようなオブジェクトは存在しません。	404
NoSuchUpload	そのようなアップロード ID は存在しません。	404
NotImplemented	メソッドが実装されていません。	501
PreconditionFailed	事前条件エラーです。	412
RequestTimeTooSkewed	OSSClient に設定されている現地時間が、OSS サーバーに設定されている時間から 15 分以上ずれています。	403
RequestTimeout	リクエストがタイムアウトされました。	400

エラーコード	説明	**HTTP** ステータスコード
SignatureDoesNotMatch	署名が不一致です。	403
InvalidEncryptionAlgorithmError	指定された暗号化アルゴリズム（エントロピー符号化タイプ）が無効です。	400

2.14 よくある質問

Jar パッケージの競合

- 原因の分析

  OSS Java SDK を使用しているときに次のエラーが発生した場合、プロジェクトでパッケージが競合している可能性があります。

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/http/ssl/TrustStrategy
    at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
    at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
    at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:77)
Caused by: java.lang.ClassNotFoundException: org.apache.http.ssl.TrustStrategy
    at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
    ... 3 more
```

  または、次のエラーが発生した場合。

```
Exception in thread "main" java.lang.NoSuchFieldError: INSTANCE
 at org.apache.http.impl.io.DefaultHttpRequestWriterFactory.<init>(DefaultHttpRequestWriterFactory.java:52)
 at org.apache.http.impl.io.DefaultHttpRequestWriterFactory.<init>(DefaultHttpRequestWriterFactory.java:56)
 at org.apache.http.impl.io.DefaultHttpRequestWriterFactory.<clinit>(DefaultHttpRequestWriterFactory.java:46)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<init>(ManagedHttpClientConnectionFactory.java:82)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<init>(ManagedHttpClientConnectionFactory.java:95)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<init>(ManagedHttpClientConnectionFactory.java:104)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<clinit>(ManagedHttpClientConnectionFactory.java:62)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager$InternalConnectionFactory.<init>(PoolingHttpClientConnectionManager.java:572)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:174)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:158)
```

```
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:149)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:125)
 at com.aliyun.oss.common.comm.DefaultServiceClient.createHttpClientConnectionManager(DefaultServiceClient.java:237)
 at com.aliyun.oss.common.comm.DefaultServiceClient.<init>(DefaultServiceClient.java:78)
 at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
 at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
 at OSSManagerImpl.upload(OSSManagerImpl.java:42)
 at OSSManagerImpl.main(OSSManagerImpl.java:63)
```

これは、OSS Java SDK は Apache httpclient 4.4.1 を使用しているのに対し、プロジェクトでは、Apache httpclient もしくは commons-httpclient jar を使用しているため、Apache httpclient 4.4.1 と競合しています。 プロジェクトで使用されている jar パッケージとそのバージョンを確認するため、OSS Java SDK を開き、`mvn dependency:tree` を実行します。 次の図は、Apache httpclient 4.3 を使用しているプロジェクトを示しています。

```
[INFO] --- maven-dependency-plugin:2.2:tree (default-cli) @ maven-demo ---
[INFO] com.aliyun.oss:maven-demo:jar:0.1.1-SNAPSHOT
[INFO] +- junit:junit:jar:4.10:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.1:test
[INFO] +- org.apache.httpcomponents:httpclient:jar:4.3:compile
[INFO] |  +- org.apache.httpcomponents:httpcore:jar:4.3:compile
[INFO] |  +- commons-logging:commons-logging:jar:1.1.3:compile
[INFO] |  \- commons-codec:commons-codec:jar:1.6:compile
[INFO] \- com.aliyun.oss:aliyun-sdk-oss:jar:2.2.1:compile
[INFO]    +- org.jdom:jdom:jar:1.1:compile
[INFO]    \- net.sf.json-lib:json-lib:jar:jdk15:2.4:compile
[INFO]       +- commons-beanutils:commons-beanutils:jar:1.8.0:compile
[INFO]       +- commons-collections:commons-collections:jar:3.2.1:compile
[INFO]       +- commons-lang:commons-lang:jar:2.5:compile
[INFO]       \- net.sf.ezmorph:ezmorph:jar:1.0.6:compile
```

- パッケージの競合を解消するには、2 つの方法があります。
  - 統合バージョンを使用します。 プロジェクトで Apache httpclient 4.4.1 と競合するバージョンを使用している場合、Apache httpclient 4.4.1 を使用します。また、pom.xml ファイル内にある、他のバージョンの Apache httpclient と依存関係にある部分を削除します。プロジェクトで commons-httpclient を使用している場合についても、jar パッケージの競合が発生する可能性があります。 この場合は、commons-httpclient を削除します。
  - 依存関係にある競合を削除します。 プロジェクトが複数のサードパーティーと依存関係にある場合、各サードパーティーは複数のバージョンの Apache httpclient を使用している可

能性があるため、プロジェクトに依存関係の競合が存在するかもしれません。 競合を解決するため、"exclusion" を使用します。 詳しくは、「Maven ガイド」をご参照ください。

OSS Java SDK は次の jar パッケージバージョンに依存しています。 競合の解決方法は、httpclien と同様です。

```
[INFO] com.aliyun.oss:maven-demo:jar:0.1.1-SNAPSHOT
[INFO] +- junit:junit:jar:4.10:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.1:test
[INFO] \- com.aliyun.oss:aliyun-sdk-oss:jar:2.2.1:compile
[INFO]    +- org.apache.httpcomponents:httpclient:jar:4.4.1:compile
[INFO]    |  +- org.apache.httpcomponents:httpcore:jar:4.4.1:compile
[INFO]    |  +- commons-logging:commons-logging:jar:1.2:compile
[INFO]    |  \- commons-codec:commons-codec:jar:1.9:compile
[INFO]    +- org.jdom:jdom:jar:1.1:compile
[INFO]    \- net.sf.json-lib:json-lib:jar:jdk15:2.4:compile
[INFO]       +- commons-beanutils:commons-beanutils:jar:1.8.0:compile
[INFO]       +- commons-collections:commons-collections:jar:3.2.1:compile
[INFO]       +- commons-lang:commons-lang:jar:2.5:compile
[INFO]       \- net.sf.ezmorph:ezmorph:jar:1.0.6:compile
```

ファイルの不足

- 原因分析

  OSS Java SDKを使用しているときに次のエラーが発生した場合、プロジェクトに OSS Java SDK のコンパイルと実行に必要なファイルが不足している可能性があります。

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/http/auth/Credentials
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
        at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:76)
Caused by: java.lang.ClassNotFoundException: org.apache.http.auth.Credentials
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
        ... 3 more
```

  または

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/http/protocol/HttpContext
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
        at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:76)
Caused by: java.lang.ClassNotFoundException: org.apache.http.protocol.HttpContext
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
```

```
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
        ... 3 more
```

または

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/jdom/input/
SAXBuilder
        at com.aliyun.oss.internal.ResponseParsers.getXmlRootElement(ResponseParsers
.java:645)
        at ... ...
        at com.aliyun.oss.OSSClient.doesBucketExist(OSSClient.java:471)
        at com.aliyun.oss.OSSClient.doesBucketExist(OSSClient.java:465)
        at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:82)
Caused by: java.lang.ClassNotFoundException: org.jdom.input.SAXBuilder
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
        ... 11 more
```

OSS Java SDK が依存関係にあるパッケージは次のとおりです。

- aliyun-sdk-oss-2.2.1.jar
- hamcrest-core-1.1.jar
- jdom-1.1.jar
- commons-codec-1.9.jar
- httpclient-4.4.1.jar
- commons-logging-1.2.jar
- httpcore-4.4.1.jar
- log4j-1.2.15.jar

上記パッケージでは、log4j-1.2.15.jar ファイル以外、すべてのファイルが必要です。 log4j-1.2.15.jar ファイルは、アクセスログを有効にする際に追加します。

- 解決方法

  プロジェクトに依存関係にある OSS Java SDK を追加します。 以下、手順をご確認ください。

  - プロジェクトで Eclipse を使用している場合、「方法2: Eclipse プロジェクトに JAR パッケージをインポート」をご参照ください。
  - プロジェクトが Ant にある場合は、OSS Java SDK の依存関係にあるパッケージを、lib ディレクトリに追加します。
  - .javac ファイルまたは .java ファイルを直接使用している場合、`-classpath` コマンドまたは `-cp` コマンドを実行して、OSS Java SDK と依存関係にあるパッケージのパスを指定するか、Java SDK が“classpath” パスと依存関係にあるパッケージを保存します。

接続タイムアウト

- 原因分析

  OSS Java SDK プログラムの実行時、次のエラーが発生した場合、考えられる原因はエンドポイントエラー、もしくはネットワークが利用できないためです。

```
com.aliyun.oss.ClientException: SocketException
    at com.aliyun.oss.common.utils.ExceptionFactory.createNetworkException(
ExceptionFactory.java:71)
    at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultSer
viceClient.java:116)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequestImpl(ServiceClient.java
:121)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequest(ServiceClient.java:67)
    at com.aliyun.oss.internal.OSSOperation.send(OSSOperation.java:92)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:140)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:111)
    at com.aliyun.oss.internal.OSSBucketOperation.getBucketInfo(OSSBucketOperation.
java:1152)
    at com.aliyun.oss.OSSClient.getBucketInfo(OSSClient.java:1220)
    at com.aliyun.oss.OSSClient.getBucketInfo(OSSClient.java:1214)
    at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:94)
Caused by: org.apache.http.conn.HttpHostConnectException: Connect to oss-test.oss
-cn-hangzhou-internal.aliyuncs.com:80 [oss-test.oss-cn-hangzhou-internal.aliyuncs.
com/10.84.135.99] failed: Connection timed out: connect
    at org.apache.http.impl.conn.DefaultHttpClientConnectionOperator.connect(
DefaultHttpClientConnectionOperator.java:151)
    at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.connect(
PoolingHttpClientConnectionManager.java:353)
    at org.apache.http.impl.execchain.MainClientExec.establishRoute(MainClientExec.
java:380)
    at org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:236
)
    at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:184)
    at org.apache.http.impl.execchain.RedirectExec.execute(RedirectExec.java:110)
    at org.apache.http.impl.client.InternalHttpClient.doExecute(InternalHttpClient.java:
184)
    at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java
:82)
    at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultSer
viceClient.java:113)
    ... 9 more
```

- 解決方法

  すばやくトラブルシューティングを行うため、ossutil を使用します。

org.apache.http.NoHttpResponseException: The target server failed to respond

- 原因分析

  OSS Java SDK プログラムの実行時、次のエラーが発生します。

```
com.aliyun.oss.ClientException: Unknown
        at com.aliyun.oss.common.utils.ExceptionFactory.createNetworkException(ExceptionFactory.java:68) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultServiceClient.java:115) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.common.comm.ServiceClient.sendRequestImpl(ServiceClient.java:121) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.common.comm.ServiceClient.sendRequest(ServiceClient.java:67) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSOperation.send(OSSOperation.java:92) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:140) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:111) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSMultipartOperation.initiateMultipartUpload(OSSMultipartOperation.java:206) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.OSSClient.initiateMultipartUpload(OSSClient.java:765) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.taobao.agoo.dump.client.OssTools.multipartUpload(OssTools.java:79) ~[agoo-dump-client-2.0.0-SNAPSHOT.jar:na]
        at com.taobao.agoo.dump.biz.manager.TaskExcutorManager$UploadTask.run(TaskExcutorManager.java:114) ~[agoo-dump-biz-2.0.0-SNAPSHOT.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471) [na:1.7.0_51]
        at java.util.concurrent.FutureTask.run(FutureTask.java:262) [na:1.7.0_51]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) [na:1.7.0_51]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615) [na:1.7.0_51]
        at java.lang.Thread.run(Thread.java:744) [na:1.7.0_51]
Caused by: org.apache.http.NoHttpResponseException: The target server failed to respond
        at org.apache.http.impl.conn.DefaultHttpResponseParser.parseHead(DefaultHttpResponseParser.java:143) ~[httpclient-4.4.jar:4.4]
        at org.apache.http.impl.conn.DefaultHttpResponseParser.parseHead(DefaultHttpResponseParser.java:57) ~[httpclient-4.4.jar:4.4]
        at org.apache.http.impl.io.AbstractMessageParser.parse(AbstractMessageParser.java:261) ~[httpcore-4.4.jar:4.4]
        at org.apache.http.impl.DefaultBHttpClientConnection.receiveResponseHeader(DefaultBHttpClientConnection.java:165) ~[httpcore-4.4.jar:4.4]
        at org.apache.http.impl.conn.CPoolProxy.receiveResponseHeader(CPoolProxy.java:167) ~[httpclient-4.4.jar:4.4]
        at org.apache.http.protocol.HttpRequestExecutor.doReceiveResponse(HttpRequestExecutor.java:272) ~[httpcore-4.4.jar:4.4]
```

  有効期限切れの接続が使用されている場合、上記エラーが発生します。 このエラーは、旧バージョンの Java SDK 2.1.2 が使用されている場合にのみ発生します。

- 解決方法

  OSS Java SDK を 2.1.2 以降のバージョンにアップグレードします。

Java プログラムの呼び出し時、OSS Java SDK が応答を停止

- 原因分析

  Java プログラムが呼び出されたとき、OSS Java SDK が応答を停止します。 `jstack -l pid` コマンドを実行し、スタックを確認します。 エラー内容は次のとおりです。

```
"main" prio=6 tid=0x000000000291e000 nid=0xc40 waiting on condition [0x00000000
02dae000]
java.lang.Thread.State: WAITING (parking)
   at sun.misc.Unsafe.park(Native Method)
   - parking to wait for  <0x00000007d85697f8> (a java.util.concurrent.locks.
AbstractQueuedSynchronizer$ConditionObject)
   at java.util.concurrent.locks.LockSupport.park(LockSupport.java:186)
   at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(
AbstractQueuedSynchronizer.java:2043)
   at org.apache.http.pool.PoolEntryFuture.await(PoolEntryFuture.java:138)
   at org.apache.http.pool.AbstractConnPool.getPoolEntryBlocking(AbstractConnPool.
java:306)
   at org.apache.http.pool.AbstractConnPool.access$000(AbstractConnPool.java:64)
   at org.apache.http.pool.AbstractConnPool$2.getPoolEntry(AbstractConnPool.java:
192)
   at org.apache.http.pool.AbstractConnPool$2.getPoolEntry(AbstractConnPool.java:
185)
   at org.apache.http.pool.PoolEntryFuture.get(PoolEntryFuture.java:107)
   at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.leaseConne
ction(PoolingHttpClientConnectionManager.java:276)
```

```
    at org.apache.http.impl.conn.PoolingHttpClientConnectionManager$1.get(
PoolingHttpClientConnectionManager.java:263)
    at org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:190
)
    at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:184)
    at org.apache.http.impl.execchain.RedirectExec.execute(RedirectExec.java:110)
    at org.apache.http.impl.client.InternalHttpClient.doExecute(InternalHttpClient.java:
184)
    at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java
:82)
    at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultSer
viceClient.java:113)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequestImpl(ServiceClient.java
:123)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequest(ServiceClient.java:68)
    at com.aliyun.oss.internal.OSSOperation.send(OSSOperation.java:94)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:146)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:113)
    at com.aliyun.oss.internal.OSSObjectOperation.getObject(OSSObjectOperation.java
:229)
    at com.aliyun.oss.OSSClient.getObject(OSSClient.java:629)
    at com.aliyun.oss.OSSClient.getObject(OSSClient.java:617)
    at samples.HelloOSS.main(HelloOSS.java:49)
```

原因は、接続プールでの接続漏れです(ossObject の使用後、終了しないままでいた可能性が高い)。

- 解決方法

  プログラムを調べて、接続漏れが発生していないことを確認します。 次のコードを使用して、ossObject を終了します。

```
// Read an object.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
// Perform operations on OSS.
// Close your ossObject.
ossObject.close();
```

接続の終了

- 原因分析

  oSSClient.getObject の使用時に、次のようなエラーが発生した場合

```
Exception in thread "main" org.apache.http.ConnectionClosedException: Premature
 end of Content-Length delimited message body (expected: 11990526; received:
202880
    at org.apache.http.impl.io.ContentLengthInputStream.read(ContentLengthInputSt
ream.java:180)
    at org.apache.http.impl.io.ContentLengthInputStream.read(ContentLengthInputSt
ream.java:200)
    at org.apache.http.impl.io.ContentLengthInputStream.close(ContentLengthInputSt
ream.java:103)
    at org.apache.http.impl.execchain.ResponseEntityProxy.streamClosed(ResponseEn
tityProxy.java:128)
    at org.apache.http.conn.EofSensorInputStream.checkClose(EofSensorInputStream.
java:228)
```

```
    at org.apache.http.conn.EofSensorInputStream.close(EofSensorInputStream.java:
174)
    at java.io.FilterInputStream.close(FilterInputStream.java:181)
    at java.io.FilterInputStream.close(FilterInputStream.java:181)
    at com.aliyun.oss.event.ProgressInputStream.close(ProgressInputStream.java:147)
    at java.io.FilterInputStream.close(FilterInputStream.java:181)
    at samples.HelloOSS.main(HelloOSS.java:39)
```

原因は、2 つのデータの読み込み間隔が 1 分間を超過したためです。 OSS では、2 つのデータの読み込み間隔が 1 分間を超過した場合、接続を終了します。

- 解決方法

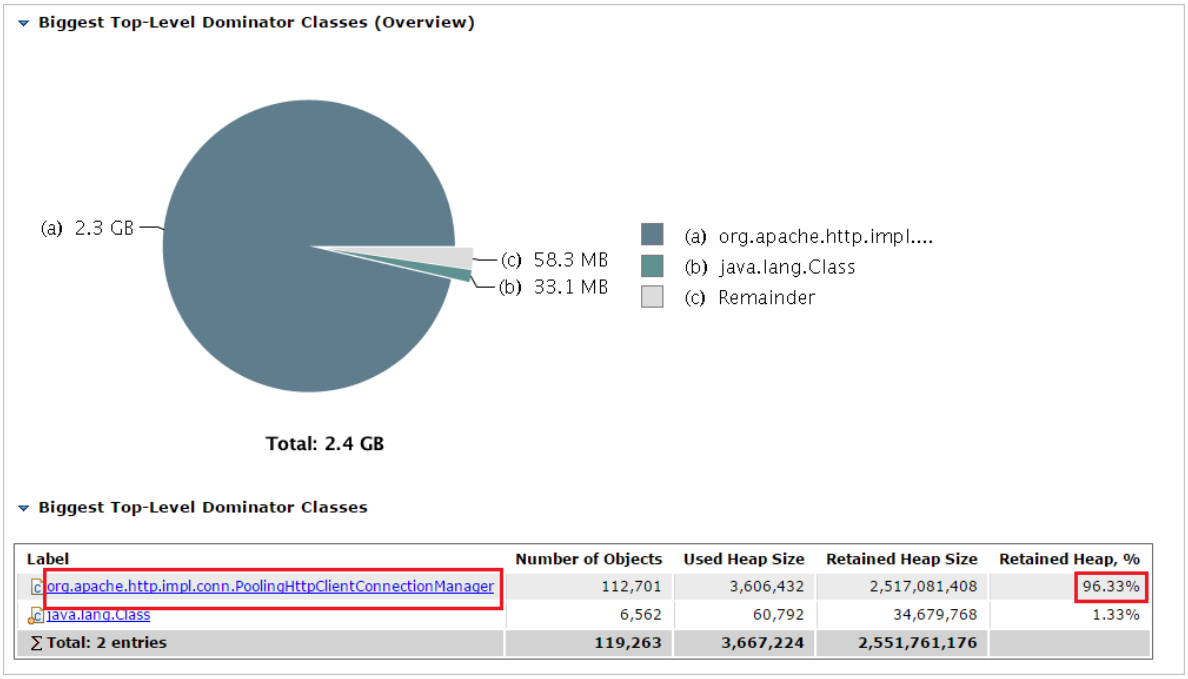
  データの処理間隔が指定されていおらず、データの一部を読み取る必要があるだけの場合は、読み取り中に接続が切断されないようにするため、「範囲のダウンロード」を使用します。

メモリリーク

- 原因分析

  メモリリークは、OSS Java SDK プログラムが一定期間 (アクセストラフィックに基づき、数時間から数日) 実行されると発生します。 メモリ使用量の分析ツールとして、Eclipse Memory Analyzer(MAT) の使用を推奨します。 使用方法の詳細については、「Use MAT for Heap Dump File Analysis」をご参照ください。

  分析結果が次の図のようになっている (PoolingHttpClientConnectionManager が96% のメモリを占有している) 場合、メモリリークの原因はossClient.shutdown が呼び出されていないのに、新しい OSSClient が複数回実行されているためです。 結果として、メモリリークが発生しています。



- a

  OSSClient が有効になった後、ossClient.shutdown が有効になっていることを確認します(併用する必要があります)。

ossClient.shutdown が呼び出し時、InterruptedException エラーが発生

- エラーの原因

  OSSClient.shutdown が呼び出されるとき、次のエラーが発生します。

```
java.lang.InterruptedException: sleep interrupted
        at java.lang.Thread.sleep(Native Method)
        at com.aliyun.oss.common.comm.IdleConnectionReaper.run(IdleConnec
tionReaper:76)
```

  原因は、ossClient のバックエンドスレッド (IdleConnectionReaper) が定期的に接続を終了するためです。 IdleConnectionReaper スレッドがスリープモードのときに、ossClient.shutdown が呼び出されると、上記のエラーが発生します。

- 解決方法

  OSS Java SDK 2.3.0 ではこのエラーの問題を修正してあります。

  OSS Java SDK 2.3.0 より前のバージョンをご使用の場合は、次のコードを使用してエラーを無視します。

```
try {
    ossClient.shutdown();
} catch(Exception e) {
}
```

その他のエラー

OSS から返されるその他のエラーの修正方法については、「エラーとトラブルシューティング」をご参照ください。

3 Python

3.1 インストール

環境要件

OSS Python SDK は、Python 2.6、2.7、3.3、3.4、および 3.5 に対応しています。

- インストール環境

  『Python 公式サイト』よりインストール方法を確認し、適切なバージョンの Python をインストールします。

- Python のバージョン確認

  python -V コマンドを実行し、 Python のバージョンを確認します。

  Windowsでは、「内部コマンドまたは外部コマンドではありません」というメッセージが表示された場合は、環境変数の Path を編集し、Python と pip のインストールパスを追加します。 通常、pip は Python のインストール先パスの Script ディレクトリにインストールされています。

  > 注：
  > コンピューターを再起動し、環境変数が有効になっていることを確認します。

SDK のダウンロード

- GitHub からダウンロード
- 旧バージョンをダウンロード

詳しくは、「OSS API のドキュメント」をご参照ください。

Python-devel のインストール

Python SDK は CRC を計算するため、Python.h ファイル に依存する crcmod ライブラリを使用します。 Python.h ファイルがない場合、crcmod C 拡張モードのインストールが失敗したとしても、Python SDK は正常にインストールすることができてしまいます。そのため、アップロードやダウンロードなどの操作の効率が大幅に低下します。 したがって、python-devel パッケージがない場合は、インストールをする必要があります。

- Windows および Mac OS X では、Python が依存するヘッダーファイルは Python と一緒にインストールされるため、python-devel パッケージをインストールする必要はありません。

- CentOS、RHEL、および Fedora の場合は、次のコマンドを実行して python-devel をインストールします。

```
yum install python-devel
```

- Debian と Ubuntu の場合は、次のコマンドを実行して python-devel をインストールします。

```
apt-get install python-dev
```

SDK のインストール

- Pip での SDK のインストール

  次のコマンドを実行します。

```
pip install oss2
```

  注：
  ご使用の環境に Pip がインストールされていない場合は、『Pip 公式サイト』をご参照のうえ、 Pip をインストールします。

- ソースコードでの SDK のインストール

  『GitHub』から適切なバージョンの SDK パッケージをダウンロードし、解凍を行います。ディレクトリに setup.py ファイルがあることを確認し、次のコマンドを実行します。

```
python setup.py install
```

検証

- SDK のバージョンの確認

  1. コマンドラインでpythonと入力し、Enter キーを押して Python 環境に入ります。
  2. 次のコマンドを実行して SDK のバージョンを確認します。

```
>>> import oss2
>>> oss2.__version__
'2.0.0'
```

  上記の情報が表示される場合、Python SDK 2.0.0 は正常にインストールされています。

- crcmod の検証

  crcmod による、OSS の CRC データ検証には、C 拡張モードと Python モードの 2 つの方法があります。なお、C 拡張モードでは、crcmod より拡張ライブラリ _crcfunext.so を呼び出し、 CRC の計算を行います。 C 拡張モードのパフォーマンスは、Python モードよりもはるか

に優れています。 oss2 をインストールすると、crcmod ライブラリが自動的にインストールされます。 crcmod の詳細については、「crcmod introduction」をご参照ください。

SDK のインストール後、アップロードとダウンロードのインターフェイスの速度が ossutil やその他の SDK などのツールと比べて著しく遅いと感じる場合、SDK の依存ライブラリ crcmod の C 拡張モードのインストールに失敗した可能性があります。その結果、アップロードとダウンロードでは、 Python モードによる CRC の計算を行います。

crcmod の C 拡張モードが正常にインストールされたことを確認する方法は次のとおりです。

1. コマンドラインでpythonと入力し、Enter キーを押して Python 環境に入ります。
2. import crcmod._crcfunextと入力して、Enter キーを押します。
   - エラーが報告されない場合、C 拡張モードが正常にインストールされたことを示します。
   - 次のメッセージが表示された場合は、C 拡張モードが正常にインストールされていません。

     ```
     >>> import crcmod._crcfunext
     Traceback (most recent call last):
     File "<stdin>", line 1, in <module>
     ImportError: No module named _crcfunext
     ```

     このエラーは、crcmod のコンパイル時に _crcfunext.so が依存している Python.h ファイルが見つからないために発生し、結果として、_crcfunext.so ライブラリの生成は失敗します。 このエラーが出た理由は、crcmod がコンパイルされるとき、_crcfunext.so は Python.h ファイルに依存するのに、ヘッダーファイルがシステムから見つからない

ため、_crcfunext.so ライブラリの生成に失敗します。 SDK は正常にインストールされています。 ただし、アップロードやダウンロードなどの操作の効率は低くなります。

この問題が発生した場合は、次の手順を実行します。

a. 次のコマンドを実行して、crcmod をアンインストールします。

```
pip uninstall crcmod
```

b. python-devel をインストールします。

c. 次のコマンドを実行して、crcmod を再インストールします。

```
pip install crcmod
```

それでもインストールが失敗する場合は、crcmod をアンインストールし、次のコマンドを実行して詳細情報を確認します。

```
pip install crcmod -v
```

SDK のアンインストール

SDK のインストールに失敗した場合は、pip を使用して SDK をアンインストールしてから再インストールすることを推奨します。 アンインストールコマンドは次のとおりです。

```
pip uninstall oss2
```

3.2 クイックスタート

このトピックでは、OSS Python SDK を使用してバケットの作成、オブジェクトのアップロード、オブジェクトのダウンロードなどのルーチン操作を実行する方法について説明します。

バケットの作成

次のコードを実行してバケットを作成します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Set the ACL of the bucket to private.
```

```
bucket.create_bucket(oss2.models.BUCKET_ACL_PRIVATE)
```

エンドポイントに関する情報は、「リージョンおよびエンドポイント」をご参照ください。

オブジェクトのアップロード

次のコードを実行してファイルを OSS にオブジェクトをアップロードします。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# <yourLocalFile> consists of a local file path and a file name with extension, for example: /users/local/myfile.txt
bucket.put_object_from_file('<yourObjectName>', '<yourLocalFile>')
```

オブジェクトのダウンロード

次のコードを実行して、指定したオブジェクトをローカルファイルにダウンロードします。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# <yourLocalFile> consists of a local file path and a file name with extension, for example: /users/local/myfile.txt
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>')
```

オブジェクトの一覧表示

次のコードを実行して、指定したバケット内の 10 オブジェクトを一覧で表示します。

```
# -*- coding: utf-8 -*-
import oss2
from itertools import islice

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We strongly recommend that you create a RAM account and use it for API access and daily O&M. Log on to https://ram.console.aliyun.com to create a RAM account.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
```

```
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName
>')

# oss2.ObjectIteratorr is used to traverse objects.
for b in islice(oss2. ObjectIterator(bucket), 10):
    print(b.key)
```

オブジェクトの削除

指定したオブジェクトを削除するには、次のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName
>')

bucket.delete_object('<yourObjectName>')
```

3.3 初期化

このトピックでは、OSS Python SDK を初期化する方法について説明します。

OSS Python SDK のほとんどの操作は oss2.Service および oss2.Bucket を介して実行されます。

- oss2.Service クラスは、バケットを一覧表示するために使用されます。
- oss2.Bucket クラスは、オブジェクトのアップロード、ダウンロード、削除、そしてバケットの設定に使用されます。

2 つのクラスを初期化するには、エンドポイントを指定しなければなりません。 oss2.Service クラスは、カスタムドメイン (CNAME) とのアクセスをサポートしていません。 エンドポイントの詳細については、「リージョンおよびエンドポイント」および「カスタムドメイン名のバインド方法」をご参照ください。

oss2.Service クラスの初期化

詳しくは、「バケットの管理」のバケット一覧をご参照ください。

oss2.Bucket クラスの初期化

- OSS ドメイン名を使用してクラスを初期化

  次のコードを実行して、OSS ドメイン名を使用して oss2.Bucket クラスを初期化します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This case uses the Hangzhou endpoint as an example. Fill in the region endpoint name according to the actual circumstances.
endpoint = 'http://oss-cn-hangzhou.aliyuncs.com'

bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>')
```

- カスタムドメイン名を使用してクラスを初期化

  次のコードを実行して、カスタムドメイン名を使用して oss2.Bucket クラスを初期化します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')

# Use a custom domain name: my-domain.com as an example. is_cname=True indicates that the CNAME is enabled. Cname means binding a custom domain name to the storage space. CNAME indicates a custom domain bound to a bucket.
cname = 'http://my-domain.com'
bucket = oss2.Bucket(auth, cname, '<yourBucketName>', is_cname=True)
```

- 接続タイムアウトの設定

  次のコードを実行して、接続タイムアウトを設定します。

```
# -*- coding: utf-8 -*-
import oss2

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
endpoint = 'http://oss-cn-hangzhou.aliyuncs.com'

# Set the connection timeout to 30 seconds.
```

```
bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>', connect_timeout=30)
```

- CRC 検証を無効化

  アップロードおよびダウンロード中の CRC 検証は、アップロードおよびダウンロード中のデータ保全性を保証するためにデフォルトで有効になっています。 CRC 検証を無効にするには、次のコードを実行します。

  警告：

  CRC 検証を無効化しないことを推奨します。 CRC 検証を無効にした場合、アップロード中およびダウンロード中のデータの整合性は保証されません。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
endpoint = 'http://oss-cn-hangzhou.aliyuncs.com'

bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>', enable_crc=False)
```

3.4 オブジェクトのアップロード

3.4.1 概要

このトピックでは、OSS Python SDK でのオブジェクトダウンロードの概要を説明します。

OSS では、オブジェクトはデータ操作の基本単位です。 OSS Python SDK は次のファイルアップロード方法を提供します。

- 簡易アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 追加アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 再開可能アップロード: 48.8 TB までのファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。 同時アップロードをサポートしており、各パーツのサイズを定義できます。
- マルチパートアップロード: 最大 48.8 TB までのファイルのアップロードをサポートします。 大きなファイルのアップロードに適用されます。

注：

各アップロード方法の使用シナリオの詳細については、「OSS 開発者ガイド」 の「ファイルのアップロード」をご参照ください。

ファイルのアップロード中は、 オブジェクトメタを設定し、アップロードの進行状況を アップロード進行状況バーに表示します。 ファイルのアップロードが完了したら、 「アップロードコールバック」を実行できます。

3.4.2 簡易アップロード

このトピックでは、簡易アップロードの使用方法について説明します。

ファイルをアップロードするために、bucket.put_object メソッドの使用が可能です。 このメソッドは、次の表に示す複数の種類の入力をサポートしています。

型	アップロード方法
文字列	直接アップロードします。
バイト	直接アップロードします。
Unicode	UTF-8 エンコードされたバイトに変換後、アップロードします。
ローカルファイル	バイナリモードで開くファイルオブジェクトとしてアップロードします。
ネットワークストリーム	Chunked Encoding メソッドで Iterable オブジェクトとしてアップロードします。

- 文字列のアップロード

  次のコードを実行し、文字列をアップロードします。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Return value.
result = bucket.put_object('<yourObjectName>', 'content of object')
# HTTP return code.
print('http status: {0}'.format(result.status))
# Request ID. The request ID is the unique identification of a request. We strongly recommend you to add this parameter in the program log.
print('request_id: {0}'.format(result.request_id))
# Etag is a unique attribute of the value returned by put_object.
```

```
print('ETag: {0}'.format(result.etag))
# HTTP response header.
print('date: {0}'.format(result.headers['date']))
```

- バイトのアップロード

  次のコードを実行し、バイトをアップロードします。

```
# -*- coding: utf-8 -*-

import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucket Name>')

bucket.put_object('<yourObjectName>', b'content of object')
```

- Unicode のアップロード

  次のコードを実行し、Unicode をアップロードします。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucket Name>')

bucket.put_object('<yourObjectName>', u'content of object')
```

- ローカルファイルのアップロード

  次のコードを実行し、ローカルファイルをアップロードします。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
```

```
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucket
Name>')

# Files must be opened in the binary mode because the number of bytes included in
the file is required for upload.
with open('<yourLocalFile>', 'rb') as fileobj:
    # The seek method specifies that the read or write operations start from the 1000th
byte. The upload starts from the 1000th byte until the end of the file.
    fileobj.seek(1000, os.SEEK_SET)
    # The tell method is used to return to the current location.
    current = fileobj.tell()
    bucket.put_object('<yourObjectName>', b'content of object')
```

OSS Python SDK はローカルファイルをアップロードするためのより便利な方法を提供します:

```
bucket.put_object_from_file('<yourObjectName>', '<yourLocalFile>')
```

- ネットワークストリームのアップロード

  次のコードを実行してネットワークストリームをアップロードできます:

```
# -*- coding: utf-8 -*-
import oss2
import requests

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucket
Name>')

# The requests.get method returns an Iterable object, and the Python SDK uploads the
network stream in the Chunked Encoding method.
input = requests.get('http://www.aliyun.com')
bucket.put_object('<yourObjectName>', input)
```

3.4.3 追加アップロード

このトピックでは、追加アップロードの使用方法について説明します。

追加アップロードでファイルをアップロードするには、次のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
```

```
# Set the position (the Position parameter) where the object is appended to 0.
result = bucket.append_object('<yourObjectName>', 0, 'content of first append')
# If the object is not uploaded for the first time, you can use the bucket.head_object
method or the next_position attribute returned in the last append upload to get the
append position.
bucket.append_object('<yourObjectName>', result.next_position, 'content of second
append')
```

ファイルが既に存在する場合は、以下の 2 つのシナリオにおいて例外が起こります。

- 追加アップロードでファイルをアップロードできない場合、 ObjectNotAppendable の例外が発生します。
- ファイルは追加アップロードでアップロードできても、追加位置が現在のファイル長と一致しない場合、 PositionNotEqualToLength の例外が発生します。

3.4.4 再開可能アップロード

再開可能アップロードを使用すると、アップロードするファイルを複数のパーツに分割して同時にアップロードします。 すべてのパーツをアップロードしたら、各パーツを組み合わせて完全なオブジェクトにすることができます。 このようにして、オブジェクト全体のアップロードを完了させます。

指定したファイルを再開可能アップロードでアップロードするには、oss2.resumable_upload メソッドを使用します。以下のパラメーターを設定します。

パラメーター	説明	必須	デフォルト値
bucket	バケット名を指定します。	必須	なし
key	オブジェクト名を指定します。	必須	なし
filename	アップロードするローカルファイルの名前を指定します。	必須	なし
store	進捗情報が格納されているディレクトリを指定します。	オプション	Directory '.py-oss-upload' under the HOME directory
headers	HTTPヘッダーを指定します。	オプション	なし

パラメーター	説明	必須	デフォルト値
multipart_threshold	オブジェクト長のしきい値を指定します。 オブジェクトの長さがしきい値よりも大きい場合は、再開可能アップロードが使用されます。	オプション	10 MB
part_size	部品サイズを指定します。	オプション	Auto-calculated
progress_callback	アップロード進捗コールバック関数を指定します。	オプション	なし
num_threads	同時アップロードスレッドの数を指定します。	オプション	1

再開可能アップロードを行うには、次のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2
# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# When the file length is larger than or be equal to the value of the optional parameter multipart_threshold (10 MB by default), the resumable upload is used. If you do not use the store parameter to specify the directory, a py-oss-upload directory is created under the HOME directory to save upload progress information.
oss2.resumable_upload(bucket, '<yourObjectName>', '<yourLocalFile>')
```

Python SDK 2.1.0以降では、次のコードに示すように、オプションのパラメーターを使用して再開可能アップロードをサポートしています。

```
# If you use the store parameter to specify the directory, the progress information is stored in the specified directory. If you use the num_threads to set the number of concurrent upload threads, ensure that the value of oss2.defaults.connection_pool_size is no less than the thread number.. The number of concurrent threads is 1 by default.
oss2.resumable_upload(bucket, '<yourObjectName>', '<yourLocalFile>',
    store=oss2. ResumableStore(root='/tmp'),
    multipart_threshold=100*1024,
    part_size=100*1024,
```

```
    num_threads=4)
```

再開可能アップロードについての詳細は、「OSS 開発者ガイド」の 「再開可能アップロード」 をご参照ください。

3.4.5 マルチパートアップロード

このトピックでは、マルチパートアップロードの使用方法について説明します。

マルチパートアップロードを有効にするには、次の手順を実行します。

1. 初期化（bucket.init_multipart_upload を使用）：アップロード ID を取得します。
2. パーツをアップロードする（bucket.upload_part を使用）：データパーツをアップロードします。 複数のパーツを同時にアップロードできます。
3. マルチパートアップロードを完了する（bucket.complete_multipart_upload を使用）：すべてのパーツをアップロードしたら、これらのパーツをまとめて完全なオブジェクトにします。

マルチパートアップロードには、次のコードを実行します。

```
# -*- coding: utf-8 -*-
import os
from oss2 import SizedFileAdapter, determine_part_size
from oss2.models import PartInfo
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

key = '<yourObjectName>'
filename = '<yourLocalFile>'

total_size = os.path.getsize(filename)
# Use determine_part_size to determine the part size.
part_size = determine_part_size(total_size, preferred_size=100 * 1024)

# Initialize a multipart upload event.
upload_id = bucket.init_multipart_upload(key).upload_id
parts = []

# Upload parts one by one.
with open(filename, 'rb') as fileobj:
    part_number = 1
    offset = 0
    while offset < total_size:
        num_to_upload = min(part_size, total_size - offset)
  # The SizedFileAdapter(fileobj, size) method generates a new object, and re-calculates
the initial append location.
        result = bucket.upload_part(key, upload_id, part_number,
                                    SizedFileAdapter(fileobj, num_to_upload))
        parts.append(PartInfo(part_number, result.etag))
```

```
        offset += num_to_upload
        part_number += 1

# Complete multipart upload.
bucket.complete_multipart_upload(key, upload_id, parts)

# Verify the multipart upload.
with open(filename, 'rb') as fileobj:
    assert bucket.get_object(key).read() == fileobj.read()
```

3.4.6 アップロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を示すことができます。

アップロードプログレスバーの完全なコードについては、『GitHub』をご参照ください。

次のコードは、PutObject を使用した進行状況を表示する方法の説明例として使用されています。

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import os, sys
import oss2
# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# If the length of the data to be uploaded cannot be determined, the value of total_bytes is None.
def percentage(consumed_bytes, total_bytes):
    if total_bytes:
        rate = int(100 * (float(consumed_bytes) / float(total_bytes)))
        print('\r{0}% '.format(rate), end='')
        sys.stdout.flush()
# progress_callback is an optional parameter used to implement progress bars.
bucket.put_object('<yourObjectName>', 'a'*1024*1024, progress_callback=percentage)
```

3.4.7 アップロードコールバック

このトピックでは、アップロードコールバックの使用方法について説明します。

アップロードコールバックの完全なコードについては、『GitHub』をご参照ください。

アップロードコールバックには次のコードを実行します。

```
# -*- coding: utf-8 -*-
import json
import base64
import os
import oss2
```

```
# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Prepare callback parameters.
callback_dict = {}
# Configure the IP address of the server you want to send the callback request to, for example, http://oss-demo.aliyuncs.com:23450 or http://127.0.0.1:9090.
callback_dict['callbackUrl'] = 'http://oss-demo.aliyuncs.com:23450'
# Configure the value of the Host field carried in the callback request header, such as oss-cn-hangzhou.aliyuncs.com.
callback_dict['callbackHost'] = 'oss-cn-hangzhou.aliyuncs.com'
# Configure the value of the body field carried in the callback request.
callback_dict['callbackBody'] = 'filename=${object}&size=${size}&mimeType=${mimeType}'
# Configure Content-Type for the callback request.
callback_dict['callbackBodyType'] = 'application/x-www-form-urlencoded'
# The callback parameter is in the JSON format and Base64-encoded.
callback_param = json.dumps(callback_dict).strip()
base64_callback_body = base64.b64encode(callback_param)
# Encoded callback parameters are carried in the request header and are sent to OSS.
headers = {'x-oss-callback': base64_callback_body}

# Upload and callback.
result = bucket.put_object('<yourObjectName>', 'a'*1024*1024, headers)
```

put_object、put_object_from_file、および complete_multipart_upload メソッドは、アップロードコールバック機能を提供します。 詳しくは、 「OSS 開発者ガイド」の「アップロードコールバック」をご参照ください。

3.5 オブジェクトのダウンロード

3.5.1 概要

このトピックでは、OSS Python SDK でのオブジェクトダウンロードの概要を説明します。

OSS Python SDK は以下のダウンロード方法を提供します。

- ストリーミングダウンロード
- ローカルファイルへのダウンロード
- 部分ダウンロード
- 再開可能なダウンロード

ダウンロードの進行状況は、ダウンロード進行状況バーに表示されます。

3.5.2 ストリーミングダウンロード

大容量のオブジェクトをダウンローズする場合、またはオブジェクト全体を一度にダウンロードするのに時間がかかる場合は、ストリーミングダウンロードを使用します。 ストリーミングダウンロードでは、オブジェクト全体をダウンロードし終わるまで、オブジェクトのパーツをその都度ダウンロードし続けます。

次のコードを実行して、ストリーミングダウンロードを行います。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
// This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# The return value of bucket.get_object is a file-like object, which is also an iterable object.
object_stream = bucket.get_object('<yourObjectName>')
print(object_stream.read())

# The get_object interface returns a stream, which must be read by read() before being used to calculate the CRC checksum of the returned object data. Therefore, you must perform CRC verification after calling the interface.
if object_stream.client_crc ! = object_stream.server_crc:
    print "The CRC checksum between client and server is inconsistent!"
```

以下のコードはデータをローカルファイルへダウンロードするために使用されます。

```
import shutil

# object_stream is a file-like object. Therefore, you can use the shutil.copyfileobj method to download data to a local file.
object_stream = bucket.get_object('<yourObjectName>')
with open('<yourLocalFile>', 'wb') as local_fileobj:
    shutil.copyfileobj(object_stream, local_fileobj)
```

以下のコードはストリーミングコピーを使用して、オブジェクトを他のオブジェクトへコピーします。

```


# object_stream is an iterable object, which can be copied to another object using streaming copy.
object_stream = bucket.get_object('<yourObjectName>')
```

```
bucket.put_object('<yourBackupObjectName>', object_stream)
```

3.5.3 ローカルファイルへのダウンロード

このトピックでは、OSS からローカルファイルにオブジェクトをダウンロードする方法について説明します。

次のコードを実行して、指定されたオブジェクトを OSS からローカルファイルにダウンロードします。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
#  This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Download an object to your local file. If the name of the object is the same as that of your local file, the object will replace the local file. If the name of the object is different from that of your local file, the object will be downloaded and a new file will be added to your local device.
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>')
```

3.5.4 部分ダウンロード

オブジェクトのデータの一部だけが必要な場合は、部分ダウンロードを使用することで、指定のコンテンツをダウンロードできます。

部分ダウンロードを行うには次のコードを実行します

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Obtain the data ranging from byte 0 to the 99th byte of the object (including byte 0 and the 99th byte , a total of 100 bytes). If the specified value range is invalid, for example, the specified range includes a negative number, or the specified value is larger than the object size, the entire object is downloaded.
object_stream = bucket.get_object('<yourObjectName>', byte_range=(0, 99))
```

3.5.5 再開可能ダウンロード

大規模なオブジェクトは、不安定なネットワークまたはその他の例外のためにダウンロードに失敗する可能性があり、オブジェクト全体を再度ダウンロードする必要があります。 ただし、オブジェクトは複数回試行してもダウンロードに失敗することがあります。 そのため、OSS は再開可能ダウンロードを提供します。

再開可能ダウンロードのプロセスは以下のとおりです。

1. 一時的なローカルファイルが作成されます。 ファイル名は、元のオブジェクト名とランダムな接尾辞で構成されています。
2. HTTP リクエストの Range ヘッダーで指定した範囲に基づいて OSS オブジェクトを読み込み、読み込んだ内容を一時ファイルの対応する位置に書き込みます。
3. ダウンロードが完了したら、一時ファイルの名前をターゲットファイルに変更します。 ターゲットファイルがすでに存在する場合は上書きされます。 ターゲットファイルが存在しない場合は、新しいファイルが作成されます。

再開可能ダウンロードを有効にするには、oss2.resumable_download を使用します。 以下のパラメーターが含まれています。

パラメーター	説明	必須またはオプション	デフォルト値
bucket	バケットの名前を指定します。	必須	なし
key	オブジェクトの名前を指定します。	必須	なし
filename	ローカルファイルの名前を指定します。 OSS オブジェクトはローカルファイルにダウンロードされます。	必須	なし

パラメーター	説明	必須またはオプション	デフォルト値
store	ダウンロードの進行状況を記録するローカルファイルを指定します。 ダウンロードの進行状況はファイルに保存されています。 ダウンロードがある時点で失敗した場合、次のダウンロードは記録された進行状況に基づいてその時点から開始されます。	オプション	HOMEディレクトリの下の作成された .py-OSS-download ディレクトリ
multipart_threshold	オブジェクトサイズの長しきい値を指定します。 オブジェクトサイズがしきい値を超えると、再開可能ダウンロードが有効になります。	オプション	10 MB
part_size	パーツのサイズを指定します。	オプション	Auto-calculated
progress_callback	ダウンロード進捗コールバック関数を指定します	オプション	なし
num_threads	同時にダウンロードする必要があるパーツの数を指定します。	オプション	1

再開可能ダウンロードを行うには、次のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
```

```
oss2.resumable_download(bucket, '<yourObjectName>', '<yourLocalFile>')
```

Python SDK 2.1.0 以降では、次のコードに示すように、再開可能ダウンロード用のオプションパラメーターをサポートしています。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Make sure that the value of oss2.defaults.connection_pool_size is no less than the number of the download threads, and the value of part_size is no less than the value of oss2.defaults.multiget_part_size.
oss2.resumable_download(bucket, '<yourObjectName>', '<yourLocalFile>',
  store=oss2.ResumableDownloadStore(root='/tmp'),
  multiget_threshold=20*1024*1024,
  part_size=10*1024*1024,
  num_threads=3)
```

注：

メソッドを呼び出して、複数のプログラム（スレッド）が同一オブジェクトを同一ターゲットファイルに同時にダウンロードすることを許可しないでください。 メソッドを呼び出して、複数のプログラム（スレッド）が同一オブジェクトを同一ターゲットファイルに同時にダウンロードすることを許可しないでください。

3.5.6 ダウンロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を表示することができます。

ダウンロードプログレスバーの完全なコードについては、『GitHub』をご参照ください。

次のコードは、ダウンロードプログレスバーの使用方法を説明するための例として使用されています。

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import os, sys
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
```

```
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# If the HTTP response header does not carry Content-Length, the value of total_bytes is
None.
def percentage(consumed_bytes, total_bytes):
    if total_bytes:
        rate = int(100 * (float(consumed_bytes) / float(total_bytes)))
        print('\r{0}% '.format(rate), end='')

        sys.stdout.flush()

# progress_callback is an optional parameter used to implement progress bar.
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>', progress_callback=
percentage)
```

3.6 オブジェクトの管理

3.6.1 オブジェクトメタの管理

このトピックでは、オブジェクトメタの管理方法について説明します。 詳しくは、「OSS 開発者ガイド」の「オブジェクトメタ」をご参照ください。

HTTP ヘッダーの設定

OSS はユーザー定義の HTTP ヘッダーをサポートしています。 HTTP ヘッダーを設定するには、以下のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

bucket.put_object('<yourObjectName>', '{"age": 1}', headers={'Content-Type': 'applicatio
n/json; charset=utf-8'})
```

HTTP ヘッダーの詳細については、 「RFC2616」をご参照ください。

ユーザー定義のオブジェクトメタの構成

オブジェクト記述にはユーザー定義のオブジェクトメタを定義できます。 ユーザー定義のオブジェクトメタを設定するには、以下のコードを実行します。

```
# -*- coding: utf-8 -*-

import oss2
```

```
# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

bucket.put_object('', 'a novel', headers={'x-oss-meta-author': 'O. Henry', 'Content-Type': 'application/json; charset=utf-8'})
```

注：

Parameters with the x-oss-meta- prefix are used as the headers of user-defined Object Meta. 詳しくは、「PutObject」をご参照ください。

オブジェクトメタの変更

オブジェクトメタを変更するには、以下のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with the AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operation and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

bucket.update_object_meta('<yourObjectName>', {'x-oss-meta-author': 'O. Henry'})
# User-defined Object Meta is cleared each time the bucket.update_object_meta is called and must be configured again.
bucket.update_object_meta('<yourObjectName>', {'x-oss-meta-price': '100 dollar'})
```

Run the following code to change Object Meta, such as Content-Type:

```
bucket.update_object_meta('<yourObjectName>', {'x-oss-meta-author': 'O. Henry'})
# User-defined Object Meta is cleared each time the bucket.update_object_meta is called and must be configured again.
bucket.update_object_meta('<yourObjectName>', {'Content-Type': 'text/plain'})
```

オブジェクトメタの取得

OSS Python SDK が提供する API を使用してオブジェクトメタを取得します。

方法	説明	利点
get_object_meta	オブジェクトの ETag、Size（オブジェクトサイズ）、および LastModified（オブジェクトが最後に変更された時刻）を取得します。	簡単でスピーディー
head_object	オブジェクトメタの取得	なし

オブジェクトメタを取得するには、以下のコードを実行します。

```
# -*- coding: utf-8 -*-

import oss2

# We recommend that you do not log on with the AccessKey of an Alibaba Cloud account because the Alibaba Cloud account has permissions on all APIs in OSS. Instead, we recommend that you log on as an authorized RAM user to access APIs or perform routine operation and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses the endpoint China East 1 (Hangzhou). In actual scenarios, specify your required endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Obtain part of Object Meta.
simplifiedmeta = bucket.get_object_meta("<yourObjectName>")
print(simplifiedmeta.headers['Last-Modified'])
print(simplifiedmeta.headers['Content-Length'])
print(simplifiedmeta.headers['ETag'])

# Obtain all Object Meta.
objectmeta = bucket.head_object("<yourObjectName>")
print(objectmeta.headers['Content-Type'])
print(objectmeta.headers['Last-Modified'])
print(objectmeta.headers['x-oss-object-type'])
```

3.7 許可されたアクセス

このトピックでは、ユーザーへのアクセスを許可する方法について説明します。

一時的アクセスを許可するための URL に署名

一時的アクセスのために訪問者に対して、署名された URL を提供することができます。 URL に署名する際、訪問者からのアクセス期間を制限するため、URL の有効期限を指定できます。

署名付き URL を持つオブジェクトをダウンロードするには、次のコードを実行します。

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM
```

```
user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')

# Set the expiration time of the  URL to one hour.
print(bucket.sign_url('GET', '<yourObjectName>', 60))
```

STS を使用した一時的アクセス許可

STS を使用する完全なコードについては、『GitHub』をご参照ください。

一時的なアクセス許可には、STS (Security Token Service) を使用できます。 STS の詳細については、STS アクセスコントロール API の「はじめに」をご参照ください。 アカウントと承認の詳細については、「ベストプラクティス」内の「STS 一時的アクセス許可」をご参照ください。

公式の Python STS クライアントをインストールする必要があります。

```
pip install aliyun-python-sdk-sts
```

バージョン 2.0.5 またはそれ以降のバージョンをお使いください。 次のコードを実行し、STS で署名リクエストを作成します。

```
# -*- coding: utf-8 -*-

from aliyunsdkcore import client
from aliyunsdksts.request.v20150401 import AssumeRoleRequest
import json
import oss2

# This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
endpoint = 'oss-cn-hangzhou.aliyuncs.com'
# It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
access_key_id = '<yourAccessKeyId>'
access_key_secret = '<yourAccessKeySecret>'
bucket_name = '<yourBucketName>'
# role_arn is the resource name of the role.
role_arn = '<yourRoleArn>'

clt = client.AcsClient(access_key_id, access_key_secret, 'cn-hangzhou')
req = AssumeRoleRequest.AssumeRoleRequest()

# Set the format of returned values to JSON.
req.set_accept_format('json')
req.set_RoleArn(role_arn)
req.set_RoleSessionName('session-name')
body = clt.do_action(req)

# Use the AccessKeyId and AccessKeySecret to apply for a temporary token from STS.
token = json.loads(body)
```

```
# Use the authentication information in the temporary token to initialize the StsAuth
instance.
auth = oss2.StsAuth(token['Credentials']['AccessKeyId'],
                    token['Credentials']['AccessKeySecret'],
                    token['Credentials']['SecurityToken'])

# Use the StsAuth instance to initialize the bucket.
bucket = oss2.Bucket(auth, endpoint, bucket_name)

# Upload a string.
bucket.put_object('object-name.txt', b'hello world')
```

3.8 Python SDK のログ機能の有効化

OSS Python SDKには、問題を簡単にトラックするためのログ機能があります。 この機能は既定では無効になっています。

ログ機能を使用すると、OSS 操作に関するログ情報を見つけ出して収集し、その情報をログファイルとしてローカルディスクに保存することができます。

注：

ログ機能は OSS Python SDK 2.6.x 以降でのみ利用可能です。

ログファイルの形式は次のとおりです。

```
<time><name><level><threadId><message>
```

ログレベルは、次の重要度順に並べられます。CRITICAL > ERROR > WARNING > INFO > DEBUG > NOTSET

注：

ログレベルを指定した後は、指定したレベル以上のレベルのログ情報だけが、ローカルのログファイルに記録されます。 たとえば、ログレベルを INFO に指定した場合、レベル CRITICAL、ERROR、WARNING、および INFO のログ情報がログファイルに記録されます。

Python SDK のログ機能の有効化

OSS Python SDK のログ機能を有効にするには、次のコードを実行します。

```
# -*- coding: utf-8 -*-

import os
import logging
import oss2
from itertools import islice

# Initialize information such as AccessKeyId, AccessKeySecret, and Endpoint.
# Replace <AccessKeyId>, <AccessKeySecret>, <Bucket>, and <Endpoint> with actual
values.
access_key_id = '<AccessKeyId>'
```

```
access_key_secret = '<AccessKeySecret>'
bucket_name = '<Bucket>'
endpoint = '<Endpoint>'

log_file_path = "log.log"
# Enable the logging function.
oss2.set_file_logger(log_file_path, 'oss2', logging.INFO)

# Create a bucket.
bucket = oss2. Bucket(oss2. Auth(access_key_id, access_key_secret), endpoint,
bucket_name)

# Traverse all objects in the bucket.
for b in islice(oss2. ObjectIterator(bucket), 10):
    print(b.key)
# Obtain the object meta.
object_meta = bucket.get_object_meta('object')
```

記録されるログ情報は以下のとおりです。

```
2018-11-20 16:37:46,325 oss2.api [INFO] 26716 : Init oss bucket, endpoint: oss-cn
-shenzhen.aliyuncs.com, isCname: False, connect_timeout: None, app_name: ,
enabled_crc: True
2018-11-20 16:37:46,326 oss2.api [INFO] 26716 : Start to List objects, bucket: funrily-
hello, prefix: , delimiter: , marker: , max-keys: 100
2018-11-20 16:37:46,430 oss2.api [INFO] 26716 : List objects done, req_id: 5BF3C7DA23
6B3A201CE645F7, status_code: 200
2018-11-20 16:37:46,430 oss2.api [INFO] 26716 : Start to get object metadata, bucket:
bucket-hello, key: object
2018-11-20 16:37:46,437 oss2.api [ERROR] 26716 : Exception: {
    'status': 404,
    'x-oss-request-id': '5BF3C7DA236B3A201CE64679',
    'details': {
     'HostId': 'bucket-hello.oss-cn-shenzhen.aliyuncs.com',
        'Message': 'The specified key does not exist.',
        'Code': 'NoSuchKey',
        'RequestId': '5BF3C7DA236B3A201CE64679',
        'Key': 'object'
    }
}
```

前述のログ情報によると、バケットが初期化される際、バケット内のすべてのオブジェクトはトラバースされています。 要求されたオブジェクトが存在しない場合は、`NoSuchKey` エラーが発生します。

3.9 漢字と時刻

本ページでは、OSS Python SDK で使用される漢字と時間について説明します。

漢字

Python コードに漢字が使用されていると、エラーが発生する可能性があります。 したがって、コードの先頭で漢字エンコードを宣言する必要があります。 以下の例をご参照ください。

```
# -*- coding: utf-8 -*-
```

- データ型

  Python 2.x は以下の 2 つのデータ型をサポートしています。

データ型	解説
str	Python 3.x の bytes 型に対応する文字列
unicode	Unicode ストリームは、長さが文字数で測定されます。たとえば、u'中文'の長さは 2 です。

  Python 3.x は以下の 2 つのデータ型をサポートしています。

データ型	説明
str	Python 2.x の Unicode 型に対応する文字列
bytes	バイトストリームは長さがバイト単位で測定されます。 たとえば、b'中文' の長さはエンコーディングに依存します。UTF-8 でエンコードされている場合は、6 になります。

- 入力と出力の規則

  次のテーブルでは、OSS Python SDK の入力パラメーター型の規則を説明しています。

入力	データ型	解説
OSS のオブジェクト名	str	入力型がバイトの場合は、UTF-8 でエンコードする必要があります。
ローカルファイル名	str, unicode	入力型がバイトの場合は、bucket.get_object_to_file 内の yourLocalFile パラメーターなどは、UTF-8 でエンコードする必要があります。

入力	データ型	解説
入力データストリーム	bytes	たとえば、bucket.put_object のデータパラメーターがあります。

次のテーブルでは、OSS Python SDK の出力パラメーター型の規則を説明しています。

出力	データ型	解説
XML で解析された結果	str	たとえば、bucket.list_object によって取得された結果内の文字列です。
ダウンロードされたコンテンツ	bytes	Python SDK では、 bytes 型は既定で UTF-8 でエンコードされているとみなします。そのため、Python のソースファイルも UTF-8 でエンコードされていることを確認する必要があります。

- 型変換関数

  OSS Python SDK は型変換のため、 3 つの関数を提供しています。

関数	説明
to_bytes	Python 2.xでは、unicode 型を str 型に変換します。他の型は変換されずに返されます。 Python 3.x では、str 型を bytes 型に変換します。他の型は変換されずに返されます。
to_unicode	Python 2.x では、str 型を unicode 型に変換します。他の型は変換されずに返されます。 Python 3.x では、bytes 型を str 型に変換します。他の型は変換されずに返されます。
to_string	Python 2.x では、to_bytes として機能します。 Python 3.x では、to_unicode として機能します。

時刻

OSS Python SDK では、サーバーから取得したすべてのタイムスタンプ文字列を Unix 時刻に変換します。つまり、1970 年 1 月 1 日の UTC 00:00 からの秒数に変換します。 たとえば、bucket.get_object によって返される結果の last_modified 時刻は、int 型の Unix 時刻です。

datetime.datetime.fromtimestamp() メソッドを使用して時刻を変換し、タイムスタンプ文字列を取得することできます。

4 Go

4.1 はじめに

このトピックでは、コードサンプルの使用方法について説明します。

SDK のソースコードと API ドキュメント

OSS Go SDK のソースコードについては、『GitHub』をご参照ください。 詳細については、『OSS Go SDK API documents』をご参照ください。

コードサンプル

OSS Go SDK には、参考用または使用目的でさまざまなコードサンプルが用意されています。 次の表にコードサンプルの内容を示します。

サンプルファイル	コンテンツ
new_bucket.go	クライアントの初期化
put_object.go	簡易アップロードおよび再開可能アップロードを含むオブジェクトアップロード
append_object.go	アップロード追加
get_object.go	ストリーミングダウンロード、条件付きダウンロード、圧縮オブジェクトへのオブジェクトダウンロードを含むオブジェクトダウンロード
delete_object.go	オブジェクトの削除
copy_object.go	バケット内のオブジェクトのコピー、バケット間でのオブジェクトのコピー、および条件付きコピー
list_objects.go	指定されたプレフィックスとオブジェクト数を持つリストオブジェクト
object_meta.go	オブジェクトメタの設定と読み取り
object_acl.go	オブジェクト ACL を設定しての読み取り
sign_url.go	署名付き URL の生成
cname_sample.go	カスタムドメインのバインド (CNAME)
create_bucket.go	バケットの作成
list_buckets.go	既定または指定されたパラメーターを使用したリストバケット

サンプルファイル	コンテンツ
bucket_acl.go	バケット ACL の設定
bucket_referer.go	バケットの anti-leech 設定の構成、表示、および削除
bucket_logging.go	バケットのアクセスログの設定、表示、削除
bucket_lifecycle.go	バケットのライフサイクルの設定の構成、表示、および削除
bucket_cors.go	バケットの CORS 設定の構成、表示、および削除

4.2 インストール

準備

- 環境要件

  Golang 1.6 以降のバージョンを使う必要があります。

  Go のコンパイルおよび実行環境をダウンロードしてインストールするには、「ソースから Go をインストールする」をご参照ください。 Go がインストールされたら、GOPATH をあなたのコードのパスに設定します。 GOPATH の詳細については、`go help gopath` コマンドを実行します。
- Go のバージョンの確認

  Go のバージョンを確認するには、`go version` コマンドを実行します。

SDK のダウンロード

- 『GitHub』からダウンロードします。
- 以前のバージョンをダウンロードします。

SDK のインストール

次のコマンドを実行して OSS Go SDK をインストールします。

```
go get github.com/aliyun/aliyun-oss-go-sdk/oss
```

注：
インストール中、メッセージはインターフェイスに表示されません。 インストールの完了をお待ちください。 インストールがタイムアウトした場合は、コマンドを再実行します。

SDK のバージョン確認

次のコードを実行して OSS Go SDK のバージョンを確認します。

```
package main

import (
  "fmt"
  "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
  fmt.Println("OSS Go SDK Version: ", oss.Version)
}
```

4.3 クイックスタート

このトピックでは、OSS Go SDK を使用してバケットの作成、オブジェクトのアップロード、オブジェクトのダウンロードなど定型の操作を実行する方法について説明します。

バケットの作成

次のコードを実行してバケットを作成します。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err ! = nil {
        handleError(err)
    }
    // Create a bucket.
    err = client.CreateBucket(bucketName)
    if err ! = nil {
        handleError(err)
    }
```

```
}
```

エンドポイントに関する情報は、「サービスエンドポイント」をご参照ください。

オブジェクトのアップロード

次のコードを実行してファイルを OSS にアップロードします。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    localFileName := "<yourLocalFileName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err ! = nil {
        handleError(err)
    }
    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        handleError(err)
    }
    // Upload a file.
    err = bucket.PutObjectFromFile(objectName, localFileName)
    if err ! = nil {
        handleError(err)
    }
}
```

オブジェクトのダウンロード

次のコードを実行してオブジェクトをローカルにダウンロードします。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
```

```
}
func main() {
    // This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    downloadedFileName := "<yourDownloadedFileName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err ! = nil {
        handleError(err)
    }
    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        handleError(err)
    }
    // Download the object.
    err = bucket.GetObjectToFile(objectName, downloadedFileName)
    if err ! = nil {
        handleError(err)
    }
}
```

オブジェクトリストの表示

次のコードを実行して、指定したバケット内のオブジェクトを表示します。 既定では最大 100 個のオブジェクトを表示できます。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        HandleError(err)
    }
    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }
    // List objects.
    marker := ""
    for {
```

```
        lsRes, err := bucket.ListObjects(oss.Marker(marker))
        if err ! = nil {
            HandleError(err)
        }
        // Print the listed objects. A maximum of 100 objects are listed by default.
        for _, object := range lsRes.Objects {
            fmt.Println("Bucket: ", object.Key)
        }
        if lsRes.IsTruncated {
            marker = lsRes.NextMarker
        } else {
            break
        }
    }
}
```

オブジェクトの削除

指定したオブジェクトを削除するには、次のコードを実行します。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err ! = nil {
        handleError(err)
    }
    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        handleError(err)
    }
    // Delete the object.
    err = bucket.DeleteObject(objectName)
    if err ! = nil {
        handleError(err)
    }
```

```
}
```

4.4 初期化

OSSClient は、バケットやオブジェクトなどの OSS リソースを管理するための OSS Java クライアントとして機能します。

OSSClient インスタンスを作成するには、エンドポイントを指定する必要があります。 エンドポイントの詳細については、「リージョンとエンドポイント」および「カスタムドメイン名をバインドする」をご参照ください。

次の表に、設定可能なパラメーターを示します。

パラメーター	説明	既定値
UseCname	CNAME を使用するかどうかを指定します。	No
Timeout	接続タイムアウトおよびソケット読み書きタイムアウトを含む、リクエストタイムアウト時間を秒単位で指定します。	接続タイムアウトは 30 秒、読み書きタイムアウトは 60 秒です。
SecurityToken	一時ユーザー用のセキュリティトークンを指定します。	Null
EnableMD5	MD5 検証を有効にするかどうかを指定します。 CRC 検証は MD5 検証よりも効率的であるため推奨されます。	No
EnableCRC	CRC 検証を有効にするかどうかを指定します。	Yes
Proxy	http: // 8.8.8.8: 3128 などのプロキシサーバーを指定します。	Null
AuthProxy	ログインにアカウントとパスワードを必要とするプロキシサーバーを指定します。	Null

OSS ドメインを使用した OSSClient インスタンスの作成

次のコードを実行して、OSS によって割り当てられたドメインを使用して OSSClient インスタンスを作成します。

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"

client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
```

カスタムドメイン (CNAME) を使用した OSSClient インスタンスの作成

顧客ドメインの完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、CNAME で OSSClient インスタンスを作成します。

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"

// oss.UseCname (true) indicates thath the CNAME is enabled. CNAME indicates a custom
 domain bound to a bucket.
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>",
oss.UseCname(true))
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
```

:

CNAME が有効になっている場合、CNAME は特定のバケットにバインドされているため、バケットを列挙できません。

STS を使用した OSSClient インスタンスの作成

以下のコードを実行して、Security Token Service (STS) を使用して OSSClient インスタンスを作成します。

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"

client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>",
oss.SecurityToken("<yourSecurityToken>"))
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
```

詳しくは、「RAM と STS の概要」および「承認済みアクセス」をご参照ください。

タイムアウトの設定

次のコードを実行して接続のタイムアウトを設定します。

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"

// oss.Timeout(10, 120) indicates that the HTTP connection timeout is set to 10 seconds (
the default value is 30 seconds) and the HTTP read-write timeout is set to 120 seconds (
the default value is 60 seconds). 値 0 はタイムアウトが発生しないことを示しますが、推奨
しません。
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>",
oss.Timeout(10, 120))
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
```

CRC 検証

データの整合性を確保するために、アップロードおよびダウンロード中の CRC 検証は既定により有効になっています。 CRC 検証を無効にするには、次のコードを実行します。

警告：

CRC 検証は無効にしないことを推奨します。 CRC 検証を無効にした場合、アップロード中およびダウンロード中のデータの整合性は保証されません。

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"

client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>",
oss.EnableCRC(false))
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
```

4.5 オブジェクトのアップロード

4.5.1 概要

OSS では、オブジェクトはデータ操作の基本単位です。 OSS Go SDK は次のファイルアップロード方法を提供します。

- 簡易アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 追加アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 再開可能アップロード: 48.8 TB までのファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。 同時アップロードをサポートしており、各パーツのサイズを定義できます。

- マルチパートアップロード: 最大 48.8 TB のファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。

> 注 :
> 各アップロード方法の使用シナリオの詳細については、「OSS Developer Guide」の「Upload files」をご参照ください。

ファイルのアップロード中、 オブジェクトメタを設定することで、アップロードの進行状況をアップロード進行状況バーに表示します。

4.5.2 簡易アップロード

簡易アップロードの完全なコードについては、『GitHub』をご参照ください。

文字列のアップロード

次のコードを実行して文字列をアップロードします。

```
package main

import (
 "fmt"
 "os"
 "strings"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Upload a string.
 err = bucket.PutObject("<yourObjectName>", strings.NewReader("yourObjectValue"))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
```

```
}
```

バイト配列のアップロード

次のコードを実行してバイト配列をアップロードします。

```
package main

import (
 "fmt"
 "os"
 "bytes"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Upload a byte array.
 err = bucket.PutObject("<yourObjectName>", bytes.NewReader([]byte("yourObject
ValueByteArrary")))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
    }
}
```

ファイルストリームのアップロード

ファイルストリームをアップロードするには、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
```

```
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Read the local file.
 fd, err := os.Open("<yourLocalFile>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 defer fd.Close()

 // Upload a file stream.
 err = bucket.PutObject("<yourObjectName>", fd)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

ローカルファイルのアップロード

次のコードを実行してローカルファイルをアップロードします。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Upload a local file.
 err = bucket.PutObjectFromFile("<yourObjectName>", "<yourLocalFile>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.5.3 追加アップロード

追加オブジェクトに対して copyObject を実行することはできません。 AppendObject を呼び出すときに、ターゲットオブジェクトが存在しない場合は、新しい追加可能オブジェクトが作成さ

れます。 ターゲットオブジェクトが存在する場合、アップロードされたコンテンツはオブジェクトの末尾に追加されます。

追加アップロードでファイルをアップロードするには、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "strings"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 var nextPos int64 = 0
 // If the object is appended for the first time, the append position is 0, and the returned value is the position for the next append. The position for the next append is the current length of the object.
 nextPos, err = bucket.AppendObject("<yourObjectName>", strings.NewReader("YourObjectAppendValue1"), nextPos)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Start the second append.
 nextPos, err = bucket.AppendObject("<yourObjectName>", strings.NewReader("YourObjectAppendValue2"), nextPos)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // You can append content to an object for multiple times.
}
```

オブジェクトメタを指定できるのは、最初にオブジェクトを追加するときのみです (つまり、追加位置は0です)。

```
 // Specify the Object Meta in the first append upload.
 nextPos, err = bucket.AppendObject("<yourObjectName>", strings.NewReader("YourObjectValue"), 0, oss.Meta("MyProp", "MyPropVal"))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
```

```
}
```

4.5.4 再開可能アップロード

再開可能アップロードを使用すると、アップロードするファイルを複数のパーツに分割して同時にアップロードします。 すべてのパーツをアップロードしたら、各パーツを組み合わせて完全なオブジェクトにすることができます。 このようにして、オブジェクト全体のアップロードを完了させます。

現在のアップロードの進行状況は、アップロード中にチェックポイントファイルに記録されます。 処理中にパーツのアップロードに失敗した場合、オブジェクトはアップロードの再開時にチェックポイントファイルに記録されているパーツとなる、再開可能なアップロードからアップロードされます。 アップロードが完了すると、チェックポイントは削除されます。

注：

- SDK はアップロードの進行状況情報をチェックポイントファイルに記録します。 プログラムにチェックポイントファイルへの書き込み権限があることを確認する必要があります。 検証情報が含まれているため、チェックポイントファイルを変更しないでください。 チェックポイントファイルが破損している場合は、すべてのパーツが再度アップロードされます。
- ファイルのアップロード中にローカルファイルが変更された場合は、すべてのパーツが再度アップロードされます。

再開可能アップロードを行うために、Bucket.UploadFile を使います。 以下のパラメーターを設定します。

パラメーター	説明
objectKey	OSS にアップロードされたオブジェクトの名前を指定します。
filePath	ローカルファイルのパスを指定します。
partSize	アップロードパーツのサイズを 100 KB から 5 GB の範囲で指定します。

パラメーター	説明
options	以下のオプションを指定します。 • ルーチン: 同時にアップロードされるパーツの数を指定します。 既定値は 1 です。これは、パーツが同時にアップロードされないことを意味します。 • チェックポイント: 再開可能アップロードを有効にするかどうかを指定し、チェックポイントファイルを構成します。 再開可能アップロードは既定では無効になっています。 たとえば、oss.Checkpoint(true, "")は 再開可能アップロードが有効であり、チェックポイントファイルはローカルファイルのアップロードが必要なディレクトリに格納されている file.cp です。 file はローカルファイルの名前です。 チェックポイントファイルを指定するために oss.Checkpoint(true, "your-cp-file.cp") を使用することもできます。 • その他のメタデータ情報: 「オブジェクトメタの管理」をご参照ください。

再開可能アップロードを行うには、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // The part size is 100 KB. Parts are uploaded through three coroutines concurrently. The
resumable upload is enabled.
 // <yourObjectName>" is the objectKey, "LocalFile" is the filePath, and 100*1024 is the
partSize.
```

```
 err = bucket.UploadFile("<yourObjectName>", "LocalFile", 100*1024, oss.Routines(3), oss
.Checkpoint(true, ""))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.5.5 マルチパートアップロード

このトピックでは、オブジェクトを複数のパーツに分けてアップロードする方法について説明します。

マルチパートアップロードを有効にするには、次の手順を実行します。

1. マルチパートアップロードイベントを開始します。

   Bucket.InitiateMultipartUpload を呼び出して、OSS で作成されたグローバルに一意な uploadId を返します。

2. マルチパートアップロードタスクを開始します。

   Bucket.UploadPart を呼び出して、データを複数のパーツに分けてアップロードします。

   :

   - 同じ uploadId を持つパーツは、それらのパーツ番号によって順序付けされます。 パーツをアップロードし、同じパーツ番号を使用して別のパーツをアップロードすると、後のパーツが前のパーツに置き換わります。
   - データパーツのMD5 値は ETag ヘッダーに含まれており、ユーザーに返されます。
   - OSS SDK は自動的に Content-MD5 を設定します。 OSS はアップロードされたデータの MD 5 値を計算し、それを SDK によって計算されたMD 5 値と比較します。 2 つの値が異なると、InvalidDigest エラーコードが返されます。

3. マルチパートアップロードタスクを完了します。

   すべてのパーツをアップロードしたら、Bucket.CompleteMultipartUpload を呼び出してこれらのパーツを組み合わせて完全なオブジェクトにします。

次のコードは、マルチパートアップロードのプロセスを説明する完全な例として使用されています。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Creates an OSSClient instance.
```

```
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    locaFilename := "<yourLocalFilename>"
    // Obtains a bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    chunks, err := oss.SplitFileByPartNum(locaFilename, 3)
    fd, err := os.Open(locaFilename)
    defer fd.Close()
    // Step 1: Initiates a multipart upload event.
    imur, err := bucket.InitiateMultipartUpload(objectName)
    // Step 2: Uploads parts.
    var parts []oss.UploadPart
    for _, chunk := range chunks {
        fd.Seek(chunk.Offset, os.SEEK_SET)
        // Calls UploadPart to upload each part.
        part, err := bucket.UploadPart(imur, fd, chunk.Size, chunk.Number)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        parts = append(parts, part)
    }
    // Step 3: Completes the multipart upload task.
    cmur, err := bucket.CompleteMultipartUpload(imur, parts)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("cmur:", cmur)
}
```

マルチパートアップロードイベントのキャンセル

次のコードを実行してマルチパートアップロードイベントをキャンセルします。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Creates an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Obtains a bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
```

```
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Initiates a multipart upload event.
    imur, err := bucket.InitiateMultipartUpload("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Cancels the multipart upload event.
    err = bucket.AbortMultipartUpload(imur)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

アップロードしたパーツの列挙

次のコードを実行して、マルチパートアップロードイベントでアップロードされたパーツを列挙します。

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Creates an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    locaFilename := "<yourLocalFilename>"
    uploadID := "<yourUploadID>"
    // Obtains a bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Splits the local file into multiple parts (3 parts in this example).
    chunks, err := oss.SplitFileByPartNum(locaFilename, 3)
    fd, err := os.Open(locaFilename)
    defer fd.Close()
    // Initiates a multipart upload event.
    imur, err := bucket.InitiateMultipartUpload(objectName)
    uploadID = imur.UploadID
    fmt.Println("InitiateMultipartUpload UploadID: ", uploadID)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Uploads parts.
    var parts []oss.UploadPart
    for _, chunk := range chunks {
        fd.Seek(chunk.Offset, os.SEEK_SET)
```

```
        // Calls UploadPart to upload each part.
        part, err := bucket.UploadPart(imur, fd, chunk.Size, chunk.Number)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        fmt.Println("UploadPartNumber: ", part.PartNumber, ", ETag: ", part.ETag)
        parts = append(parts, part)
    }
    // Lists uploaded parts based on InitiateMultipartUploadResult
    lsRes, err := bucket.ListUploadedParts(imur)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Prints the uploaded parts.
    fmt.Println("\nParts:", lsRes.UploadedParts)
    for _, upload := range lsRes.UploadedParts {
        fmt.Println("List PartNumber:  ", upload.PartNumber, ", ETag: " ,upload.ETag, ",
LastModified: ", upload.LastModified)
    }
    // Generates InitiateMultipartUploadResult based on objectName and UploadID, and
then list all uploaded parts, which applies to scenarios where you know the objectName
and UploadID.
    var imur_with_uploadid oss.InitiateMultipartUploadResult
    imur_with_uploadid.Key = objectName
    imur_with_uploadid.UploadID = uploadID
    // Lists uploaded parts.
    lsRes, err = bucket.ListUploadedParts(imur_with_uploadid)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Prints the uploaded parts.
    fmt.Println("\nListUploadedParts by UploadID: ", uploadID)
    for _, upload := range lsRes.UploadedParts {
        fmt.Println("List PartNumber:  ", upload.PartNumber, ", ETag: " ,upload.ETag, ",
LastModified: ", upload.LastModified)
    }
    // Completes the multipart upload task.
    cmur, err := bucket.CompleteMultipartUpload(imur, parts)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("cmur:", cmur)
}
```

バケットのすべてのパートアップロードイベントの列挙

Bucket.ListMultipartUploads を呼び出して、進行中のすべてのマルチパートアップロードイベント (開始されたが完了していないイベント、またはキャンセルされたイベント) を列挙します。 以下のパラメータを設定できます。

パラメーター	説明
Delimiter	オブジェクト名をグループ化するために使用されるスラッシュ (/) の区切り記号を指定します。 名前に指定されたプレフィックスが含まれ、その間に区切り記号が最初に現れるオブジェクトはすべて、エレメントグループを形成します。
MaxUploads	パートアップロードイベントの最大数を指定します。 設定できる最大値 (これも既定値) は 1,000 です。
KeyMarker	オブジェクト名が "KeyMarker" 値の後に続く文字でアルファベット順に始まるすべてのマルチパートアップロードイベントを列挙します。このパラメーターを "UploadIDMarke"r と一緒に使用して、返される結果の始点を指定できます。
Prefix	返されるオブジェクト名に含める必要があるプレフィックスを指定します。 クエリにプレフィックスを使用すると、返されるオブジェクト名にはそのプレフィックスが含まれます。
UploadIDMarker	"UploadIDMarker "は、返された結果の始点を指定するため、 "KeyMarker" パラメーターと一緒に使用されます。 • "KeyMarker" が設定されていない場合、このパラメーターは無視されます。 • "KeyMarker" が設定されている場合、クエリの結果には以下が含まれます。 - すべてのオブジェクト名がアルファベット順で "KeyMarker" 値の後に続く文字で始まるマルチパートアップロードイベント。 - オブジェクト名は "KeyMarker" の値に等しいが、オブジェクトのアップロード ID は "UploadIDMarker" の値よりも大きいマルチパートアップロードイベント。

- 既定値のパラメーターを使用します。

```
package main
import (
    "fmt"
    "os"
```

```
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Creates an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccess
KeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Obtains a bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Lists all multipart upload events.
    keyMarker := ""
    uploadIdMarker := ""
    for {
        // 1000 records are listed at a time by default.
        lsRes, err := bucket.ListMultipartUploads(oss.KeyMarker(keyMarker), oss.
UploadIDMarker(uploadIdMarker))
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        // Prints multipart upload events.
        for _, upload := range lsRes.Uploads {
            fmt.Println("Upload: ", upload.Key, ", UploadID: ",upload.UploadID)
        }
        if lsRes.IsTruncated {
            keyMarker = lsRes.NextKeyMarker
            uploadIdMarker = lsRes.NextUploadIDMarker
        } else {
            break
        }
    }
}
```

- 次のコードを実行して、指定されたプレフィックスを持つマルチパートアップロードイベントを列挙します。

```
    lsRes, err := bucket.ListMultipartUploads(oss.Prefix("<yourObjectNamePrefix>"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Uploads:", lsRes.Uploads)
```

- 以下のコードを実行して、一度に最大 100 個のマルチパートアップロードイベントを列挙します。

```
    lsRes, err := bucket.ListMultipartUploads(oss.MaxUploads(100))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
```

```
    fmt.Println("Uploads:", lsRes.Uploads)
```

- 次のコードを実行して、指定されたプレフィックスと最大数を持つマルチパートアップロードイベントを列挙します。

```
    lsRes, err := bucket.ListMultipartUploads(oss.Prefix("<yourObjectNamePrefix>"), oss.
MaxUploads(100))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Uploads:", lsRes.Uploads)
```

4.5.6 プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を示すことができます。

次のコードは、Bucket.PutObjectFromFile を使用した進行状況を表示する方法を説明するための例として、使用されています。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

// Define a progress bar listener.
type OssProgressListener struct {
}

// Define the progress event handler.
func (listener *OssProgressListener) ProgressChanged(event *oss.ProgressEvent) {
 switch event.EventType {
 case oss.TransferStartedEvent:
  fmt.Printf("Transfer Started, ConsumedBytes: %d, TotalBytes %d.\n",
   event.ConsumedBytes, event.TotalBytes)
 case oss.TransferDataEvent:
  fmt.Printf("\rTransfer Data, ConsumedBytes: %d, TotalBytes %d, %d%%.",
   event.ConsumedBytes, event.TotalBytes, event.ConsumedBytes*100/event.TotalBytes)
 case oss.TransferCompletedEvent:
  fmt.Printf("\nTransfer Completed, ConsumedBytes: %d, TotalBytes %d.\n",
   event.ConsumedBytes, event.TotalBytes)
 case oss.TransferFailedEvent:
  fmt.Printf("\nTransfer Failed, ConsumedBytes: %d, TotalBytes %d.\n",
   event.ConsumedBytes, event.TotalBytes)
 default:
 }
}

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
```

```
 bucketName := "<yourBucketName>"
 objectName := "<yourObjectName>"
 localFile := "<yourLocalFile>"

 // Obtain a bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Upload with a progress bar displayed.
 err = bucket.PutObjectFromFile(objectName, localFile, oss.Progress(&OssProgres
sListener{}))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.6 オブジェクトのダウンロード

4.6.1 概要

オブジェクトダウンロードの完全なコードについては、『GitHub』をご参照ください。

OSS Go SDK は以下のダウンロード方法を提供します。

- ストリーミングダウンロード
- ユーザーのローカルファイルへのダウンロード
- 再開可能なダウンロード
- 条件付きダウンロード
- 圧縮オブジェクトのダウンロード

ダウンロードの進行状況は、[ダウンロード進行状況バー]に表示されます。

4.6.2 ストリーミングダウンロード

オブジェクトのストリームダウンロード

次のコードを実行して、指定したオブジェクトをストリームにダウンロードします。

```
package main

import (
 "fmt"
 "os"
 "io/ioutil"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
```

```
func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Download an object to a stream.
 body, err := bucket.GetObject("<yourObjectName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 // If you do not close the reader after the data is read, connection leaks may occur.
Consequently, no connection is available and the program cannot run properly.
 defer body.Close()

 data, err := ioutil.ReadAll(body)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 fmt.Println("data:", string(data))
}
```

オブジェクトのローカルキャッシュへのダウンロード

次のコードを実行して、指定したオブジェクトをローカルキャッシュにダウンロードします。

```
package main

import (
 "fmt"
 "os"
 "io"
 "bytes"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
```

```
 // Download the object to the local cache.
 body, err := bucket.GetObject("<yourObjectName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 defer body.Close()

 buf := new(bytes.Buffer)
 io.Copy(buf, body)
 fmt.Println("buf:", buf)
}
```

オブジェクトのローカルファイルストリームへのダウンロード

次のコードを実行して、指定したオブジェクトをローカルファイルストリームにダウンロードします。

```
package main

import (
 "fmt"
 "os"
 "io"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Download an object to a local file stream.
 body, err := bucket.GetObject("<yourObjectName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 defer body.Close()

 fd, err := os.OpenFile("LocalFile", os.O_WRONLY|os.O_CREATE, 0660)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 defer fd.Close()

 io.Copy(fd, body)
```

```
}
```

4.6.3 ローカルファイルへのダウンロード

このトピックでは、OSS からローカルファイルにオブジェクトをダウンロードする方法について説明します。

次のコードを実行して、指定されたオブジェクトを OSSClient からローカルファイルにダウンロードします。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Download an object to your local file.
 err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.6.4 再開可能なダウンロード

ネットワークが不安定であるかプログラムが異常終了し、オブジェクト全体を再度ダウンロードする必要があるため、大きなサイズのオブジェクトをダウンロードできない可能性があります。ただし、オブジェクトは複数回試行してもダウンロードに失敗することがあります。 したがって、OSS は再開可能なダウンロードを提供します。

再開可能ダウンロードを有効にするには、ダウンロードするオブジェクトを複数のパーツに分割して別々にダウンロードする必要があります。 すべてのパーツをダウンロードした後に、これらのパーツは完全なオブジェクトに結合されます。

現在のダウンロードの進行状況はチェックポイントファイルに記録されます。 あるパーツがダウンロードに失敗した場合、次のダウンロードはチェックポイントファイルに記録された進行状況に基づいてそのパーツから始まり、再開可能なダウンロードを実現します。 オブジェクトがダウンロードされた後、チェックポイントファイルは削除されます。

注：

- OSS SDK はダウンロードの進行状況をチェックポイントファイルに記録します。 したがって、プログラムにチェックポイントファイルへの書き込み権限があることを確認してください。 検証情報が含まれているため、チェックポイントファイルを変更しないでください。 オブジェクトのチェックポイントファイルが破損している場合は、オブジェクト全体が再度ダウンロードされます。
- ダウンロード中にダウンロード中のオブジェクトが変更された (オブジェクトのタグが変更された)、またはパーツファイルが見つからないか変更された場合、オブジェクト全体が再度ダウンロードされます。

再開可能なダウンロードを有効にするために、Bucket.DownloadFile を使うことができます。 DownloadFileRequest には、以下のパラメーターが含まれています。

パラメーター	説明
objectKey	ダウンロードするオブジェクトの名前を指定します。
filePath	オブジェクトをダウンロードするローカルファイルのパスを指定します。
partSize	パーツのサイズを指定します。 値は 1 バイトから 5 GB までです。

パラメーター	説明
options	以下のダウンロードオプションを指定します。 • ルーチン: 同時にダウンロードする必要があるパーツの番号を指定します。 既定値は 1 です。 • チェックポイント: 再開可能アップロードを有効にするかどうかを指定し、チェックポイントファイルを構成します。 再開可能アップロードは既定では無効になっています。 たとえば、oss.Checkpoint(true, "") は再開可能ダウンロードが有効であり、チェックポイントファイルが file.cp であることを示します。ここで、file はローカルファイルの名前です。 チェックポイントファイルは、ローカルファイルがあるディレクトリにあります。 チェックポイントファイルを指定するには、oss.Checkpoint(true, "your-cp-file.cp")を使用できます。 • ダウンロードの条件を設定したい場合は、「条件付きダウンロード」をご参照ください。

再開可能ダウンロードのため、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Start the resumable download. Three parts are downloaded simultaneously.
 // <yourObjectName>" is the objectKey, "LocalFile" is the filePath, and 100*1024 is the
partSize.
```

```
 err = bucket.DownloadFile("<yourObjectName>", "LocalFile", 100*1024, oss.Routines(3),
oss.Checkpoint(true, ""))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.6.5 条件付きダウンロード

ダウンロードに対して 1 つ以上の条件を設定できます。 指定した条件が満たされると、オブジェクトをダウンロードします。 指定された条件が満たされないと、エラーコードが返され、オブジェクトをダウンロードできません。 次の条件を設定します。

パラメーター	説明	設定方法
IfModifiedSince	指定された時間がオブジェクトが変更された時間より早い場合は、オブジェクトをダウンロードします。 そうでない場合は、エラー (304 Not modified) が返されます。	oss.IfModifiedSince
IfUnmodifiedSince	指定された時間がオブジェクトが変更された時間以降である場合は、オブジェクトをダウンロードします。 そうでない場合は、エラー (412 Precondition failed) が返されます。	oss.IfUnmodifiedSince
IfMatch	指定された ETag がオブジェクトのものと一致すれば、そのオブジェクトをダウンロードします。 そうでない場合は、エラー (412 Precondition failed) が返されます。	oss.IfMatch
IfNoneMatch	指定された ETag がオブジェクトのものと一致しない場合は、オブジェクトをダウンロードします。 そうでない場合は、エラー (304 Not modified) が返されます。	oss.IfNoneMatch

"If-Modified-Since" と "If-Unmodified-Since" の両方が、オブジェクトのダウンロード条件と同時に存在することがあります。 If-Match と If-None-Match の両方が、オブジェクトのダウンロード条件と同時に存在することがあります。

ETag を取得するには Bucket.GetObjectDetailedMeta を使用します。

```
package main

import (
 "fmt"
 "os"
 "time"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Configure time conditions.
 date := time.Date(2015, time.November, 10, 23, 0, 0, 0, time.UTC)

 // The condition is not met, and the object will not be downloaded.
 err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile", oss.IfModifiedSince(date
))
 if err == nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // The conditions are met, and the object will be downloaded.
 err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile", oss.IfUnmodifiedSince(
date))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
```

```
}
```

4.6.6 圧縮オブジェクトのダウンロード

このトピックでは、圧縮オブジェクトをダウンロードする方法について説明します。

圧縮オブジェクトをダウンロードします。 現在、GZIP はオブジェクト圧縮でサポートされています。 Bucket.GetObject と Bucket.GetObjectToFile はダウンロード時の圧縮をサポートします。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Download an compressed object.
 err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile.gzip", oss.AcceptEncoding
("gzip"))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.6.7 プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を示すことができます。

次のコードは、Bucket.GetObjectToFile を使用した進行状況を表示する方法の説明例になります。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
```

```
// Define a progress bar listener.
type OssProgressListener struct {
}

// Define the progress event handler.
func (listener *OssProgressListener) ProgressChanged(event *oss.ProgressEvent) {
 switch event.EventType {
 case oss.TransferStartedEvent:
  fmt.Printf("Transfer Started, ConsumedBytes: %d, TotalBytes %d.\n",
   event.ConsumedBytes, event.TotalBytes)
 case oss.TransferDataEvent:
  fmt.Printf("\rTransfer Data, ConsumedBytes: %d, TotalBytes %d, %d%%.",
   event.ConsumedBytes, event.TotalBytes, event.ConsumedBytes*100/event.TotalBytes)
 case oss.TransferCompletedEvent:
  fmt.Printf("\nTransfer Completed, ConsumedBytes: %d, TotalBytes %d.\n",
   event.ConsumedBytes, event.TotalBytes)
 case oss.TransferFailedEvent:
  fmt.Printf("\nTransfer Failed, ConsumedBytes: %d, TotalBytes %d.\n",
   event.ConsumedBytes, event.TotalBytes)
 default:
 }
}

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"
 objectName := "<yourObjectName>"
 localFile := "<yourLocalFile>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Download with a progress bar displayed.
 err = bucket.GetObjectToFile(objectName, localFile, oss.Progress(&OssProgressListener
{}))
 if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
 }
 fmt.Println("Transfer Completed.")
}
```

4.7 オブジェクトの管理

4.7.1 概要

次の操作を含む API の使用により、バケットに格納されたオブジェクトを管理します。

- オブジェクトの存在有無の確認
- オブジェクトの ACL を管理
- オブジェクトメタの管理
- オブジェクトの一覧表示
- オブジェクトの削除
- オブジェクトのコピー
- アーカイブオブジェクトの復元
- シンボリックリンクの管理

4.7.2 指定したオブジェクトの存在有無の確認

このトピックでは、指定されたオブジェクトが存在するかどうかを判断する方法について説明します。

次のコードを実行して、指定したオブジェクトが存在するかどうかを確認します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Determine whether the specified object exists.
 isExist, err := bucket.IsObjectExist("<yourObjectName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 fmt.Println("Exist:", isExist)
}
```

4.7.3 オブジェクトの ACL の管理

ACL の完全なコードデモは、『GitHub』をご参照ください。

次の表は、オブジェクトのアクセス制御リスト (ACL) に含まれるアクセス許可について説明しています。

アクセス許可	説明	値
既定値	オブジェクトの ACL は、バケットの ACL と同一です。	oss.ACLDefault
非公開	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。	oss.ACLPrivate
一般公開	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。 他のユーザーはオブジェクトを読むことのみできます。 この許可は慎重に承認してください。	oss.ACLPublicRead
公開読み書き	すべてのユーザーがオブジェクトを読み書きできます。 この許可は慎重に承認してください。	oss.PublicReadWrite

オブジェクトの ACL 特権はバケットの ACL 特権よりも優先されます。 たとえば、バケットの ACL がプライベートで、オブジェクトの ACL がパブリックの読み書きである場合、すべてのユーザーがそのオブジェクトを読み書きできます。 オブジェクトが ACL で設定されていない場合、その ACL は既定ではバケットの ACL と同じです。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Obtain the bucket.
 bucket, err := client.Bucket("<yourBucketName>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
```

```
// Configure an ACL for an object.
err = bucket.SetObjectACL("<yourObjectName>", oss.ACLPublicReadWrite)
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}

// Obtain the ALC for an object.
aclRes, err := bucket.GetObjectACL("<yourObjectName>")
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
fmt.Println("Object ACL:", aclRes.ACL)
}
```

4.7.4 オブジェクトメタの管理

オブジェクトメタの詳細については、「OSS 開発者ガイド」内の「オブジェクトメタ」をご参照ください。 オブジェクトメタの完全なコードについては、『 GitHub』をご参照ください。

オブジェクトメタの設定

オブジェクトをアップロードする際に、オブジェクトメタを設定します。 次の表は、ユーザーが構成できるオブジェクトメタを示しています。

パラメーター	説明
CacheControl	オブジェクトがダウンロードされたときの Web ページのキャッシュアクションを指定します。
ContentDisposition	// ダウンロードされたときのオブジェクトの名前を指定します。
ContentEncoding	オブジェクトがダウンロードされるときのエンコード形式を指定します。
Expires	キャッシュの有効期限を指定します (GMT が使用されます)。
ServerSideEncryption	OSS がオブジェクトを作成するときに、サーバー側の暗号化アルゴリズムを指定します。有効値は AES256 です。
ObjectACL	OSS によって作成されたオブジェクトの ACL を指定します。
Meta	プレフィックスとして X-Oss-Meta- を使用して、カスタムオブジェクトメタを指定します。

次のコードを実行してオブジェクトメタを設定します。

```
package main

import (
    "fmt"
    "os"
    "time",
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Obtain the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Configure Object Meta: The expiration time is 2049-01-10, 23:00:00 in the GMT
 format. ACL は公開読み取りです。 The custom Object Meta is MyProp (the value is
MyPropVal).
    expires := time.Date(2049, time.January, 10, 23, 0, 0, 0, time.UTC)
    options := []oss.Option{
        oss.Expires(expires),
        oss.ObjectACL(oss.ACLPublicRead),
        oss.Meta("MyProp", "MyPropVal"),
    }

    // Use the data stream to upload.
    err = bucket.PutObject("<yourObjectName>", strings.NewReader("MyObjectValue"),
options...)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Obtain Object Meta.
    props, err := bucket.GetObjectDetailedMeta("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object Meta:", props)
}
```

オブジェクトメタの変更

次のコードを実行して、一度に 1 つ以上のオブジェクトメタを変更します。

```
package main
```

```
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"

    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Modify one Object Meta at a time.
    err = bucket.SetObjectMeta(objectName, oss.Meta("MyMeta", "MyMetaValue1"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Modify multiple Object Metas at one time.
    options := []oss.Option{
        oss.Meta("MyMeta", "MyMetaValue2"),
        oss.Meta("MyObjectLocation", "HangZhou"),
    }
    err = bucket.SetObjectMeta(objectName, options...)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Obtain Object Meta.
    props, err := bucket.GetObjectDetailedMeta(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object Meta:", props)
}
```

オブジェクトメタの取得

次のコードを実行してオブジェクトメタを取得します。

```
package main

import (
    "fmt"
```

```
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"

    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Obtain Object Meta.
    props, err := bucket.GetObjectDetailedMeta(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object Meta:", props)
}
```

4.7.5 オブジェクトの一覧表示

オブジェクトはアルファベット順に一覧表示されています。 Bucket.ListObjects を使って、バケット内のオブジェクトを一覧表示します。 以下のパラメーターを使用します。

パラメーター	説明
delimiter	オブジェクトをグループ化するために使用される区切り記号を指定します。 指定されたプレフィックスとスラッシュ (/) の最初の区切り記号の間のサブストリングは commonPrefixes です。
prefix	必須オブジェクトを一覧表示するために構成するプレフィックスを指定します。
maxKeys	一覧表示できるオブジェクトの最大数を指定します。 既定値は 100 です。 最大値は 1,000 です。
marker	リストの最初のオブジェクトを指定します。

ListObjects の完全なコードについては、『GitHub』をご参照ください。

簡易リスト

既定のパラメーターを使用して、バケットのオブジェクトリストを取得できます。 既定では 100 個のオブジェクトを一覧表示できます。 次のコードを実行してバケット内のオブジェクトを一覧表示します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
 fmt.Println("Error:", err)
 os.Exit(-1)
}

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  HandleError(err)
 }

 // Obtain the bucket.
 bucketName := "<yourBucketName>"
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  HandleError(err)
 }

 // List all objects.
 marker := ""
 for {
  lsRes, err := bucket.ListObjects(oss.Marker(marker))
  if err ! = nil {
   HandleError(err)
  }

  // Prints the object list. 100 records are returned by default.
  for _, object := range lsRes.Objects {
   fmt.Println("Bucket: ", object.Key)
  }

  if lsRes.IsTruncated {
   marker = lsRes.NextMarker
  } else {
   break
  }
 }
}
```

指定された数のオブジェクトの一覧表示

次のコードを実行して、指定した数のオブジェクトを一覧表示します。

```
package main
```

```
import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Configure the maximum number of lthe listed object and list objects.
 lsRes, err := bucket.ListObjects(oss.MaxKeys(200))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Print the results.
 fmt.Println("Objects:", lsRes.Objects)
 for _, object := range lsRes.Objects {
  fmt.Println("Object:", object.Key)
 }
}
```

指定したプレフィックスによるオブジェクトの一覧表示

指定されたプレフィックスを持つオブジェクトを一覧表示するには、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
```

```
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // List objects with a specified prefix. 100 objects are listed by default.
 lsRes, err := bucket.ListObjects(oss.Prefix("my-object-"))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Print the results.
 fmt.Println("Objects:", lsRes.Objects)
 for _, object := range lsRes.Objects {
  fmt.Println("Object:", object.Key)
 }
}
```

"marker" によるオブジェクトの一覧表示

"marker" パラメーターは、リストの作成元のオブジェクトの名前を示します。 次のコードを実行して、リストの開始後にオブジェクト ("marker" で指定) を指定します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // List objects placed after the object (specified with marker). 100 objects are listed by
default.
 lsRes, err := bucket.ListObjects(oss.Marker("my-object-xx"))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Print the results.
 fmt.Println("Objects:", lsRes.Objects)
 for _, object := range lsRes.Objects {
  fmt.Println("Object:", object.Key)
 }
```

```
}
```

1 ページ以上のすべてのオブジェクトの一覧表示

次のコードを実行して、1 ページ 以上のすべてのオブジェクトを一覧表示します。 各ページに一覧表示されるオブジェクトの最大数は、"maxKeys" で指定されています。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // List all objects on multiple pages. 200 objects are listed in one page.
 marker := oss.Marker("")
 for {
  lsRes, err := bucket.ListObjects(oss.MaxKeys(200), marker)
  if err ! = nil {
   fmt.Println("Error:", err)
   os.Exit(-1)
  }
  marker = oss.Marker(lsRes.NextMarker)

  fmt.Println("Objects:", lsRes.Objects)

  if ! lsRes.IsTruncated {
   break
  }
 }
}
```

" marker" による、1 ページ以上のオブジェクトの一覧表示

次のコードを実行して、1 つ以上のページで一覧表示を開始するオブジェクト ("marker" で指定) を指定します。各ページに一覧表示されるオブジェクトの最大数は、"maxKeys" で指定されます。

```
package main
```

```
import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // List objects placed after the object (specified with marker) on multiple pages. 200
objects are listed in one page.
 marker = oss.Marker("my-object-xx")
 for {
  lsRes, err := bucket.ListObjects(oss.MaxKeys(50), marker)
  if err ! = nil {
   fmt.Println("Error:", err)
   os.Exit(-1)
  }

  marker = oss.Marker(lsRes.NextMarker)

  fmt.Println("Objects:", lsRes.Objects)

  if ! lsRes.IsTruncated {
   break
  }
 }
}
```

1 つ以上のページに指定されたプレフィックスによるオブジェクトの一覧表示

次のコードを実行して、1 つ以上のページに指定されたプレフィックスでオブジェクトを一覧表示します。各ページに一覧表示されるオブジェクトの最大数は、"maxKeys" で指定されます。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
```

```
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // List objects with a specified prefix on multiple pages. 80 objects are listed in one page
.
 prefix := oss.Prefix("my-object-")
 marker := oss.Marker("")
 for {
  lsRes, err := bucket.ListObjects(oss.MaxKeys(80), marker, prefix)
  if err ! = nil {
   fmt.Println("Error:", err)
   os.Exit(-1)
  }

  prefix = oss.Prefix(lsRes.Prefix)
  marker = oss.Marker(lsRes.NextMarker)

  fmt.Println("Objects:", lsRes.Objects)

  if ! lsRes.IsTruncated {
   break
  }
 }
}
```

4.7.6 オブジェクトの削除

警告：

削除されたオブジェクトは回復できないため、オブジェクトは慎重に削除してください。

オブジェクト削除の完全なコードについては、『GitHub』をご参照ください。

オブジェクトの削除

単一オブジェクトを削除するには、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
```

```
bucketName := "<yourBucketName>"
objectName := "<yourObjectName>"

// Obtain the bucket.
bucket, err := client.Bucket(bucketName)
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}

// Delete an object.
err = bucket.DeleteObject(objectName)
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}
}
```

複数のオブジェクトの削除

"Bucket.DeleteObjects" を使用して複数のオブジェクトを削除し、"DeleteObjectsQuiet" パラメーターを指定して削除操作の結果が返されるかどうかを判断します。 削除されたオブジェクトの結果が既定では返されます。

次のコードを実行して複数のオブジェクトを削除します。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // The results of deleted objects are returned.
 delRes, err := bucket.DeleteObjects([]string{"my-object-1", "my-object-2"})
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
 fmt.Println("Deleted Objects:", delRes.DeletedObjects)
```

```
 // The deletion results are not returned.
 _, err = bucket.DeleteObjects([]string{"my-object-3", "my-object-4"},
 oss.DeleteObjectsQuiet(true))
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

4.7.7 オブジェクトのコピー

オブジェクトをコピーする完全なコードについては、『GitHub』をご参照ください。

同じバケット内のオブジェクトのコピー

次のコードを実行して、同じバケット内のオブジェクトをコピーします。

```
package main

import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"
 objectName := "<yourObjectName>"
 destObjectName := "<yourDestObjectName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Copy an object to another object in the same bucket.
 _, err = bucket.CopyObject(objectName, destObjectName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }
}
```

別のバケットへのオブジェクトのコピー

現在のバケットから別のバケットへ、または他のバケットから現在のバケットへオブジェクトをコピーできます。 次のコードを実行してオブジェクトを別のバケットにコピーします。

```
package main
```

```
import (
 "fmt"
 "os"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"
 srcBucketName := "<yourSrcBucketName>"
 dstBucketName := "<yourDstBucketName>"
 srcObjectName := "<yourSrcObjectName>"
 dstObjectName := "<yourDstObjectName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // Copy the source object (srcObjectName) from other buckets (srcBucketName) to the
current bucket.
 bucket.CopyObjectFrom(srcBucketName, srcObjectName, dstObjectName)
 if err ! = nil {
  fmt.Println("CopyObjectFrom Error:", err)
  os.Exit(-1)
 }

 // Copy the source object (srcObjectName) from the current bucket to another bucket (
dstBucketName).
 bucket.CopyObjectTo(dstBucketName, dstObjectName, srcObjectName)
 if err ! = nil {
  fmt.Println("CopyObjectTo Error:", err)
  os.Exit(-1)
 }
}
```

- コピー中のオブジェクトメタデータの処理

  "MetadataDirective "パラメーターを指定して、オブジェクトをコピーするときにオブジェクトメタデータを処理します。

  - oss.MetaCopy: 既定値。ターゲットオブジェクトのメタデータがソース・オブジェクトのメタデータと同じであることを示し、ソースオブジェクトのメタデータをコピーすることを示します。
  - oss.MetaReplace: ソースオブジェクトのメタデータを新しいメタデータに置き換えます。

  :

"MetadataDirective" パラメーターを oss.MetaReplace に指定した場合、ソースオブジェクトのメタデータは上書きされ、復元できません。 この操作は慎重に行ってください。

"MetadataDirective "の値が oss.MetaReplace の場合、次のメタデータを指定できます。

Metadata	説明
CacheControl	新しいオブジェクトがダウンロードされたときの Web ページのキャッシュアクションを指定します。
ContentDisposition	ダウンロード時の新しいオブジェクトの名前を指定します。
ContentEncoding	新しいオブジェクトがダウンロードされたときのコンテンツのエンコード形式を指定します。
Expires	新しいオブジェクトの有効期限 (GMT) を指定します。
ServerSideEncryption	OSS がオブジェクトを作成するときに使用されるサーバー側の暗号化アルゴリズムを指定します。 有効値: AES256
ObjectACL	作成時に OSS オブジェクトの ACL を指定します。
Meta	X-Oss-Meta- というプレフィックスが付いたカスタムパラメーターです。

コピー中にオブジェクトのメタデータを処理するには、次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "time"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 bucketName := "<yourBucketName>"
 objectName := "<yourObjectName>"
 destObjectName := "<yourDestObjectName>"
```

```
// Obtain the bucket.
bucket, err := client.Bucket(bucketName)
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
}

// Specify the metadata of the target object.
expires := time.Date(2049, time.January, 10, 23, 0, 0, 0, time.UTC)
options := []oss.Option{
 oss.MetadataDirective(oss.MetaReplace),
 oss.Expires(expires),
 oss.ObjectACL(oss.ACLPublicRead),
 oss.Meta("MyMeta", "MyMetaValue")}

// Overwrites the metadata of the source object with the specified metadata.
_, err = bucket.CopyObject(objectName, destObjectName, options...)
if err ! = nil {
 fmt.Println("Error:", err)
 os.Exit(-1)
 }
}
```

条件付きコピー

オブジェクトコピーの条件を設定できます。 オブジェクトは、条件が満たされたときにのみコピーされます。 それ以外の場合は、エラーが報告され、オブジェクトはコピーされません。 利用可能な条件は次のとおりです。

パラメーター	説明
CopySourceIfMatch	ソースオブジェクトの ETag 値がユーザーによって提供された ETag と同じ場合、コピー操作が実行されます。 それ以外の場合は、エラーが返されます。
CopySourceIfNoneMatch	ソースオブジェクトの ETag 値がユーザーによって提供された ETag と同じでない場合は、コピー操作が実行されます。 それ以外の場合は、エラーが返されます。
CopySourceIfModifiedSince	入力パラメーターで指定された時間がソースオブジェクトの実際の修正時間と同じかそれより遅い場合、オブジェクトは通常通りコピーされます。 それ以外の場合は、エラーが返されます。
CopySourceIfUnmodifiedSince	ユーザーが指定した時間が経過してもソースオブジェクトが変更されている場合は、そのオブジェクトがコピーされます。 それ以外の場合は、エラーが返されます。

"CopySourceIfMatch" と "CopySourceIfNoneMatch" は同時に条件として使用できます。 "CopySourceIfModifiedSince" と "CopySourceIfUnmodifiedSince" は同時に条件として使用できます。

条件付きコピーには次のコードを実行します。

```
package main

import (
 "fmt"
 "os"
 "time"
 "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
 // Create an OSSClient instance.
 client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 // bucketName := "<yourBucketName>"
 // objectName := "<yourObjectName>"
 bucketName := "<yourBucketName>"
 objectName := "<yourObjectName>"
 destMatchObjectName := "<yourMatchDestObjectName>"
 destUnMatchObjectName := "<yourUnmatchDestObjectName>"

 // Obtain the bucket.
 bucket, err := client.Bucket(bucketName)
 if err ! = nil {
  fmt.Println("Error:", err)
  os.Exit(-1)
 }

 date := time.Date(2011, time.November, 10, 23, 0, 0, 0, time.UTC)

 // The conditions are satisfied, copy the object.
 _, err = bucket.CopyObject(objectName, destMatchObjectName, oss.CopySource
IfModifiedSince(date))
 if err ! = nil {
  fmt.Println("CopyObject CopySourceIfModifiedSince Error:", err)
 }

 // The conditions are not met, the object is not copied.
 _, err = bucket.CopyObject(objectName, destUnMatchObjectName, oss.CopySource
IfUnmodifiedSince(date))
 if err ! = nil {
  fmt.Println("CopyObject CopySourceIfUnmodifiedSince Error:", err)
 }
}
```

4.7.8 アーカイブオブジェクトの復元

アーカイブオブジェクトを読み込む前に復元する必要があります。 非アーカイブオブジェクトに対して、"restoreObject "を呼び出さないでください。

アーカイブオブジェクトの状態変換プロセスは次のとおりです。

- アーカイブオブジェクトはフリーズ状態です。
- 復元のために送信した後、サーバーはオブジェクトを復元します。 オブジェクトは復元中の状態です。
- 1 分後、オブジェクトは復元状態になり、読み取ることができます。 復元された状態は、既定では 1 日続き、最大 7 日間まで延長できます。 この期間が終了すると、オブジェクトはフリーズ状態に戻ります。

アーカイブストレージクラスの詳細については、「ストレージクラスの概要」をご参照ください。関連するステータスコードの詳細については、「RestoreObject」をご参照ください。 アーカイブオブジェクトを復元するための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行してオブジェクトを復元します。

```
package main

import (
    "fmt"
    "os"
    "time"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        HandleError(err)
    }

    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    localFilename := "<yourLocalFilename>"

    // Create a bucket.
    err = client.CreateBucket(bucketName, oss.StorageClass(oss.StorageArchive))
    if err ! = nil {
        HandleError(err)
    }

    // Obtain the bucket.
    archiveBucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }

    // Upload the archived object.
    var val = "More than just cloud."
```

```
    err = archiveBucket.PutObject(objectName, strings.NewReader(val))
    if err ! = nil {
        HandleError(err)
    }

    // Check whether the object is an archive object.
    meta, err := archiveBucket.GetObjectDetailedMeta(objectName)
    if err ! = nil {
        HandleError(err)
    }

    fmt.Println("X-Oss-Storage-Class : ", meta.Get("X-Oss-Storage-Class"))
    if meta.Get("X-Oss-Storage-Class") == string(oss.StorageArchive) {
        // Restore the archive object for read.
        err = archiveBucket.RestoreObject(objectName)
        if err ! = nil {
            HandleError(err)
        }

        // Wait until the object is restored.
        meta, err = archiveBucket.GetObjectDetailedMeta(objectName)
        for meta.Get("X-Oss-Restore") == "ongoing-request=\"true\"" {
            fmt.Println("x-oss-restore:" + meta.Get("X-Oss-Restore"))
            time.Sleep(1000 * time.Second)
            meta, err = archiveBucket.GetObjectDetailedMeta(objectName)
        }
    }

    // Download the restored object.
    err = archiveBucket.GetObjectToFile(objectName, localFilename)
    if err ! = nil {
        HandleError(err)
    }

    // Restore the object again.
    err = archiveBucket.RestoreObject(objectName)

    fmt.Println("ArchiveSample completed")
}
```

4.7.9 シンボリックリンクの管理

シンボリックリンクは、Windows で使用されるショートカットのように、オブジェクトにマップする特殊なオブジェクトです。 シンボリックリンクには、ユーザー定義のオブジェクトメタを構成できます。

次のコードを実行してシンボリックリンクを作成します。

```
package main

import (
    "fmt"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
    // Create an OSSClient instance.
```

```
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    bucketName := "<yourBucketName>"
    // The symbolic link.
    objectName := "<yourSymlinkName>"
    // The target object that the symbolic link directs to
    targetObjectName := "<yourSymlinkTargetName>"

    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Create a symbolic link.
    err = bucket.PutSymlink(objectName, targetObjectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Upload a file to the object that the symbolic link directs to.
    err = bucket.PutObject(targetObjectName, strings.NewReader("target"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    err = bucket.PutSymlink(objectName, targetObjectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Obtain the content of the object that the symbolic link directs to.
    meta, err := bucket.GetSymlink(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println(meta.Get(oss.HTTPHeaderOssSymlinkTarget))
}
```

シンボリックリンク作成の詳細については、「PutSymlink」をご参照ください。

シンボリックリンクのオブジェクトコンテンツの取得

シンボリックリンクを取得するには、読み取り許可を持っていなければなりません。 次のコードを実行して、シンボリックリンクのオブジェクトコンテンツを取得します。

```
package main

import (
    "fmt"
```

```
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    bucketName := "<yourBucketName>"
    // Symbolic Link name.
    objectName := "<yourSymlinkObjectName>"

    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }

    // Obtain the content of the object that the symbolic link directs to.
    meta, err := bucket.GetSymlink(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println(meta.Get(oss.HTTPHeaderOssSymlinkTarget))
}
```

シンボリックリンクの取得の詳細については、「GetSymlink」をご参照ください。

4.8 画像処理

Image Processing (IMG) は、大規模で安全、低コストで、信頼性の高い画像処理サービスです。ソースイメージを OSS にアップロードすることで、簡単な RESTful API を通じて、いつでも、どこでも、どんなインターネットデバイスでも画像を処理できます。

IMG の詳細については、「Image Processing」をご参照ください。

画像処理の基本機能

OSS は以下の画像処理機能を提供します。

- ドミナントトーンの取得
- 画像形式の変換
- 画像サイズの変更
- 内接円の調整
- 向きの調整

- 画像輝度
- イメージ、テキスト、およびテキストとイメージ透かしをイメージに追加します。
- 画像処理
- 画像処理のアクセスルール: 複数の IMG 関数を呼び出します。

画像処理の使用方法

IMG は標準の HTTP GET を使用します。 URL の QueryString に IMG パラメーターを設定します。

イメージ オブジェクトの ACL が非公開読み書きの場合、許可されたユーザーだけがアクセスを許可されます。

- 匿名アクセス

  処理済みイメージにアクセスするには、次の形式の第 3 レベルドメインを使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
  ```

パラメーター	説明
bucket	バケットの名前を指定します。
endpoint	リージョンへのアクセスに使用されるエンドポイントを指定します。
object	画像オブジェクトの名前を指定します。
image	IMGの予約済み識別子を指定します。
action	拡大縮小、トリミング、回転など、画像に対する操作を指定します。

パラメーター	説明
param	画像に対する操作を示すパラメーターを指定します。

- 基本操作

  たとえば、画像を 100 ピクセルの幅に拡大縮小します。 比率に基づいて高さを調整します。

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- カスタマイズされたイメージスタイル

  処理済みイメージにアクセスするには、次の形式の第 3 レベルドメインを使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  ■ style: カスタマイズされたイメージスタイルの予約済み識別子を指定します。

  ■ yourStyleName: カスタムイメージスタイルの名前を指定します。 OSS コンソールで作成されたルールで指定された名前です。

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- カスケード処理

  カスケード処理では、画像に対して複数の操作を順番に実行できます。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- HTTPS アクセス

  IMG は HTTPS を通じたアクセスをサポートしています。 例:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

• 許可されたアクセス

アクセス許可により、イメージスタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

次のコードを使用して IMG の署名付き URL を生成します。

```
package main

import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccess
KeySecret>")
    if err ! = nil {
 HandleError(err)
    }

    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
 HandleError(err)
    }

    ossImageName := "<yourImageName>"
    signedURL, err := bucket.SignURL(ossImageName, oss.HTTPGet, 600, oss.Process("
image/format,png"))
    if err ! = nil {
 HandleError(err)
    } else {
 fmt.Println(signedURL)
    }
}
```

- SDK を使ったアクセス

  SDK を使用すると、画像オブジェクトにアクセスして処理することができます。

  SDKを使用すると、イメージスタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

  - 基本操作

    次のコードを実行して画像に対して基本操作を実行します。

    ```
    package main

    import (
        "fmt"
    ```

```
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccess
KeySecret>")
    if err ! = nil {
 HandleError(err)
    }

    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
 HandleError(err)
    }

    // Scale the image.
    sourceImageName := "example.png"
    targetImageName := "example-resize.png"
    style := "image/resize,m_fixed,w_100,h_100"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }

    // Crop the image.
    targetImageName = "example-crop.png"
    style = "image/crop,w_100,h_100,x_100,y_100,r_1"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }

    // Rotate the image.
    targetImageName = "example-rotate.png"
    style = "image/rotate,90"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }

    // Sharpen the image.
    targetImageName = "example-sharpen.png"
    style = "image/sharpen,100"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }

    // Add watermarks.
    targetImageName = "example-watermark.png"
    style = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ"
```

```
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }

    // Convert the image format.
    targetImageName = "example-formatconvert.jpg"
    style = "image/format,jpg"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }
}
```

- イメージスタイルのカスタマイズ

  次のコードを実行して画像スタイルをカスタマイズします。

```
package main

import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccess
KeySecret>")
    if err ! = nil {
 HandleError(err)
    }

    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
  HandleError(err)
    }

    // Customize the image style.
    sourceImageName := "example.png"
    targetImageName := "example-custom.png"
    style := "style/<yourCustomStyleName>"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
 HandleError(err)
    }
```

```
}
```

- カスケード処理

  次のコードを実行して画像に対してカスケード処理を実行します。

```
package main

import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccess
KeySecret>")
    if err ! = nil {
 HandleError(err)
    }

    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
 HandleError(err)
     }

    // Customize an image style.
    sourceImageName := "example.png"
    targetImageName := "example-resize.png"
    style := "image/resize,m_fixed,w_100,h_100/rotate,90"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(
style))
    if err ! = nil {
  HandleError(err)
    }
}
```

画像処理の永続化

画像処理の永続化のために次のコードを実行します。

```
package main

import (
    "fmt"
    "os"
    "encoding/base64"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)

func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
```

```
}

func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
    if err ! = nil {
 HandleError(err)
    }

    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
  HandleError(err)
    }


    // Image processing persistence: Scale the image. The target image is stored in the
same bucket by default.
    sourceImageName := "example.png"
    targetImageName := "example-resize.png"
    style := "image/resize,m_fixed,w_100,h_100"
    process := fmt.Sprintf("%s|sys/saveas,o_%v", style, base64. URLEncoding.EncodeToSt
ring([]byte(targetImageName)))
    result, err := bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
        fmt.Println(result)
    }

    // Image processing persistence: Scale the image. The target image is stored in the
different bucket.
    targetBucketName := "<yourTargetBucketName>"
    targetImageName = "example-crop-different-bucket.png"
    style = "image/resize,m_fixed,w_100,h_100"
    process = fmt.Sprintf("%s|sys/saveas,o_%v,b_%v", style, base64. URLEncoding.
EncodeToString([]byte(targetImageName)), base64. URLEncoding.EncodeToString([]byte(
targetBucketName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
        fmt.Println(result)
    }

    // Image processing persistence: Crop the image.
    targetImageName = "example-crop.png"
    style = "image/crop,w_100,h_100,x_100,y_100,r_1"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64. URLEncoding.EncodeToSt
ring([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
        fmt.Println(result)
    }

    // Image processing persistence: Rotate the image.
    targetImageName = "example-rotate.png"
    style = "image/rotate,90"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64. URLEncoding.EncodeToSt
ring([]byte(targetImageName)))
```

```
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
        fmt.Println(result)
    }

    // Image processing persistence: Sharpen the image.
    targetImageName = "example-sharpen.png"
    style = "image/sharpen,100"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64. URLEncoding.EncodeToSt
ring([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
      fmt.Println(result)
    }

    // Image processing persistence: Add watermarks.
    targetImageName = "example-watermark.png"
    style = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64. URLEncoding.EncodeToSt
ring([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
        fmt.Println(result)
    }

    // Image processing persistence: Covert the image format.
    targetImageName = "example-formatconvert.jpg"
    style = "image/format,jpg"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64. URLEncoding.EncodeToSt
ring([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
 HandleError(err)
    } else {
        fmt.Println(result)
    }
}
```

4.9 エラー処理

クライアントエラー

Go SDK の呼び出しエラーは、統合エラーインターフェイスを通して返されます。 エラーインターフェイスは次のように定義されています。

```
type error interface {
    Error() string
```

```
}
```

他のエラーもこのエラーインタフェイスを通して返されます。 たとえば、HTTP リクエストエラーは次のように返されます。

```
//net.Error
type Error interface {
    error
    Timeout() bool   // Is the error a timeout
    Temporary() bool // Is the error temporary
}
```

サーバーエラー

OSS Go SDK の使用時にリクエストエラーが発生した場合は、対応するエラーが返されます。 HTTP リクエストエラーおよび I / O エラーの場合は、Go で事前設定されたエラーが返されます。 OSS サーバーのリクエスト処理エラーの場合、エラーインターフェイスを通して次のエラーが返されます。

```
type ServiceError struct {
    Code       string    // Error code returned by OSS to a user
    Message    string    // Error details provided by OSS
    RequestId  string    // UUID that uniquely identifies the request
    HostId     string    // OSS cluster that identifies the access
    StatusCode int       // HTTP status code
}
```

OSS から返された HTTP ステータスコードが予想されたものと一致しない場合、同様にエラーインタフェイスを通して次のエラーが返されます。

```
type UnexpectedStatusCodeError struct {
    allowed []int    // Expected HTTP status code returned by OSS
    got     int      // Actual HTTP status code returned by OSS
}
```

エラー処理の例

次のコードはエラー処理を説明するために使用されます。

```
package main
import (
    "fmt"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
// Error handling function
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret
>")
```

```
    if err ! = nil {
        HandleError(err)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    // Obtain the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }
    // Simple upload: string upload
    err = bucket.PutObject(objectName, strings.NewReader("yourObjectValueContentString
"))
    if err ! = nil {
        HandleError(err)
}
```

OSS エラーコード

次の表に OSS のエラーコードを示します。

エラーコード	説明	**HTTP** ステータスコード
AccessDenied	アクセス拒否	403
BucketAlreadyExists	バケットが既に存在します。	409
BucketNotEmpty	バケットが空ではありません。	409
EntityTooLarge	エンティティが大きすぎます。	400
EntityTooSmall	エンティティが小さすぎます。	400
FileGroupTooLarge	オブジェクトグループが大きすぎます。	400
InvalidLinkName	オブジェクトリンクの名前は、リンクされたオブジェクトと同じです。	400
LinkPartNotExist	オブジェクトリンクがリンクされているオブジェクトが存在しません。	400
ObjectLinkTooLarge	オブジェクトリンクに含まれるオブジェクトが多すぎます。	400
FieldItemTooLong	POST リクエストのテーブルフィールドが大きすぎます。	400
FilePartInterity	オブジェクトのパーツが変更されました。	400

エラーコード	説明	**HTTP** ステータスコード
FilePartNotExist	オブジェクトのパーツが存在しません。	400
FilePartStale	オブジェクトパーツは期限切れです。	400
IncorrectNumberOfFilesInPOSTRequest	POST リクエスト内のオブジェクトの数が正しくありません。	400
InvalidArgument	パラメーターの形式が無効です。	400
InvalidAccessKeyId	AccessKeyID は存在しません。	403
InvalidBucketName	バケット名が無効です。	400
InvalidDigest	ダイジェストは無効です。	400
InvalidEncryptionAlgorithmError	指定されたエントロピー暗号化アルゴリズムは無効です。	400
InvalidObjectName	オブジェクト名が無効です。	400
InvalidPart	そのパーツは無効です。	400
InvalidPartOrder	パーツオーダーが無効です。	400
InvalidPolicyDocument	ポリシー文書が無効です。	400
InvalidTargetBucketForLogging	ログ記録用のターゲットバケットが無効です。	400
InternalError	OSS でエラーが発生しました。	500
MalformedXML	XML 形式が無効です。	400
MalformedPOSTRequest	POST リクエストの本文の形式が無効です。	400
MaxPOSTPreDataLengthExceededError	POST リクエストでアップロードされたオブジェクトのコンテンツを除く本文が大きすぎます。	400
MethodNotAllowed	メソッドがサポートされていません。	405
MissingArgument	パラメータがありません。	411
MissingContentLength	コンテンツ長がありません。	411

エラーコード	説明	**HTTP** ステータスコード
NoSuchBucket	バケットが存在しません。	404
NoSuchKey	オブジェクトが存在しません。	404
NoSuchUpload	マルチパートアップロード ID が存在しません。	404
NotImplemented	メソッドが実装できません。	501
PreconditionFailed	事前条件が失敗しました。	412
RequestTimeTooSkewed	クライアントの現地時間と OSS サーバー時間は 15 分以上異なります。	403
RequestTimeout	リクエストはタイムアウトしました。	400
RequestIsNotMultiPartContent	POST リクエストのコンテンツタイプが無効です。	400
SignatureDoesNotMatch	署名が正しくありません。	403
TooManyBuckets	ユーザーのバケット数が制限を超過しています。	400
InvalidEncryptionAlgorithmError	指定されたエントロピー暗号化アルゴリズムは無効です。	400

注：

上記表内で、エラーコードは OssServiceError.Code、**HTTP** ステータスコードは OssServiceError.StatusCode です。

5 C++

6 PHP

6.1 はじめに

ソースコード

SDK のソースコードについては、『GitHub』をご参照ください。

コードサンプル

PHP OSS SDK には、参考用または使用目的でさまざまなコードサンプルが用意されています。次の表にコードサンプルの内容を示します。

サンプルファイル	コンテンツ
Object.php	オブジェクトのアップロード、オブジェクトのダウンロード、オブジェクトの管理
MultipartUpload.php	マルチパートアップロード
Signature.php	署名付きURLを使用した許可アクセス
Callback.php	アップロードコールバック
Image.php	画像処理
Bucket.php	バケットの作成、バケットの削除、バケットの一覧表示、ACL
BucketLifecycle.php	バケットのライフサイクルルールの設定、表示、削除
BucketLogging.php	バケットのアクセスログファイルの設定、表示、削除
BucketReferer.php	バケットのanti-leech の設定、表示、削除
BucketWebsite.php	バケットの静的 Web サイトホスティングの設定、表示、削除
BucketCors.php	バケットの CORS の設定、表示、削除

6.2 インストール

準備

OSS PHP SDK は、PHP 5.3 以降のバージョンに適用されます。 このトピックでは例として PHP 5.6.22 を使用します。

- 設置環境

  PHP の cURL 拡張機能をインストールする必要があります。

  - Windows では、PHP cURL 拡張モジュールをインストールするため、「Windows でAlibaba Cloud OSS PHP SDK をコンパイルして使用する」をご参照ください。
    Windowsでは、必要なモジュールが見つからない場合は、 php.ini. の extension_dir を "C:/Windows/System32/" に指定してください。
  - Ubuntu では、`sudo apt-get install php-curl` を実行することで apt-get パッケージマネージャを使用して PHP cURL extension をインストールできます。
  - CentOSでは、"`sudo yum install php-curl`" を実行することで、 yum パッケージマネージャを使用して PHP cURL エクステンションをインストールできます。
- PHP のバージョン確認
  - `php -v`を実行し、現在の PHP のバージョンを確認します。
  - `php -m` を実行し、 cURL エクステンションがインストールされているかどうかを確認します。

SDK のダウンロード

- 『GitHub』からダウンロード
- 以前のバージョンのダウンロード

詳しくは、「OSS APIドキュメント」をご参照ください。

注：
最新バージョンの SDK を使用することを推奨します。 OSS PHP SDK 2.0.0 以前のバージョンのドキュメントは、「ここ」からダウンロードできます。

SDK のインストール

SDK をインストールする方法は 3 つあります。

- Composer

  次のように composer を通して SDK をインストールします。

  1. プロジェクトのルートディレクトリの `composer require aliyuncs/oss-sdk-php` を実行、または `composer.json` で、次のように Alibaba Cloud OSS SDK PHP 上での依存関係を宣言します。

```
"require": {
   "aliyuncs/oss-sdk-php": "~2.x.x"
```

```
}
```

2. composer install を実行し、依存関係をインストールします。 依存関係がインストールされたら、ディレクトリ構造が次の構造に準拠しているかどうかを確認します。

```
.
├──── app.php
├──── composer.json
├──── composer.lock
└──── vendor
```

app.php はユーザーアプリケーションです。 ベンダ/ ディレクトリは、依存ライブラリを含みます。 app.php に依存関係を導入する必要があります。

```
require_once __DIR__ . '/vendor/autoload.php';
```

注：

- プロジェクトが autoload.php を参照している場合は、SDK 依存関係を追加した後で、ファイルを再度導入する必要はありません。
- Composer依存関係マネージャーを使用しているときにネットワークエラーが発生した場合は、Composer China リージョンで次のコマンドを実行して、「Image Source」を使うことができます。 composer config -g repositories.packagist composer http://packagist.phpcomposer.com

- Phar

  次のように phar を通して SDK をインストールできます。

  1. 必要なバージョンを選択し、Phar パッケージを『GitHub』からダウンロードします。
  2. コード内でそのファイルを導入してください。

```
require_once '/path/to/oss-sdk-php.phar';
```

ソースコード

次のようにソースコードから SDK をインストールします。

1. 必要なバージョンを選択し、zip パッケージを『GitHub』からダウンロードします。

2. autoload.php のファイルを解凍したパッケージのルート ディレクトリから見つけ、コードにそのファイルを追加します。

```
require_once '/path/to/oss-sdk/autoload.php';
```

6.3 クイックスタート

このトピックでは、OSS PHP SDK を使用してバケットの作成、オブジェクトのアップロード、オブジェクトのダウンロードなどのルーチン操作を実行する方法について説明します。

バケットの作成

次のコードを実行してバケットを作成します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket = "<yourBucketName>";

try {
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
 $ossClient->createBucket($bucket);
} catch (OssException $e) {
 print $e->getMessage();
}
```

バケット命名規則の詳細については、「基本概念」の「命名の慣例」をご参照ください。 バケットの作成方法については、「バケットの管理」をご参照ください。

エンドポイントに関する情報は、「サービスエンドポイント」をご参照ください。

オブジェクトのアップロード

次のコードを実行してファイルを OSS にアップロードします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
```

```
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket= " <yourBucketName>";
// Object name
$object = " <yourObjectName>";
$content = "Hi, OSS.";

try {
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
 $ossClient->putObject($bucket, $object, $content);
} catch (OssException $e) {
 print $e->getMessage();
}
```

詳しくは、「オブジェクトのアップロード」をご参照ください。

オブジェクトのダウンロード

次のコードを実行してオブジェクトをダウンロードします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket= "<yourBucketName>";
// Object name
$object = "<yourObjectName>";
```

```
try {
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
 $content = $ossClient->getObject($bucket, $object);
   print("object content: " . $content);
} catch (OssException $e) {
 print $e->getMessage();
}
```

詳しくは、「SDK のダウンロード」をご参照ください。

オブジェクトのリスト

次のコードを実行して、指定したバケット内のオブジェクトを一覧表示します。 既定値では最大100 個のオブジェクトを一覧にできます。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
   require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
   require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket= "<yourBucketName>";

try {
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

 $listObjectInfo = $ossClient->listObjects($bucket);
 $objectList = $listObjectInfo->getObjectList();
 if (! empty($objectList)) {
  foreach ($objectList as $objectInfo) {
  print($objectInfo->getKey() . "\t" . $objectInfo->getSize() . "\t" . $objectInfo->
getLastModified() . "\n");
  }
 }
} catch (OssException $e) {
 print $e->getMessage();
}
```

詳しくは、「オブジェクトの管理」内の「リストオブジェクト」をご参照ください。

オブジェクトの削除

指定したオブジェクトを削除するには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket= "<yourBucketName>";
// Object name
$object = "<yourObjectName>";

try {
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
 $ossClient->deleteObject($bucket, $object);
} catch (OssException $e) {
 print $e->getMessage();
}
```

詳しくは、「オブジェクトの管理」内の「オブジェクトの削除」をご参照ください。

6.4 初期化

OSSClient は、バケット や オブジェクト などの OSS リソースを管理するための OSS Java クライアントとして機能します。

OSSClient インスタンスの作成

OSSClient インスタンスを作成するには、エンドポイントを指定する必要があります。 エンドポイントの詳細については、「リージョンとエンドポイント」および「カスタムドメイン名をバインドする」をご参照ください。

- OSS ドメインを使用した OSSClient インスタンスの作成

  次のコードを実行して、OSS によって割り当てられたドメインを使用して OSSClient インスタンスを作成します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

try {
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
} catch (OssException $e) {
 print $e->getMessage();
}
```

- カスタムドメイン (CNAME) を使用した OSSClient インスタンスの作成

  次のコードを実行して、CNAME で OSSClient インスタンスを作成します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
$endpoint = "<yourEndpoint>";

try {
 // The true parameter indicates that the CNAME is enabled. CNAME indicates a custom
 domain bound to a bucket.
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, true);
} catch (OssException $e) {
 print $e->getMessage();
```

```
}
```

注：

CNAME が使用されている場合、ossClient.listBuckets メソッドは使用できません。

- STS を 使用した OSSClient インスタンスの作成

  以下のコードを実行して、Security Token Service (STS) を使用して OSSClient インスタンスを作成します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$securityToken = "<yourSecurityToken>";

try {
 $ossClient = new OssClient(
       $accessKeyId, $accessKeySecret, $endpoint, false, $securityToken);
} catch (OssException $e) {
 print $e->getMessage();
}
```

  詳しくは、「RAM および STS の概要」および「承認されたアクセス」をご参照ください。

- プロキシサーバーを使用した OSSClient インスタンスの作成

  次のコードを実行して、プロキシサーバーを含む OSSClient インスタンスを作成します。これは、PHP 5.3 以降でサポートされています。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The address of the proxy server, for example, http://<username>:<password>@<
proxy ip>:<proxy port>
$requestProxy = "<yourRequestProxy>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based
on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com>";

try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false, $
requestProxy);
} catch (OssException $e) {
    print $e->getMessage();
}
```

ネットワークパラメーターの設定

次のコードを実行して、OSSClient インスタンスのネットワークパラメーターを設定します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on
your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

try {
    $ossClient = new OssClient(
        $accessKeyId, $accessKeySecret, $endpoint);
 // Set the timeout time in seconds for data transmission at the sockets layer. The default
value is 5184000.
 $ossClient->setTimeout(3600);
 // Set the timeout time in seconds when connections are established. The default value
is 10.
 $ossClient->setConnectTimeout(10);
} catch (OssException $e) {
    print $e->getMessage();
}
```

6.5 オブジェクトのアップロード

6.5.1 概要

OSS では、オブジェクトはデータ操作の基本単位です。 OSS PHP SDK は次のファイルアップロード方法を提供します。

- 簡易アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 追加アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- マルチパートアップロード: 最大 48.8 TB までのファイルのアップロードをサポートします。大きなファイルのアップロードに適用されます。

注 :

各アップロード方法の使用シナリオの詳細については、『OSS Developer Guide』 の「Upload files」をご参照ください。 ファイルのアップロードが完了したら、 「アップロードコールバック」を実行できます。

6.5.2 簡易アップロード

簡易アップロードは、文字列のアップロードとローカルファイルのアップロードという 2 つのシナリオに適用されます。

文字列のアップロード

次のコードを実行して文字列を OSS にアップロードできます。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket= "<yourBucketName>";
// Object name
$object = "<yourObjectName>";
// Object content
$content = "Hello OSS";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
```

```
    $ossClient->putObject($bucket, $object, $content);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

ローカルファイルのアップロード

次のコードを実行して、ローカルファイルを OSS にアップロードできます。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Bucket name
$bucket= "<yourBucketName>";
// Object name
$object = "<yourObjectName>";
// <yourLocalFile> consists of the local file path and the file name (including the extension), such as /users/local/myfile.txt
$filePath = "<yourLocalFile>";

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->uploadFile($bucket, $object, $filePath);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

6.5.3 追加アップロード

OSS オブジェクトに文字列またはファイルを追加することができます。

追加オブジェクトに対して copyObject を実行することはできません。

文字列の追加

次のコードを実行して、OSS オブジェクトに文字列を追加します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the bucket name.
$bucket= "<yourBucketName>";
// Specify the object name.
$object = "<yourObjectName>";
// Specify the content of the string.
$content_array = array('Hello OSS', 'Hi OSS', 'OSS OK');
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $position = $ossClient->appendObject($bucket, $object, $content_array[0], 0);
    $position = $ossClient->appendObject($bucket, $object, $content_array[1], $position);
    $position = $ossClient->appendObject($bucket, $object, $content_array[2], $position);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

ファイルの追加

次のコードを実行して、ローカルファイルを OSS オブジェクトに追加します。

```
 <? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
```

```
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the bucket name.
$bucket= "<yourBucketName>";
// Specify the object name.
String objectName = "<yourObjectName>";
// Specify the first local file.
$filePath = "<yourLocalFile1>";
// Specify the second local file
$filePath1 = "<yourLocalFile2>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $position = $ossClient->appendFile($bucket, $object, $filePath, 0);
    $position = $ossClient->appendFile($bucket, $object, $filePath1, $position);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

6.5.4 マルチパートアップロード

マルチパートアップロードを有効にするには、次の手順を実行します。

1. マルチパートアップロードイベントを開始します。

   $ ossClient-> initMultipartUpload を呼び出して、OSS で作成されたグローバルな一意の uploadId を返すことができます。

2. パーツアップロード

   パーツデータをアップロードするには、$ ossClient-> uploadPart を呼び出します。

   :

   - 同じ uploadId を持つパーツの場合、パーツはそれらのパーツ番号によって順序付けされます。 パーツをアップロードし、同じパーツ番号を使用して別のパーツをアップロードすると、後のパーツが前のパーツに置き換わります。
   - 最後のパーツを除いて、他のサイズの最小サイズは 100 KB です。 最後のパーツのサイズは制限されていません。

3. マルチパートアップロードを完了します。

   これらのパーツを組み合わせて完全なオブジェクトにするには、$ ossClient-> completeMultipartUpload を呼び出します。

マルチパートアップロードの完全なコードについては、『GitHub』をご参照ください。

次のコードは、マルチパートアップロードのプロセスを説明する完全な例として使用されています。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Core\OssUtil;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$uploadFile = "<yourLocalFile>";

/**
 *  Step 1: Initiate a multipart upload event and obtain the uploadId.
 */
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    // An uploadId is returned. It is the unique identifier for a part upload event. You can initiate related operations (such as part upload cancelation and query) based on the uploadId.
    $uploadId = $ossClient->initiateMultipartUpload($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": initiateMultipartUpload FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": initiateMultipartUpload OK" . "\n");
/*
 * Step 2: Upload parts.
 */
$partSize = 10 * 1024 * 1024;
$uploadFileSize = filesize($uploadFile);
$pieces = $ossClient->generateMultiuploadParts($uploadFileSize, $partSize);
$responseUploadPart = array();
$uploadPosition = 0;
$isCheckMd5 = true;
foreach ($pieces as $i => $piece) {
    $fromPos = $uploadPosition + (integer)$piece[$ossClient::OSS_SEEK_TO];
    $toPos = (integer)$piece[$ossClient::OSS_LENGTH] + $fromPos - 1;
    $upOptions = array(
        $ossClient::OSS_FILE_UPLOAD => $uploadFile,
        $ossClient::OSS_PART_NUM => ($i + 1),
        $ossClient::OSS_SEEK_TO => $fromPos,
```

```
        $ossClient::OSS_LENGTH => $toPos - $fromPos + 1,
        $ossClient::OSS_CHECK_MD5 => $isCheckMd5,
    );
 // MD5 check.
    if ($isCheckMd5) {
        $contentMd5 = OssUtil::getMd5SumForFile($uploadFile, $fromPos, $toPos);
        $upOptions[$ossClient::OSS_CONTENT_MD5] = $contentMd5;
    }
    try {
  // Upload parts.
        $responseUploadPart[] = $ossClient->uploadPart($bucket, $object, $uploadId, $
upOptions);
    } catch(OssException $e) {
        printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} FAILED\n");
        printf($e->getMessage() . "\n");
        return;
    }
    printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} OK\n");
}
// $uploadParts is an array composed by the ETags and part number (PartNumber) of
each part.
$uploadParts = array();
foreach ($responseUploadPart as $i => $eTag) {
    $uploadParts[] = array(
        'PartNumber' => ($i + 1),
        'ETag' => $eTag,
    );
}
/**
 * Step 3: Complete multipart upload.
 */
try {
 // You must provide all valid $uploadParts when you perform this operation. OSS verifies
 the validity of all parts one by one after it receives $uploadParts. After part verification is
 successful, OSS combines these parts into a complete object.
    $ossClient->completeMultipartUpload($bucket, $object, $uploadId, $uploadParts);
}  catch(OssException $e) {
    printf(__FUNCTION__ . ": completeMultipartUpload FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
printf(__FUNCTION__ . ": completeMultipartUpload OK\n");
```

上記の例では、$ options には次のパラメーターが含まれています。

パラメーター	説明
$ossClient::OSS_FILE_UPLOAD	ファイルのアップロード
OssClient::OSS_PART_NUM	品番
OssClient::OSS_SEEK_TO	開始位置の指定
OssClient::OSS_LENGTH	オブジェクト長
OssClient::OSS_PART_SIZE	パーツサイズ
OssClient::OSS_CHECK_MD5	MD5 チェックを有効にするかどうかを決定します。 値が true の場合、MD5 チェックは有効になります。

マルチパートアップロードイベントのキャンセル

マルチパートアップロードイベントをキャンセルするには、$ossClient の abortMultipartUpload を呼び出します。 マルチパートアップロードイベントをキャンセルすると、この uploadId を使用して他の操作を実行することはできなくなります。 アップロードされたパーツは削除されます。

次のコードを実行してマルチパートアップロードイベントをキャンセルします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$upload_id = "<yourUploadId>";

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->abortMultipartUpload($bucket, $object, $upload_id);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

マルチパートアップロードを使用したローカルファイルのアップロード

マルチパートアップロードを使用してローカルファイルをアップロードするには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
```

```
use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$file = "<yourLocalFile>";

$options = array(
    OssClient::OSS_CHECK_MD5 => true,
    OssClient::OSS_PART_SIZE => 1,
);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->multiuploadFile($bucket, $object, $file, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ":  OK" . "\n");
```

マルチパートアップロードを使用したディレクトリのアップロード

次のコードを実行して、マルチパートアップロードを使用してローカルディレクトリ (その中のすべてのファイルを含む) をアップロードします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$localDirectory = ".";
$prefix = "samples/codes";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
```

```
    $ossClient->uploadDir($bucket, $prefix, $localDirectory);
}  catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ":  OK" . "\n");
```

アップロードしたパーツの一覧表示

次のコードを実行してアップロードされたパーツを一覧表示します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$uploadId = "<yourUploadId>";

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $listPartsInfo = $ossClient->listParts($bucket, $object, $uploadId);
    foreach ($listPartsInfo->getListPart() as $partInfo) {
        print($partInfo->getPartNumber() . "\t" . $partInfo->getSize() . "\t" . $partInfo->
getETag() . "\t" . $partInfo->getLastModified() . "\n");
    }
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

すべてのマルチパートアップロードイベントの一覧表示

次のコードを実行して、マルチパートアップロードイベントをすべて一覧表示します。

```
<? php
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
```

```
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$options = array(
    'delimiter' => '/',
    'max-uploads' => 100,
    'key-marker' => '',
    'prefix' => '',
    'upload-id-marker' => ''
);
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $listMultipartUploadInfo = $ossClient->listMultipartUploads($bucket, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": listMultipartUploads FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
printf(__FUNCTION__ . ": listMultipartUploads OK\n");
$listUploadInfo = $listMultipartUploadInfo->getUploads();
var_dump($listUploadInfo);
```

パラメーターは次のように記述されます。

パラメーター	説明
delimiter	オブジェクト名をグループ化するために使用されるスラッシュ (/) のデリミターを指定します。 指定されたプレフィックスとスラッシュ (/) の最初の区切り記号の間のオブジェクトは共通のプレフィックスです。
key-marker	オブジェクトの名前が、アルファベット順で "key-marker" 値の後にくる文字で始まるすべてのパートアップロードイベントを列挙します。 このパラメーターを "upload-id-marker" パラメーターと一緒に使用して、指定された戻り結果の初期位置を指定できます。
max-uploads	パートアップロードイベントの最大数を指定します。 設定できる最大値 (または既定値) は 1,000です。

パラメーター	説明
prefix	返されるオブジェクト名に含める必要があるプレフィックスを指定します。 クエリにプレフィックスを使用した場合、返されるオブジェクト名にはそのプレフィックスが含まれます。
upload-id-marker	このパラメーターを "key-marker" パラメーターと一緒に使用して、指定された戻り結果の初期位置を指定できます。 "keyMarker" を構成しない場合、"uploadIdMarker" パラメーターは無効です。 "key-marker" を設定した場合、クエリ結果には以下が含まれます。 • "key-marker" 値の後にくる文字で名前が始まるすべてのオブジェクト。 • 名前がアルファベット順の "key-marker" の値と同じ文字で始まり、"uploadId" の値が "upload-id-marker" の値より大きいすべてのオブジェクト。

6.5.5 コールバックのアップロード

簡易アップロード用のコールバックを設定する

次のコードを実行して、簡易アップロード用のコールバックを設定します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

// Configure callback for uploads.
// callbackUrl is the IP address of the callback server, such as http://oss-demo.aliyuncs.com:23450 or http://127.0.0.1:9090.
```

```
// callbackHost is the value of the Host field carried in the callback request, such as oss-cn-hangzhou.aliyuncs.com.
$url =
    '{
        "callbackUrl":"<yourCallbackServerUrl>",
        "callbackHost":"<yourCallbackServerHost>",
        "callbackBody":"bucket=${bucket}&object=${object}&etag=${etag}&size=${size}&mimeType=${mimeType}&imageInfo.height=${imageInfo.height}&imageInfo.width=${imageInfo.width}&imageInfo.format=${imageInfo.format}&my_var1=${x:var1}&my_var2=${x:var2}",
        "callbackBodyType":"application/x-www-form-urlencoded"
    }';

// Configure the custom parameters used to initiate a callback request. Each parameter consists of a key and a value. The key must start with x:.
$var =
    '{
        "x:var1":"value1",
        "x:var2":"value2"
    }';
$options = array(OssClient::OSS_CALLBACK => $url,
    OssClient::OSS_CALLBACK_VAR => $var
);
$result = $ossClient->putObject($bucket, $object, file_get_contents(__FILE__), $options);
print_r($result['body']);
print_r($result['info']['http_code']);
```

マルチパートアップロードのコールバックの設定

次のコードを実行し、マルチパートアップロードのコールバックを設定します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Core\OssUtil;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$uploadFile = "<yourLocalFile>";

/**
 * Step 1: Initiate a multipart upload event and get the uploadId.
 */

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
```

```
// An uploadId is returned. It is the unique identifier for a multipart upload event. You can initiate related operations (such as cancel or query a multipart upload event) based on the uploadId.
$uploadId = $ossClient->initiateMultipartUpload($bucket, $object);

print(__FUNCTION__ . ": initiateMultipartUpload OK" . "\n");
/*
 /* Step 2: Upload parts.
 */
$partSize = 10 * 1024 * 1024;
$uploadFileSize = filesize($uploadFile);
$pieces = $ossClient->generateMultiuploadParts($uploadFileSize, $partSize);
$responseUploadPart = array();
$uploadPosition = 0;
$isCheckMd5 = true;
foreach ($pieces as $i => $piece) {
    $fromPos = $uploadPosition + (integer)$piece[$ossClient::OSS_SEEK_TO];
    $toPos = (integer)$piece[$ossClient::OSS_LENGTH] + $fromPos - 1;
    $upOptions = array(
        $ossClient::OSS_FILE_UPLOAD => $uploadFile,
        $ossClient::OSS_PART_NUM => ($i + 1),
        $ossClient::OSS_SEEK_TO => $fromPos,
        $ossClient::OSS_LENGTH => $toPos - $fromPos + 1,
        $ossClient::OSS_CHECK_MD5 => $isCheckMd5,
    );
    // Perform the MD5 verification.
    if ($isCheckMd5) {
        $contentMd5 = OssUtil::getMd5SumForFile($uploadFile, $fromPos, $toPos);
        $upOptions[$ossClient::OSS_CONTENT_MD5] = $contentMd5;
    }
    try {
        // Upload the parts.
        $responseUploadPart[] = $ossClient->uploadPart($bucket, $object, $uploadId, $upOptions);
    } catch(OssException $e) {
        printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} FAILED\n");
        printf($e->getMessage() . "\n");
        return;
    }
    printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} OK\n");
}
// $uploadParts is an array that consists of the ETags and the PartNumbers of each part.
$uploadParts = array();
foreach ($responseUploadPart as $i => $eTag) {
    $uploadParts[] = array(
        'PartNumber' => ($i + 1),
        'ETag' => $eTag,
    );
}

/**
 * Step 3: Complete the upload.
 */
// The callbackUrl is the IP address of the callback server, such as http://oss-demo.aliyuncs.com:23450 or http://127.0.0.1:9090.
// callbackHost is the value of the Host field carried in the callback request, such as oss-cn-hangzhou.aliyuncs.com.
$json =
    '{
        "callbackUrl":"<yourCallbackServerUrl>",
        "callbackHost":"<yourCallbackServerHost>",
        "callbackBody":"{\"mimeType\":${mimeType},\"size\":${size},\"x:var1\":${x:var1},\"x:var2\":${x:var2}}",
        "callbackBodyType":"application/json"
```

```
    }';

// Configure the custom parameters used to initiate a callback request. Each parameter consists of a key and a value. The key must start with x:.
$var =
    '{
        "x:var1":"value1",
        "x:var2":"value2"
    }';
$options = array(OssClient::OSS_CALLBACK => $json,
    OssClient::OSS_CALLBACK_VAR => $var);
// To perform the operation, all valid $uploadParts must be provided. OSS verifies the validity of all parts one by one after it receives the $uploadParts. After all the data parts are verified, the OSS combines these parts into a complete object.
$ossClient->completeMultipartUpload($bucket, $object, $uploadId, $uploadParts, $options);

printf(__FUNCTION__ . ": completeMultipartUpload OK\n");
```

6.6 オブジェクトのダウンロード

6.6.1 概要

このトピックは、OSS PHP SDK でのオブジェクトダウンロードの概要です。

オブジェクトダウンロードの完全なコードについては、『GitHub』をご参照ください。

OSS PHP SDK は以下のダウンロード方法を提供します。

- ユーザーのローカルファイルへのダウンロード
- ローカルメモリへのダウンロード
- 部分ダウンロード
- 条件付きダウンロード

6.6.2 ローカルファイルへのダウンロード

指定した OSS オブジェクトをローカルファイルにダウンロードするには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
```

```
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
//This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$localfile = "<yourLocalFile>";
$options = array(
        OssClient::OSS_FILE_DOWNLOAD => $localfile
    );

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->getObject($bucket, $object, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK, please check localfile: 'upload-test-object-name.txt'" . "\n");
```

6.6.3 ローカルメモリへのダウンロード

次のコードを実行して、指定したオブジェクトをローカルメモリにダウンロードし、そのオブジェクトの内容を印刷します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $content = $ossClient->getObject($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print($content);
print(__FUNCTION__ . ": OK" . "\n");
```

6.6.4 部分ダウンロード

オブジェクト内のデータの一部だけが必要な場合は、部分ダウンロードを使用して特定のコンテンツをダウンロードできます。

次のコードを実行して、[0、4] の範囲のオブジェクトの内容をローカルメモリにダウンロードします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

//It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
//This example uses the endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
// Obtain data between the starting byte value of 0 and the maximum byte value of 4 inclusive (that is, a total of 5 bytes). If the specified value range is invalid, for example, the specified range includes a negative number, or the specified value is larger than the object size, the entire object is downloaded.
$options = array(OssClient::OSS_RANGE => '0-4');
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $content = $ossClient->getObject($bucket, $object, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print ($content);
print(__FUNCTION__ . ": OK" . "\n");
```

6.6.5 条件付きダウンロード

ダウンロードに対して 1 つ以上の条件を設定できます。 指定した条件が満たされると、オブジェクトをダウンロードできます。 指定された条件が満たされないと、エラーコードが返され、オブジェクトをダウンロードできません。 以下の条件を設定できます。

パラメーター	説明	設定方法
If-Modified-Since	指定された時間がオブジェクトが変更された時間より早い場合は、オブジェクトをダウンロードします。 そうでない場合は、エラー ("304 Not modified") が返されます。	OssClient::OSS_IF_MODIFIED_SINCE
If-Unmodified-Since	指定された時間がオブジェクトが変更された時間以降である場合は、オブジェクトをダウンロードします。 そうでない場合は、エラー ("412 Precondition failed") が返されます。	OssClient::OSS_IF_UNMODIFIED_SINCE
If-Match	指定された ETag がオブジェクトのものと一致すれば、そのオブジェクトをダウンロードします。 そうでない場合は、エラー ("412 Precondition failed") が返されます。	OssClient::OSS_IF_MATCH
If-None-Match	指定された ETag がオブジェクトのものと一致しない場合は、オブジェクトをダウンロードします。 そうでない場合は、エラー ("304 Not modified") が返されます。	OssClient::OSS_IF_NONE_MATCH

注：

- "If-Modified-Since" と "If-Unmodified-Since" の両方が、オブジェクトのダウンロード条件と同時に存在することがあります。 "If-Match" と "If-None-Match" の両方が、オブジェクトのダウンロード条件と同時に存在することがあります。
- ETag を取得するには、$ossClient- の getObjectMeta を使用します。

条件付きダウンロードでは、OSS オブジェクトをローカルメモリまたはローカルファイルにダウンロードします。 条件付きダウンロード用に次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
```

```
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

try{
    $options = array(
        OssClient::OSS_HEADERS => array(
            OssClient::OSS_IF_MODIFIED_SINCE => "Fri, 13 Nov 2015 14:47:53 GMT"),
    );

    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $content = $ossClient->getObject($bucket, $object, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print ($content);
print(__FUNCTION__ . ": OK" . "\n");
```

6.7 オブジェクトの管理

6.7.1 概要

API を使ってバケット内のオブジェクトに対して次の操作を実行できます。

- オブジェクトが存在するかどうかの確認
- オブジェクトの ACL の管理
- オブジェクトメタの管理
- オブジェクトの列挙
- オブジェクトの削除
- オブジェクトの削除
- アーカイブオブジェクトの復元
- シンボリックリンクの管理
- MD5 検証の有効化

6.7.2 指定したオブジェクトの存在有無の確認

指定したオブジェクトが存在するかを判断するには、次のコードを使用します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM  user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $exist = $ossClient->doesObjectExist($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
var_dump($exist);
```

6.7.3 オブジェクの ACL の管理

次の表は、オブジェクトのアクセス制御リスト (ACL) に含まれるアクセス許可について説明しています。

アクセス許可	説明	値
既定値	オブジェクトの ACL は、そのバケットの ACL と同じです。	default
非公開	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。	private

アクセス許可	説明	値
一般公開	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。 他のユーザーはオブジェクトを読むことしかできません。 この許可は慎重に承認してください。	public-read
公開読み書き	すべてのユーザーがオブジェクトを読み書きできます。 この許可は慎重に承認してください。	public-read-write

オブジェクトの ACL はバケットの ACL よりも優先されます。 たとえば、バケットの ACL が非公開で、オブジェクトの ACL が公開読み書きである場合、すべてのユーザーがそのオブジェクトを読み書きできます。 オブジェクトが ACL で設定されていない場合、その ACL は既定値でそのバケットの ACL と同じです。

オブジェクトの ACL の設定

次のコードを実行して、指定したオブジェクトの ACL を設定できます。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
// Set the ACL for the object to public read, which is inherited from the bucket by default.
$acl = "public-read";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->putObjectAcl($bucket, $object, $acl);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
```

```
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

オブジェクトの ACL の取得

指定したオブジェクトの ACL を取得するには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $objectAcl = $ossClient->getObjectAcl($bucket, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
var_dump($objectAcl);
```

6.7.4 オブジェクトメタの管理

オブジェクトメタの詳細については、『OSS Developer Guide』内の「オブジェクトメタ」をご参照ください。

オブジェクトメタの設定

次のコードを実行してオブジェクトメタを設定します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
```

```
use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$content = file_get_contents(__FILE__);
$options = array(
    OssClient::OSS_HEADERS => array(
        'Expires' => '2012-10-01 08:00:00',
        'Content-Disposition' => 'attachment; filename="xxxxxx"',
        'x-oss-meta-self-define-title' => 'user define meta info',
    ));
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->putObject($bucket, $object, $content, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

オブジェクトメタの変更

次のコードを実行してオブジェクトメタを変更します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$fromBucket = "<yourFromBucketName>";
$fromObject = "<yourFromObjectName>";
$toBucket = "<yourToBucketName>";
$toObject = "<yourToObjectName>";
$copyOptions = array(
```

```
    OssClient::OSS_HEADERS => array(
        'Expires' => '2018-10-01 08:00:00',
        'Content-Disposition' => 'attachment; filename="xxxxxx"',
        'x-oss-meta-location' => 'location',
    ),
);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->copyObject($fromBucket, $fromObject, $toBucket, $toObject, $copyOption
s);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

オブジェクトメタの取得

次のコードを実行してオブジェクトメタを取得します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all APIs in OSS. We recommend that you log on as a RAM
user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ Objectmeta = $ ossclient-> getobjectmeta ($ bucket, $ object );
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
if (isset($objectMeta[strtolower('Content-Disposition')]) &&
    'attachment; filename="xxxxxx"' === $objectMeta[strtolower('Content-Disposition')]
) {
    print(__FUNCTION__ . ": ObjectMeta checked OK" . "\n");
} else {
    print(__FUNCTION__ . ": ObjectMeta checked FAILED" . "\n");
```

```
}
```

6.7.5 オブジェクトの一覧表示

バケット内のすべてのオブジェクトを一覧表示する

次のコードを実行して、バケット内のすべてのオブジェクトを一覧表示します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
account has permissions on all the APIs in OSS. We recommend that you log on as a RAM
 user to access APIs or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

while (true) {
    try {
        $listObjectInfo = $ossClient->listObjects($bucket, $options);
    } catch (OssException $e) {
        printf(__FUNCTION__ . ": FAILED\n");
        printf($e->getMessage() . "\n");
        return;
    }
    // Obtain the nextMarker. Obtain the object list from the object next to the last object
read by listObjects.
    $nextMarker = $listObjectInfo->getNextMarker();
    $listObject = $listObjectInfo->getObjectList();
    $listPrefix = $listObjectInfo->getPrefixList();

    if (! empty($listObject)) {
        print("objectList:\n");
        foreach ($listObject as $objectInfo) {
            print($objectInfo->getKey() . "\n");
        }
    }
    if (! empty($listPrefix)) {
        print("prefixList: \n");
        foreach ($listPrefix as $prefixInfo) {
            print($prefixInfo->getPrefix() . "\n");
        }
    }
    if ($nextMarker === '') {
        break;
    }
```

```
}
```

指定された条件を持つオブジェクトの一覧表示

次のコードを実行して、指定した条件を持つオブジェクトを一覧表示します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

$prefix = 'dir/';
$delimiter = '/';
$nextMarker = '';
$maxkeys = 10;
$options = array(
    'delimiter' => $delimiter,
    'prefix' => $prefix,
    'max-keys' => $maxkeys,
    'marker' => $nextMarker,
);
try {
    $listObjectInfo = $ossClient->listObjects($bucket, $options);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
$objectList = $listObjectInfo->getObjectList(); // object list
$prefixList = $listObjectInfo->getPrefixList(); // directory list
if (! empty($objectList)) {
    print("objectList:\n");
    foreach ($objectList as $objectInfo) {
        print($objectInfo->getKey() . "\n");
    }
}
if (! empty($prefixList)) {
    print("prefixList: \n");
    foreach ($prefixList as $prefixInfo) {
        print($prefixInfo->getPrefix() . "\n");
    }
```

```
}
```

上記の例では、$options には次のパラメータが含まれています。

パラメーター	説明	必須かどうか
delimiter	オブジェクトをグループ化するために使用される区切り記号を指定します。CommonPrefixes は、区切り記号で終わり、同じプレフィックスを持つ一連のオブジェクトを示します。	いいえ
prefix	必須オブジェクトを一覧表示するために構成するプレフィックスを指定します。	いいえ
max-keys	一覧表示できるオブジェクトの最大数を指定します。 既定値は 100 です。 最大値は 1,000 です。	いいえ
marker	一覧表示する最初のオブジェクトを指定します。	いいえ

6.7.6 オブジェクトの削除

オブジェクトを削除します

単一のオブジェクトを削除するには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. RAM ユーザーとしてログオンして API にアクセスするか、定期的な運用とメンテナンスを実行することを推奨します。 To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
```

```
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->deleteObject($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

複数のオブジェクトの削除

複数のオブジェクトを同時に削除するには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$objects = array();
$objects[] = "<yourObjectName1>";
$objects[] = "<yourObjectName2>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->deleteObjects($bucket, $objects);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

6.7.7 オブジェクトのコピー

簡易コピー

以下のコードを実行して、1 GB より小さいオブジェクトをコピーします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
```

```
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$from_bucket = "<yourFromBucketName>";
$from_object = "<yourFromObjectName>";
$to_bucket = $bucket;
$to_object = $from_object . '.copy';

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->copyObject($from_bucket, $from_object, $to_bucket, $to_object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

大規模オブジェクトのコピー

1 GB を超えるオブジェクトをコピーするには、マルチパートコピーを使用する必要があります。マルチパートコピーを使用するには、次の手順に従います。

1. UploadPartCopy タスクを開始するには、$ossClient->initiateMultipartUpload を使用します。
2. マルチパートコピーを開始するには、$ossClient->uploadPartCopy を使用します。 最後のパーツを除いて、すべてのパーツは 100 KB より大きくなければなりません。
3. $ossClient->completeMultipartUpload を使用してマルチパートコピータスクを送信します。

マルチパートコピーを行うには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
```

```
use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

$src_bucket = "<yourSourceBucketName>";
$src_object = "<yourSourceObjectName>";
$dst_bucket = "<yourDestinationBucketName>";
$dst_object = "<yourDestinationObjectName>";

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    // Initialize a multipart copy.
    $upload_id = $ossClient->initiateMultipartUpload($dst_bucket, $dst_object);

    $copyId = 1;
    // Copy the parts one-by-one.
    $eTag = $ossClient->uploadPartCopy( $src_bucket, $src_object, $dst_bucket, $dst_object,$copyId, $upload_id);
    $upload_parts[] = array(
        'PartNumber' => $copyId,
        'ETag' => $eTag,
    );

    // Complete the multipart copy.
    $result = $ossClient->completeMultipartUpload($dst_bucket, $dst_object, $upload_id, $upload_parts);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

6.7.8 アーカイブオブジェクトの復元

アーカイブオブジェクトを読み込む前に復元する必要があります。 非アーカイブオブジェクトに対して restoreObject を呼び出さないでください。

アーカイブオブジェクトの状態変換プロセスは次のとおりです。

1. アーカイブオブジェクトはフリーズ状態です。
2. 復元のために送信した後、サーバーはオブジェクトを復元します。 オブジェクトは復元中の状態です。
3. 復元後にオブジェクトを読むことができます。 オブジェクトの復元された状態は、既定で 1 日続きます。 この期間は最大 7 日間まで延長できます。 この期間が終了すると、オブジェクトはフリーズ状態に戻ります。

次のコードを実行してオブジェクトを復元します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. RAM ユーザーとしてログオンして API にアクセスするか、定期的な運用とメンテナンスを実行することを推奨します。 To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). 要件に基づいて実際のエンドポイントを指定してください。
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->restoreObject($bucket, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

アーカイブストレージクラスの詳細については、「ストレージクラスの概要」をご参照ください。

6.7.9 シンボリックリンクの管理

シンボリックリンクの作成

シンボリックリンクは、Windows で使用されるショートカットに似たオブジェクトにマップされる特殊なオブジェクトです。 シンボリックリンクには、ユーザー定義のオブジェクトメタを構成します。

次のコードを実行してシンボリックリンクを作成します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
```

```
use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$symlink = "<yourSymlink>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->putSymlink($bucket, $symlink, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

シンボリックリンクのオブジェクトコンテンツの取得

シンボリックリンクを取得するには、読み取り許可を持っていなければなりません。 次のコードを実行して、シンボリックリンクのオブジェクトコンテンツを取得します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$symlink = "<yourSymlink>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $Symlinks = $ossClient->getSymlink($bucket, $symlink);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
```

```
}
print(__FUNCTION__ . ": OK" . "\n");
var_dump($Symlinks);
```

6.7.10 MD5 を有効にする

MD5 検証は、送信中のデータの完全性を保証するために実行されます。 MD5 検証はパフォーマンスの低下をもたらします。 ファイルのアップロード中、MD5 検証は既定では無効になっています。

ファイルのアップロード中にMD5 検証を有効にするには、次のコードを実行します。

```
<? php
if (is_file(__DIR__ . '/../Autopoload. php ')){
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the account has permissions on all the APIs in OSS. We recommend that you log on as a RAM user to access APIs or perform routine operations and maintenance. To create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

$options = array(OssClient::OSS_CHECK_MD5 => true);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

    $ossClient->uploadFile($bucket, $object, __FILE__, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

putObject、uploadFile、appendObject、appendFile、および multiuploadFile メソッドは MD5 検証をサポートします。

6.8 画像処理

Image Processing (IMG) は、大規模で安全、低コストで、信頼性の高い画像処理サービスです。ソース画像 が OSS にアップロードされると、簡易な RESTful API を通して、いつでも、どこからでも、どのインターネットデバイス上の画像でも処理できます。

IMG の詳細については、「Image Processing」をご参照ください。 画像処理の完全なコードについては、『GitHub』をご参照ください。

基本機能

OSS は以下の IMG 機能を提供します。

- ドミナントイメージトーンの取得
- 画像フォーマット変換
- 画像サイズの変更
- 内接円の調整
- 画像の向きの調整
- 画像輝度
- 透かしの追加: 画像、テキスト、または画像テキストの透かしを別の画像に追加します。
- 画像処理
- 画像処理アクセスルール: 複数の IMG 関数を呼び出します。

使用方法

IMG"は標準の HTTP GET を使用します。 URL の QueryString に IMG パラメーターを設定します。

イメージオブジェクトの ACL が "非公開読み書き" の場合、承認されたユーザーだけがアクセスを許可されます。

- 匿名アクセス

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
  ```

パラメーター	説明
bucket	バケットの名前を指定します。
endpoint	リージョンへのアクセスに使用されるエンドポイントを指定します。

パラメーター	説明
object	画像オブジェクトの名前を指定します。
image	IMG の予約済み識別子を指定します。
action	拡大縮小、トリミング、回転など、画像に対する操作を指定します。

パラメーター	説明
param	画像に対する操作を示すパラメーターを指定します。

- 基本操作

  たとえば、画像を 100 PX の幅に拡大縮小します。 比率に基づいて高さを調整します。

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- カスタマイズされた画像スタイル

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  ■ style: カスタマイズされた画像スタイルの予約済み識別子を指定します。

  ■ yourStyleName：カスタム画像スタイルの名前を指定します。 OSS コンソールで作成されたルールで指定された名前です。

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- カスケード処理

  カスケード処理では、画像に対して複数の操作を順番に実行します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- HTTPS アクセス

  IMG は HTTPS を介したアクセスをサポートしています。 例:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

• アクセス承認

  アクセス承認により、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

次のコードを実行して IMG のサイン済み URL を生成します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because the
 account has permissions on all the APIs in OSS. We recommend that you log on as a
RAM user to access APIs or perform routine operations and maintenance. To create a
RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual endpoint
based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

// Generate a signed URL that is valid for 3600 seconds and can be directly accessed
by browsers.
$timeout = 3600;
$options = array(
    OssClient::OSS_PROCESS => "image/resize,m_lfit,h_100,w_100" );
$signedUrl = $ossClient->signUrl($bucket, $object, $timeout, "GET", $options);

print("rtmp url: \n" . $signedUrl);
```

- SDK を使用したアクセス

  SDK を使用して、画像オブジェクトにアクセスして処理します。

  SDK を使用すると、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

  - 基本操作

    次のコードを実行して、画像に対して基本操作を実行します。

    ```
    <? php
    if (is_file(__DIR__ . '/../autoload.php')) {
        require_once __DIR__ . '/../autoload.php';
    }
    if (is_file(__DIR__ . '/../vendor/autoload.php')) {
        require_once __DIR__ . '/../vendor/autoload.php';
    }

    use OSS\OssClient;

    // It is highly risky to log on with AccessKey of an Alibaba Cloud account because
     the account has permissions on all APIs in OSS. We recommend that you log on
    ```

```
as a RAM user to access APIs or perform routine operations and maintenance. To
create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket = "<yourBucketName>";
$object = "<yourObjectName>";
$download_file = "download.jpg";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

// Upload a sample image.
$ossClient->uploadFile($bucket, $object, "example.jpg");

// Scale the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/resize,m_fixed,h_100,w_100" );
$ossClient->getObject($bucket, $object, $options);

// Crop the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/crop,w_100,h_100,x_100,y_100,r_1" );
$ossClient->getObject($bucket, $object, $options);

// Rotate the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/rotate,90" );
$ossClient->getObject($bucket, $object, $options);

// Sharpen the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/sharpen,100" );
$ossClient->getObject($bucket, $object, $options);

// Add watermarks.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5Y
qhIQ" );
$ossClient->getObject($bucket, $object, $options);

// Convert the image format.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/format,png" );
$ossClient->getObject($bucket, $object, $options);

// Obtain image information.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/info" );
$ossClient->getObject($bucket, $object, $options);

// Delete the sample image.
```

```
$ossClient->deleteObject($bucket, $object);
```

- 画像スタイルのカスタマイズ

  次のコードを実行して画像スタイルをカスタマイズします。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because
 the account has permissions on all APIs in OSS. We recommend that you log on
as a RAM user to access APIs or perform routine operations and maintenance. To
create a RAM account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket = "<yourBucketName>";
$object = "<yourObjectName>";
$download_file = "download.jpg";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);

// Upload a sample image.
$ossClient->uploadFile($bucket, $object, "example.jpg");

// Customize an image style
$style = "style/oss-pic-style-w-300";
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => $style);
$ossClient->getObject($bucket, $object, $options);

// Delete the sample image.
$ossClient->deleteObject($bucket, $object);
```

- カスケード処理

  画像に対してカスケード処理を実行するには、次のコードを使用します。

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;

// It is highly risky to log on with AccessKey of an Alibaba Cloud account because
 the account has permissions on all APIs in OSS. We recommend that you log on
as a RAM user to access APIs or perform routine operations and maintenance. To
create a RAM account, log on to https://ram.console.aliyun.com.
```

```
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket = "<yourBucketName>";
$object = "<yourObjectName>";
$download_file = "download.jpg";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false);

// Upload a sample image.
$ossClient->uploadFile($bucket, $object, "example.jpg");

// Perform cascade operations.
$style = "image/resize,m_fixed,w_100,h_100/rotate,90";
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => $style);
$ossClient->getObject($bucket, $object, $options);

// Delete the sample image.
$ossClient->deleteObject($bucket, $object);
```

IMG ツール

IMG ビューアー (ImageStyleViever) を使用すると、IMG の結果を直接表示できます。

6.9 例外処理

OSS PHP SDK の例外 (OssException) には、無効なパラメーターやファイルの欠落など、さまざまなエラーが含まれます。 エラーメッセージを取得するには、getMessage メソッドを使用します。

OssException の詳細については、『GitHub』をご参照ください。

例外処理の例

次のコードは、既存のバケットを作成するときに例外を処理する方法を示し、エラーメッセージを出力します。

```
  try {
     $ossClient->createBucket($bucket);
  } catch (OssException $e) {
print("Exception:" . $e->getMessage() . "\n");
  }
```

以下の情報も取得できます。

パラメーター	説明
HTTPStatus	HTTP status code. getHTTPStatus メソッドを使ってステータスコードを取得します。

パラメーター	説明
ErrorCode	OSS から返されたエラーコードを指定します。エラーコードは getErrorCode メソッドを使用して取得します。
ErrorMessage	OSS から返されるエラーメッセージを指定します。 getErrorMessage メソッドを使用してエラーメッセージを取得します。
RequestId	UUID を指定します。 これは固有のものであり、リクエストを識別するために使用されます。 問題が解決しない場合は、OSS 開発者に RequestId を送ってヘルプを求めてください。getRequestId メソッドを使用して RequestId を取得します。
Details	OSS から返される詳細なエラーメッセージを指定します。 getDetails メソッドを使用して詳細なエラーメッセージを取得します。

一般的な OSS エラーコード

エラーコード	説明	**HTTP** ステータスコード
AccessDenied	アクセスが拒否されました。	403
BucketAlreadyExists	バケットが既に存在します。	409
BucketNotEmpty	バケットは空ではありません。	409
EntityTooLarge	エンティティサイズが上限を超えています。	400
EntityTooSmall	エンティティサイズが最小制限を下回っています。	400
FileGroupTooLarge	ファイルグループの合計サイズが上限を超えています。	400
FilePartNotExist	そのようなパーツはありません。	400
FilePartStale	そのパーツは期限切れです。	400
InvalidArgument	パラメーターの形式が無効です。	400
InvalidAccessKeyId	そのような AccessKeyId は存在しません。	403

エラーコード	説明	**HTTP** ステータスコード
InvalidBucketName	バケット名が無効です。	400
InvalidDigest	ダイジェストが無効です。	400
InvalidObjectName	オブジェクト名が無効です。	400
InvalidPart	そのパーツは無効です。	400
InvalidPartOrder	パーツオーダーが無効です。	400
InvalidTargetBucketForLogging	ログファイルを格納するバケットが無効です。	400
InternalError	OSS のエラーです。	500
MalformedXML	XML 形式が無効です。	400
MethodNotAllowed	このメソッドはサポートされていません。	405
MissingArgument	パラメーターが設定されていません。	411
MissingContentLength	コンテンツの長さが設定されていません。	411
NoSuchBucket	そのようなバケットは存在しません。	404
NoSuchKey	そのようなオブジェクト (キー) は存在しません。	404
NoSuchUpload	そのような uploadId は存在しません。	404
NotImplemented	そのメソッドは実装されていません。	501
PreconditionFailed	事前条件のエラーです。	412
RequestTimeTooSkewed	OSSClient に設定された現地時間が、OSS サーバーに設定された時間から 15 分以上ずれています。	403
RequestTimeout	リクエストはタイムアウトしました。	400
SignatureDoesNotMatch	署名が間違っています。	403
InvalidEncryptionAlgorithmError	指定された暗号化アルゴリズム (エントロピー符号化タイプ) が無効です。	400

7 C

7.1 はじめに

この文書は OSS C SDK 3.5.0 に基づき作成されています。

互換性

- SDK シリーズ 3.*.*: 互換性があります。
- SDK シリーズ 2.*.*:
  - Windows: 互換性があります。
  - Linux API: リンクリスト (aos_list_t) トラバーサルインターフェイスを除き、互換性があります。
    - os_list_for_each_entry
    - aos_list_for_each_entry_reverse
    - aos_list_for_each_entry_safe
    - aos_list_for_each_entry_safe_reverse
- SDK シリーズ 1.0.0: 以下の API にのみ互換性がありません。
  - oss_request_options_t
  - oss_get_object_to_buffer
  - oss_get_object_to_file
  - oss_get_object_to_buffer_by_url
  - oss_get_object_to_file_by_url
  - oss_init_multipart_upload
  - oss_complete_multipart_upload
- For SDK series of 0.0.*: 互換性がありません。

SDK のソースコード

Java SDK のソースコードについては、『GitHub』をご参照ください。

コードサンプル

OSS SDK for C は、参考用または使用目的でさまざまなコードサンプルを提供しています。 次の表にコードサンプルの内容を示します。

サンプルファイル	コンテンツ
oss_put_object_sample	オブジェクトのアップロード
oss_get_object_sample.c	オブジェクトのダウンロード
oss_append_object_sample.c	アップロードの追加
oss_multipart_upload_sample.c	マルチパートアップロード
oss_resumable_sample.c	再開可能アップロード、再開可能ダウンロード
oss_head_object_sample.c	オブジェクトメタデータの管理
oss_list_object_sample.c	オブジェクトの列挙
oss_delete_object_sample.c	オブジェクトの削除
oss_callback_sample.c	コールバックのアップロード
oss_progress_sample.c	プログレスバー、プログレスバー
oss_crc_sample.c	アップロードまたはダウンロード中の CRC チェック
oss_image_sample.c	イメージ処理

7.2 インストール

Linux への OSS C SDK のインストール

- CMake とサードパーティーのライブラリをインストール

  Linux に OSS C SDK をインストールするときは、CMake と以下のサードパーティーライブラリをインストールする必要があります。

  環境をインストールするには、以下のパラメーターが必要です。

パラメーター	説明	バージョン要件
CMake	ビルドとインストールツール。	バージョン 2.6.0 以降
curl	ネットワークの問題に焦点を当てています。	バージョン 7.32.0 以降
apr-util	メモリ管理とクロスプラットフォーム操作に焦点を当てています。	バージョン 1.5.2 以降

パラメーター	説明	バージョン要件
minixml	リクエストによって返された XML を解析します。	バージョン 2.9 を使用することを推奨します。

CMake とサードパーティーのライブラリをインストールする必要があるオペレーティングシステムを選択します。

- Ubuntu / Debian
  - CMake をインストール

    ```
    sudo apt-get install apache2
    ```

  - サードパーティーのライブラリをインストール

    ```
    sudo apt-get install libcurl4-openssl-dev libapr1-dev libaprutil1-dev libmxml-dev
    ```

    mxml インストールパッケージが yum ソースに含まれていない場合は、RPM を使用してライブラリをインストールします。 ご使用のオペレーティングシステムに基づいて rpm インストールパッケージをダウンロードします。

    - 64-bit: mxml-2.9-1.x86_64.rpm
    - 32-bit: mxml-2.9-1.i386.rpm

    次のコードを実行して、rpm インストールパッケージと共に mxml をインストールします。

    ```
    rpm -ivh mxml-2.9-1.x86_64.rpm
    ```

- RedHat/Aliyun/CentOS
  - CMake をインストール

    ```
    sudo yum install cmake
    ```

  - サードパーティーのライブラリをインストール

    ```
    sudo yum install curl-devel apr-devel apr-util-devel  mxml mxml-devel
    ```

- SuSE

■ CMake をインストール

```
zypper install cmake
```

■ サードパーティーのライブラリをインストールします。

```
zypper install libcurl-devel libapr1-devel libapr-util1-devel mxml-devel
```

- Other Linux

■ CMake をインストール： ダウンロードリンク

一般的なインストール方法は次のとおりです。

```
./configure
make
make install
```

注：

./configure を実行している場合、既定のインストールパスは /usr/local/ です。 別のパスを指定したい場合は、./configure --prefix option を使用します。

■ libcurl をインストール: ダウンロードリンク

一般的なインストール方法は次のとおりです。

```
./configure
make
make install
```

■ apr をインストール: ダウンロードリンク

一般的なインストール方法は次のとおりです。

```
./configure
make
make install
```

■ apr-util をインストール: ダウンロードリンク

一般的なインストール方法は次のとおりです。

```
// You must specify the --with-apr option in the installation.
./configure --with-apr=/your/apr/install/path
make
make install
```

■ minixml をインストール：ダウンロードリンク

一般的なインストール方法は次のとおりです。

```
./configure
make
```

```
sudo make install
```

- SDK をダウンロード
  - SDK を直接ダウンロード
  - 『GitHub』 から SDK をダウンロード
  - 旧バージョンの SDK をダウンロード
- SDK をインストール

  cmake を使用します。 url、apr、apr-util、mxml などのサードパーティーライブラリが、コンパイルプロセス中に既定の方法でインストールされない場合は、それらのヘッダーファイルとライブラリファイルのパスを指定する必要があります。

  次のように SDK をインストールします。

```
cmake .
make
make install
```

  CentOS でのインストール例は以下の通りです。

```
cmake -f CMakeLists.txt
// The compilation type is Release. Common compilation types include: Debug, Release, RelWithDebInfo, and MinSizeRel, in which Debug is the default type.
-DCMAKE_BUILD_TYPE=Release
// Customize the installation directory.
-DCMAKE_INSTALL_PREFIX=/usr/local/
// Specify the path where the header files and library files of third party libaries, such as curl, apr, apr-util, and xml, are located,
-DCURL_INCLUDE_DIR=/usr/include/curl
-DCURL_LIBRARY=/usr/lib64/libcurl.so
-DAPR_INCLUDE_DIR=/usr/include/apr-1
-DAPR_LIBRARY=/usr/lib64/libapr-1.so
-DAPR_UTIL_INCLUDE_DIR=/usr/include/apr-1
-DAPR_UTIL_LIBRARY=/usr/lib64/libaprutil-1.so
-DMINIXML_INCLUDE_DIR=/usr/include
-DMINIXML_LIBRARY=/usr/lib64/libmxml.so
// If the "Could not find apr-config/apr-1-config" error occurs during the compiling process, it indicates that the apr-1-config file cannot be found in the default path. Add this option.
-DAPR_CONFIG_BIN=/path/to/bin/apr-1-config
// If the "Could not find apu-config/apu-1-config" error occurs, add this option.
-DAPU_CONFIG_BIN=/path/to/bin/apu-1-config
```

Windows への OSS C SDK のインストール

- SDK をダウンロード
  - SDK を直接ダウンロード
  - 『GitHub』 から SDK をダウンロード
  - 旧バージョンの SDK をダウンロード

- SDK をインストール

  Visual Studio を使用してOSS C SDK をコンパイルする際の詳細な手順と一般的な問題については、「Windows での Alibaba Cloud OSS C SDK のコンパイルと使用」をご参照ください。

  注：
  Visual Studio 2012 以降のバージョンを使用している場合は、プロジェクトを最新のコンパイラとライブラリにアップグレードするように求められます。 プロジェクトに基づいてアップグレードするかどうかを決定することを推奨します。 プロジェクトで最新のコンパイラとライブラリを使用している場合はアップグレードできます。

7.3 クイックスタート

このトピックでは、OSS C SDK を使用してバケットの作成、オブジェクトのアップロード、およびオブジェクトのダウンロードなどの日常的な操作を完了する方法について説明します。

デモ

- Linux デモ

  Linuxベースのデモ: aliyun-oss-c-sdk-demo.tar.gz

  1. デモプロジェクトをダウンロードして解凍します。
  2. oss-c-sdk-demo 指定インストールのインストールディレクトリを指定する必要があります。/home/your/oss/csdk 他のデモプロジェクトは、OSS C SDK および他の依存するサードパーティライブラリに基づいて自動的にインストールされるため、インストールディレクトリを指定する必要はありません。
  3. デモの "OSS_ENDPOINT"、"ACCESS_KEY_ID"、"ACCESS_KEY_SECRET"、"BUCKET_NAME" を有効な値に置き換えます。 システムディレクトリにない場合は、`LD_LIBRARY_PATH` を使用して、OSS C SDK のディレクトリと従属ライブラリの動的ライブラリを指定します。
  4. プロジェクトのディレクトリ (oss-c-sdk-demo-xxx) を入力し、`make` コマンドを実行してデモプロジェクトをコンパイルします。 次に、 `./main` コマンドを実行して、実行可能プログラムを実行します。 プロジェクトを再コンパイルする必要がある場合は、 `make clean` を実行します。

  コンパイルする前に、まずコマンドを実行します。https://yq.aliyun.com/articles/60924

- Windows デモ

  Windows ベースのデモ: aliyun- os-c-sdk-sample.zip 『GitHub』から他のデモプロジェクトをダウンロードします。

  1. デモプロジェクトをダウンロードして解凍します。
  2. Visual Studio を使用してデモプロジェクトを開き、oss_config.c の次のパラメーターを有効な値で置き換えま
     す。"OSS_ENDPOINT"、"ACCESS_KEY_ID"、"ACCESS_KEY_SECRET"、"BUCKET_NAME"、"OBJECT_NAM
     プロジェクトをコンパイルして実行します。 デモプロジェクトに基づき、ユーザー自身のプロジェクトを開発することができます。

     注 :
     Visual Studio でデモプロジェクトを実行する手順の詳細については、関連文書をご参照ください。 https://yq.aliyun.com/articles/57947

バケットの作成

バケットは、オブジェクトを格納するために使用される OSS 上のグローバルネームスペースです。 データコンテナーに相当します。 次のコードはバケットを作成します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 the network and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Create the bucket. */
    aos_status_t *resp_status;
    aos_table_t *resp_headers;
    oss_acl_e oss_acl = OSS_ACL_PRIVATE;
    aos_string_t bucket;
    /* Assign a char* to the bucket. */
    aos_str_set(&bucket, bucket_name);
    resp_status = oss_create_bucket(oss_client_options, &bucket, oss_acl, &resp_headers);
    /* Identify whether the request succeeds */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

エンドポイントに関する詳細は、「リージョンとエンドポイント」をご参照ください。

オブジェクトのアップロード

次のコードを実行してオブジェクトを OSS にアップロードします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) { = AOSE_OK) {
        exit(1);
    }
```

```
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memory in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_string_t object;
    aos_status_t *resp_status;
    aos_table_t *headers;
    aos_table_t *resp_headers;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    const char *data = "More than just cloud.";
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers = aos_table_make(pool, 0);
    /* Convert the char* type data to the aos_list_t type data. */
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, data, strlen(data));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload an object. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer
, headers, &resp_headers);
    /* Determine whether the upload request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("put file succeeded\n");
    } else {
        printf("put file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

オブジェクトのダウンロード

指定したオブジェクトをダウンロードするには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *download_filename = "<yourDownloadedFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
```

```
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_status_t *resp_status;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, download_filename);
    headers = aos_table_make(pool, 0);
    /* Download an object. */
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers,
params, &file, &resp_headers);
    /* Determine whether the download request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("get object to local file succeeded\n");
    } else {
        printf("get object to local file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

バケット内のオブジェクトの一覧表示

次のコードを実行して、指定したバケットに格納されているオブジェクトを一覧表示します。

```
#include "oss_api.h"
```

```
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool = NULL;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options = NULL;
    /* Allocate memory in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    /* List objects */
    resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
    /* Determine whether the list request is successful. */
    if (! aos_status_is_ok(resp_status))
    {
        printf("list object failed\n");
    }
    else{
        /* Print objects. */
        printf("Object\tSize\tLastModified\n");
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list,
node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data
,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
```

```
        }
        printf("Total %d\n", size);
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

オブジェクトの削除

次のコードを実行して、オブジェクトを削除します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool = NULL;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options = NULL;
    /* Allocate memory in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_string_t object;
    aos_status_t *resp_status = NULL;
    aos_table_t *resp_headers = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Delete an object. */
```

```
    resp_status = oss_delete_object(oss_client_options, &bucket, &object, &resp_headers);
    /* Determine whether the delete request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("delete object succeeded\n");
    } else {
        printf("delete object failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.4 初期化

OSS C SDK を使用するには、リクエストオプション (oss_request_options_t) を初期化し、エンドポイントを指定する必要があります。 エンドポイントの詳細については、「リージョンおよびエンドポイント」を参照し、「カスタムドメイン名」をバインドします。

OSS ドメイン名を使用したリクエストオプションの初期化

OSS ドメイン名を使用して次のコードを実行することで、リクエストオプションを初期化します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main() {
    aos_pool_t *p;
    oss_request_options_t *options;
    /* Initialize global variables, which are run first before the program starts. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        return -1;
    }
    /* Initialize the memory pool and options. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    /* Logical code, which is not listed here. */
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(p);
```

```
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

注：

リクエストオプションの設定の詳細については、関連ドキュメントをご参照ください。 http://bbs.aliyun.com/read/261421.html?spm=5176.bbsr248504.0.0.NoFhRX

カスタムドメイン名を使用したリクエストオプションの初期化

次のコードを実行することで、リクエストオプションをドメイン名で初期化します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourCustomEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options) {
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Enable the CNAME and bind the custom domain name to the bucket. */
    options->config->is_cname = 1;
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main() {
    aos_pool_t *p;
    oss_request_options_t *options;
    /* Initialize global variables, which are run before the program starts. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        return -1;
    }
    /* Initialize the memory pool and options. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    /* Logical code, which is not listed here. */
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

STS を使用したリクエストオプションの初期化

STS を使用してリクエストオプションを初期化するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *sts_token = "<yourSecurityToken>";
```

```
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Configure STS */
    aos_str_set(&options->config->sts_token, sts_token);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main() {
    aos_pool_t *p;
    oss_request_options_t *options;
    /* Initialize global variables, which are run first before the program starts. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        return -1;
    }
    /* Initialize the memory pool and options. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    /* Logical code, which is not listed here. */
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

タイムアウト値の設定

タイムアウト値を設定するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
    /* Set the connection timeout value, which is 10 seconds by default. */
    options->ctl->options->connect_timeout = 10;
    /* Set the DNS timeout value, which is 60 seconds by default. */
    options->ctl->options->dns_cache_timeout = 60;
    /* Set the request timeout value:
    Set the value of speed_limit to specify the minimum speed that can be accepted, which
 is 1,024 Bytes/s by default.
    Set the value of speed_time to specify the maximum time period that can be accepted,
which is 15 seconds by default.
```

```
    The following code indicates that if the trasmission speed is continuously lower than 1,
024 Bytes/s for 15 seconds, the request is time-out. */
    options->ctl->options->speed_limit = 1024;
    options->ctl->options->speed_time = 15;
}
```

7.5 オブジェクトのアップロード

7.5.1 概要

OSS では、オブジェクトはデータ操作の基本単位です。 OSS C SDK は次のファイルアップロード方法を提供します。

- 簡易アップロード: 最大 5 GB までのファイルのアップロードをサポートします。 ストリーミングアップロードとファイルアップロードが含まれます。
- 追加アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 再開可能アップロード: 最大 48.8 TB のファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。 同時アップロードもサポートしており、各パーツのサイズを定義できます。
- マルチパートアップロード: 最大 48.8 TB のファイルのアップロードをサポートします。 大容量ファイルのアップロードに適用されます。

注 :
各アップロード方法の使用シナリオの詳細については、「OSS 開発者ガイド」の「ファイルのアップロード」をご参照ください。

ファイルのアップロード中は、オブジェクトメタを設定することで、アップロードの進行状況をアップロード進行状況バーで確認できます。 ファイルのアップロードが完了すると、アップロードコールバックを実行できます。

7.5.2 簡易アップロード

簡易アップロードの完全なコードについては、『 GitHub』をご参照ください。

メモリから OSS へのデータアップロード

次のコードを実行することで、メモリから OSS にデータをアップロードします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
```

```
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_con
tent));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload a file. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer
, headers, &resp_headers);
    /* Determine whether the upload is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("put object from buffer succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

ローカルファイルを OSS へアップロード

次のコードを実行して、ローカルファイルを OSS にアップロードします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Upload a file. */
    resp_status = oss_put_object_from_file(oss_client_options, &bucket, &object, &file,
headers, &resp_headers);
    /* Determine whether the upload is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
```

```
    } else {
        printf("put object from file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.5.3 追加アップロード

追加オブジェクトに対して copyObject は実行できません。

メモリからのデータの追加

次のコードを実行し、メモリからデータを追加します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is not
used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The implementa
tion code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the new
pool does not inherit any other memory pool. */
    aos_pool_create(&p, NULL);
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
```

```
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    int64_t position = 0;
    char *next_append_position = NULL;
    aos_buf_t *content = NULL;
    aos_table_t *headers1 = NULL;
    aos_table_t *headers2 = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers1 = aos_table_make(pool, 0);
    /* Obtain the position where the object is appended for the first time. */
    resp_status = oss_head_object(oss_client_options, &bucket, &object, headers1, &
resp_headers);
    if (aos_status_is_ok(resp_status)) {
        next_append_position = (char*)(apr_table_get(resp_headers, "x-oss-next-append-
position"));
        position = atoi(next_append_position);
    }
    /* Append the file. */
    headers2 = aos_table_make(pool, 0);
    aos_list_init(&buffer);
    content = aos_buf_pack(pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    resp_status = oss_append_object_from_buffer(oss_client_options, &bucket, &object,
position, &buffer, headers2, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("append object from buffer succeeded\n");
    } else {
        printf("append object from buffer failed\n");
    }
    /* Release the memory pool, that is, the memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

ローカルファイルからのデータの追加

ローカルファイルからデータを追加するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is not
used. */
```

```
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The implementa
tion code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the new
pool does not inherit any other memory pool. */
    aos_pool_create(&p, NULL);
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    int64_t position = 0;
    char *next_append_position = NULL;
    aos_table_t *headers1 = NULL;
    aos_table_t *headers2 = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Obtain the position where the object is appended for the first time. */
    resp_status = oss_head_object(oss_client_options, &bucket, &object, headers1, &
resp_headers);
    if(aos_status_is_ok(resp_status)) {
        next_append_position = (char*)(apr_table_get(resp_headers, "x-oss-next-append-
position"));
        position = atoi(next_append_position);
    }
    /* Append the file. */
    resp_status = oss_append_object_from_file(oss_client_options, &bucket, &object,
position, &file, headers2, &resp_headers);
    /* Determine whether the append upload is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("append object from file succeeded\n");
    } else {
        printf("append object from file failed\n");
    }
    /* Release the memory pool, that is, the memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

ファイルが既に存在する場合は、以下のことが起こります。

- 追加アップロードでファイルをアップロードできない場合、ObjectNotAppendable の例外が発生します。
- 追加アップロードでファイルをアップロードできる場合、ファイルは Put Object メソッドによってアップロードされ、既存のオブジェクトに上書きされます。 オブジェクトタイプが Normal Object に変更されます。 Head Object 操作が実行されると、オブジェクトタイプを示す x-oss-object-type をシステムが返します。

7.5.4 再開可能アップロード

再開可能アップロードでは、オブジェクトは複数のパーツに分割され、別々にアップロードされます。 すべてのパーツがアップロードされると、各パーツは完全オブジェクトにマージされてアップロードが完了します。

現在のアップロードの進行状況は、アップロード中にチェックポイントファイルに記録されます。 処理中にパーツのアップロードに失敗した場合、オブジェクトはアップロードの再開時にチェックポイントファイルに記録されている再開可能なアップロードパーツからアップロードされます。 アップロードが完了すると、チェックポイントは削除されます。

再開可能アップロードを実現するには、oss_resumable_clt_params_t メソッドを使用します。このメソッドで使用される一般的なパラメーターは次のとおりです。

パラメーター	説明
thread_num	同時スレッド数。既定では 1 です。
enable_checkpoint	再開可能アップロードを有効にするかどうか。既定では無効になっています。
partSize	パーツサイズ。100 KB から 5 GB の範囲です。
checkpoint_path	チェックポイントファイルが保存されているパス。既定では "{upload_file_path} .cp" です。

次のコードを実行することで、再開可能アップロードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
```

```
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Deternube whether the CNAME is used. 0 indicates that it is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_list_t resp_body;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_resumable_clt_params_t *clt_params;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    aos_list_init(&resp_body);
    /* Resumable upload. */
    clt_params = oss_create_resumable_clt_params_content(pool, 1024 * 100, 3, AOS_TRUE
, NULL);
    resp_status = oss_resumable_upload_file(oss_client_options, &bucket, &object, &file,
headers, NULL, clt_params, NULL, &resp_headers, &resp_body);
    if (aos_status_is_ok(resp_status)) {
        printf("resumable upload succeeded\n");
    } else {
        printf("resumable upload failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.5.5 マルチパートアップロード

マルチパートアップロードを有効にするには、次の手順を実行します。

1. マルチパートアップロードイベントを開始します。

   ossClient.initiateMultipartUpload を呼び出して、OSS で作成されたグローバルに一意の uploadId を返します。

2. パーツのアップロード

   パーツデータをアップロードするには、 ossClient.uploadPart を呼び出します。

   :

   - 同じ uploadId を持つパーツの場合、パーツはそれらのパーツ番号によって順序付けされます。 パーツをアップロードし、同じパーツ番号を使用して別のパーツをアップロードすると、後のパーツが前のパーツに置き換わります。
   - OSS はパーツデータの MD5 値を ETag に格納し、MD5 値をユーザーに返します。
   - SDK は自動的に Content-MD5 を設定します。 OSS はアップロードされたデータの MD5 値を計算し、それを SDK によって計算された MDS 値と比較します。 2 つの値が異なると、InvalidDigest のエラーコードが返されます。

3. マルチパートアップロードの完了

   すべてのパーツをアップロードすると、 partossClient.completeMultipartUpload を呼び出し、各パーツを完全なオブジェクトにまとめます。

マルチパートアップロードのプロセスを説明する完全なコードの例は次のとおりです。

```
#include "oss_api.h"
#include "aos_http_io.h"
#include <sys/stat.h>
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    ptions->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int64_t get_file_size(const char *file_path)
{
    int64_t filesize = -1;
    struct stat statbuff;
    if(stat(file_path, &statbuff) < 0){
        return filesize;
```

```
    } else {
        filesize = statbuff.st_size;
    }
    return filesize;
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
nformation, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    oss_upload_file_t *upload_file = NULL;
    aos_string_t upload_id;
    aos_table_t *headers = NULL;
    aos_table_t *complete_headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_null(&upload_id);
    headers = aos_table_make(p, 1);
    complete_headers = aos_table_make(p, 1);
    int part_num = 1;
    /* Initialize multipart upload and obtain an upload ID (upload_id). */
    resp_status = oss_init_multipart_upload(oss_client_options, &bucket, &object, &
upload_id, headers, &resp_headers);
    /* Determines whether the slice upload Initialization is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("Init multipart upload succeeded, upload_id:%.*s\n",
            upload_id.len, upload_id.data);
    } else {
        printf("Init multipart upload failed, upload_id:%.*s\n",
            upload_id.len, upload_id.data);
    }
    /* Upload the parts. */
    int64_t file_length = 0;
    int64_t pos = 0;
    file_length = get_file_size(local_filename);
    while(pos < file_length) {
        upload_file = oss_create_upload_file(pool);
        aos_str_set(&upload_file->filename, local_filename);
        upload_file->file_pos = pos;
        pos += 100 * 1024;
        upload_file->file_last = pos < file_length ? pos : file_length;
        aos_list_for_each_entry（oss_list_part_content_t、part_content、&amp; params-&
gt; part_list、node）{
        if (aos_status_is_ok(resp_status)) {
```

```
            printf("Multipart upload part from file succeeded\n");
        } else {
            printf("Multipart upload part from file failed\n");
        }
    }
    oss_list_upload_part_params_t *params = NULL;
    aos_list_t complete_part_list;
    /* Obtain uploaded parts. */
    params = oss_create_list_upload_part_params(pool);
    params->max_ret = 1000;
    aos_list_init(&complete_part_list);
    resp_status = oss_list_upload_part(oss_client_options, &bucket, &object, &upload_id,
params, &resp_headers);
    /* Determine whether the part list is successfully obtained. */
    if (aos_status_is_ok(resp_status)) {
        printf("List multipart succeeded\n");
    } else{
        printf("List multipart failed\n");
    }
    oss_complete_part_content_t *complete_part_content = NULL;
    oss_list_part_content_t *part_content = NULL;
    aos_list_for_each_entry(oss_list_part_content_t, part_content, &params->part_list,
node) {
        complete_part_content = oss_create_complete_part_content(pool);
        aos_str_set(&complete_part_content->part_number, part_content->part_number.
data);
        aos_str_set(&complete_part_content->etag, part_content->etag.data);
        aos_list_add_tail(&complete_part_content->node, &complete_part_list);
    }
    /* Complete multipart upload. */
    resp_status = oss_complete_multipart_upload(oss_client_options, &bucket, &object, &
upload_id,
            &complete_part_list, complete_headers, &resp_headers);
    /* Determine whether the multipart upload is complete. */
    if (aos_status_is_ok(resp_status)) {
        printf("Complete multipart upload from file succeeded, upload_id:%.*s\n",
               upload_id.len, upload_id.data);
    } else {
        printf("Complete multipart upload from file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

注：

アップロードされたパーツを取得するため、 oss_complete_multipart_upload を呼び出すときには、すべての有効な PartETag を指定する必要があります。 ETag は 2 種類の方法で取得できます。1: パーツがアップロードされると、そのパーツの ETag が返される結果の中に含まれます。 2: アップロードしたパーツの "ETag" を取得するには、 oss_list_upload_part を呼び出します。 この例では、2 つ目のメソッドが使用されています。

マルチパートアップロードイベントのキャンセル

次のコードを実行してマルチパートアップロードイベントをキャンセルします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as


networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration

information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t upload_id;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_null(&upload_id);
    /* Initialize multipart upload and obtain an upload ID (upload_id). */
    resp_status = oss_init_multipart_upload(oss_client_options, &bucket, &object, &
upload_id, headers, &resp_headers);
    /* Determine whether the multipart upload is initialized successfully. */
```

```
    if (aos_status_is_ok(resp_status)) {
        printf("Init multipart upload succeeded, upload_id:%.*s\n",
            upload_id.len, upload_id.data);
    } else {
        printf("Init multipart upload failed, upload_id:%.*s\n",
            upload_id.len, upload_id.data);
    }
    /* Cancel the multipart upload. */
    resp_status = oss_abort_multipart_upload(oss_client_options, &bucket, &object, &
upload_id, &resp_headers);
    /* Determine whether the multipart upload is canceled successfully. */
    if (aos_status_is_ok(resp_status)) {
        printf("Abort multipart upload succeeded, upload_id::%.*s\n",
            upload_id.len, upload_id.data);
    } else {
        printf("Abort multipart upload failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

注：

マルチパートアップロードイベントをキャンセルすると、この upload_id を使用して他の操作を実行することはできなくなります。 アップロードされたパーツは削除されます。

7.5.6 アップロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を示すことができます。

アップロードプログレスバーの完全なコードについては、『GitHub』をご参照ください。

次のコードは、PutObject を使用した進行状況を表示する方法の説明例として使用されています。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
```

```
    options->ctl = aos_http_controller_create(options->pool, 0);
}
void percentage(int64_t consumed_bytes, int64_t total_bytes)
{
    assert(total_bytes >= consumed_bytes);
    printf("%%%" APR_INT64_T_FMT "\n", consumed_bytes * 100 / total_bytes);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_list_t resp_body;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Upload with a progress bar displayed. */
    resp_status = oss_do_put_object_from_file(oss_client_options, &bucket, &object, &file,
NULL, NULL, percentage, &resp_headers, &resp_body);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
    } else {
        printf("put object from file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

oss_put_object_from_file と oss_put_object_from_buffer はプログレスバーをサポートしません。

7.5.7 アップロードコールバック

次のコードを実行して、アップロードコールバックを行います。

```
#include "oss_api.h"
```

```
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is not
used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    aos_pool_t *p = NULL;
    aos_status_t *s = NULL;
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *headers = NULL;
    oss_request_options_t *options = NULL;
    aos_table_t *resp_headers = NULL;
    aos_list_t resp_body;
    aos_list_t buffer;
    aos_buf_t *content;
    char *buf = NULL;
    int64_t len = 0;
    int64_t size = 0;
    int64_t pos = 0;
    char b64_buf[1024];
    int b64_len;
    /* Use the JSON format. */
    char *callback =  "{"
        "\"callbackUrl\":\"http://oss-demo.aliyuncs.com:23450\","
        "\"callbackHost\":\"oss-cn-hangzhou.aliyuncs.com\","
        "\"callbackBody\":\"bucket=${bucket}&object=${object}&size=${size}&mimeType=${
mimeType}\","
        "\"callbackBodyType\":\"application/x-www-form-urlencoded\""
        "}";
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Initialize parameters. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&resp_body);
    aos_list_init(&buffer);
    content = aos_buf_pack(options->pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    /* Add callback into the header. */
    b64_len = aos_base64_encode((unsigned char*)callback, strlen(callback), b64_buf);
    b64_buf[b64_len] = '\0';
```

```
    headers = aos_table_make(p, 1);
    apr_table_set(headers, OSS_CALLBACK, b64_buf);
    /* Start upload callback. */
    s = oss_do_put_object_from_buffer(options, &bucket, &object, &buffer,
        headers, NULL, NULL, &resp_headers, &resp_body);
    if (aos_status_is_ok(s)) {
        printf("put object from buffer succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Obtain the buffer length. */
    len = aos_buf_list_len(&resp_body);
    buf = (char *)aos_pcalloc(p, (apr_size_t)(len + 1));
    buf[len] = '\0';
    /* Copy the content in the buffer to the memory. */
    aos_list_for_each_entry(aos_buf_t, content, &resp_body, node) {
        size = aos_buf_size(content);
        memcpy(buf + pos, content->pos, (size_t)size);
        pos += size;
    }
    /* Release the memory pool, that is, the memories allocated to resources during the
request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

アップロードコールバックの詳細については、「アップロードコールバック」をご参照ください。 詳細なデバッグ方法とトラブルシューティング情報については、「アップロードコールバックエラーと除外」をご参照ください。

7.6 オブジェクトのダウンロード

7.6.1 概要

オブジェクトダウンロードの完全なコードについては、『GitHub』をご参照ください。

OSS C SDK は以下のダウンロード方法を提供します。

- ローカルファイルへのダウンロード
- ローカルメモリへのダウンロード
- 部分ダウンロード
- 再開可能なダウンロード

ダウンロードの進行状況は、ダウンロード進行状況バーに表示されます。

7.6.2 ローカルファイルへのオブジェクトのダウンロード

OSS オブジェクトをダウンロードするための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、指定した OSS オブジェクトをローカルファイルにダウンロードします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *params;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    params = aos_table_make(pool, 0);
    /* Download the object. A new local file is created if the specified file does not exist. If
the specified local file exists, it is overwritten. */
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers,
params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("Get object from file succeeded\n");
    } else {
        printf("Get object from file failed\n");
    }
```

```
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

注：

1.0.0 バージョンと比較して、"params" パラメーターが oss_get_object_to_file インターフェイスに追加され、"headers" および" params" パラメーターの値は "NULL" になることがあります。

7.6.3 ローカルメモリへのダウンロード

オブジェクトダウンロードの完全なコードについては、『GitHub』をご参照ください。

次のコードを実行し、指定したオブジェクトをローカルメモリにダウンロードします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
```

```
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *params = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    char *buf = NULL;
    int64_t len = 0;
    int64_t size = 0;
    int64_t pos = 0;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    /* Download an object to local memory. */
    resp_status = oss_get_object_to_buffer(oss_client_options, &bucket, &object,
                                headers, params, &buffer, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to buffer succeeded\n");
        /* Copy the downloaded content to the buffer. */
        len = aos_buf_list_len(&buffer);
        buf = aos_pcalloc(pool, len + 1);
        buf[len] = '\0';
        aos_list_for_each_entry(aos_buf_t, content, &buffer, node) {
        size = aos_buf_size(content);
            memcpy(buf + pos, content->pos, size);
            pos += size;
        }
    }
    else {
        printf("get object to buffer failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.6.4 部分ダウンロード

オブジェクトのデータの一部だけが必要な場合は、部分ダウンロードを使用して指定したコンテンツをダウンロードします。

部分ダウンロードは次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
```

```
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *params = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    char *buf = NULL;
    int64_t len = 0;
    int64_t size = 0;
    int64_t pos = 0;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    headers = aos_table_make(pool, 1);
    /* Set the download range, which ranges from the 20th byte to the 100th byte. */
    apr_table_set(headers, "Range", "bytes=20-100");
    /* Download the object to memory. */
    resp_status = oss_get_object_to_buffer(oss_client_options, &bucket, &object, headers,
params, &buffer, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to buffer succeeded\n");
        /* Obtain the buffer length. */
        aos_list_for_each_entry(aos_buf_t, content, &buffer, node) {
            len += aos_buf_size(content);
        }
        buf = aos_pcalloc(pool, (apr_size_t)(len + 1));
        buf[len] = '\0';
        /* Copy the content in the buffer to memory. */
        aos_list_for_each_entry(aos_buf_t, content, &buffer, node) {
            size = aos_buf_size(content);
            memcpy(buf + pos, content->pos, (size_t)size);
            pos += size;
        }
    }
```

```
    else {
        printf("get object to buffer failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.6.5 再開可能ダウンロード

大規模なオブジェクトは、不安定なネットワークまたはその他の例外のためにダウンロードに失敗する可能性があり、オブジェクト全体を再度ダウンロードする必要があります。 ただし、オブジェクトは複数回試行してもダウンロードに失敗することがあります。 そのため、OSS は再開可能ダウンロードを提供します。

再開可能ダウンロードを有効にするには、ダウンロードするオブジェクトを複数のパーツに分割して、別々にダウンロードする必要があります。 すべての部分をダウンロードした後に、各パーツは完全オブジェクトに結合されます。

再開可能ダウンロードを有効にするには、oss_resumable_clt_params_t を使用します。 oss_resumable_clt_params_t には、以下のパラメーターが含まれています。

パラメーター	説明
thread_num	同時スレッド数。既定では 1 です。
enable_checkpoint	再開可能ダウンロードを有効にするかどうかを決定します。既定では無効になっています。
partSize	パーツのサイズを指定します。 値は 1 バイトから 5 GB までで、単位はバイトです。
checkpoint_path	チェックポイントファイルが保存されているパスです。既定では {upload_file_path} .cp の下にあります。

このチェックポイントファイルには、ダウンロードの進行状況に関する情報が格納されます。 パーツをダウンロードできなかった場合、ダウンロードはチェックポイントファイルに記録された進行状況に基づいて続行されます。 オブジェクトがダウンロードされると、チェックポイントファイルは削除されます。

再開可能ダウンロードは、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
```

```
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_resumable_clt_params_t *clt_params;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Download the object using resumable download. */
    clt_params = oss_create_resumable_clt_params_content(pool, 1024 * 100, 3, AOS_TRUE
, NULL);
    resp_status = oss_resumable_download_file(oss_client_options, &bucket, &object, &file
, headers, NULL, clt_params, NULL, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("download succeeded\n");
    } else {
        printf("download failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

7.6.6 ダウンロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を表示することができます。

ダウンロードの進捗プログレスバーの完全なコードについては、『GitHub』をご参照ください。

次のコードは、ダウンロードプログレスバーの使用方法を説明するための例として使用されています。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
const char *download_filename = "<yourDownloadFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
void percentage(int64_t consumed_bytes, int64_t total_bytes)
{
    assert(total_bytes >= consumed_bytes);
    printf("%%%" APR_INT64_T_FMT "\n", consumed_bytes * 100 / total_bytes);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
```

```
    aos_string_t object;
    aos_string_t file;
    aos_table_t *resp_headers = NULL;
    aos_list_t resp_body;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Upload with a progress bar displayed. */
    resp_status = oss_do_put_object_from_file(oss_client_options, &bucket, &object, &file,
NULL, NULL, percentage, &resp_headers, &resp_body);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
    } else {
        printf("put object from file failed\n");
    }
    /* Download with a progress bar displayed. */
    aos_pool_create(&pool, NULL);
    oss_client_options = oss_request_options_create(pool);
    init_options(oss_client_options);
    aos_str_set(&file, download_filename);
    resp_status = oss_do_get_object_to_file(oss_client_options, &bucket, &object, NULL,
NULL, &file, percentage, NULL);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

oss_get_object_from_file と oss_get_object_from_buffer はプログレスバーをサポートしません。

7.7 オブジェクトの管理

7.7.1 概要

API を使用して、バケット内のオブジェクトに対して次の操作を実行します。

- オブジェクトのACLの管理
- オブジェクトメタの管理
- オブジェクトの列挙
- オブジェクトのコピー
- オブジェクトの削除
- アーカイブオブジェクトの復元
- シンボリックリンクの管理

7.7.2 オブジェクトのACLの管理

次の表は、オブジェクトのアクセス制御リスト (ACL) に含まれるアクセス許可について説明しています。

アクセス許可	説明	値
default	オブジェクトの ACL は、そのバケットの ACL と同一です。	OSS_ACL_DEFAULT
Private	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。	OSS_ACL_PRIVATE
Public read	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。 他のユーザーはオブジェクトを読むことしかできません。 この許可は慎重に承認してください。	OSS_ACL_PUBLIC_READ
Public read-write	すべてのユーザーがオブジェクトを読み書きできます。 この許可は慎重に承認してください。	OSS_ACL_PUBLIC_READ_WRITE

オブジェクトの ACL はバケットの ACL に優先されます。 たとえば、バケットの ACL が "private" で、オブジェクトの ACL が "public read-write"である場合、すべてのユーザーがそのオブジェクトを読み書きできます。 オブジェクトが ACL で設定されていない場合、その ACL は既定ではそのバケットの ACL と同一です。

オブジェクトの ACL の設定

次のコードを実行して、オブジェクトの ACL を設定をします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
```

```
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    oss_acl_e oss_acl = OSS_ACL_PUBLIC_READ;
    /* Configure an ACL for an object. */
    resp_status = oss_put_object_acl(oss_client_options, &bucket, &object, oss_acl, &
resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put object acl success! \n");
    } else {
        printf("put object acl failed! \n");
    }
    /* Obtain the ACL for an object. */
    aos_string_t oss_acl_string;
    resp_status = oss_get_object_acl(oss_client_options, &bucket, &object, &oss_acl_string
, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object acl success! \n");
        printf("acl: %s \n", oss_acl_string.data);
    } else {
        printf("get object acl failed! \n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

7.7.3 オブジェクトメタの管理

オブジェクトメタには、HTTP ヘッダーとユーザー定義のオブジェクトメタが含まれます。 詳しくは、「OSS 開発者ガイド」の「オブジェクトメタ」をご参照ください。

オブジェクトメタの取得

OSS にファイルをアップロードした後、または OSS からオブジェクトを読み取る前に、`oss_get_object_meta` インターフェイスを使用して、長さやタイプなど、オブジェクトに関するメタ情報を取得します。 次のコードを実行して、オブジェクトメタを設定および取得します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is not
used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The implementa
tion code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the new
pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *headers;
    aos_list_t buffer;
    aos_table_t *resp_headers = NULL;
```

```
    aos_status_t *resp_status = NULL;
    aos_buf_t *content = NULL;
    char *content_length_str = NULL;
    char *object_type = NULL;
    char *object_author = NULL;
    int64_t content_length = 0;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers = aos_table_make(pool, 2);
    // Configure custom Object Meta. */
    apr_table_set(headers, "Expires", "Fri, 28 Feb 2032 05:38:42 GMT");
    apr_table_set(headers, "x-oss-meta-author", "oss");
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_con
tent));
    aos_list_add_tail(&content->node, &buffer);
    /* Set object permissions and upload the object from the cache. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object,
            &buffer, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from buffer with md5 succeeded\n");
    } else {
        printf("put object from buffer with md5 failed\n");
    }
    /* Obtain the object permission. */
    resp_status = oss_get_object_meta(oss_client_options, &bucket, &object, &resp_heade
rs);
    if (aos_status_is_ok(resp_status)) {
        content_length_str = (char*)apr_table_get(resp_headers, OSS_CONTENT_LENGTH);
        if (content_length_str ! = NULL) {
            content_length = atol(content_length_str);
        }
        object_author = (char*)apr_table_get(resp_headers, OSS_AUTHORIZATION);
        object_type = (char*)apr_table_get(resp_headers, OSS_OBJECT_TYPE);
        printf("get object meta succeeded, object author:%s, object type:%s, content_length
:%ld\n", object_author, object_type, content_length);
    } else {
        printf("req:%s, get object meta failed\n", resp_status->req_id);
    }
    /* Release the memory pool, that is, the memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

注：

HTTP ヘッダーの詳細については、「RFC2616」をご参照ください。

7.7.4 オブジェクトの一覧表示

オブジェクトを一覧表示するための完全なコードについては、『GitHub』をご参照ください。

OSS オブジェクトはアルファベット順に一覧表示されています。 `oss_list_object` を使用すると、バケット内のすべてのオブジェクトを一覧表示できます。 次の表は、オブジェクトの一覧表示に使用される一般的なパラメーターについて説明しています。

パラメーター	説明
delimiter	オブジェクトをグループ化するために使用される区切り記号を指定します。 名前に指定されたプレフィックスが含まれ、その後に区切り記号が最初に現れるオブジェクトはすべて、要素のグループ、つまり commonPrefixes として機能します。
prefix	一覧表示されたオブジェクトが含まなければならないプレフィックスを指定します。
max_ret	一覧表示できるオブジェクトの最大数を指定します。 既定値は 1000 です。
marker	一覧表示されている初期オブジェクトを指定します。

すべてのオブジェクトの一覧表示

現在のバケット内のすべてのオブジェクトを一覧表示するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
```

```
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *prefix = "";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (! aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list,
node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data
,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->
next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

注：

既定では、バケットに 1,000 個を超えるオブジェクトが含まれる場合、最初の 1,000 個のオブジェクトだけが返され、返される結果の切り捨てられたパラメーターは "true" です。 さらに、"next_marker" が返されて、次の読み込みの始点として機能します。 返されるオブジェクトの数を調整するには、"max_ret" パラメーターを変更するか、または複数の読み込みに対して "marker" パラメーターを使用します。

指定された数のオブジェクトの一覧表示

次のコードを実行し、指定した数のオブジェクトを一覧表示します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* aos_str_set uses a char* string to initialize the aos_string_t type.*/
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used.*/
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.*/
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories.*/
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Memories for options are allocated by the memory pool and are released after the
memory pool is released. Therefore, you do not need to release the memory for options
individually. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *prefix = "";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    /* Set the maximum number of listed objects to 200. */
    params->max_ret = 200;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects */
    resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
```

```
    if (! aos_status_is_ok(resp_status))
    {
        printf("list object failed\n");
    }
    aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node
) {
        ++size;
        line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
            content->size.len, content->size.data,
            content->last_modified.len, content->last_modified.data);
        printf("%s", line);
    }
    printf("Total %d\n", size);
    /* Release the memory pool after a request is executed, which means that memories
allocated for all resources during the request are released. */
    aos_pool_destroy(pool);
    /* Before the project ends, call the aos_http_io_deinitialize method to release allocated
 global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

指定されたプレフィックスを持つオブジェクトの一覧表示

指定されたプレフィックスを持つオブジェクトを一覧表示するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* aos_str_set uses a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources,
including networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */ */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
```

```
    /* Memories for options are allocated by the memory pool and are released after the
memory pool is released. Therefore, you do not need to release the memory for options
individually. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    /* List objects with specified prefix. */
    char *prefix = "<yourObjectPefix>";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (! aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list,
node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data
,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->
next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after a request is executed, which means that memories
allocated for all resources during the request are released. */
    aos_pool_destroy(pool);
    /* Before the project ends, call the aos_http_io_deinitialize method to release allocated
 global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

一覧表示されている初期オブジェクトの指定

"marker" の値は初期オブジェクトの名前です。 アルファベット順のマーカーの後のオブジェクトを一覧表示するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
```

```
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* aos_str_set uses a char* string to initialize the aos_string_t type.*/
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used.*/
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.*/
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 network and memories.*/
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Memories for options are allocated by the memory pool and are released after the
memory pool is released. Therefore, you do not need to release the memory for options
individually. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *prefix = "";
    Specifies the initial object that is listed. */
    char *nextMarker = "<yourObjectMarker>";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (! aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list,
node) {
```

```
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data
,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->
next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after a request is executed, which means that memories
allocated for all resources during the request are released. */
    aos_pool_destroy(pool);
    /* Before the project ends, call the aos_http_io_deinitialize method to release allocated
 global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.7.5 オブジェクトの削除

警告：

削除されたオブジェクトは回復できないため、オブジェクトは慎重に削除してください。

オブジェクト削除の完全なコードについては、『GitHub』をご参照ください。

オブジェクトの削除

単一オブジェクトを削除するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
```

```
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign char* data to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Delete an object. */
    resp_status = oss_delete_object(oss_client_options, &bucket, &object, &resp_headers);
    /* Determine whether the object is deleted successfully. */
    if (aos_status_is_ok(resp_status)) {
        printf("delete object succeed\n");
    } else {
        printf("delete object failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

複数オブジェクトの削除

複数オブジェクトを同時に削除するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name1 = "<yourObjectName1>";
const char *object_name2 = "<yourObjectName2>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
```

```
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object1;
    aos_string_t object2;
    int is_quiet = 1;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object1, object_name1);
    aos_str_set(&object2, object_name2);
    /* Construct the list of objects to be deleted. */
    aos_list_t object_list;
    aos_list_t deleted_object_list;
    oss_object_key_t *content1;
    oss_object_key_t *content2;
    aos_list_init(&object_list);
    aos_list_init(&deleted_object_list);
    content1 = oss_create_oss_object_key(pool);
    aos_str_set(&content1->key, object_name1);
    aos_list_add_tail(&content1->node, &object_list);
    content2 = oss_create_oss_object_key(pool);
    aos_str_set(&content2->key, object_name2);
    aos_list_add_tail(&content2->node, &object_list);
    /* Delete objects in the list. `is_quiet` is used to determine whether the result of
deletion is returned. */
    resp_status = oss_delete_objects(oss_client_options, &bucket, &object_list, is_quiet, &
resp_headers, &deleted_object_list);
    if (aos_status_is_ok(resp_status)) {
        printf("delete objects succeeded\n");
    } else {
        printf("delete objects failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

指定されたプレフィックスを持つ複数オブジェクトの削除

指定したプレフィックスを持つ複数オブジェクトを削除することにより、ディレクトリを削除します。 たとえば、プレフィックス "dir" を持つオブジェクトを削除することで、**"dir"** ディレクトリを削除します。 次のコードを実行して、指定したプレフィックスを持つ複数オブジェクトを同時に削除します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_prefix = "<yourObjectNamePrefix>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t prefix;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&prefix, object_prefix);
    /* Delete objects with a specified prefix. */
    resp_status = oss_delete_objects_by_prefix(oss_client_options, &bucket, &prefix);
    if (aos_status_is_ok(resp_status)) {
        printf("delete objects by prefix succeeded\n");
    } else {
        printf("delete objects by prefix failed\n");
```

```
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

警告：

プレフィックスが空の文字列または "null" の場合、バケット内のすべてのオブジェクトが削除されます。 慎重に操作を実行してください。

7.7.6 オブジェクトのコピー

小さいサイズのオブジェクトのコピー

小さいサイズのオブジェクトをコピーする前に、次の制限事項に注意する必要があります。

- ユーザーはソースオブジェクトに対する読み取りおよび書き込み権限を持っている必要があります。
- あるリージョンから別のリージョンにオブジェクトをコピーすることはできません。 たとえば、中国 (杭州) リージョンのバケットから中国 (青島) リージョンのバケットにオブジェクトをコピーすることはできません。
- オブジェクトサイズは 1 GB より小さくなければなりません。

小さいサイズのオブジェクトをコピーするには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *source_bucket_name = "<yourSourceBucketName>";
const char *source_object_name = "<yourSourceObjectName>";
const char *dest_bucket_name = "<yourDestBucketName>";
const char *dest_object_name = "<yourDestObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
```

```
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t source_bucket;
    aos_string_t source_object;
    aos_string_t dest_bucket;
    aos_string_t dest_object;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&source_bucket, source_bucket_name);
    aos_str_set(&source_object, source_object_name);
    aos_str_set(&dest_bucket, dest_bucket_name);
    aos_str_set(&dest_object, dest_object_name);
    headers = aos_table_make(pool, 0);
    /* Copy an object. */
    resp_status = oss_copy_object(oss_client_options, &source_bucket, &source_object, &
dest_bucket, &dest_object, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("copy object succeeded\n");
    } else {
        printf("copy object failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

大きいサイズのオブジェクトのコピー

1 GB を超えるオブジェクトの場合、マルチパートコピーを実行するには **oss_upload_part_copy** を使用することを推奨します (UploadPartCopy)。 マルチパートコピーを実行するには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
#include <sys/stat.h>
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *source_bucket_name = "<yourSourceBucketName>";
const char *source_object_name = "<yourSourceObjectName>";
```

```
const char *dest_bucket_name = "<yourDestBucketName>";
const char *dest_object_name = "<yourDestObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int64_t get_file_size(const char *file_path)
{
    int64_t filesize = -1;
    struct stat statbuff;
    if(stat(file_path, &statbuff) < 0){
        return filesize;
    } else {
        filesize = statbuff.st_size;
    }
    return filesize;
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t dest_bucket;
    aos_string_t dest_object;
    aos_string_t upload_id;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_list_upload_part_params_t *list_upload_part_params;
    oss_upload_part_copy_params_t *upload_part_copy_params1;
    oss_upload_part_copy_params_t *upload_part_copy_params2;
    oss_list_part_content_t *part_content;
    aos_list_t complete_part_list;
    oss_complete_part_content_t *complete_content;
    aos_table_t *list_part_resp_headers = NULL;
    aos_table_t *complete_resp_headers = NULL;
    int part1 = 1;
    int part2 = 2;
    int64_t range_start1 = 0;
```

```
    int64_t range_end1 = 6000000;//not less than 5MB
    int64_t range_start2 = 6000001;
    int64_t range_end2;
    int max_ret = 1000;
    headers = aos_table_make(pool, 0);
    aos_str_set(&dest_bucket, dest_bucket_name);
    aos_str_set(&dest_object, dest_object_name);
    resp_status = oss_init_multipart_upload(oss_client_options, &dest_bucket, &dest_objec
t, &upload_id, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("init multipart upload succeeded, upload_id is %s\n", upload_id.data);
    } else {
        printf("init multipart upload failed\n");
    }
    /* Copy the first part of the object. */
    upload_part_copy_params1 = oss_create_upload_part_copy_params(pool);
    aos_str_set(&upload_part_copy_params1->source_bucket, source_bucket_name);
    aos_str_set(&upload_part_copy_params1->source_object, source_object_name);
    aos_str_set(&upload_part_copy_params1->dest_bucket, dest_bucket_name);
    aos_str_set(&upload_part_copy_params1->dest_object, dest_object_name);
    aos_str_set(&upload_part_copy_params1->upload_id, upload_id.data);
    upload_part_copy_params1->part_num = part1;
    upload_part_copy_params1->range_start = range_start1;
    upload_part_copy_params1->range_end = range_end1;
    headers = aos_table_make(pool, 0);
    resp_status = oss_upload_part_copy(oss_client_options, upload_part_copy_params1,
headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("upload part 1 copy succeeded\n");
    } else {
        printf("upload part 1 copy failed\n");
    }
    /* Copy the second part of the object. */
    range_end2 = get_file_size(local_filename) - 1;
    upload_part_copy_params2 = oss_create_upload_part_copy_params(pool);
    aos_str_set(&upload_part_copy_params2->source_bucket, source_bucket_name);
    aos_str_set(&upload_part_copy_params2->source_object, source_object_name);
    aos_str_set(&upload_part_copy_params2->dest_bucket, dest_bucket_name);
    aos_str_set(&upload_part_copy_params2->dest_object, dest_object_name);
    aos_str_set(&upload_part_copy_params2->upload_id, upload_id.data);
    upload_part_copy_params2->part_num = part2;
    upload_part_copy_params2->range_start = range_start2;
    upload_part_copy_params2->range_end = range_end2;
    headers = aos_table_make(pool, 0);
    resp_status = oss_upload_part_copy(oss_client_options, upload_part_copy_params2,
headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("upload part 2 copy succeeded\n");
    } else {
        printf("upload part 2 copy failed\n");
    }
    /* Lists the parts. */
    headers = aos_table_make(pool, 0);
    list_upload_part_params = oss_create_list_upload_part_params(pool);
    list_upload_part_params->max_ret = max_ret;
    aos_list_init(&complete_part_list);
    resp_status = oss_list_upload_part(oss_client_options, &dest_bucket, &dest_object, &
upload_id, list_upload_part_params, &list_part_resp_headers);
    aos_list_for_each_entry(oss_list_part_content_t, part_content, &list_upload_part_par
ams->part_list, node) {
        complete_content = oss_create_complete_part_content(pool);
        aos_str_set(&complete_content->part_number, part_content->part_number.data);
        aos_str_set(&complete_content->etag, part_content->etag.data);
        aos_list_add_tail(&complete_content->node, &complete_part_list);
```

```
    }
    /* Complete the multipart copy. */
    resp_status = oss_complete_multipart_upload(oss_client_options, &dest_bucket, &
dest_object, &upload_id, &complete_part_list, headers, &complete_resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("complete multipart upload succeeded\n");
    } else {
        printf("complete multipart upload failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.7.7 アーカイブオブジェクトの復元

アーカイブオブジェクトを読み込む前にアーカイブオブジェクトを復元する必要があります。

アーカイブオブジェクトではないオブジェクトに対しては、restoreObject を呼び出さないでください。

アーカイブオブジェクトの状態変換プロセスは次のとおりです。

- アーカイブオブジェクトはフリーズ状態です。
- 復元のために送信した後、サーバーはオブジェクトを復元します。 オブジェクトは復元中の状態です。
- 復元後にオブジェクトを読むことができます。 オブジェクトの復元された状態は、既定では 1 日続きます。 この期間は最大 7 日間まで延長可能です。 この期間が終了すると、オブジェクトはフリーズ状態に戻ります。

次のコードを実行してオブジェクトを復元します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
```

```
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    oss_acl_e oss_acl = OSS_ACL_PRIVATE;
    aos_buf_t *content = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers = aos_table_make(pool, 0);
    /* Create a bucket of the archive storage class. */
    resp_status = oss_create_bucket_with_storage_class(oss_client_options, &bucket,
oss_acl, OSS_STORAGE_CLASS_ARCHIVE, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_con
tent));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload a file. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer
, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from buffer succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Restore the object. */
    do {
        headers = aos_table_make(pool, 0);
        resp_status = oss_restore_object(oss_client_options, &bucket, &object, headers, &
resp_headers);
        printf("restore object resp_status->code: %d \n", resp_status->code);
        if (resp_status->code ! = 409) {
            break;
        } else {
            printf("restore object is already in progress, resp_status->code: %d \n", resp_statu
s->code);
            apr_sleep(5000);
        }
```

```
    } while (1);
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

アーカイブストレージクラスの詳細については、「ストレージクラスの概要」をご参照ください。 関連するステータスコードの詳細については、「RestoreObject」をご参照ください。

7.7.8 シンボリックリンクの管理

シンボリックリンクの作成

シンボリックリンクは、Windows で使用されるショートカットに似た、オブジェクトにマップされる特殊なオブジェクトです。 シンボリックリンクに対してユーザー定義のオブジェクトメタを構成できます。

次のコードを実行してシンボリックリンクを作成します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *link_object_name = "<yourLinkObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
```

```
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t sym_object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&sym_object, link_object_name);
    resp_status = oss_put_symlink(oss_client_options, &bucket, &sym_object, &object, &
resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put symlink succeeded\n");
    } else {
        printf("put symlink failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

シンボリックリンクのオブジェクトコンテンツの取得

シンボリックリンクを取得するには、読み取り許可を持っていなければなりません。 次のコードを実行して、シンボリックリンクのオブジェクトコンテンツを取得します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *link_object_name = "<yourLinkObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
 networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
```

```
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t link_object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    char *target_object_name = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&link_object, link_object_name);
    resp_status = oss_get_symlink(oss_client_options, &bucket, &link_object, &resp_heade
rs);
    if (aos_status_is_ok(resp_status)) {
        printf("get symlink succeeded\n");
        target_object_name = (char*)(apr_table_get(resp_headers, OSS_CANNONICALIZED_H
EADER_SYMLINK));
        printf("target_object_name: %s \n", target_object_name);
    } else {
        printf("get symlink failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.8 許可されたアクセス

一時的アクセス許可に STS アクセスを使用

OSS は一時的アクセスの許可のため、Alibaba Cloud Security Token Service (STS) をサポートしています。 STS は、クラウドコンピューティングユーザーに一時的アクセストークンを提供する Web サービスです。 STS を通して、サードパーティー製アプリケーションまたは RAM ユーザー (ユーザー ID を管理できます) に、カスタムの有効期間とアクセス許可を持つアクセス資格情報を割り当てることができます。 STS の詳細については、「STSの概要」をご参照ください。

STS の利点:

- ユーザーの長期キー (AccessKey) は第三者のアプリケーションには公開されません。 アクセストークンを生成し、アクセストークンをサードパーティー製アプリケーションに送信する必要があるだけです。 アクセス許可とトークンの有効性をカスタマイズできます。
- 権限を取り消す必要はありません。 有効期限が切れると、アクセストークンは自動的に無効になります。

STS による OSS へのアクセスプロセスの詳細については、「OSS 開発者ガイド」の「RAM および STS のシナリオの実例」をご参照ください。

STS を使った署名リクエストの作成

次のコードを実行して STS で署名リクエストを作成します。

```
#include "oss_api.h"
#include "aos_http_io.h"

const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *sts_token = "<yourStsToken>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";

void init_options(oss_request_options_t *options)
{
 options->config = oss_config_create(options->pool);
 /* Use a char* string to initialize the aos_string_t type. */
 aos_str_set(&options->config->endpoint, endpoint);
 aos_str_set(&options->config->access_key_id, access_key_id);
 aos_str_set(&options->config->access_key_secret, access_key_secret);
 aos_str_set(&options->config->sts_token, sts_token);
 /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
 options->config->is_cname = 0;
 /* Used to configure network parameters, such as timeout. */
 options->ctl = aos_http_controller_create(options->pool, 0);
}

int main(int argc, char *argv[])
{
 /* Call the aos_http_io_initialize method in main() to initialize global resources, such as
networks and memories. */
 if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
  exit(1);
 }

 /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
 aos_pool_t *pool;
 /* Re-create a new memory pool. The second parameter is NULL, indicating that it does
not inherit from any other memory pools. */
 aos_pool_create(&pool, NULL);
 /* Create and initialize options. This parameter mainly includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl. */
 oss_request_options_t *oss_client_options;
 /* Allocate memories in the memory pool to options. */
 oss_client_options = oss_request_options_create(pool);
 /* Use oss_client_options to initialize client options. */
 init_options(oss_client_options);

   /*  Initialization parameters. */
 aos_string_t bucket;
 aos_string_t object;
 aos_list_t buffer;
 aos_buf_t *content = NULL;
   aos_table_t *headers = NULL;
 aos_table_t *resp_headers = NULL;
```

```
 aos_status_t *resp_status = NULL;
 /* Assign the char* data to the bucket. */
 aos_str_set(&bucket, bucket_name);
   aos_str_set(&object, object_name);

   aos_list_init(&buffer);
   content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_con
tent));
   aos_list_add_tail(&content->node, &buffer);

 /* Upload an object. */
   resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer
, headers, &resp_headers);
    /* Determine whether the upload is successful. */
   if (aos_status_is_ok(resp_status)) {
      printf("put object from buffer succeeded\n");
   } else {
      printf("put object from buffer failed\n");
   }

 /* Release the memory pool, that is, memories allocated to resources during the request
. */
 aos_pool_destroy(pool);

 /* Release allocated global resources. */
 aos_http_io_deinitialize();

 return 0;
}
```

一時的アクセスを許可するための URL への署名

一時的アクセスができるよう、訪問者に署名された URL を提供することができます。 URL に署名する際、訪問者のアクセス期間を制限するために URL の有効期限を指定できます。

- 署名付き URL を使用したオブジェクトのアップロード

  署名付き URL を持つオブジェクトをアップロードするには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
```

```
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such
 as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
 information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl.
*/
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_http_request_t *req;
    apr_time_t now;
    char *url_str;
    aos_string_t url;
    int64_t expire_time;
    int one_hour = 3600;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    expire_time = now / 1000000 + one_hour;
    headers = aos_table_make(pool, 0);
    req = aos_http_request_create(pool);
    req->method = HTTP_PUT;
    now = apr_time_now();  /* Unit: microseconds */
    expire_time = now / 1000000 + one_hour;
    /* Generate the signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &object, expire_time, req);
    aos_str_set(&url, url_str);
    printf("Temporary upload URL: %s\n", url_str);
    /* Use the signed URL to upload an object. */
    resp_status = oss_put_object_from_file_by_url(oss_client_options, &url, &file,
headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put objects by signed url succeeded\n");
    } else {
        printf("put objects by signed url failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

- 署名付き URL を使用したオブジェクトのダウンロード

  署名付き URL を持つオブジェクトをアップロードするには、次のコードを実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such
 as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL, indicating that it
does not inherit from any other memory pools. */
    aos_pool_create(&p, NULL);
    /* Create and initialize options. This parameter mainly includes global configuration
 information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl.
 */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /*  Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_http_request_t *req;
    apr_time_t now;
    char *url_str;
    aos_string_t url;
    int64_t expire_time;
    int one_hour = 3600;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
```

```
    aos_str_set(&file, local_filename);
    expire_time = now / 1000000 + one_hour;
    headers = aos_table_make(pool, 0);
    params = aos_table_make(pool, 0);
    req = aos_http_request_create(pool);
    req->method = HTTP_GET;
    now = apr_time_now();  /* Unit: microseconds */
    expire_time = now / 1000000 + one_hour;
    /* Generate the signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &object, expire_time, req);
    aos_str_set(&url, url_str);
    /* Use the signed URL to download an object. */
    resp_status = oss_get_object_to_file_by_url(oss_client_options, &url, headers,
params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object succeeded\n");
    } else {
        printf("get object failed\n");
    }
    /* Release the memory pool, that is, memories allocated to resources during the
request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

7.9 画像処理

Image Processing (IMG)は、大規模で安全、低コストで、信頼性の高い画像処理サービスです。ソース画像が OSS にアップロードされると、簡単な RESTful API を通して、いつでも、どこからでも、どのインターネットデバイスからでも、画像を処理できます。

IMG の詳細については、「Image Processing」をご参照ください。

IMG の完全なコードについては『GitHub』をご参照ください。

基本機能

OSS は以下の IMG 機能を提供します。

- ドミナントトーンの取得
- 画像形式の変換
- 画像イメージのサイズ変更
- 内接円の調整
- 向きの調整
- 画像輝度
- 透かしの追加: 画像、テキスト、または画像テキストの透かしを、別の画像に追加します。
- 画像処理
- 画像処理アクセスルール: 複数の IMG 機能を呼び出します。

使用方法

IMG は標準の HTTP GET を使用します。 URL のクエリ文字列 に IMG パラメーターを設定します。

イメージオブジェクトの ACL が "非公開読み書き" の場合、許可されたユーザーだけがアクセスできます。

- 匿名アクセス

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
  ```

パラメーター	説明
bucket	バケットの名前を指定します。
endpoint	リージョンへのアクセスに使用されるエンドポイントを指定します。
object	画像オブジェクトの名前を指定します。
image	IMG の 予約済み識別子を指定します。
action	拡大縮小、トリミング、回転など、画像に対する操作を指定します。

パラメーター	説明
param	画像に対する操作を示すパラメーターを指定します。

- 基本操作

  たとえば、画像を 100 ピクセルの幅に拡大縮小します。 比率に基づいて高さを調整します。

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- カスタマイズされた画像スタイル

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  ■ style: カスタマイズされた画像スタイルの予約済み識別子を指定します。

  ■ yourStyleName: カスタム画像スタイルの名前を指定します。 OSS コンソールで作成されたルールで指定された名前です。

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- カスケード処理

  カスケード処理では、画像に対して複数の処理を順番に実行します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- HTTPS アクセス

  IMG は HTTPS を通したアクセスをサポートしています。 例:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

• 許可されたアクセス

許可されたアクセスにより、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

次のコードを使用して IMG の署名付き URL を生成します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is
not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources, such
 as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the new
pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl.
 */
    oss_request_options_t *oss_client_options;
    /* Allocate the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *params = NULL;
    aos_http_request_t *req;
    char *url_str;
    apr_time_t now;
    int64_t expire_time;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Start image processing. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100,h_100");
    req = aos_http_request_create(pool);
    req->method = HTTP_GET;
```

```
    req->query_params = params;
    /* Specify the expiration time (expire_time) in seconds. */
    now = apr_time_now();
    expire_time = now / 1000000 + 10 * 60;
    /* Generate a signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &object, expire_time, req);
    printf("url: %s\n", url_str);
    /* Release the memory pool, that is, the memories allocated to resources during the
 request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- SDK を使用したアクセス

  SDK を使用して、画像オブジェクトにアクセスして処理します。

  SDK を使用すると、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

  - 基本操作

    次のコードを実行して、画像に対して基本操作を実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is
 not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources,
such as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the
new pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl
. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Scale the image.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100,h_100");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-resize.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Crop the image.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/crop,w_100,h_100,x_100,y_100,r_1
");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-crop.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Rotate the image.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/rotate,90");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-rotate.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Sharpen the image.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/sharpen,100");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-sharpen.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
```

```
    } else {
        printf("get object to file failed\n");
    }
    /* Add watermarks.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/watermark,text_SGVsbG8g5Zu-
54mH5pyN5YqhIQ");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-watermark.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Convert the image format.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/format,png");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-format.png");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Obtain image information.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/info");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-info.txt");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, that is, the memories allocated to resources during
the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- 画像スタイルのカスタマイズ

  次のコードを実行して画像スタイルをカスタマイズします。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
```

```
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is
 not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources,
such as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the
new pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl
. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Customize the image style. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "style/<yourCustomStyleName>");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-custom-style.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, that is, the memories allocated to resources during
the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

- カスケード処理

次のコードを実行して、画像に対してカスケード処理を実行します。

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Determine whether the CNAME is used. The value 0 indicates that the CNAME is
 not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources,
such as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t. The
implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL, indicating that the
new pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration
information, such as endpoint, access_key_id, acces_key_secret, is_cname, and curl
. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Perform cascade operations. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100,h_100");
    /* Download processed images to a local file. */
    aos_str_set(&file, "example-cascade.jpg");
```

```
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object,
headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, that is, the memories allocated to resources during
the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

IMG ツール

IMG ビューア ImageStyleViever を使用して、IMG 結果を直接表示できます。

7.10 エラー処理

例外処理

OSS C SDK の使用中にリクエストエラーが発生した場合、対応するエラーメッセージが aos_status_s に出力されます。 aos_status_s の属性は次のとおりです。

エラーコード	説明	タイプ
code	エラーリクエストの HTTP ステータスコード	整数
error_code	OSS により返されたエラーコード	文字列
error_msg	OSS により返されたエラーメッセージ	文字列
req_id	リクエストを識別する UUID OSS 開発エンジニアに req_id を提供することでヘルプを求めることができます。	文字列

タイムアウト処理

返された aos_status_t のコードが "2XX" ではなく、error_code が "-992" または "-995" の場合は、接続タイムアウトエラーまたはリクエストタイムアウトエラーが発生したことを示します。もう一度試すことができます。

aos_status.h の aos_should_retry(aos_status_t *) を使用して、返されたエラーコードが再試行を必要とするかどうかを判断できます。 戻り値が 1 の場合は、再試行が必要です。

一般的なエラーコードは次のとおりです。

エラーコード	説明	**HTTP** ステータスコード
AccessDenied	アクセスが拒否されました。	403
BucketAlreadyExists	バケットが既に存在します。	409
BucketNotEmpty	バケットは空ではありません。	409
EntityTooLarge	エンティティサイズが上限を超えています。	400
EntityTooSmall	エンティティサイズが下限を下回っています。	400
FileGroupTooLarge	オブジェクトグループの合計サイズが上限を超えています。	400
FilePartNotExist	オブジェクトパートが存在しません。	400
FilePartStale	オブジェクトパートが期限切れです。	400
InvalidArgument	パラメーターの形式が無効です。	400
InvalidAccessKeyId	AccessKeyID が存在しません。	403
InvalidBucketName	バケット名が無効です。	400
InvalidDigest	ダイジェストが無効です。	400
InvalidObjectName	オブジェクト名が無効です。	400
InvalidPart	そのパーツは無効です。	400
InvalidPartOrder	パーツ順番が無効です。	400
InvalidTargetBucketForLogging	ログファイルを格納するバケットが無効です。	400
InternalError	内部 OSS エラーです。	500
MalformedXML	XML 形式が無効です。	400
MethodNotAllowed	メソッドがサポートされていません。	405
MissingArgument	パラメータが設定されていません。	411

エラーコード	説明	HTTP ステータスコード
MissingContentLength	コンテンツの長さが設定されていません。	411
NoSuchBucket	バケットが存在しません。	404
NoSuchKey	オブジェクトが存在しません。	404
NoSuchUpload	マルチパートアップロード ID が存在しません。	404
NotImplemented	メソッドを処理できません。	501
PreconditionFailed	前処理エラーです。	412
RequestTimeTooSkewed	OSSClient に設定されている現地時間が、OSS サーバーに設定されている時間から 15 分以上ずれています。	403
RequestTimeout	リクエストのタイムアウトです。	400
SignatureDoesNotMatch	署名が不一致です。	403
TooManyBuckets	バケット数が上限を超えています。	400

7.11 FAQ

このトピックでは、OSS C SDK に関する FAQ を紹介します。

- OSS C SDK でアプリケーションログの出力を設定する方法
- OSS C SDK で cURL ベースの通信用のパラメーターを設定する方法
- OSS C SDK の cURL のコールバックを使用してデータをアップロードする方法
- OSS C SDK の cURL のコールバックを使用してデータをアップロードする方法
- OSS C SDK の cURL のコールバックを使用してデータをダウンロードする方法
- Windows 用の OSS C SDK で中国語文字を含む名前のオブジェクトをアップロードおよびダウンロードする方法

8 . NET

8.1 インストール

このトピックでは、OSS C# SDK のインストール方法について説明します。

準備

- Windows
  - NET 2.0 以降に対応
  - Visual Studio 2010 以降に対応
- Linux Mac
  - Mono 3.12 以降に対応

SDK のダウンロード

- 直接ダウンロード
- GitHub からダウンロード
- 旧バージョンをダウンロード

SDK のインストール

- Windows に OSS C#SDK をインストール
  - NuGet を通して SDK をインストール
    1. Visual Studio に NuGet がインストールされていない場合は、まず NuGet をインストールします。
    2. プロジェクトを作成、またはVisual Studio で既存のプロジェクトを開くには、ツール > **NuGet** パッケージマネージャー > **NuGet** パッケージソリューション管理 をクリックします。
    3. aliyun.oss.sdk を検索し、検索結果から Aliyun.OSS.SDK を見つけます。 最新のバージョンを選択して、インストールをクリックします。
  - DLL リファレンスを通して SDK をインストール
    1. C#SDK パッケージをダウンロードして解凍します。
    2. Visual Studioで、ソリューションリソースマネージャー にアクセスし、プロジェクトを選択して、[プロジェクト名] を右クリックします。 リファレンス > 追加リファレンスをクリックし、プロンプトダイアログボックスで 参照 をクリックします。
    3. SDK パッケージが解凍されたディレクトリを探し、bin ディレクトリでAliyun.OSS.dllを選択して **OK** をクリックします。
  - プロジェクトイントロダクションを通して SDK をインストール

    GitHub から SDK パッケージまたはソースコードをダウンロードし、そのソースコードを使用して SDK パッケージをインストールする場合は、
    1. Visual Studio でソリューションを右クリックし、ダイアログボックスのメニューで 追加 をクリックして、既存のプロジェクトを追加します。
    2. プロンプトダイアログボックスで、aliyun-oss-sdk.csproj を選択して 開く をクリックします。
    3. ユーザーのプロジェクトを右クリックして、リファレンス > リファレンスの追加をクリックします。 プロンプトダイアログボックスで、プロジェクト タブをクリックし、aliyun-oss-sdk プロジェクトを選択して **OK** をクリックします。

- Windows に OSS C#SDK をインストール
  - NuGet を通して SDK をインストール
    1. Xamarinで、プロジェクトを作成するか既存のプロジェクトを開き、[ツール] から **NuGet** パッケージの追加 を選択します。
    2. aliyun.oss.sdk を検索し、最新のバージョンを選択して、パッケージの追加 をクリックし、パッケージをプロジェクトアプリケーションに追加します。
  - DLL リファレンスを通して SDK をインストール
    1. C#SDK パッケージをダウンロードして解凍します。
    2. Xamarin内の ソリューション でユーザーのプロジェクトを選択し、リファレンス を右クリックして、プロンプトメニューのリファレンスの編集をクリックします。
    3. [リファレンスの編集] ダイアログボックスで、 .Net #####を選択し、参照 をクリックします。 SDK が解凍されたディレクトリを探し、bin ディレクトリで Aliyun.OSS.dll を選択して、開く をクリックします。

8.2 クイックスタート

このトピックでは、OSS の使用方法について説明します。NET SDK は、バケットの作成、オブジェクトのアップロード、オブジェクトのダウンロードなどのルーチン操作を実行します。

バケットを作成

バケットは OSS のグローバルネームスペースです。 ファイルを格納するデータコンテナーのようなものです。

次のコードを実行してバケットを作成します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Create a bucket.
    var bucket = client.CreateBucket(bucketName);
    Console.WriteLine("Create bucket succeeded, {0} ", bucket.Name);
}
} catch (Exception e) {
{
    Console.WriteLine("Create bucket failed, {0}", ex.Message);
}
```

バケットの命名ルールの詳細については、「基本概念」の命名ルールをご参照ください。

エンドポイントに関する情報は、「リージョンおよびエンドポイント」をご参照ください。

オブジェクトのアップロード

次のコードを実行してファイルを OSS にオブジェクトをアップロードします。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Upload a file
    var result = client.PutObject(bucketName, objectName, localFilename);
    Console.WriteLine("Put object succeeded, ETag: {0} ", result.ETag);
}
} catch (Exception e) {
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

詳しくは、「Upload an object」をご参照ください。

オブジェクトのダウンロード

次のコードを実行して、指定したオブジェクトをローカルファイルにダウンロードします。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    client.PutObject(bucketName, objectName, localFilename);
    // Download the object.
    var result = client.GetObject(bucketName, objectName);
    using (var requestStream = result.Content)
    {
        using (var fs = File.Open(downloadFilename, FileMode.OpenOrCreate))
        {
            int length = 4 * 1024;
            var buf = new byte[length];
            do
            {
                length = requestStream.Read(buf, 0, length);
                fs.Write(buf, 0, length);
            } while (length ! = 0);
        }
```

```
    }
    Console.WriteLine("Get object succeeded");
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

オブジェクトの一覧表示

次のコードを実行して、指定したバケット内のオブジェクトを一覧で表示します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var objects = new List<string>();
    ObjectListing result = null;
    String nextmarker = string. empty;
    do
    {
        var listObjectsRequest = new ListObjectsRequest(bucketName)
        {
            Marker = nextMarker,
        };
        // List objects.
        result = client.ListObjects(listObjectsRequest);
        foreach (var summary in result.ObjectSummaries)
        {
            Console.WriteLine(summary.Key);
            objects.Add(summary.Key);
        }
        nextMarker = result.NextMarker;
    } while (result.IsTruncated);
    Console.WriteLine("List objects of bucket:{0} succeeded ", bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

オブジェクトの削除

指定したオブジェクトを削除するには、次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Delete an object.
    client.DeleteObject(bucketName, objectName);
    Console.WriteLine("Delete object succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Delete object failed, {0}", ex.Message);
}
```

8.3 初期化

OSSClient は、バケットやオブジェクトなどの OSS リソースを管理するための OSS C# クライアントとして機能します。

OSSClient インスタンスの作成

OSSClient インスタンスを作成するには、エンドポイントを指定する必要があります。 エンドポイントの詳細については、「リージョンおよびエンドポイント」および「カスタムドメイン名のバインド方法」をご参照ください。

- OSS ドメインを使用して OSSClient インスタンスを作成

  次のコードを実行して、OSS によって割り当てられたドメインを使用して OSSClient インスタンスを作成します。

  ```
  using Aliyun.OSS;

  const string accessKeyId = "<yourAccessKeyId>";
  const string accessKeySecret = "<yourAccessKeySecret>";
  const string endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

  // Create a new OSSClient instance with the OSS access address specified by the user and the AccessKeyId/AccessKeySecret granted by Alibaba Cloud.
  var ossClient = new OssClient(endpoint, accessKeyId, accessKeySecret);
  ```

- カスタムドメイン (CNAME) を使用して OSSClient インスタンスを作成

  次のコードを実行して、CNAME で OSSClient インスタンスを作成します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

const string accessKeyId = "<yourAccessKeyId>";
const string accessKeySecret = "<yourAccessKeySecret>";
const string endpoint = "<yourDomain>";

// Create a ClientConfiguration instance. Modify parameters as required.
var conf = new ClientConfiguration();

// Enable CNAME. CNAME indicates a custom domain bound to a bucket.
conf.IsCname = true;

// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret, conf);
```

  注 :

  CNAME が使用されている場合、ossClient.listBuckets メソッドは使用できません。

OSSClient インスタンスの設定

ClientConfiguration は OSSClient の設定クラスです。 ClientConfiguration は、ユーザーエージェント、ホストプロキシ、接続タイムアウト、最大接続数などのパラメーターを設定するために使用されます。 以下のパラメーターを設定します。

パラメーター	説明	設定方法
ConnectionLimit	有効にできる HTTP 接続の最大数を指定します。	512
MaxErrorRetry	要求エラーが発生した場合の最大再試行回数を指定します。	3
ConnectionTimeout	接続が確立されるときのタイムアウト時間をミリ秒単位で指定します。 既定値は -1 です。これは、接続がタイムアウトしていないことを表します。	-1
IsCname	CNAME をエンドポイントとして使用できるかどうかを指定します。	false

パラメーター	説明	設定方法
ProgressUpdateInterval	プログレスバーの更新間隔をバイト単位で指定します。	8096

コード例は次のとおりです。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var conf = new ClientConfiguration();
conf.ConnectionLimit = 512;
conf.MaxErrorRetry = 3;
conf.ConnectionTimeout = 300;

var client = new OssClient(endpoint, accessKeyId, accessKeySecret, conf);
```

- データ検証

  MD5 データ検証には、次のコードを実行します。

  ```
  using Aliyun.OSS;
  using Aliyun.OSS.Common;

  var conf = new ClientConfiguration();
  // Enable MD5 verification. Set EnableMD5Check to perform MD5 verification on the uploaded or downloaded data. この機能はデフォルトで無効です。
  conf.EnalbeMD5Check = true;

  var client = new OssClient(endpoint, accessKeyId, accessKeySecret, conf);
  ```

  注 :

  MD5 検証を無効にすると、システムパフォーマンスが低下します。

- Proxy network

  プロキシネットワークを使用している場合は、OSS にアクセスするために次のパラメータを設定します。

パラメーター	説明	既定値
ProxyHost	8.8.8.8 や abc.def.com などのプロキシサーバー。	Null
ProxyPort	3128 や 8080 などのプロキシポート。	Null
ProxyUserName	プロキシサービスアカウント。オプションです。	Null

パラメーター	説明	既定値
ProxyPassword	プロキシサービスのパスワード。オプションです。	Null

アカウントとパスワードを使用せずに、プロキシネットワークにアクセスする例は次のとおりです。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var conf = new ClientConfiguration();
conf.ProxyHost = "8.8.8.8";
conf.ProxyPort = 3128;

var client = new OssClient(endpoint, accessKeyId, accessKeySecret, conf);
```

アカウントとパスワードを使用した、プロキシネットワークアクセスの例は次のとおりです。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var conf = new ClientConfiguration();
conf.ProxyHost = "8.8.8.8";
conf.ProxyPort = 3128;
conf.ProxyUserName = "user";
conf.ProxyPassword = "6666";

var client = new OssClient(endpoint, accessKeyId, accessKeySecret, conf);
```

8.4 オブジェクトのアップロード

8.4.1 概要

OSS では、オブジェクトはデータ操作の基本単位です。 OSS NET SDK は次のファイルアップロード方法を提供します。

- 簡易アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 追加アップロード: 最大 5 GB までのファイルのアップロードをサポートします。
- 再開可能アップロード: 同時アップロードをサポートし、各パーツのサイズを定義できます。大きなファイルのアップロードに適用されます。 再開可能アップロードは、最大 48.8 TB のサイズのファイルのアップロードをサポートします。
- マルチパートアップロード: 最大 48.8 TB のファイルのアップロードをサポートします。 大きなファイルのアップロードに適用されます。

注 :

各アップロード方法の使用シナリオの詳細は、「OSS 開発者ガイド」内の「ファイルのアップロード」をご参照ください。

ファイルのアップロード中に、「オブジェクトメタ」を設定することで、「アップロードの進行状況」を表示することができます。 ファイルのアップロードが完了したら、「アップロードコールバック」を実行することできます。

8.4.2 簡易アップロード

文字列のアップロード

文字列をアップロードするための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行し、文字列をアップロードします。

```
using System.Text;
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var objectContent = "More than just cloud." ;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    byte[] binaryData = Encoding.ASCII.GetBytes(objectContent);
    MemoryStream requestContent = new MemoryStream(binaryData);
    // Upload a file.
    client.PutObject(bucketName, objectName, requestContent);
    Console.WriteLine("Put object succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

指定されたローカルファイルのアップロード

指定のローカルファイルをアップロードするための完全なコードについては、『GitHub』をご参照ください。

指定したローカルファイルをアップロードするには、次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
```

```
    // Upload a local file.
    client.PutObject(bucketName, objectName, localFilename);
    Console.WriteLine("Put object succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

MD5 検証を使用したファイルのアップロード

SDK によって送信されたデータと OSS クライアントによって受信されたデータの間の一貫性を保証するために、ObjectMeta に Content-MD5 値を追加できます。 その後、OSS は受信データを MD5 値と照合します。

MD5 検証を使用してファイルをアップロードするための完全なコードについては、『GitHub』をご参照ください。

```
using Aliyun.OSS;
using Aliyun.OSS.Util;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Calculate the MD5 value.
    string md5;
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        md5 = OssUtils.ComputeContentMd5(fs, fs.Length);
    }
    var objectMeta = new ObjectMetadata
    {
        ContentMd5 = md5
    };
    // Upload a file.
    client.PutObject(bucketName, objectName, localFilename, objectMeta);
    Console.WriteLine("Put object succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

注：

MD5 検証はパフォーマンスの低下を引き起こす可能性があります。

非同期アップロード

非同期アップロードのための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行し、非同期アップロードを実行します。

```
using System;
using System.IO;
using System.Threading;

using Aliyun.OSS;
using Aliyun.OSS.Common;

// Upload a file asynchronously.
namespace AsyncPutObject
{
   class Program
   {
      static string endpoint = "<yourEndpoint>";
      static string accessKeyId = "<yourAccessKeyId>";
      static string accessKeySecret = "<yourAccessKeySecret>";
      static string bucketName = "<yourBucketName>";
      static string objectName = "<yourObjectName>";
      static string localFilename = "<yourLocalFilename>";

      static AutoResetEvent _event = new AutoResetEvent(false);

      // Create an OSSClient instance.
      static OssClient client = new OssClient(endpoint, accessKeyId, accessKeySecret);

      private static void PutObjectCallback(IAsyncResult ar)
      {
         try
         {
            client.EndPutObject(ar);
            Console.WriteLine(ar.AsyncState as string);
            Console.WriteLine("Put object succeeded");
         }
         } catch (Exception e) {
         {
            Console.WriteLine(ex.Message);
         }
         finally
         {
            _event.Set();
         }
      }

      public static void AsyncPutObject()
      {
         try
         {
            using (var fs = File.Open(localFilename, FileMode.Open))
            {
               var metadata = new ObjectMetadata();
    // Add custom meta information.
               metadata.UserMetadata.Add("mykey1", "myval1");
               metadata.UserMetadata.Add("mykey2", "myval2");
               metadata.CacheControl = "No-Cache";
               metadata.ContentType = "text/html";
               string result = "Notice user: put object finish";

               // Upload a file asynchronously.
               client.BeginPutObject(bucketName, objectName, fs, metadata, PutObjectC
allback, result.ToCharArray());

               _event.WaitOne();
```

```
            }
        }
        catch (OssException ex)
        {
            Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\
tHostID:{3}",
                ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
        }
        } catch (Exception e) {
        {
            Console.WriteLine("Failed with error info: {0}", ex.Message);
        }
    }

    static void Main(string[] args)
    {
        Program.AsyncPutObject();
    }
  }
}
```

注：

非同期アップロードを使用する場合、独自のコールバックハンドラーを実装する必要があります。

8.4.3 追加アップロード

追加アップロードの完全なコードについては、『GitHub』をご参照ください。

追加オブジェクトに対して copyObject を実行することはできません。 AppendObject を呼び出してデータをアップロードするときに、ターゲットオブジェクトが存在しない場合は、追加可能なオブジェクトが作成されます。 オブジェクトがすでに存在する場合は、データがオブジェクトに追加されます。

追加アップロードするには次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// If the object is appended for the first time, the append position is 0, and the returned
value is the position for the next append. The position for the next append is the length
of object before the append.
long position = 0;
try
{
    var metadata = client.GetObjectMetadata(bucketName, objectName);
    position = metadata.ContentLength;
}
catch (Exception) { }
try
```

```
{
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        var request = new AppendObjectRequest(bucketName, objectName)
        {
            ObjectMetadata = new ObjectMetadata(),
            Content = fs,
            Position = position
        };
        // Append the object.
        var result = client.AppendObject(request);
        // Configure the position where the object is appended
        position = result.NextAppendPosition;
        Console.WriteLine("Append object succeeded, next append position:{0}", position);
    }
    // Obtain the start position for the next append.
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        var request = new AppendObjectRequest(bucketName, objectName)
        {
            ObjectMetadata = new ObjectMetadata(),
            Content = fs,
            Position = position
        };
        var result = client.AppendObject(request);
        position = result.NextAppendPosition;
        Console.WriteLine("Append object succeeded, next append position:{0}", position);
    }
}
} catch (Exception e) {
{
    Console.WriteLine("Append object failed, {0}", ex.Message);
}
```

8.4.4 再開可能アップロード

再開可能アップロードを使用すると、アップロードするファイルを複数のパーツに分割して同時にアップロードできます。 すべてのパーツをアップロードしたら、パーツを組み合わせて完全なオブジェクトにすることができます。 こうして、オブジェクト全体をアップロードすることができます。

現在のアップロードの進行状況は、アップロード中にチェックポイントファイルに記録されます。 処理中にパーツのアップロードに失敗した場合、オブジェクトはアップロードの再開時にチェックポイントファイルに記録され、再開可能アップロードのパーツからアップロードされます。 アップロードが完了すると、チェックポイントは削除されます。

再開可能アップロードの完全なコードについては、『GitHub』をご参照ください。

再開可能アップロードを行うには、次のコードを実行します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
```

```
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
string checkpointDir = "<yourCheckpointDir>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    //  Configure parameters with UploadFileRequest.
    UploadObjectRequest request = new UploadObjectRequest(bucketName, objectName,
localFilename)
    {
        // Specify the size of a part you want to upload.
        PartSize = 8 * 1024 * 1024,
        // Specify the number of threads for simultaneous upload.
        ParallelThreadCount = 3,
        // The progress information of a resumable upload is stored in the checkpointDir
directory.  If a part fails to be uploaded, the progress information is used to continue the
upload. If the checkpointDir directory is null, the resumable upload does not take effect,
and the file that failed to be uploaded is uploaded all over again.
        CheckpointDir = checkpointDir,
    };
    // Start resumable upload.
    client.ResumableUploadObject(request);
    Console.WriteLine("Resumable upload object:{0} succeeded", objectName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

8.4.5 マルチパートアップロード

マルチパートアップロードの完全なコードについては、『GitHub』をご参照ください。

マルチパートアップロードを有効にするには、次の手順を実行します。

1. マルチパートアップロードイベントを開始します。

   OssClient.initiateMultipartUpload を呼び出して、OSS で作成されたグローバルに一意な uploadId を返します。

2. パーツをアップロード

   OssClient.uploadPart を呼び出してパーツデータをアップロードします。

   :

   - 同じ uploadId を持つパーツの場合、パーツはそれらのパーツ番号によって順序付けされます。 パーツをアップロードし、同じパーツ番号を使用して別のパーツをアップロードすると、後のパーツが前のパーツに置き換わります。

- OSSは パーツデータの MD5 値を ETag に格納し、MD5 値をユーザーに返します。
- SDK は自動的に Content-MD5 を設定します。 OSS はアップロードされたデータの MD5 値を計算し、それを SDK によって計算された MD5 値と比較します。 2 つの値が異なると、InvalidDigest のエラーコードが返されます。

3. マルチパートアップロードの完了

すべてのパーツをアップロードした後、partossClient.completeMultipartUpload を呼び出して各パーツを完全なオブジェクトにまとめます。

次のコードは、マルチパートアップロードのプロセスを説明する完全な例です。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// Initiate the multipart upload.
var uploadId = "";
try
{
    // Specifiy the name and the bucket of the uploaded object. You can configure
ObjectMeta when using InitiateMultipartUploadRequest. However, you do not need to
specify the ContentLength.
    var request = new InitiateMultipartUploadRequest(bucketName, objectName);
    var result = client.InitiateMultipartUpload(request);
    uploadId = result.UploadId;
    // Print the UploadId
    Console.WriteLine("Init multi part upload succeeded");
    Console.WriteLine("Upload Id:{0}", result.UploadId);
}
} catch (Exception e) {
{
    Console.WriteLine("Init multi part upload failed, {0}", ex.Message);
}
// Calculate the total number of parts.
var partSize = 100 * 1024;
var fi = new FileInfo(localFilename);
var fileSize = fi.Length;
var partCount = fileSize / partSize;
if (fileSize % partSize ! = 0)
{
    partCount++;
}
// Start the multipart upload. partETags is a list of partETags. OSS verifies the validity of
 all parts one by one after it receives the partETags. After part verification is successful,
OSS combines these parts into a complete object.
var partETags = new List<PartETag>();
try
{
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        for (var i = 0; i < partCount; i++)
        {
            var skipBytes = (long)partSize * i;
```

```
            // Locate to the start position of the upload.
            fs.Seek(skipBytes, 0);
            // Calculate the size of the uploaded part. The size of the last part is the size of the
 data remained after being splited by the part size.
            var size = (partSize < fileSize - skipBytes) ? partSize : (fileSize - skipBytes);
            var request = new UploadPartRequest(bucketName, objectName, uploadId)
            {
                InputStream = fs,
                PartSize = size,
                PartNumber = i + 1
            };
            // Call the UploadPart interface to upload the object. The returned results contain
the ETag value of the uploaded part.
            var result = client.UploadPart(request);
            partETags.Add(result.PartETag);
            Console.WriteLine("finish {0}/{1}", partETags.Count, partCount);
        }
        Console.WriteLine("Put multi part upload succeeded");
    }
}
} catch (Exception e) {
{
    Console.WriteLine("Put multi part upload failed, {0}", ex.Message);
}
// List uploaded parts.
try
{
    var listPartsRequest = new ListPartsRequest(bucketName, objectName, uploadId);
    var listPartsResult = client.ListParts(listPartsRequest);
    Console.WriteLine("List parts succeeded");
    // Upload each part simultaneously until all parts are uploaded.
    var parts = listPartsResult.Parts;
    foreach (var part in parts)
    {
        Console.WriteLine("partNumber: {0}, ETag: {1}, Size: {2}", part.PartNumber, part.ETag
, part.Size);
    }
}
} catch (Exception e) {
{
    Console.WriteLine("List parts failed, {0}", ex.Message);
}
// Complete the multipart upload.
try
{
    var completeMultipartUploadRequest = new CompleteMultipartUploadRequest(
bucketName, objectName, uploadId);
    foreach (var partETag in partETags)
    {
        completeMultipartUploadRequest.PartETags.Add(partETag);
    }
    var result = client.CompleteMultipartUpload(completeMultipartUploadRequest);
    Console.WriteLine("complete multi part succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("complete multi part failed, {0}", ex.Message);
```

```
}
```

マルチパートアップロードイベントのキャンセル

マルチプラットフォームアップロードイベントをキャンセルするための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行してマルチパートアップロードイベントをキャンセルします。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var uploadId = "";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// Initiate the multipart upload.
try
{
    var request = new InitiateMultipartUploadRequest(bucketName, objectName);
    var result = client.InitiateMultipartUpload(request);
    uploadId = result.UploadId;
    // Print the UploadId.
    Console.WriteLine("Init multi part upload succeeded");
    Console.WriteLine("Upload Id:{0}", result.UploadId);
}
} catch (Exception e) {
{
    Console.WriteLine("Init multi part upload failed, {0}", ex.Message);
}
// Cancel the multipart upload.
try
{
    var request = new AbortMultipartUploadRequest(bucketName, objectName, uploadId);
    client.AbortMultipartUpload(request);
    Console.WriteLine("Abort multi part succeeded,  {0}", uploadId);
}
} catch (Exception e) {
{
    Console.WriteLine("Abort multi part failed, {0}", ex.Message);
}
```

アップロードしたパーツの一覧表示

アップロードされたパーツを一覧表示する完全なコードについては、『GitHub』をご参照ください。

次のコードを実行してアップロードされたパーツを一覧表示します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var uploadId = "<yourUploadId>";
```

```
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    PartListing listPartsResult = null;
    var nextMarker = 0;
    do
    {
        var listPartsRequest = new ListPartsRequest(bucketName, objectName, uploadId)
        {
            PartNumberMarker = nextMarker,
        };
        // List uploaded parts.
        listPartsResult = client.ListParts(listPartsRequest);
        Console.WriteLine("List parts succeeded");
        // Upload each part simultaneously until all parts are uploaded.
        var parts = listPartsResult.Parts;
        foreach (var part in parts)
        {
            Console.WriteLine("partNumber: {0}, ETag: {1}, Size: {2}", part.PartNumber, part.
ETag, part.Size);
        }
        nextMarker = listPartsResult.NextPartNumberMarker;
    } while (listPartsResult.IsTruncated);
}
} catch (Exception e) {
{
    Console.WriteLine("List parts failed, {0}", ex.Message);
}
```

マルチパートアップロードイベントの一覧表示

ossClient.listMultipartUploads を呼び出して、進行中のすべてのパーツアップロードイベント (開始されたが完了していないかキャンセルされたイベント) を一覧表示します。 次のパラメーターをモニタリングできます。

パラメーター	説明	設定方法
prefix	返されるオブジェクト名に含める必要があるプレフィックスを指定します。 クエリにプレフィックスを使用した場合、返されるオブジェクト名にはそのプレフィックスが含まれます。	ListMultipartUploadsRequest.setPrefix(String prefix)
delimiter	オブジェクト名をグループ化するために使用されるスラッシュ (/) の区切り記号を指定します。 指定されたプレフィックスとスラッシュ (/) の最初の区切り記号の間のオブジェクトは commonPrefixes です。	ListMultipartUploadsRequest.setDelimiter(String delimiter)

パラメーター	説明	設定方法
maxUploads	パートアップロードイベントの最大数を指定します。 設定できる最大値 (または既定値) は1,000です。	ListMultipartUploadsRequest.setMaxUploads(Integer maxUploads)
keyMarker	オブジェクト名が、keyMarker 値のアルファベットより後のアルファベットで始まるオブジェクトのマルチパートアップロードイベントをすべてリストします。 このパラメーターを uploadIdMarker パラメーターと一緒に使用して、指定された戻り結果の初期位置を指定します。	ListMultipartUploadsRequest.setKeyMarker(String keyMarker)
uploadIdMarker	このパラメーターを keyMarker パラメーターと一緒に使用して、指定された戻り結果の初期位置を指定できます。 keyMarker を構成しない場合、uploadIdMarker パラメーターは無効です。 keyMarker を設定した場合、クエリ結果には以下が含まれます。 • keyMarker 値の後の文字で名前が始まるすべてのオブジェクト。 • オブジェクトの名前が、アルファベット順の keyMarker 値と同じ文字から始まり、uploadId の値が uploadIdMarker よりも大きい文字で始まるすべてのオブジェクト	ListMultipartUploadsRequest.setUploadIdMarker(String uploadIdMarker)

マルチプラットフォームアップロードイベント一覧表示の完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、マルチパートアップロードイベントをすべて一覧表示します。

```
using Aliyun.OSS;
```

```
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    MultipartUploadListing multipartUploadListing = null;
    var nextMarker = string.Empty;
    do
    {
        // List multipart upload events.
        var request = new ListMultipartUploadsRequest(bucketName)
        {
            KeyMarker = nextMarker,
        };
        multipartUploadListing = client.ListMultipartUploads(request);
        Console.WriteLine("List multi part succeeded");
        // List the information about multipart upload events.
        foreach (var mu in multipartUploadListing.MultipartUploads)
        {
            Console.WriteLine("Key: {0},  UploadId: {1}", mu.Key, mu.UploadId);
        }
        // If the returned result shows that the value of isTruncated is false, the values of
nextKeyMarker and nextUploadIdMarker are returned and used as the initial position for
the next object reading.
        nextMarker = multipartUploadListing.NextKeyMarker;
    } while (multipartUploadListing.IsTruncated);
}
catch (Exception ex)
{
    Console.WriteLine("List multi part uploads failed, {0}", ex.Message);
}
```

返される結果のプレフィックスと最大数を指定

次のコードを実行して、指定されたプレフィックスと返される結果の最大数を指定してマルチパートアップロードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Define prefix.
var prefix = "<yourObjectPrefix>";
// Set the maximum number of returned result to 100.
var maxUploads = 100;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    MultipartUploadListing multipartUploadListing = null;
    var nextMarker = string.Empty;
    do
    {
        // List multipart upload events. By default, 1,000 parts are listed.
        var request = new ListMultipartUploadsRequest(bucketName)
        {
            KeyMarker = nextMarker,
            // Specify the prefix.
```

```
            Prefix = prefix,
            // Specify the maximum number of returned result.
            MaxUploads = maxUploads,
        };
        multipartUploadListing = client.ListMultipartUploads(request);
        Console.WriteLine("List multi part succeeded");
        // List the information about each multipart upload event.
        foreach (var mu in multipartUploadListing.MultipartUploads)
        {
            Console.WriteLine("Key: {0},  UploadId: {1}", mu.Key, mu.UploadId);
        }
        nextMarker = multipartUploadListing.NextKeyMarker;
    } while (multipartUploadListing.IsTruncated);
}
catch (Exception ex)
{
    Console.WriteLine("List multi part uploads failed, {0}", ex.Message);
}
```

8.4.6 アップロードの進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を表示することができます。

プログレスバーの完全なコードについては、『GitHub』をご参照ください。

次のコードは、PutObject を使用して進行状況を表示する方法を説明するための例です。

```
using System;
using System.IO;
using System.Text;
using Aliyun.OSS;
using Aliyun.OSS.Common;
namespace PutObjectProgress
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.PutObjectProgress();
            Console.ReadKey();
        }
        public static void PutObjectProgress()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            var localFilename = "<yourLocalFilename>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
            // Upload with a progress bar displayed.
            try
            {
                using (var fs = File.Open(localFilename, FileMode.Open))
                {
                    var putObjectRequest = new PutObjectRequest(bucketName, objectName, fs);
                    putObjectRequest.StreamTransferProgress += streamProgressCallback;
                    client.PutObject(putObjectRequest);
                }
```

```
            Console.WriteLine("Put object:{0} succeeded", objectName);
        }
        catch (OssException ex)
        {
            Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID: {2}\
tHostID: {3}",
                ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
        }
        } catch (Exception e) {
        {
            Console.WriteLine("Failed with error info: {0}", ex.Message);
        }
    }
    // Obtain the upload progress.
    private static void streamProgressCallback(object sender, StreamTransferProgre
ssArgs args)
    {
        System.Console.WriteLine("ProgressCallback - Progress: {0}%, TotalBytes:{1},
TransferredBytes:{2} ",
            args.TransferredBytes * 100 / args.TotalBytes, args.TotalBytes, args.Transferre
dBytes);
    }
  }
}
```

8.4.7 アップロードコールバック

アップロードコールバックの完全なコードについては、『GitHub』をご参照ください。

アップロードコールバックを行うには次のコードを実行します。

```
using System;
using System.IO;
using System.Text;
using Aliyun.OSS;
using Aliyun.OSS.Common;
using Aliyun.OSS.Util;
namespace Callback
{
  class Program
  {
    static void Main(string[] args)
    {
        Program.PutObjectCallback();
        Console.ReadKey();
    }
    public static void PutObjectCallback()
    {
        var endpoint = "<yourEndpoint>";
        var accessKeyId = "<yourAccessKeyId>";
        var accessKeySecret = "<yourAccessKeySecret>";
        var bucketName = "<yourBucketName>";
        var objectName = "<yourObjectName>";
        var localFilename = "<yourLocalFilename>";
        // Specify the IP address of the server you want to send the callback request to.
        const string callbackUrl = "http://oss-demo.aliyuncs.com:23450";
        # Configure the value of the body field carried in the callback request.
        const string callbackBody = "bucket=${bucket}&object=${object}&etag=${etag}&size
=${size}&mimeType=${mimeType}&" +
                                    "my_var1=${x:var1}&my_var2=${x:var2}";
        // Create an OSSClient instance.
        var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
```

```
            try
            {
                string responseContent = "";
                var metadata = BuildCallbackMetadata(callbackUrl, callbackBody);
                using (var fs = File.Open(localFilename, FileMode.Open))
                {
                    var putObjectRequest = new PutObjectRequest(bucketName, objectName, fs,
metadata);
                    var result = client.PutObject(putObjectRequest);
                    responseContent = GetCallbackResponse(result);
                }
                Console.WriteLine("Put object:{0} succeeded, callback response content:{1}",
objectName, responseContent);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID: {2}\
tHostID: {3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
            }
            } catch (Exception e) {
            {
                Console.WriteLine("Failed with error info: {0}", ex.Message);
            }
        }
        // Configure upload callback.
        private static ObjectMetadata BuildCallbackMetadata(string callbackUrl, string
callbackBody)
        {
            string callbackHeaderBuilder = new CallbackHeaderBuilder(callbackUrl,
callbackBody). Build();
            string CallbackVariableHeaderBuilder = new CallbackVariableHeaderBuilder().
                AddCallbackVariable("x:var1", "x:value1"). AddCallbackVariable("x:var2", "x:
value2"). Build();
            var metadata = new ObjectMetadata();
            metadata.AddHeader(HttpHeaders.Callback, callbackHeaderBuilder);
            metadata.AddHeader(HttpHeaders.CallbackVar, CallbackVariableHeaderBuilder);
            return metadata;
        }
        // Read the message returned from upload callback.
        private static string GetCallbackResponse(PutObjectResult putObjectResult)
        {
            string callbackResponse = null;
            using (var stream = putObjectResult.ResponseStream)
            {
                var buffer = new byte[4 * 1024];
                var bytesRead = stream.Read(buffer, 0, buffer.Length);
                callbackResponse = Encoding.Default.GetString(buffer, 0, bytesRead);
            }
            return callbackResponse;
        }
    }
}
```

注：

アップロードコールバックの詳細については、『開発者ガイド』内の「アップロードコールバック」をご参照ください。

8.5 オブジェクトのダウンロード

8.5.1 概要

OSSNET SDK は以下のダウンロード方法を提供します。

- ストリーミングダウンロード
- 部分ダウンロード
- 再開可能なダウンロード

ダウンロードの進行状況は「ダウンロードプログレスバー」に表示されます。

8.5.2 ストリーミングダウンロード

大容量のオブジェクトをダウンローズする場合、またはオブジェクト全体を一度にダウンロードするのに時間がかかる場合は、ストリーミングダウンロードを使用します。 ストリーミングダウンロードでは、オブジェクト全体をダウンロードし終わるまで、オブジェクトのパーツをその都度ダウンロードし続けます。

ストリーミングダウンロードの完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、指定した OSS オブジェクトをストリームにダウンロードします。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Download an object to a stream. OssObject includes object information, such as the
bucket that contains the object, object name, metadata, and an input stream.
    var obj = client.GetObject(bucketName, objectName);
    using (var requestStream = obj.Content)
    {
        byte[] buf = new byte[1024];
        var fs = File.Open(downloadFilename, FileMode.OpenOrCreate);
        var len = 0;
        // Use the input stream to read the object content into a file or the memory.
        while ((len = requestStream.Read(buf, 0, 1024)) ! = 0)
        {
            fs.Write(buf, 0, len);
        }
        rs.close();
    }
    Console.WriteLine("Get object succeeded");
}
} catch (Exception e) {
{
```

```
    Console.WriteLine("Get object failed. {0}", ex.Message);
}
```

8.5.3 部分ダウンロード

オブジェクトのデータの一部だけが必要な場合は、部分ダウンロードを使用することで、指定のコンテンツをダウンロードできます。

部分ダウンロードの完全なコードについては、『GitHub』をご参照ください。

部分ダウンロードを行うには次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var getObjectRequest = new GetObjectRequest(bucketName, objectName);
    // Set the range, which is from the 20th byte to the 100th byte.
    getObjectRequest.SetRange(20, 100);
    // Start range download. The setRange of getObjectRequest can be used to implement
multipart download and resumable download.
    var obj = client.GetObject(getObjectRequest);
    // Download the data and write it into a file.
    using (var requestStream = obj.Content)
    {
        byte[] buf = new byte[1024];
        var fs = File.Open(downloadFilename, FileMode.OpenOrCreate);
        var len = 0;
        while ((len = requestStream.Read(buf, 0, 1024)) ! = 0)
        {
            fs.Write(buf, 0, len);
        }
        rs.close();
    }
    Console.WriteLine("Get object succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Get object failed. {0}", ex.Message);
}
```

GetObjectRequest の以下のパラメーターを設定します。

パラメーター	説明
Range	オブジェクトの送信範囲を指定します。

パラメーター	説明
ModifiedSinceConstraint	指定された時間がファイルの実際の変更時間より早い場合、ファイルは正常に送信されます。それ以外の場合は、304 Not Modified エラーが返されます。
UnmodifiedSinceConstraint	パラメーターに渡された時間がファイルの実際の変更時刻と同じかそれより遅い場合、ファイルは通常どおりに送信されます。それ以外の場合、412 事前条件失敗エラーが返されます。
MatchingETagConstraints	ETag のグループ内でパスします。 渡される ETag がオブジェクトファイルの ETag と一致する場合、ファイルは正常に送信されます。それ以外の場合、412 事前条件失敗エラーが返されます。
NonmatchingEtagConstraints	ETag のグループ内でパスします。 渡される ETag がオブジェクトファイルの ETag と一致しない場合、ファイルは正常に送信されます。それ以外の場合は、304 Not Modifiedエラーが返されます。
ResponseHeaderOverrides	OSS から返されるリクエスト内のいくつかのヘッダーをカスタマイズします。

8.5.4 再開可能ダウンロード

サイズが大きいオブジェクトは、ネットワークが不安定であったり、Java プログラムが異常終了することがあるため、ダウンロードに失敗することがあります。その際は、オブジェクト全体を再度ダウンロードする必要があります。 オブジェクトは複数回試行してもダウンロードに失敗することがあります。 そのため、OSS は再開可能ダウンロードを提供しています。

再開可能ダウンロードは次のコードを実行します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
var downloadFilename = "<yourDownloadFilename>";
var checkpointDir = "<yourCheckpointDir>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Configure parameters with DownloadObjectRequest.
```

```
    DownloadObjectRequest request = new DownloadObjectRequest(bucketName,
objectName, downloadFilename)
    {
        // Specify the size of a part you want to download.
        PartSize = 8 * 1024 * 1024,
        // Specifies the number of concurrent download threads.
        ParallelThreadCount = 3,
        // Specify the checkpointDir to store the download progress information. ダウンロードが失敗した場合は、進行状況に基づいて続行できます。 checkpointDir ディレクトリが null の場合、再開可能アップロードは有効になりません。ダウンロードに失敗したオブジェクトはもう一度ダウンロードされます。
        CheckpointDir = checkpointDir,
    };
    // Start resumable download.
    client.ResumableDownloadObject(request);
    Console.WriteLine("Resumable download object:{0} succeeded", objectName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

再開可能ダウンロードの詳細については、「OSS 開発者ガイド」内の「再開可能ダウンロード」をご参照ください。

8.5.5 ダウンロード進捗プログレスバー

プログレスバーを使用して、アップロードまたはダウンロードの進行状況を表示することができます。

ダウンロード進捗プログレスバーについての完全なコードについては、『GitHub』をご参照ください。

次のコードは、GetObject を使用して進行状況を表示する方法の説明例です。

```
using System;
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
namespace GetObjectProgress
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.GetObjectProgress();
            Console.ReadKey();
        }
        public static void GetObjectProgress()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
```

```
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
            try
            {
                var getObjectRequest = new GetObjectRequest(bucketName, objectName);
                getObjectRequest.StreamTransferProgress += streamProgressCallback;
                // Download an object.
                var ossObject = client.GetObject(getObjectRequest);
                using (var stream = ossObject.Content)
                {
                    var buffer = new byte[1024 * 1024];
                    var bytesRead = 0;
                    while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
                    {
                        // Process the read data (detailed code is not listed here).
                    }
                }
                Console.WriteLine("Get object:{0} succeeded", objectName);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\
tHostID:{3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
            }
            } catch (Exception e) {
            {
                Console.WriteLine("Failed with error info: {0}", ex.Message);
            }
        }
        private static void streamProgressCallback(object sender, StreamTransferProgre
ssArgs args)
        {
            System.Console.WriteLine("ProgressCallback - Progress: {0}%, TotalBytes:{1},
TransferredBytes:{2} ",
                args.TransferredBytes * 100 / args.TotalBytes, args.TotalBytes, args.Transferre
dBytes);
        }
    }
}
```

8.6 オブジェクトの管理

8.6.1 概要

このトピックは、OSS におけるオブジェクト管理の概要です。NET SDK

API を使用して、バケット内のオブジェクトに対して次の操作を実行できます。

- オブジェクトが存在するかの確認
- オブジェクトの ACL を管理
- オブジェクトメタの管理
- オブジェクトの列挙

- オブジェクトの削除
- オブジェクトのコピー
- アーカイブオブジェクトの復元
- シンボリックリンクの管理

8.6.2 指定したオブジェクトの存在有無の確認

次のコードを実行して、指定したオブジェクトが存在するかを確認します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";

// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    //  Determine whether a specified object exists.
    var exist = client.DoesObjectExist(bucketName, objectName);
    Console.WriteLine("Object exist ? " + exist);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

8.6.3 オブジェクトの ACL の管理

次の表は、オブジェクトのアクセス制御リスト (ACL) に含まれるアクセス許可について説明しています。

権限	説明	値
default	オブジェクトの ACL とバケットの ACL は同一です。	CannedAccessControlList.Default
Private	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。	CannedAccessControlList.Private

権限	説明	値
Public read	オブジェクトを読み書きできるのは、オブジェクト所有者と許可ユーザーだけです。 他のユーザーはオブジェクトを読むことしかできません。 この許可は慎重に承認してください。	CannedAccessControlList.PublicRead
Public read-write	すべてのユーザーがオブジェクトを読み書きできます。 この許可は慎重に承認してください。	CannedAccessControlList.PublicReadWrite

オブジェクトの ACL はバケットの ACL よりも優先されます。 たとえば、バケットの ACL が "Private" で、オブジェクトの ACL が "Public read-write" である場合、すべてのユーザーがそのオブジェクトを読み書きできます。 オブジェクトが ACL で設定されていない場合、既定ではオブジェクトの ACL はバケットの ACL です。

オブジェクトの ACL を設定する完全なコードについては、『GitHub』をご参照ください。 オブジェクトの ACL を取得するための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行してオブジェクトの ACL を設定し、オブジェクトの ACL を取得できます。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";

// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// Configure an ACL for an object.
try
{
    // Use SetObjectAcl to configure an ACL for an object.
    client.SetObjectAcl(bucketName, objectName, CannedAccessControlList.PublicRead);
    Console.WriteLine("Set Object:{0} ACL succeeded ", objectName);
}
} catch (Exception e) {
{
    Console.WriteLine("Set Object ACL failed with error info: {0}", ex.Message);
}
// Obtain the ACL for an object.
try
{
    // Use GetObjectAcl to obtain the ACL for an object.
    var result = client.GetObjectAcl(bucketName, objectName);
    Console.WriteLine("Get Object ACL succeeded, Id: {0}  ACL: {1}",
```

```
        result.Owner.Id, result.ACL.ToString());
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID: {2}\tHostID:
 {3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

オブジェクト ACL の詳細については、 「アクセス制御」をご参照ください。

8.6.4 オブジェクトメタの管理

オブジェクトメタには、HTTP ヘッダーとユーザー定義のオブジェクトメタが含まれます。 詳しくは、「OSS 開発者ガイド」内の「オブジェクトメタ」をご参照ください。

オブジェクトメタを設定するための完全なコードについては、『GitHub』をご参照ください。 オブジェクトメタを変更するための完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、オブジェクトメタ を設定、変更、および取得します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        // Create Object Meta. You can configure HTTP headers with Object Meta.
        var metadata = new ObjectMetadata()
        {
            // Specify the object type.
            ContentType = "text/html",
            // Configure the expiration time (GMT is used).
            ExpirationTime = DateTime.Parse("2025-10-12T00:00:00.000Z"),
        };
        // Configure the length of the object you want to upload. If the object length is
greater than the configured length, only the configured length of the object is uploaded
. If the configured length is greater than the actual object length, the entire object is
uploaded.
        metadata.ContentLength = fs.Length;
        // Configure the cache action of a webpage when an object is downloaded.
        metadata.CacheControl = "No-Cache";
        // Set the value of metadata mykey1 to myval1.
        metadata.UserMetadata.Add("mykey1", "myval1");
        // Set the value of metadata mykey2 to myval2.
        metadata.UserMetadata.Add("mykey2", "myval2");
        var saveAsFilename = "Filetest123.txt";
```

```
        var contentDisposition = string.Format("attachment;filename*=utf-8''{0}", HttpUtils.
EncodeUri(saveAsFilename, "utf-8"));
        // Provide a default file name when saving the requested content as a file.
        metadata.ContentDisposition = contentDisposition;
        // Upload a file and configure the Object Meta.
        client.PutObject(bucketName, objectName, fs, metadata);
        Console.WriteLine("Put object succeeded");
        //Obtain Object Meta.
        var oldMeta = client.GetObjectMetadata(bucketName, objectName);
        // Configure new Object Meta.
        var newMeta = new ObjectMetadata()
        {
            ContentType = "application/octet-stream",
            ExpirationTime = DateTime.Parse("2035-11-11T00:00:00.000Z"),
            // Specify the encoding format of a download object.
            ContentEncoding = null,
            CacheControl = ""
        };
        // Add custom Object Meta.
        newMeta.UserMetadata.Add("author", "oss");
        newMeta.UserMetadata.Add("flag", "my-flag");
        newMeta.UserMetadata.Add("mykey2", "myval2-modified-value");
        // Use the ModifyObjectMeta method to modify Object Meta.
        client.ModifyObjectMeta(bucketName, objectName, newMeta);
    }
}
} catch (Exception e) {
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

注：

- HTTP ヘッダーの詳細については、「RFC2616」をご参照ください。
- カスタム HTTP ヘッダーを持つオブジェクトメタのサイズは 8 KB 以下です。

8.6.5 オブジェクトの一覧表示

オブジェクトを一覧表示するコード全体については、『GitHub』をご参照ください。

オブジェクトはアルファベット順に一覧表示されています。 ListObjects を使用すると、バケット内のオブジェクトを一覧表示できます。 次の表に ListObjects のパラメーターを示します。

パラメーター	詳細
Delimiter	オブジェクト名をグループ化するために使用されるスラッシュ (/) の区切り記号を指定します。 CommonPrefixes は、区切り記号で終わり、同じプレフィックスを持つオブジェクトのグループです。
Marker	リスト内の最初のオブジェクトを指定します。

パラメーター	詳細
MaxKeys	一覧表示できるオブジェクトの最大数を指定します。 MaxKeys の既定値は 100 です。 MaxKeys の最大値は 1,000 です。
Prefix	必須オブジェクトを一覧表示するために設定するプレフィックスを指定します。

簡易リスト

簡易リストの完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、指定したバケット内のオブジェクトを一覧表示します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var listObjectsRequest = new ListObjectsRequest(bucketName);
    // Simply list the objects in a specified bucket. 100 records are returned by default.
    var result = client.ListObjects(listObjectsRequest);
    Console.WriteLine("List objects succeeded");
    foreach (var summary in result.ObjectSummaries)
    {
        Console.WriteLine("File name:{0}", summary.Key);
    }
}
catch (Exception ex)
{
    Console.WriteLine("List objects failed. {0}", ex.Message);
}
```

指定された数のオブジェクトを一覧表示

指定された数のオブジェクトを一覧表示するには、次のコードを使用します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var listObjectsRequest = new ListObjectsRequest(bucketName)
    {
        // A maximum of 200 records can be returned.
        MaxKeys = 200,
    };
    var result = client.ListObjects(listObjectsRequest);
    Console.WriteLine("List objects succeeded");
```

```
    foreach (var summary in result.ObjectSummaries)
    {
        Console.WriteLine(summary.Key);
    }
}
catch (Exception ex)
{
    Console.WriteLine("List objects failed, {0}", ex.Message);
}
```

指定したプレフィックスでオブジェクトを一覧表示

指定されたプレフィックスを持つオブジェクトを一覧表示するには、次のコードを使用します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var prefix = "<yourObjectPrefix>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var keys = new List<string>();
    ObjectListing result = null;
    string nextMarker = string.Empty;
    do
    {
        var listObjectsRequest = new ListObjectsRequest(bucketName)
        {
            Marker = nextMarker,
            MaxKeys = 100,
            Prefix = prefix,
        };
        result = client.ListObjects(listObjectsRequest);
        foreach (var summary in result.ObjectSummaries)
        {
            Console.WriteLine(summary.Key);
            keys.Add(summary.Key);
        }
        nextMarker = result.NextMarker;
    } while (result.IsTruncated);
    Console.WriteLine("List objects of bucket:{0} succeeded ", bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

marker によるオブジェクトの一覧表示

marker パラメーターは、リストの作成元のオブジェクトの名前を表します。 次のコードを実行して、リストの開始後にオブジェクトを指定します (marker で指定)。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var marker = "<yourObjectMarker>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var keys = new List<string>();
    ObjectListing result = null;
    string nextMarker = marker;
    do
    {
        var listObjectsRequest = new ListObjectsRequest(bucketName)
        // To increase the number of returned objects, you can modify the MaxKeys
parameter or use the Marker parameter for multiple reads.
        {
            Marker = nextMarker,
            MaxKeys = 100,
        };
        result = client.ListObjects(listObjectsRequest);
        foreach (var summary in result.ObjectSummaries)
        {
            Console.WriteLine(summary.Key);
            keys.Add(summary.Key);
        }
        nextMarker = result.NextMarker;
    // If the value of IsTruncated is true, the next read starts from NextMarker.
    } while (result.IsTruncated);
    Console.WriteLine("List objects of bucket:{0} succeeded ", bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\tHostID:{
3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

非同期モードでオブジェクトを一覧表示

非同期モードでオブジェクトを一覧表示する完全なコードについては、『GitHub』をご参照ください。

次のコードを実行して、非同期モードでオブジェクトを一覧表示します。

```
using System;
using System.IO;
using System.Threading;
using Aliyun.OSS;
namespace AsyncListObjects
{
    class Program
    {
        static string endpoint = "<yourEndpoint>";
        static string accessKeyId = "<yourAccessKeyId>";
        static string accessKeySecret = "<yourAccessKeySecret>";
        static string bucketName = "<yourBucketName>";
        static AutoResetEvent _event = new AutoResetEvent(false);
        // Create an OSSClient instance.
        static OssClient client = new OssClient(endpoint, accessKeyId, accessKeySecret);
        static void Main(string[] args)
        {
            Program.AsyncListObjects();
            Console.ReadKey();
        }
        public static void AsyncListObjects()
        {
            try
            {
                var listObjectsRequest = new ListObjectsRequest(bucketName);
                client.BeginListObjects(listObjectsRequest, ListObjectCallback, null);
                _event.WaitOne();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Async list objects failed. {0}", ex.Message);
            }
        }
        // The ListObjectCallback method is implemented after an asynchronous call. Similar
 interfaces must be used when an asynchronous interface is called to list objects.
        private static void ListObjectCallback(IAsyncResult ar)
        {
            try
            {
                var result = client.EndListObjects(ar);
                foreach (var summary in result.ObjectSummaries)
                {
                    Console.WriteLine ("Object name: 0", summary.Key);
                }
                _event.Set();
                Console.WriteLine("Async list objects succeeded");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Async list objects failed. {0}", ex.Message);
            }
        }
    }
```

```
}
```

バケット内のすべてのオブジェクトを一覧表示

次のコードを実行して、バケット内のすべてのオブジェクトを一覧表示します。

```
using Aliyun.OSS;
// Initialize OssClient.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// List all objects in the bucket.
public void ListObject(string bucketName)
{
    try
    {
        ObjectListing result = null;
        string nextMarker = string.Empty;
        do
        {
            // The number of objects listed in each page is specified by the maxKeys
parameter. If the object number is larger than the specified value, other files are listed in
 another page.
            var listObjectsRequest = new ListObjectsRequest(bucketName)
            {
                Marker = nextMarker,
                MaxKeys = 100
            };
            result = client.ListObjects(listObjectsRequest);
            Console.WriteLine("File:");
            foreach (var summary in result.ObjectSummaries)
            {
                Console.WriteLine("Name:{0}", summary.Key);
            }
            nextMarker = result.NextMarker;
        } while (result.IsTruncated);
    }
    catch (Exception ex)
    {
        Console.WriteLine("List object failed. {0}", ex.Message);
    }
}
```

8.6.6 オブジェクトの削除

このトピックでは、オブジェクトを削除する方法について説明します。

警告：

削除されたオブジェクトは回復できないため、オブジェクトは慎重に削除してください。

オブジェクトを削除するための完全なコードについては、『GitHub』をご参照ください。

オブジェクトの削除

1 つのオブジェクトを削除するには、次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
```

```
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    /// Delete an object.
    client.DeleteObject(bucketName, objectName);
    Console.WriteLine("Delete object succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Delete object failed. {0}", ex.Message);
}
```

複数のオブジェクトの削除

同時に最大 1,000 個のオブジェクトを削除できます。 オブジェクトの削除は、詳細モード (verbose) および簡易モード (quiet) の 2 つの方法があります。

- verbose: 削除に成功したオブジェクトのリストを返します。 既定のモードは verbose です。
- quiet: 削除に失敗したオブジェクトのリストを返します。

複数のオブジェクトを同時に削除するには、次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var keys = new List<string>();
    var listResult = client.ListObjects(bucketName);
    foreach (var summary in listResult.ObjectSummaries)
    {
        keys.Add(summary.Key);
    }
    // If the quietMode is true, the quiet mode is used. If the quietMode is false, the
verbose mode is used. The default mode is verbose.
    var quietMode = false;
    // The third parameter of the DeleteObjectsRequest specifies the return mode.
    var request = new DeleteObjectsRequest(bucketName, keys, quietMode);
    // Delete multiple objects.
    var result = client.DeleteObjects(request);
    if ((! quietMode) && (result.Keys ! = null) {
    {
        foreach (var obj in result.Keys)
        {
            Console.WriteLine("Delete successfully : {0} ", obj.Key);
        }
    }
    Console.WriteLine("Delete objects succeeded");
}
} catch (Exception e) {
{
    Console.WriteLine("Delete objects failed. {0}", ex.Message);
```

```
}
```

8.6.7 オブジェクトのコピー

小さいサイズのオブジェクトをコピーする

小さいサイズのオブジェクトをコピーする完全なコードについては、『GitHub』をご参照ください。

小さいサイズのオブジェクトをコピーする前に、次の制限事項に注意してください。

- ソースオブジェクトに対する読み取りおよび書き込み権限が必要です。
- あるリージョンから別のリージョンにオブジェクトをコピーすることはできません。 たとえば、中国 (杭州) リージョンのバケットから中国 (青島) リージョンのバケットにオブジェクトをコピーすることはできません。
- オブジェクトサイズは最大 1 GB に制限されています。

小さいサイズのオブジェクトをコピーするには、次のコードを実行します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var sourceBucket = "<yourSourceBucketName>";
var sourceObject = "<yourSourceObjectName>";
var targetBucket = "<yourDestBucketName>";
var targetObject = "<yourDestObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var metadata = new ObjectMetadata();
    metadata.AddHeader("mk1", "mv1");
    metadata.AddHeader("mk2", "mv2");
    var req = new CopyObjectRequest(sourceBucket, sourceObject, targetBucket,
targetObject)
    {
        // If the value of NewObjectMetadata is null, the COPY mode is used (the metadata
 of the source object is copied). If the value of NewObjectMetadata is not null, the
REPLACE mode is used (the metadata of the source object is replaced).
        NewObjectMetadata = metadata
    };
    // Copy an object.
    client.CopyObject(req);
    Console.WriteLine("Copy object succeeded");
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID: {2} \tHostID:
 {3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

大きいサイズのオブジェクトをコピー

大きいサイズのオブジェクトをコピーするための完全なコードについては、『GitHub』をご参照ください。

- UploadPartCopy

  1 GB を超えるオブジェクトをコピーするには、UploadPartCopy を使用する必要があります。

  1. UploadPartCopy イベントを開始するには、InitiateMultipartUploadRequest を使用します。
  2. UploadPartCopy メソッドを使用してマルチパートコピーを実行します。
  3. CompleteMultipartUpload メソッドを使用してオブジェクトのコピーを完了します。

  UploadPartCopy タスクに対して次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var sourceBucket = "<yourSourceBucketName>";
var sourceObject = "<yourSourceObjectName>";
var targetBucket = "<yourDestBucketName>";
var targetObject = "<yourDestObjectName>";
var uploadId = "";
var partSize = 50 * 1024 * 1024;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Initiate a copy task. You can use the InitiateMultipartUploadRequest method to
specify the metadata information about the target object.
    var request = new InitiateMultipartUploadRequest(targetBucket, targetObject);
    var result = client.InitiateMultipartUpload(request);
    // Print the uploadId.
    uploadId = result.UploadId;
    Console.WriteLine("Init multipart upload succeeded,  Upload Id: {0}", result.
UploadId);
    // Calculate the number of parts.
    var metadata = client.GetObjectMetadata(sourceBucket, sourceObject);
    var fileSize = metadata.ContentLength;
    var partCount = (int)fileSize / partSize;
    if (fileSize % partSize ! = 0)
    {
        partCount++;
    }
    // Start the UploadPartCopy task.
    var partETags = new List<PartETag>();
    for (var i = 0; i < partCount; i++)
    {
        var skipBytes = (long)partSize * i;
        var size = (partSize < fileSize - skipBytes) ? partSize : (fileSize - skipBytes);
```

```
        // Create UploadPartCopyRequest. You can specify conditions with UploadPart
CopyRequest.
        var uploadPartCopyRequest = new UploadPartCopyRequest(targetBucket,
targetObject, sourceBucket, sourceObject, uploadId)
          {
            PartSize = size,
            PartNumber = i + 1,
            // Beginindex is used to locate the location that the last UploadPartCopy
starts.
            BeginIndex = skipBytes
          };
        // Call the uploadPartCopy method to copy each part.
        var uploadPartCopyResult = client.UploadPartCopy(uploadPartCopyRequest);
        Console.WriteLine("UploadPartCopy : {0}", i);
        partETags.Add(uploadPartCopyResult.PartETag);
    }
    // Complete the UploadPartCopy task.
    var completeMultipartUploadRequest =
    new CompleteMultipartUploadRequest(targetBucket, targetObject, uploadId);
    // The partETags refers to the list of partETags saved during the multipart upload
. After OSS receives the part list submitted by the user, it verifies the validity of each
data part one by one. After the verification is passed, OSS conbine these parts into a
complete object.
    foreach (var partETag in partETags)
    {
        completeMultipartUploadRequest.PartETags.Add(partETag);
    }
    var completeMultipartUploadResult = client.CompleteMultipartUpload(completeMu
ltipartUploadRequest);
    Console.WriteLine("CompleteMultipartUpload succeeded");
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID: {2} \
tHostID: {3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

- 再開可能コピー

  コピーが中断された場合は、再開可能コピーを使用してコピーを続行できます。 再開可能コピーの完全なコードについては、『GitHub』をご参照ください。

  再開可能コピーには、次のコードを実行します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var sourceBucket = "<yourSourceBucketName>";
var sourceObject = "<yourSourceObjectName>";
var targetBucket = "<yourDestBucketName>";
var targetObject = "<yourDestObjectName>";
var checkpointDir = @"<yourCheckpointDir>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
```

```
    var request = new CopyObjectRequest(sourceBucket, sourceObject, targetBucket,
targetObject);
    // The checkpointDir directory stores the intermediate state information of a
resumable upload. The information will be used when a failed upload task is resumed
. // If the checkpointDir directory is null, resumable copy will not take effect, and the
object that previously failed to be copied will be copied all over again.
    client.ResumableCopyObject(request, checkpointDir);
    Console.WriteLine("Resumable copy new object:{0} succeeded", request.Destinatio
nKey);
}
} catch (Exception e) {
{
    Console.WriteLine("Resumable copy new object failed, {0}", ex.Message);
}
```

8.6.8 アーカイブオブジェクトの復元

アーカイブオブジェクトを復元するための完全なコードについては、『GitHub』をご参照ください。

アーカイブオブジェクトを読み込む前に復元する必要があります。 非アーカイブオブジェクトに対して restoreObject を呼び出さないでください。

アーカイブオブジェクトの状態変換プロセスは次のとおりです。

- アーカイブオブジェクトはフリーズ状態です。
- 復元のために送信した後、サーバーはオブジェクトを復元します。 オブジェクトは復元中の状態です。
- 復元後にオブジェクトを読むことができます。 オブジェクトが復元された状態は、既定で 1 日続きます。 この期間は最大 7 日間まで延長できます。 この期間が終了すると、オブジェクトはフリーズ状態に戻ります。

次のコードを実行してオブジェクトを復元します。

```
using Aliyun.OSS;
using Aliyun.OSS.Model;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var objectContent = "More than just cloud.";
int maxWaitTimeInSeconds = 600;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Create a bucket of the archive storage class..
    var bucket = client.CreateBucket(bucketName, StorageClass.Archive);
    Console.WriteLine("Create Archive bucket succeeded, {0} ", bucket.Name);
}
} catch (Exception e) {
{
    Console.WriteLine("Create Archive bucket failed, {0}", ex.Message);
```

```
}
// Upload a file.
try
{
    byte[] binaryData = Encoding.ASCII.GetBytes(objectContent);
    MemoryStream requestContent = new MemoryStream(binaryData);
    client.PutObject(bucketName, objectName, requestContent);
    Console.WriteLine("Put object succeeded, {0}", objectName);
}
} catch (Exception e) {
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
var metadata = client.GetObjectMetadata(bucketName, objectName);
string storageClass = metadata.HttpMetadata["x-oss-storage-class"] as string;
if (storageClass ! = "Archive")
{
    Console.WriteLine("StorageClass is {0}", storageClass);
    return;
}
// Restore the object.
RestoreObjectResult result = client.RestoreObject(bucketName, objectName);
Console.WriteLine("RestoreObject result HttpStatusCode : {0}", result.HttpStatusCode);
if (result.HttpStatusCode ! = HttpStatusCode.Accepted)
{
    throw new OssException(result.RequestId + ", " + result.HttpStatusCode + " ,");
}
while (maxWaitTimeInSeconds > 0)
{
    var meta = client.GetObjectMetadata(bucketName, objectName);
    string restoreStatus = meta.HttpMetadata["x-oss-restore"] as string;
    if (restoreStatus ! = null && restoreStatus.StartsWith("ongoing-request=\"false\"",
StringComparison.InvariantCultureIgnoreCase))
    {
        ブレーク;
    }
    Thread.sleep(1000);
    maxWaitTimeInSeconds--;
}
if (maxWaitTimeInSeconds == 0)
{
    Console.WriteLine("RestoreObject is timeout. ");
    throw new TimeoutException();
}
else
{
    Console.WriteLine("RestoreObject is successful. ");
}
```

アーカイブストレージクラスの詳細については、「ストレージクラスの概要」をご参照ください。 関連するステータスコードの詳細については、「RestoreObject」をご参照ください。

8.6.9 シンボリックリンクの管理

シンボリックリンクは、Windows で使用されるショートカットのように、オブジェクトにマップする特殊なオブジェクトです。 シンボリックリンクには、ユーザー定義のオブジェクトメタを構成できます。 シンボリックリンクを取得するには、読み取り許可を持っている必要があります。

次のコードを実行して、シンボリックリンクを作成して取得します。

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var targetObjectName = "<yourTargetObjectName>";
var symlinkObjectName = "<yourSymlinkObjectName>";
var objectContent = "More than just cloud." ;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Upload the target file.
    byte[] binaryData = Encoding.ASCII.GetBytes(objectContent);
    MemoryStream requestContent = new MemoryStream(binaryData);
    client.PutObject(bucketName, targetObjectName, requestContent);
    // Create a symbolic link.
    client.CreateSymlink(bucketName, symlinkObjectName, targetObjectName);
    // Obtain a symbolic link.
    var ossSymlink = client.GetSymlink(bucketName, symlinkObjectName);
    Console.WriteLine("Target object is {0}", ossSymlink.Target);
}
} catch (Exception e) {
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

シンボリックリンク作成についての詳細は、「PutSymlink」をご参照ください。

シンボリックリンク取得についての詳細は、「GetSymlink」をご参照ください。

8.7 画像処理

Image Processing (IMG) は、大規模で安全、低コストで、信頼性の高い画像処理サービスです。ソース画像が OSS にアップロードされると、簡単な RESTful API を通して、いつでも、どこからでも、どのインターネットデバイス上からでも画像を処理できます。

IMG の詳細については、「 Image Processing 」をご参照ください。

基本機能

OSS は以下の IMG 機能を提供します。

- ドミナントトーンの取得
- フォーマット変換
- 画像形式の変換
- 内接円の調整
- 向きの調整
- 輝度

- 透かしの追加: 画像、テキスト、および画像とテキストの透かしを別の画像に追加します。
- 画像処理
- 画像処理アクセスルール: 複数の IMG 関数を呼び出します。

使用方法

IMG は標準の HTTP GET を使用します。 URL のクエリ文字列に IMG パラメータ を設定します。

イメージオブジェクトの ACL が非公開読み書きの場合、許可されたユーザーだけがアクセスを許可されます。

- 匿名アクセス

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
  ```

パラメーター	説明
bucket	バケットの名前を指定します。
endpoint	リージョンへのアクセスに使用されるエンドポイントを指定します。
object	画像オブジェクトの名前を指定します。
image	IMG の予約済み識別子を指定します。
action	拡大縮小、トリミング、回転など、画像に対する操作を指定します。

パラメーター	説明
param	画像に対する操作を示すパラメータを指定します。

- 基本操作

  たとえば、画像を 100 ピクセルの幅に拡大縮小します。 比率に基づいて高さを調整します。

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- カスタマイズされた画像スタイル

  処理済みイメージにアクセスするには、第 3 レベルドメインで次の形式を使用します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  ■ style: カスタマイズされた画像スタイルの予約済み識別子を指定します。

  ■ yourStyleName: カスタム画像スタイルの名前を指定します。 OSS コンソールで作成されたルールで指定された名前です。

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- カスケード処理

  カスケード処理では、画像に対して複数の操作を順番に実行します。

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  例:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- HTTPS アクセス

  IMG は HTTPS を通したアクセスをサポートしています。 例:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

• 許可されたアクセス

許可されたアクセスにより、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

次のコードを実行して IMG の署名付き URL を生成します。

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
   var process = "image/resize,m_fixed,w_100,h_100";
   var req = new GeneratePresignedUriRequest(bucketName, objectName, SignHttpMe
thod.Get)
   {
      Expiration = DateTime.Now.AddHours(1),
      Process = process
   };
   // Generat a signed URI.
   var uri = client.GeneratePresignedUri(req);
   Console.WriteLine("Generate Presigned Uri:{0} with process:{1} succeeded ", uri,
process);
}
catch (OssException ex)
{
   Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{2}\
tHostID:{3}",
      ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
   Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

- SDK を使用したアクセス

  SDK を使用して、画像オブジェクトにアクセスして処理します。

  SDK を使用すると、画像スタイル、HTTPS アクセス、およびカスケード処理をカスタマイズできます。

  - 基本操作

    次のコードを実行して画像に対して基本操作を実行します。

    ```
    using System;
    using System.IO;
    using Aliyun.OSS;
    using Aliyun.OSS.Common;
    using Aliyun.OSS.Util;
    namespace ImageProcess
    {
       class Program
    ```

```
  {
    static void Main(string[] args)
    {
      Program.ImageProcess();
      Console.ReadKey();
    }
    public static void ImageProcess()
    {
      var endpoint = "<yourEndpoint>";
      var accessKeyId = "<yourAccessKeyId>";
      var accessKeySecret = "<yourAccessKeySecret>";
      var bucketName = "<yourBucketName>";
      var objectName = "<yourObjectName>";
      var localImageFilename = "<yourLocalImageFilename>";
      var downloadDir = "<yourDownloadDir>";
      // Create an OSSClient instance.
      var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
      try
      {
        client.PutObject(bucketName, objectName, localImageFilename);
        // Scale the image.
        var process = "image/resize,m_fixed,w_100,h_100";
        var ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-resize.jpg", ossObject.Content);
        // Crop the image.
        process = "image/crop,w_100,h_100,x_100,y_100,r_1";
        ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-crop.jpg", ossObject.Content);
        // Rotate the image.
        process = "image/rotate,90";
        ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-rotate.jpg", ossObject.Content);
        // Sharpen the image.
        process = "image/sharpen,100";
        ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-sharpen.jpg", ossObject.Content);
        // Add watermarks.
        process = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ";
        ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-watermark.jpg", ossObject.Content);
        // Convert the image format.
        process = "image/format,png";
        ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-format.png", ossObject.Content);
        // Obtain image information.
        process = "image/info";
        ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-info.txt", ossObject.Content);
        Console.WriteLine("Get Object:{0} with process:{1} succeeded ",
objectName, process);
      }
      catch (OssException ex)
      {
        Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{
2}\tHostID:{3}",
          ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
      }
```

```
        } catch (Exception e) {
        {
            Console.WriteLine("Failed with error info: {0}", ex.Message);
        }
    }
    private static void WriteToFile(string filePath, Stream stream)
    {
        using (var requestStream = stream)
        {
            using (var fs = File.Open(filePath, FileMode.OpenOrCreate))
            {
                IoUtils.WriteTo(stream, fs);
            }
        }
    }
  }
}
```

- 画像スタイルのカスタマイズ

  次のコードを使用して画像スタイルをカスタマイズします。

```
using System;
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
using Aliyun.OSS.Util;
namespace ImageProcessCustom
{
  class Program
  {
    static void Main(string[] args)
    {
      Program.ImageProcessCustomStyle();
      Console.ReadKey();
    }
    public static void ImageProcessCustomStyle()
    {
      var endpoint = "<yourEndpoint>";
      var accessKeyId = "<yourAccessKeyId>";
      var accessKeySecret = "<yourAccessKeySecret>";
      var bucketName = "<yourBucketName>";
      var objectName = "<yourObjectName>";
      var localImageFilename = "<yourLocalImageFilename>";
      var downloadDir = "<yourDownloadDir>";
      // Create an OSSClient instance.
      var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
      try
      {
        client.PutObject(bucketName, objectName, localImageFilename);
        // Customize the image style: "style/<yourCustomStyleName>".
        var process = "style/oss-pic-style-w-100";
        var ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-style.jpg", ossObject.Content);
        Console.WriteLine("Get Object:{0} with process:{1} succeeded ",
objectName, process);
      }
      catch (OssException ex)
      {
        Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{
2}\tHostID:{3}",
          ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine("Failed with error info: {0}", ex.Message);
        }
    }
    private static void WriteToFile(string filePath, Stream stream)
    {
        using (var requestStream = stream)
        {
            using (var fs = File.Open(filePath, FileMode.OpenOrCreate))
            {
                IoUtils.WriteTo(stream, fs);
            }
        }
    }
  }
}
```

- カスケード処理

  画像に対してカスケード処理を実行するには、次のコードを使用します。

```
using System;
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
using Aliyun.OSS.Util;
namespace ImageProcessCascade
{
  class Program
  {
    static void Main(string[] args)
    {
      Program.ImageProcessCascade();
      Console.ReadKey();
    }
    public static void ImageProcessCascade()
    {
      var endpoint = "<yourEndpoint>";
      var accessKeyId = "<yourAccessKeyId>";
      var accessKeySecret = "<yourAccessKeySecret>";
      var bucketName = "<yourBucketName>";
      var objectName = "<yourObjectName>";
      var localImageFilename = "<yourLocalImageFilename>";
      var downloadDir = "<yourDownloadDir>";
      // Create an OSSClient instance.
      var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
      try
      {
        client.PutObject(bucketName, objectName, localImageFilename);
        // Perform cascade operations.
        var process = "image/resize,m_fixed,w_100,h_100/rotate,90";
        var ossObject = client.GetObject(new GetObjectRequest(bucketName,
objectName, process));
        WriteToFile(downloadDir + "/sample-style.jpg", ossObject.Content);
        Console.WriteLine("Get Object:{0} with process:{1} succeeded ",
objectName, process);
      }
      catch (OssException ex)
      {
        Console.WriteLine("Failed with error code: {0}; Error info: {1}. \nRequestID:{
2}\tHostID:{3}",
```

```
                ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}", ex.Message);
            }
        }
        private static void WriteToFile(string filePath, Stream stream)
        {
            using (var requestStream = stream)
            {
                using (var fs = File.Open(filePath, FileMode.OpenOrCreate))
                {
                    IoUtils.WriteTo(stream, fs);
                }
            }
        }
    }
}
```

IMG ツール

IMG ビューアー (ImageStyleViever) を使用すると、IMG の結果を直接表示できます。

8.8 例外処理

OSS.NET SDK には、ClientException と OSSException の 2 種類の例外があります。 ともに RuntimeException のプロパティを共有します。

ClientException

ClientException 例外は、クライアントがリクエストを送信しようとしたとき、およびデータが OSS に送信されたときに発生します。 たとえば、リクエストが送信されると、利用できないネットワーク接続が原因で ClientException が発生することがあります。 I/O 例外のため、ファイルのアップロード中に ClientException が発生することがあります。

OSSException

OSSException は、サーバー例外を示します。 サーバーエラーメッセージが解析されることが原因で、例外が発生します。 OSSException には、OSS から返されたエラーコードと情報が含まれます。 OSSException は問題を見つけて処理するために使用されます。

次の表に、一般的な OSSException の情報を示します。

パラメーター	説明
Code	OSS から返されたエラーコードを特定します。
Message	OSS から返される詳細なエラー情報を特定します。

パラメーター	説明
RequestId	UUID を特定します。 UUID は固有のものであり、リクエストを識別するために使用されます。 問題が解決しない場合は、OSS 開発者に "RequestId" を送り、問い合わせます。
HostId	OSS クラスターを識別します。 "HostId" の値が、リクエストで使用されているホスト (バケットにアクセスするためのエンドポイント) と一致していることを確認してください。

OSS エラーコード

エラーコード	説明
AccessDenied	アクセスが拒否されました。
BucketAlreadyExists	バケットが既に存在します。
BucketNotEmpty	バケットが空ではありません。
EntityTooLarge	エンティティサイズが上限を超えています。
EntityTooSmall	エンティティサイズが下限を下回っています。
FileGroupTooLarge	ファイルグループの合計サイズが上限を超えています。
FilePartNotExist	そのようなオブジェクトパーツは存在しません。
FilePartStale	オブジェクトパーツは有効期限切れです。
InvalidArgument	パラメーターの形式が無効です。
InvalidAccessKeyId	そのような AccessKeyId は存在しません。
InvalidBucketName	バケット名が無効です。
InvalidDigest	ダイジェストが無効です。
InvalidObjectName	オブジェクト名が無効です。
InvalidPart	パーツが無効です。
InvalidPartOrder	パーツの順序が無効です。
InvalidTargetBucketForLogging	ログファイルを格納するバケットが無効です。
InternalError	内部 OSS エラーです。
MalformedXML	XML 形式が無効です。
MethodNotAllowed	メソッドがサポートされていません。

エラーコード	説明
MissingArgument	パラメーターが設定されていません。
MissingContentLength	コンテンツの長さが設定されていません。
NoSuchBucket	そのようなバケットは存在しません。
NoSuchKey	そのようなオブジェクトは存在しません。
NoSuchUpload	そのようなアップロード ID は存在しません。
NotImplemented	メソッドが実装されていません。
PreconditionFailed	事前条件エラーです。
RequestTimeTooSkewed	OSSClient に設定されている現地時間が、OSS サーバーに設定されている時間から 15 分以上ずれています。
RequestTimeout	リクエストがタイムアウトされました。
SignatureDoesNotMatch	署名が不一致です。
TooManyBuckets	バケット数が上限を超えています。