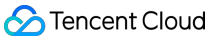
Cloud Object Storage

セキュリティとコンプライアンス

製品ドキュメント

著作権声明

商標に関する声明



サービス声明

本ドキュメントは、お客様にTencent Cloudの全部また一部の製品・サービスの概要をご紹介することを目的としておりますが、一部の製品・サービス内容は変更される可能性があります。お客様がご購入されるTencent Cloud製品・サービスの種類やサービス基準などは、お客様とTencent Cloudとの間の締結された商業契約に基づきます。別段の合意がない限り、Tencent Cloudは本ドキュメントの内容に関して、明示または黙示の一切保証もしません。

カタログ:

セキュリティとコンプライアンス
データ災害復帰
バージョン管理
バージョン管理の概要

最終更新日：：2024-06-26 10:57:13

概要

バージョン管理は同一のバケット内に同一のオブジェクトの複数のバージョンを保存する場合に使用します。例えば、1つのバケット内に、オブジェクトキーは同じpicture.jpgでも、バージョンIDが100000、100101、120002のように異なる複数のオブジェクトを保存することができます。ユーザーがあるバケットでバージョン管理機能を有効にすると、バケット内に保存したオブジェクトをバージョンIDに基づいて照会、削除または復元できるようになります。ユーザーが誤って削除したデータや、アプリケーションプログラムの障害によって消失したデータの回復に役立ちます。例えば、ユーザーがバージョン管理されたオブジェクトの削除操作を行う場合は、次のようになります。

- オブジェクトを削除したい（完全削除ではない）場合、COSは削除されるオブジェクトに削除タグを挿入し、このタグが現在のオブジェクトのバージョンとなります。この削除タグに基づいて以前のバージョンを復元することができます。
- オブジェクトを置き換えたい場合、COSは新たにアップロードされたオブジェクトに新しいバージョンIDを挿入します。バージョンIDに基づいて、置き換える前のオブジェクトを復元することも引き続き可能です。

バージョン管理の状態

バケットには3種類のバージョン管理状態があります。バージョン管理を有効にしていない状態、バージョン管理を有効にしている状態、バージョン管理の一時停止状態です。

- バージョン管理を有効にしていない状態：バケットのデフォルトの初期状態であり、このときバージョン管理機能はオフになっています。
- バージョン管理を有効にしている状態：バケットのバージョン管理機能をオンにすることを指し、このときバージョン管理は有効な状態になっています。バージョン管理状態はこのバケット内のすべてのオブジェクトに適用されます。バケットのバージョン管理を最初に有効にした時点から、このバケットに新たにアップロードされるオブジェクトには一意のバージョンIDが付与されます。
- バージョン管理の一時停止状態：バケットのバージョン管理をオンにした状態から一時停止状態に変更することを指します（バージョン管理を有効にしていない状態に戻すことはできません）。この後にバケットにアップロードされるオブジェクトについては、バージョン管理されたオブジェクトが保存されなくなります。

> ⚠ 注意：

1. 一度バージョン管理を有効にしたバケットは、バージョン管理を有効にしていない状態（初期状態）に戻すことはできません。ただし、このバケットのバージョン管理を一時停止することはでき、その後に新たにアップロードされるオブジェクトでは複数のバージョンが生成されなくなります。
2. バージョン管理を有効にする前にバケット内に保存されたオブジェクトのバージョンIDは、すべてnullとなっています。
3. バージョン管理を有効にする、または一時停止すると、COSがこれらのオブジェクトを処理する際のリクエスト方式が変更されます。オブジェクト自体は変更されません。
4. バケットのバージョン管理を一時停止できるのはルートアカウントと権限を持つサブアカウントのみです。

バージョン管理状態下のオブジェクトの管理

バケットがどのバージョン管理状態にある場合でも、その状態のバケット内のオブジェクトに対し、アップロード、照会および削除操作を行うことができます。バージョン管理を有効にしていない状態を除き、バージョン管理を有効にしている状態およびバージョン管理の一時停止状態であれば、バケット内のオブジェクトの照会および削除操作は、バージョンIDを指定してもしなくても行うことができます。

- バージョン管理を有効にしていない状態：オブジェクトのアップロード、照会および削除などの操作方法に変更はありません。詳細については、オブジェクト管理ディレクトリ下のドキュメントをご参照ください。
- バージョン管理を有効にしている状態およびバージョン管理の一時停止状態：オブジェクトのアップロード、照会および削除などの操作方法が従来の方法と異なる点は、バージョンIDが導入されている点です。オブジェクトの削除操作の実行には「削除タグ」の概念も加わっています。

バージョン管理を有効にしている状態のオブジェクトの管理

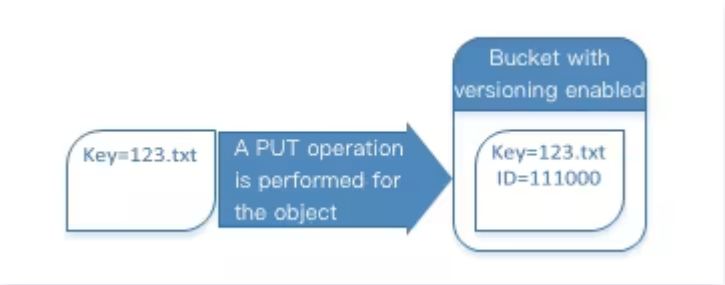
バケットバージョン管理を有効にする前にすでにバケット内に保存されていたオブジェクトについては、そのバージョンIDはnullとなります。バージョン管理を有効にした後も、バケット内の既存のオブジェクトは変更されず、既存オブジェクトに対するCOSの処理方式（リクエスト方式など）だけが変更されます。この時点で、新たにアップロードされた同名のオブジェクトは異なるバージョンとして同一のバケット内に存在することになります。バージョン管理を有効にしたバケット内でオブジェクトがどのように管理されるかについて、次にご説明します。

> ⚠ 注意：
> バージョン管理を有効にしていないバケットと、有効にしているバケットでは、ユーザーのオブジェクトのアップロード方法は同じですが、バージョンIDが異なります。バージョン管理を有効にしている場合、COSはオブジェクトに特定のバージョンIDを割り当てますが、バージョン管理を有効にしていない場合、バージョンIDは常にnullとなります。

オブジェクトのアップロード

バケットのバージョン管理を有効にすると、ユーザーがPUT、POSTまたはCOPY操作を実行した際、COSがこのバケットに保存するオブジェクトには一意のバージョンIDが自動的に追加されます。

下図のように、バージョン管理を有効にしたバケットにオブジェクトをアップロードすると、COSはそのオブジェクトに一意のバージョンIDを追加します。



バージョン管理オブジェクトのリストアップ

COSはバケットにバインドしたversionsパラメータにオブジェクトのバージョン情報を保存します。COSは保存時刻の順序に従ってオブジェクトのバージョンを返します。その際、直近のバージョンが最初に返されます。

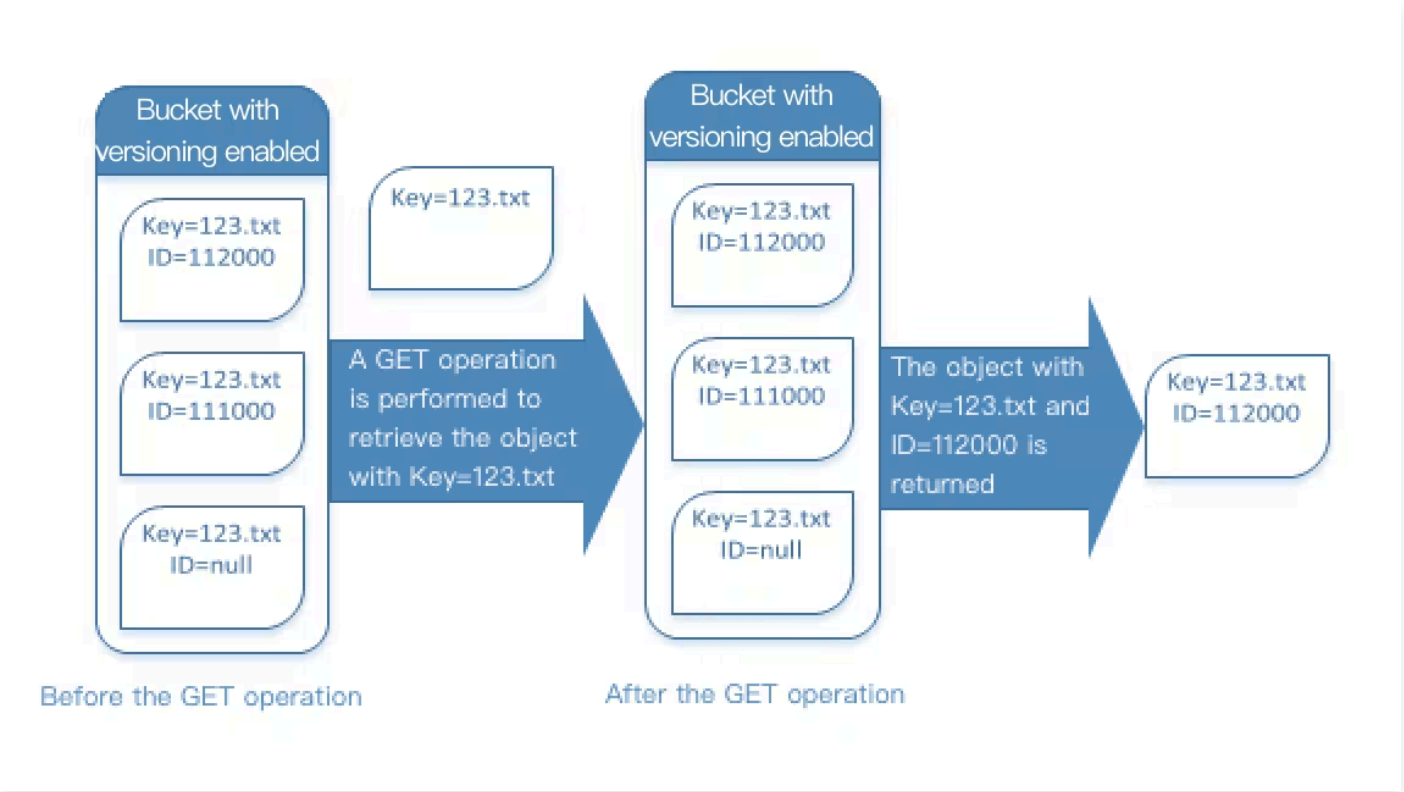
特定のオブジェクトの全バージョン照会

次のプロセスによって、versionsパラメータおよびprefixリクエストパラメータを使用して、あるオブジェクトの全バージョンを照会できます。prefixに関するその他の情報については、GET Bucket Object versionsのドキュメントをご参照ください。
あるオブジェクトの全バージョンを照会する場合の、リクエストの例は次のとおりです。

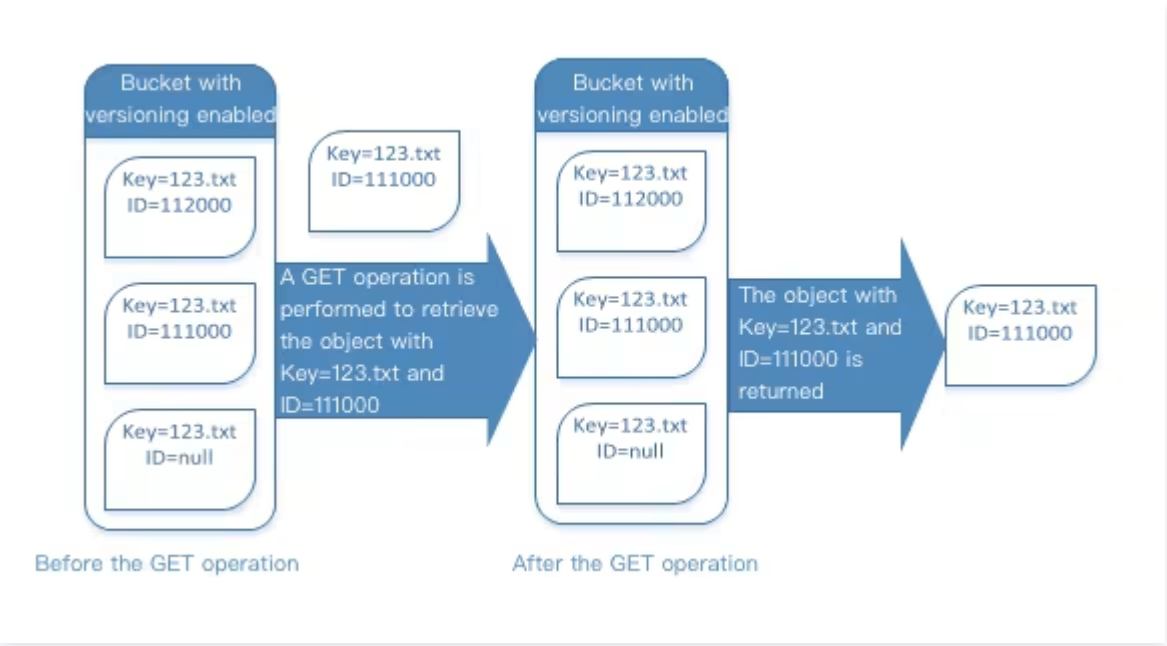
```
GET /?versions&prefix=ObjectKey HTTP/1.1
```

データのメタバージョンの照会

ユーザーがGETリクエストを使用する際にバージョンIDを指定しなければ、オブジェクトの現在のバージョンを照会します。下図のように、GETリクエストには123.txtオブジェクトの現在のバージョン（直近のバージョン）が返されます。



ユーザーがGETリクエストを使用する際にバージョンIDを指定した場合は、指定したバージョンIDのオブジェクトを照会します。下図のように、GET versionIdリクエストによって指定したバージョン（現在のバージョンでも可）のオブジェクトを照会します。



オブジェクトバージョンのメタデータの照会

オブジェクトの内容ではなく、メタデータのみを照会したい場合は、HEAD操作を使用することができます。デフォルトでは最新バージョンのメタデータを取得します。指定したオブジェクトのバージョンのメタデータを照会

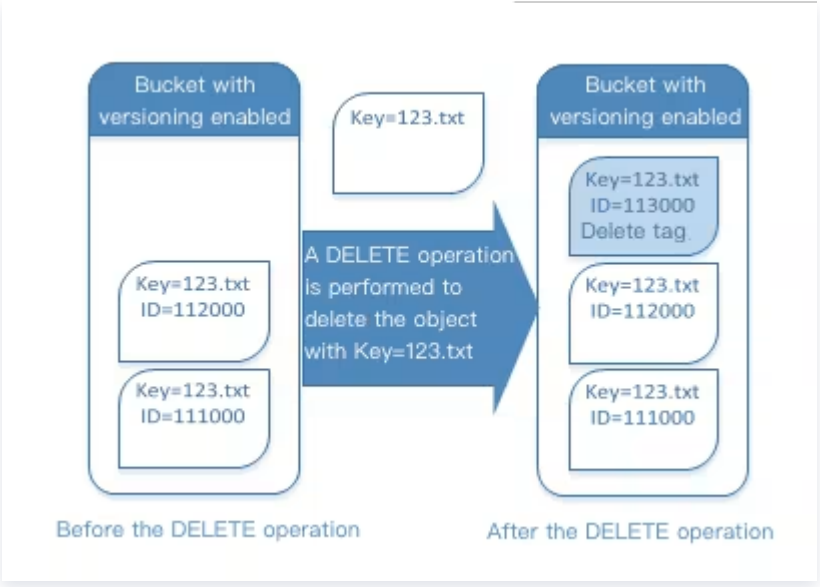
したい場合は、リクエストの送信時にバージョンIDを指定します。
指定したバージョンのオブジェクトのメタデータを照会する手順は次のとおりです。

- versionIdを照会するオブジェクトメタデータのバージョンIDに設定します。
- 指定したversionIdのHEAD操作リクエストを送信します。

オブジェクトの削除

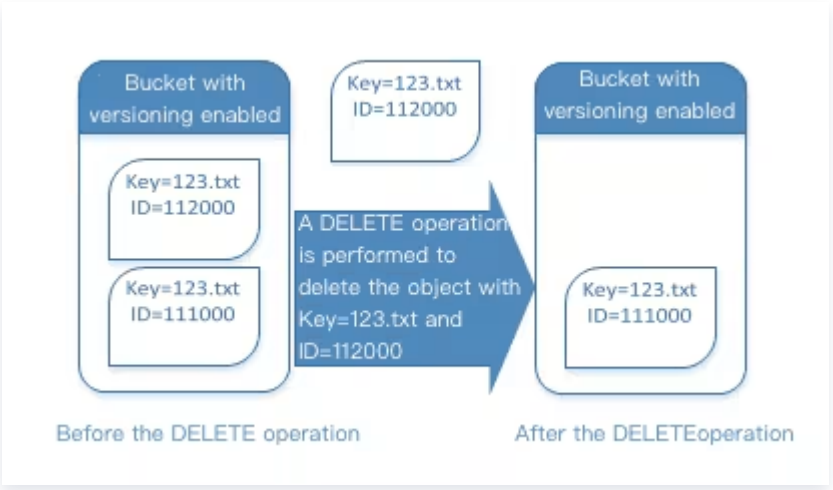
必要に応じて、不要なオブジェクトのバージョンを随時削除することができます。ユーザーがバージョン管理を有効にしている状態でDELETEリクエストを使用するケースには次の2つがあります。

1. ユーザーがバージョンIDを指定せず、通常のDELETE操作を実行する場合。
   この操作のケースは削除されたオブジェクトを「ごみ箱」に入れる場合に似ていますが、オブジェクトを完全に消去してはおらず、その後ユーザーが必要とする場合はデータを復元できます。
   下図のように、ユーザーがDELETE操作の際にバージョンIDを指定しなかった場合、実際にはKey=123.txtのオブジェクトは削除されず、新しい削除タグが挿入され、新しいバージョンIDが追加されます。



> ⚠ 注意：
> COSはバケット内で、削除されるオブジェクトに新しいバージョンIDを持つ削除タグを挿入し、この削除タグが削除されるオブジェクトの現在のバージョンとなります。この削除タグのあるオブジェクトに対しGET操作の実行を試すと、COSはこのオブジェクトが存在しないと認識し、404エラーを返します。

2. ユーザーがバージョンIDを指定して、オブジェクトバージョンの削除操作を実行した場合、このケースではバージョン管理されたオブジェクトを永久に削除することができます。



初期バージョンの復元

バージョン管理によってオブジェクトの初期バージョンを復元することができます。この操作を実行するには2つの方法があります。

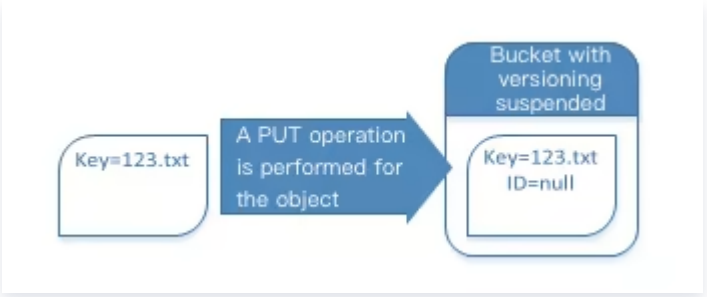
1. オブジェクトの初期バージョンを同一のバケットにコピーする
   コピーしたオブジェクトはそのオブジェクトの現在のバージョンとなり、なおかつオブジェクトのすべてのバージョンが維持されます。
2. オブジェクトの現在のバージョンを永久に削除する
   オブジェクトの現在のバージョンを削除すると、実際には1つ前のバージョンがオブジェクトの現在のバージョンに置き換わります。

バージョン管理が一時停止状態にあるオブジェクトの管理

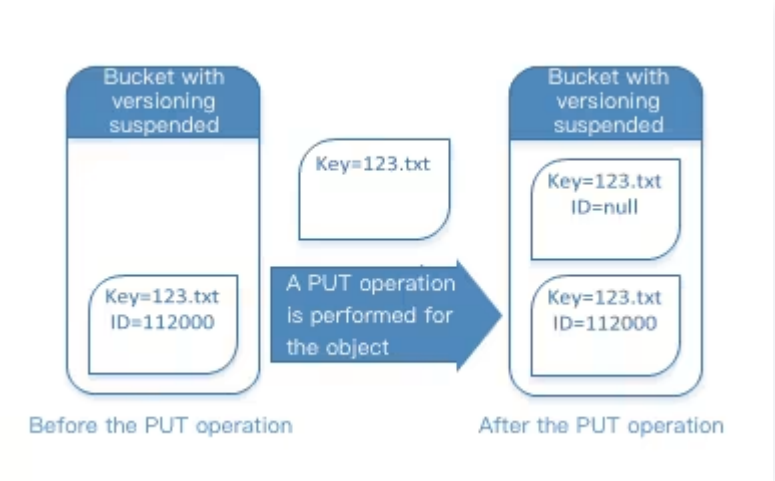
バージョン管理を一時停止した場合、バケット内の既存のオブジェクトは変更されません。変更されるのはCOSがそれ以降のリクエストにおいてオブジェクトを処理する方式です。バージョン管理を一時停止しているバケット内でオブジェクトがどのように管理されるかについて、次にご説明します。

オブジェクトのアップロード

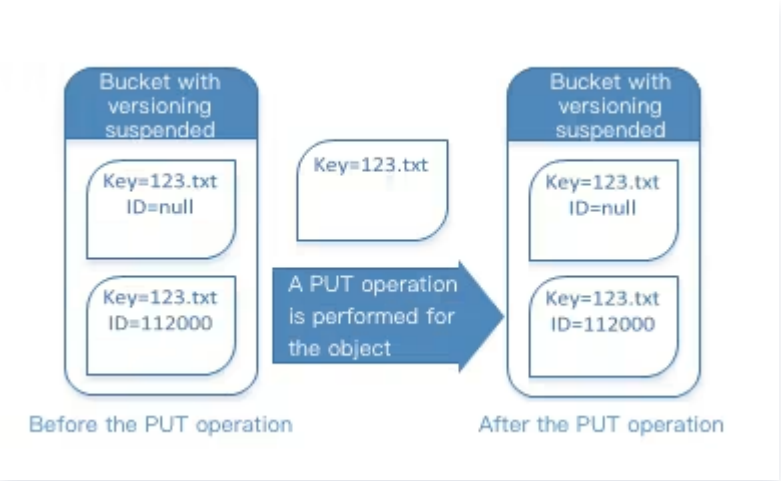
バケット上でバージョン管理を一時停止すると、ユーザーがPUT、POSTまたはCOPY操作を実行した際、COSはこのバケット内に保存されるオブジェクトに、バージョンIDをnullとして自動的に追加します。下図に示します。



バケット内にバージョン管理を行っているオブジェクトが存在する場合、バケットにアップロードされたオブジェクトが現在のバージョンとなり、バージョンIDはnullとなります。下図に示します。



バケット内に空のバージョンがすでに存在する場合、この空のバージョンは上書きされ、それまでのオブジェクトの内容もそれに応じて置き換えられます。下図に示します。



データのメタバージョンの照会

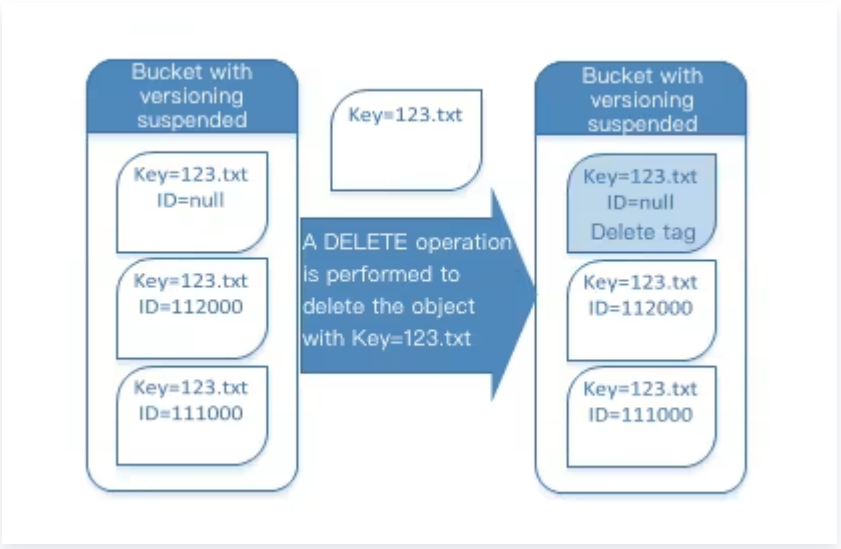
バージョン管理を一時停止したバケットで、ユーザーがGET Objectリクエストを送信すると、オブジェクトの現在のバージョンが返されます。

オブジェクトの削除

バージョン管理を一時停止した状態でDELETEリクエストを実行すると、次のようになります。

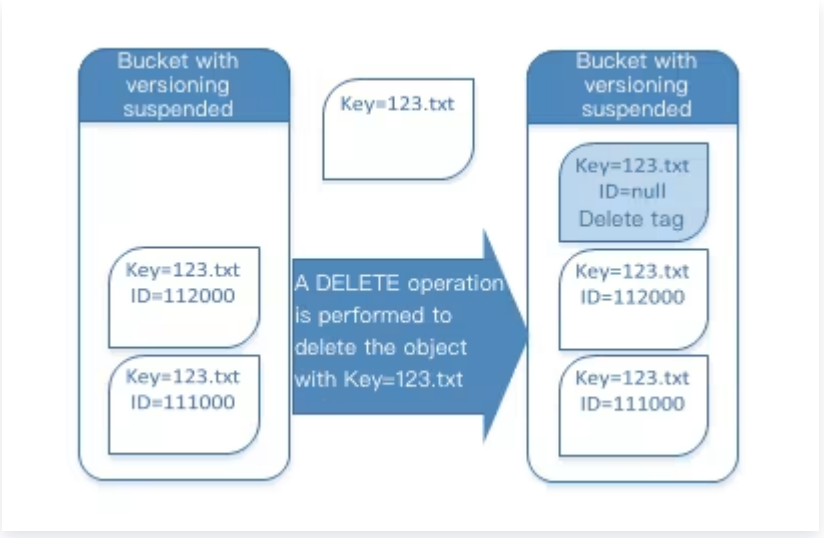
- バケット内に空のバージョンのオブジェクトが存在する場合は、バージョンIDがnullのオブジェクトを削除します。
  下図のように、ユーザーが通常のDELETE操作を実行した際、COSは空のバージョンのオブジェクトに削除タ

グを挿入します。



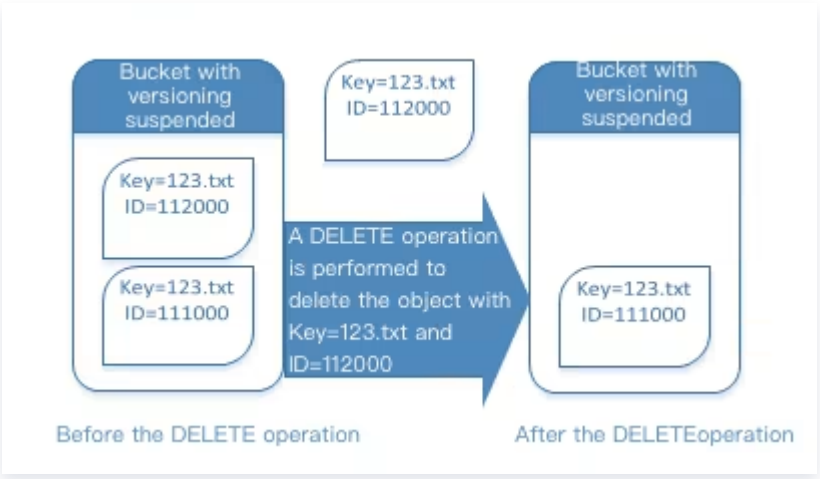
> ⚠ 注意：
> 削除タグには内容が存在しません。削除タグが空のバージョンに置き換わった時点で、空のバージョンの元の内容は失われます。

- バケット内に空のバージョンのオブジェクトが存在しない場合は、バケット内に新たに削除タグが追加されます。
  下図のように、バケットに空のバージョンが存在しない場合、ユーザーがDELETE操作を実行してもいかなる内容も削除されず、COSが削除タグを挿入するだけとなります。



- バージョン管理を一時停止しているバケットであっても、ルートアカウントであれば指定したバージョンを永久に削除することができます。

下図のように、指定したオブジェクトのバージョンを削除すると、そのオブジェクトは永久に削除されます。



注意:
指定したオブジェクトのバージョンを削除できるのは、ルートアカウントまたはルートアカウントから権限を付与されたアカウントのみです。

タグの削除

最終更新日：：2024-06-26 10:57:13

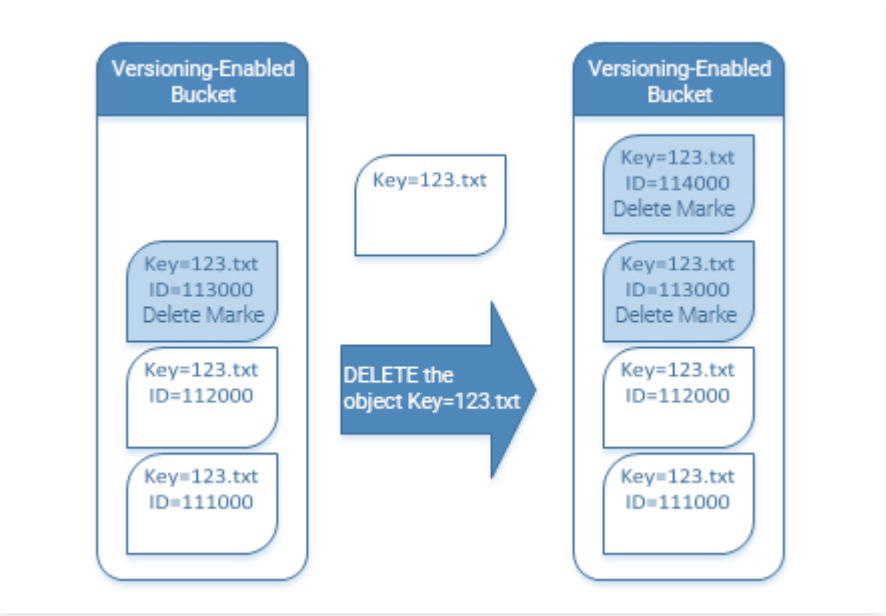
概要

削除マーカーはバージョン管理されているオブジェクトに用いられます。削除マーカーはCOS内で「オブジェクトがすでに削除されている」ことを示すマーカーとみなすことができます。削除マーカーにはオブジェクトと同様に、オブジェクトキー（Key）とバージョンIDがあります。その違いは次のいくつかの点にあります。

- 削除マーカーは内容が空です。
- 削除マーカーにはACL値が存在しません。
- 削除マーカーがGETリクエストを実行するとエラーコード404が返されます。
- 削除マーカーはDELETE操作のみサポートされます（ルートアカウントによるリクエスト送信が必要）。

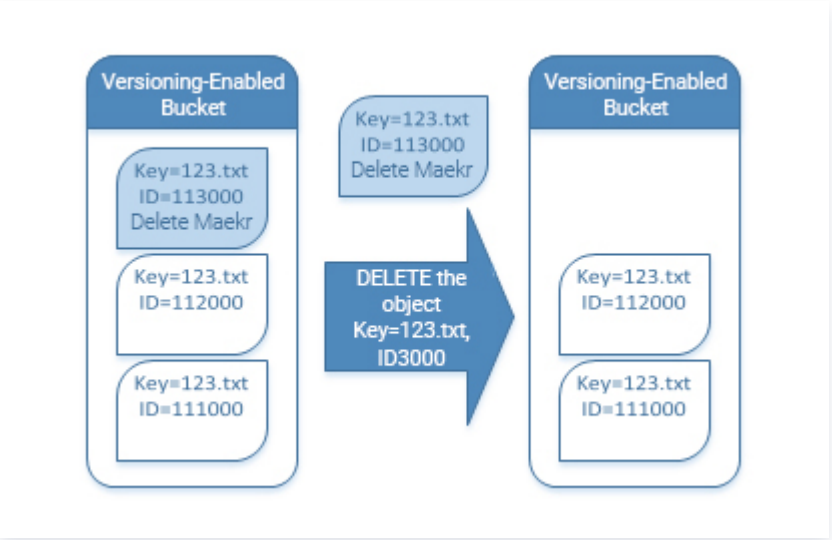
「削除マーカー」の削除

ユーザーが「削除マーカー」を削除したい場合は、DELETE Object versionIdリクエストでそのバージョンIDを指定することで、「削除マーカー」を完全に削除することができます。削除マーカーのバージョンIDを指定せず、削除マーカーに対しDELETEリクエストを送信した場合、COSはその削除マーカーを削除せず、新たな削除マーカーを挿入します。
下図のように、削除マーカーに対し通常のDELETEリクエストを実行しても、いかなる内容も削除されず、バケット内には新たな削除マーカーが追加されます。



バージョン管理を有効にしているバケットでは、新たに追加される削除マーカーは一意のバージョンIDを有しています。このため、1つのバケット内にある同一のオブジェクトが複数の削除マーカーを持つ可能性があります。
「削除マーカー」を完全に削除したい場合は、DELETE Object versionIdリクエストに必ずそのバージョンIDを含めなければなりません。

下図のように、DELETE Object versionIdリクエストを実行して「削除マーカー」を完全に削除します。



> 説明:
> 「削除マーカー」を削除できるのは、ルートアカウントによる権限承認を受けて `DeleteObject` 操作を行った場合のみです。

「削除マーカー」を完全に削除する手順は次のとおりです。

1. versionIdを削除マーカーのバージョンIDに設定します。
2. DELETE Object versionIdリクエストを送信します。

バージョン管理の使用
バージョン管理の設定

最終更新日：：2025-12-08 16:27:55

適用シナリオ

バージョニング機能を利用することで、バケット内にオブジェクトの複数バージョンを保存し、指定バージョンの検索、削除、復元が可能です。
バージョニングの詳細については、バージョニングの概要 ドキュメントをご参照ください。

> ⚠ 注意：
> ルートアカウントおよび権限あるサブアカウントのみがバケットのバージョニング状態を設定できます。

使用方法

COSコンソールの使用

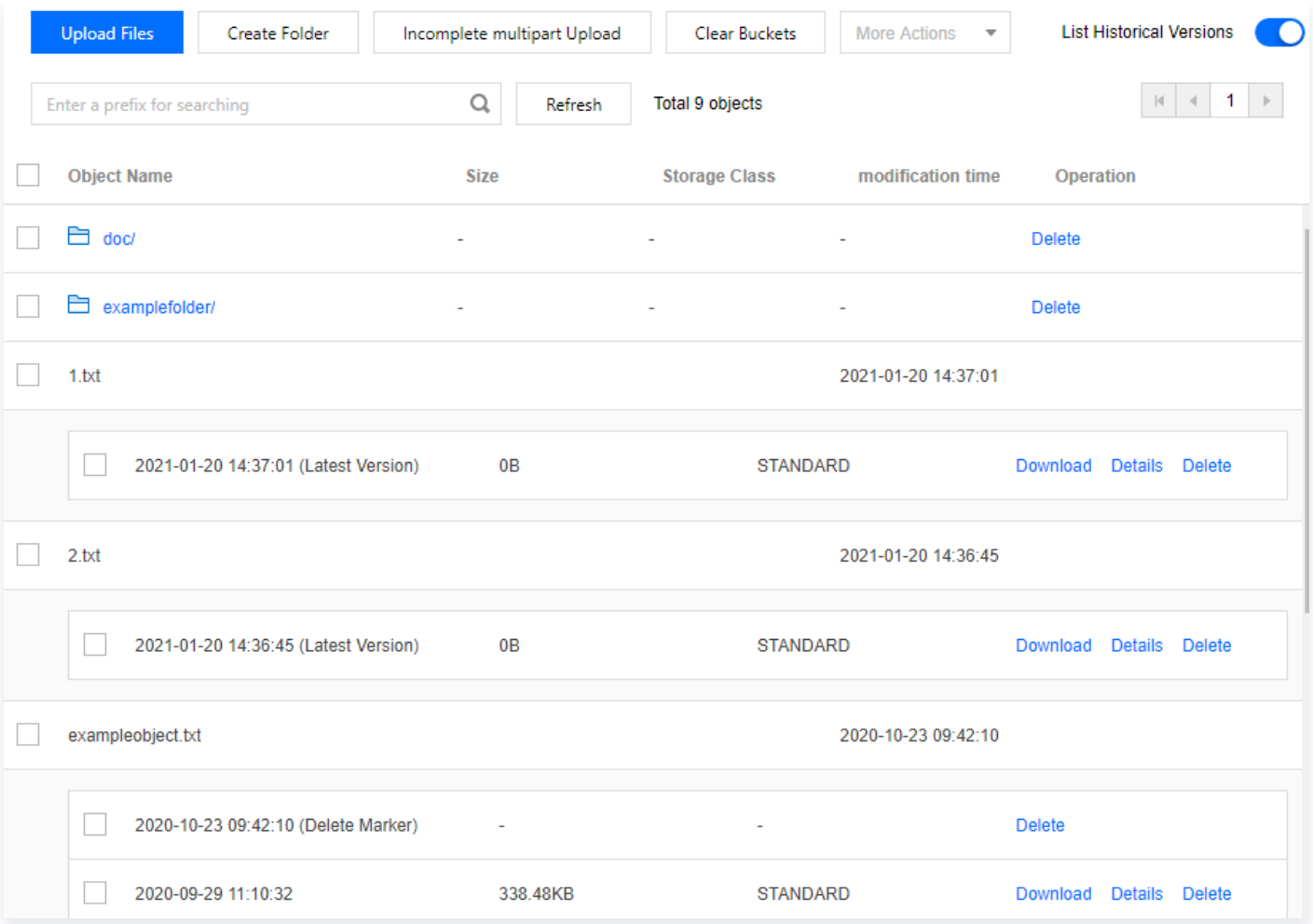
1. COS コンソールにログインします。
2. 左側のナビゲーションメニューで、バケットリストをクリックし、バケットリストのページに移動します。
3. バージョニングを設定したいバケットをクリックし、バケットの詳細ページに移動します。
4. フォールトトレランスとディザスタリカバリ管理 > バージョニングを選択し、バージョニングの設定項目を見つけて、現在のステータスを「有効」に変更し、保存をクリックします。確認のダイアログで確定をクリックすると、バージョニングが有効になります。

> ⚠ 注意：
> - バケットでバージョニングを有効にすると、未設定状態（初期状態）に戻すことはできません。ただし、バケットのバージョニングを一時停止（現在のステータスのスイッチを「有効」から「無効」に変更）することが可能です。これにより、以降のオブジェクトの上書き、削除、または変更操作では、過去のバージョンは生成されなくなります。

- バージョニングを有効にすると、オブジェクトの過去のバージョンがストレージ容量を消費するため、これらの過去バージョンのオブジェクトにもストレージ容量料金が発生します。古いオブジェクトのバージョンを長期間保持したくない場合は、過去バージョンの定期的なクリーンアップを有効にすることをお勧めします。このショートカットを使用すると、過去バージョンを定期的に削除するライフサイクルルールが作成されます（デフォルトではファイル変更3日後に実行）。ルール名は `RegularlyClean_HistoricalVersions_タイムスタンプ` の形式で表示されます。ライフサイクルページでこのルールを確認できます。また、ビジネス要件に応じて、より詳細なライフサイクルルールをカスタマイズすることも可能です。詳細はライフサイクルの設定をご参照ください。

5. バージョニングを有効にすると、ファイルリスト画面に移動し、過去バージョンを一覧表示スイッチをオンにすると、同名ファイルのすべてのバージョンを表示・管理できます。
   - 同名のファイルをこのバケットにアップロードすると、異なる時点でアップロードされた同名ファイルを確認できます。
   - ファイルを削除する際にバージョンIDを指定しない場合、削除を実行すると、現在のバージョン（最新バージョン）のオブジェクトに「削除マーク」が挿入されます。この削除マークは「オブジェクトが削除された」ことを示し、過去のバージョンを一覧表示すると、ファイルの削除記録として表示されますが、削除前の同名ファイルは引き続き保持されます。バージョンIDや変更時間に基づいて過去のファイルを検索して復元、ダウンロード、削除などの操作を行うことができます。

     説明：
     - 特定の過去バージョンの削除について、バージョンIDを指定して検索し、完全に削除します。コンソールのバケットのファイルリストで、過去バージョンを一覧表示した後、指定バージョンのファイルを削除します。
     - オブジェクトの過去バージョンの数に制限はありません。



REST API/SDKを使用

REST APIを使用して直接バケットのバージョン管理を設定し、バージョン管理状態にあるバケット内のオブジェクトを管理できます。以下のAPIドキュメントを参照してください:

- PUT Bucket versioning
- GET Bucket versioning
- GET Bucket Object versions
- PUT Object
- GET Object
- DELETE Object
- DELETE Multiple Objects

SDKのバージョン管理メソッドを直接呼び出すことができます。詳細は、以下の各言語のSDKドキュメントを参照してください:

Android、C、C++、.NET、Go、iOS、Java、JavaScript、Node.js、PHP、Python、ミニプログラム

ツールを使用する

ツールを使用してバージョン管理を設定できます。例えば COSBrowser、COSCMD、COSCLIなど。その他のツールの詳細については、ツール概要を参照してください。

バージョン履歴オブジェクトの復元

最終更新日：：2024-06-26 09:48:11

概要

COSコンソールから、オブジェクトの旧バージョンを最新バージョンに復元することができます。ここでは、コンソールでバケットオブジェクトの旧バージョンを復元する方法についてご説明します。バージョン管理の詳細については、バージョン管理の概要をご参照ください。

説明：
- オブジェクトの復元とは、旧バージョンを最新バージョンに復元することで、旧バージョンは引き続き保持されます。
- 旧バージョンを削除する必要がある場合は、バージョン管理の設定、オブジェクトの削除の操作をご参照ください。
- 単一オブジェクトの復元操作をサポートしていますが、バッチ復元はサポートしていません。

ご使用にあたっての注意事項

次に、復元されたオブジェクトの適用ケースとルールについてご説明します。

適用ケース	オブジェクトの復元操作は、バージョン管理を有効にしているバケットでサポートされます。 バージョン管理を有効化した後に再度一時停止したバケットについても、オブジェクトの復元操作がサポートされます。 バージョン管理を有効にしたことがないバケットでは、オブジェクトの復元操作はサポートされません。
復元ルール	旧バージョン：復元がサポートされます。 最新バージョン：復元できません。 削除マーカー付きのバージョン：復元できません。

前提条件

オブジェクトを復元する前に、バケットでバージョン管理が有効化されていることを確認してください。有効化されていない場合は、バージョン管理の設定ドキュメントをご参照のうえ、操作を行ってください。

操作手順

1. COSコンソールにログインします。
2. 左側ナビゲーションバーでバケットリストをクリックし、バケットリストページに進みます。
3. オブジェクトがあるバケットを見つけ、そのバケット名をクリックし、バケット管理ページに進みます。

4. 左側ナビゲーションバーでファイルリストを選択し、ファイルリストページに進みます。
5. 旧バージョンの一覧表示スイッチをオンにして、復元したいオブジェクトを見つけます。その右側の操作バーの下にある復元をクリックします。
6. 復元ファイルのポップアップウィンドウで、復元オブジェクトのバージョン情報をチェックします。正しいことを確認した後、OKをクリックすれば完了です。
7. この時点で、オブジェクトの復元操作は完了です。ファイルリストから、旧バージョンの復元に成功していることが確認できます。

バケットコピー
バケットコピーの概要

最終更新日：：2025-08-29 10:23:49

概要

バケットレプリケーションは、バケットに対する設定であり、レプリケーションルールを設定することで、異なるバケット間でインクリメンタルオブジェクトを自動的かつ非同期的に複製できます。バケットレプリケーションが有効になると、クラウドオブジェクトストレージ（COS）は、ソースバケット内のオブジェクト内容（オブジェクトメタデータやバージョンIDなど）をターゲットバケットに正確に複製し、複製されたオブジェクトのコピーは完全に一致する属性情報を持ちます。また、ソースバケット内のオブジェクトに対する操作（オブジェクトの追加や削除など）もターゲットバケットに複製されます。

> ⚠ 注意：
> - バケットのコピー機能を有効にするには、同時にソースバケットとターゲットバケットの両方でバージョン管理機能を有効にしておく必要があります。
> - バケットコピーを有効にする際は、データのコピー時にオブジェクトレプリカのストレージタイプを明確に指定した場合を除き、オブジェクトレプリカはソースオブジェクトと同じストレージタイプとなります。
> - COSはコピー時にソースバケットのアクセス制御リスト（ACL）をコピーします。現時点でCOSでは、異なる2つのアカウントのバケットをソースバケットとターゲットバケットとすることはできません。

ユースケース

- リモートディザスタリカバリ：COSはオブジェクトデータに対し99.9999999999%の可用性をご提供していますが、戦争、自然災害などの様々な不可抗力要素によるデータ消失の可能性は存在します。データ消失に伴う損失が許容できないものであり、異なるバケットでデータレプリカを明示的に保全したい場合は、バケットコピーによってデータのリモートディザスタリカバリを実現できます。あるデータセンターが不可抗力要素によって破壊された場合でも、もう1つのバケットのデータセンターが使用可能なレプリカデータを提供することができます。
- コンプライアンス要件：COSはデフォルトでは物理ディスク内で、マルチレプリカおよびイレージャーコーディング方式でデータの可用性を保障していますが、業界によってはコンプライアンス要件が存在し、異なるバケット間でデータレプリカを保存するよう規定されている場合があります。このため、バケットコピーを有効にすることで、異なるバケットのデータコピーを実現し、これらのコンプライアンス要件を満たすことができます。

- アクセス遅延の減少：お客様の顧客が異なる地理的位置からオブジェクトにアクセスする際、バケットコピーによって、顧客の地理的位置に最も近いバケット内でオブジェクトレプリカを保全することで、顧客のアクセス遅延を最大限に短縮でき、製品体験の向上に役立てることができます。
- 操作上の理由：2つの異なるバケットのどちらにもコンピューティングクラスターがあり、かつこれらのコンピューティングクラスターが同じデータセットを処理する必要がある場合は、バケットコピーによってこれらの異なるバケット内でオブジェクトレプリカを保全することができます。
- データマイグレーションおよびバックアップ：業務の発展による必要性に応じて、業務データをあるバケットから別のバケットにコピーすることで、データマイグレーションおよびデータバックアップを実現できます。

注意事項

料金説明

- オブジェクトをコピーする際、読み取りおよび書き込みリクエストが発生します。COSはリクエスト回数を計算し、リクエスト料金を請求します。詳細については、リクエスト料金をご参照ください。
- 同一リージョン内でのコピーは無料です。クロスリージョンコピーには、クロスリージョン転送のデータ転送料金が発生します。COSはクロスリージョン転送のデータ量を計算し、ソースバケットのリージョンごとに単価で料金を請求します。詳細については、トラフィック料金をご参照ください。
- オブジェクトのコピーが完了した後、ストレージ容量に対する料金が発生します。COSはオブジェクトのサイズを計算し、ストレージ容量の料金はターゲットオブジェクトのストレージタイプおよびリージョンによって決定されます。詳細については、ストレージ容量料金をご参照ください。
- バケットのコピーには、ユーザーがバージョン管理機能を有効にする必要があります。バージョン管理機能を有効にすると、バケット内にオブジェクトの複数の履歴バージョンが保存され、ストレージ消費が増加します。バケットコピーおよびバージョン管理に伴うコストを削減したり、データ保持方法をカスタマイズしたりする場合は、ライフサイクル管理を活用し、業界に応じたコスト管理やデータ保持方法を実現できます。

コピー時間の制限

COSがオブジェクトのコピーに要する時間は、オブジェクトのサイズ、バケットのリージョン間の距離、オブジェクトのアップロード方式などの要素に左右されます。同期時間はこれらの要因に基づいて異なり、数分から数時間、あるいは数日にも及ぶことがあります。

- オブジェクトのサイズ。大きなオブジェクトのコピーにはより多くの時間が必要です。大きなオブジェクトについてはマルチパートアップロード方式を利用することで、オブジェクトのアップロードおよび同期時間を短縮することをお勧めします。
- バケットのリージョン間の距離。リージョン間の距離が遠いほど、同期の際にデータ転送時間がより長くかかります。
- オブジェクトのアップロード方式。シンプルアップロード方式は同時実行ができず、1本の接続上でデータをシリアルにアップロードまたはダウンロードするだけですが、マルチパートアップロード方式では同時実行が可能なため、大容量ファイルのアップロードの際はマルチパートアップロードを使用することで、アップロードおよびバケットコピーの速度を速めることができます。オブジェクトのアップロード方式に関する詳細な説明については、シンプルアップロードおよびマルチパートアップロードのドキュメントをご参照ください。

ライフサイクル関連

バケットコピーには、ユーザーがバージョン管理機能を有効にする必要があります。バージョン管理機能が有効になると、バケット内にオブジェクトの複数の履歴バージョンが保存され、ストレージ消費が増大します。バケットコピーおよびバージョン管理に伴うコスト削減のため、ライフサイクルを使用して定期的にファイルを削除したり、アーカイブしたりできます。

- ソースバケットのライフサイクルに基づくアーカイブおよび削除操作は、ターゲットバケットに自動的に同期されません。ターゲットバケット内のオブジェクトレプリカがソースバケット内と同様のライフサイクルルールに従うようにしたい場合は、ターゲットバケットにソースバケットと同一のライフサイクルルールを追加してください。
- ターゲットバケットにライフサイクルルールを設定する場合、バケットコピー後のオブジェクトレプリカの作成時間は、それがターゲットバケット内に生成された時間ではなく、ソースバケット内での作成時間に対応することに注意する必要があります。
- オブジェクトの作成時刻。バケットのルールが設定されている場合、ターゲットバケット内のオブジェクトコピーの作成時刻はソースバケットと一致し、ターゲットバケットにオブジェクトが表示された時刻とは異なります。ターゲットバケットにライフサイクルルールを設定した場合、そのルールは作成時刻に基づいて適用されます。

バージョン管理関連

バケットコピーの設定には、ユーザーがソースバケットとターゲットバケットの両方でバージョン管理機能を設定しておく必要があります。バージョン管理機能の詳細な内容については、バージョン管理の概要をご参照ください。バージョン管理をオンにした場合は、バージョン管理をオフにすることでバケットコピー機能に生じる影響に注意する必要があります。

- バケットコピー機能を有効にしているバケット内でバージョン管理を無効にすることを試してみると、COSはエラーを返し、「先にバケットコピールールを削除してからバージョン管理を無効にする必要があります」とのメッセージを表示します。
- あるターゲットバケット内でバージョン管理を無効にすることを試してみると、COSは、「バージョン管理をオフにするとバケットコピー機能に影響があります。バージョン管理を引き続きオフにする場合、このバケットをターゲットバケットとするCOSのバケットコピールールは失効します」とのメッセージを表示します。

コピーアクションの説明

最終更新日：：2024-06-26 10:57:13

このドキュメントでは主に、ユーザーがバケットのバケットコピー機能を有効にした場合に、COSがコピーするコンテンツとコピーしないコンテンツについてご説明します。

コピーするコンテンツ

バケットのコピー機能を有効にしたソースバケットにおいて、COSは次のコンテンツをコピーします。

- バケットコピールールを追加した後に、ユーザーがソースバケットに新たにアップロードしたあらゆるオブジェクト。
- オブジェクトのメタデータおよびバージョンIDなどのオブジェクトの属性情報。
- オブジェクトの操作に関する情報。新たに追加された同名のオブジェクト（新規追加オブジェクト）、削除されたオブジェクトなど。

> 説明：
> - ソースバケット内で特定のオブジェクトのバージョンを削除するよう指定（バージョンIDを指定）した場合、その操作はコピーされません。
> - ソースバケットに、例えばライフサイクルルールのような、バケットレベルの設定を追加している場合、これらの設定によって発生したオブジェクトの操作もターゲットバケットにはコピーされません。

バケットコピーにおける削除操作

ソースバケットからオブジェクトを削除する場合、バケットのコピーアクションは次のとおりとなります。

- オブジェクトのバージョンIDを指定せずにDELETEリクエストを実行した場合、COSはソースバケットに削除タグを追加します。同期削除タグを選択した場合、バケットコピーはこのタグをターゲットバケットにコピーします。非同期削除タグを選択した場合、ターゲットバケットは削除タグを新たに追加しません。どちらの状況でも、ターゲットバケットは対応するファイルを削除せず、ユーザーはバージョンIDを指定してオブジェクトの過去バージョンにアクセスすることができます。バージョン管理と削除タグの詳細情報については、バージョン管理の概要のドキュメントをご参照ください。
- オブジェクトのバージョンIDを指定してDELETEリクエストを実行した場合、COSはソースバケット内の指定されたオブジェクトのバージョンを削除しますが、ターゲットバケットではこの削除操作をコピーしません。すなわち、COSがターゲットバケット内で指定されたオブジェクトのバージョンを削除することはありません。これにより悪意あるデータ削除を防止することができます。

コピーしないコンテンツ

ソースバケットがバケットコピー機能を有効にしている場合、COSは次のコンテンツをコピーしません。

- バケットコピー機能を有効にする前にすでに存在したオブジェクトの内容、すなわち既存データ。

- 暗号化されたオブジェクトの暗号化情報。暗号化されたオブジェクトがコピーされると、暗号化情報は失われます。
- ソースバケット内に新たに追加されたデータが、他のバケットからコピーされたオブジェクトデータの場合。
- バケットレベルの設定更新アクション。
- ライフサイクルの設定実行後の結果。

説明:
- オブジェクトデータのバケット間でのバケットコピーは伝達性を有しません。例えば、バケットAをソースバケット、バケットBをターゲットバケットとするものと、バケットBをソースバケット、バケットCをターゲットバケットとする、2つのバケットコピールールを同時に設定したとします。この場合、バケットAに新たに追加されたオブジェクトデータはバケットBにのみコピーされます。そこからさらにバケットCにコピーされることはありません。
- 例えばライフサイクル設定の場合、ソースバケットのライフサイクル設定を更新しても、COSがこのライフサイクル設定をターゲットバケットに同期的に適用することはありません。
- ソースバケットに対してのみライフサイクルルールを設定している場合、COSは期限切れのオブジェクトに削除タグを追加しますが、ターゲットバケットがこれらのタグをコピーすることはありません。ターゲットバケットで期限切れのオブジェクトを削除できるようにしたい場合は、ターゲットバケットに対し単独で、ソースバケットと同一のライフサイクルルールを設定する必要があります。

バケットコピーの設定

最終更新日：：2025-10-15 17:15:03

適用ケース

バケットコピールールを設定することで、ユーザーはオブジェクトデータをソースバケットから別の指定するターゲットバケットにコピーすることができます。バケットコピー機能は、リモート障害復旧、業界のコンプライアンス要件への適合、データマイグレーションおよびバックアップ、顧客アクセス遅延の低減、異なるリージョン間のクラスターのデータアクセス利便性向上などのケースに適しています。
特殊なケース：

- マルチリージョンバックアップ：ソースバケットに複数のレプリケーションルールを設定し、オブジェクトを異なるリージョンのバケットにコピーすることで、マルチリージョナルなバックアップと障害復旧を実現できます。
- 双方向レプリケーション：ソースバケットとターゲットバケットでそれぞれレプリケーションルールを作成することにより、2つのバケット間での双方向レプリケーションを実現し、バケットデータの同期を実現することができます。

> ⚠ 注意：
> バージョン管理機能を有効にしている場合、新たにアップロードするオブジェクトには複数のバージョンが生成され、ストレージスペースを占有します。このためこれらのバージョンのオブジェクトも同様にストレージ料金の課金対象となります。

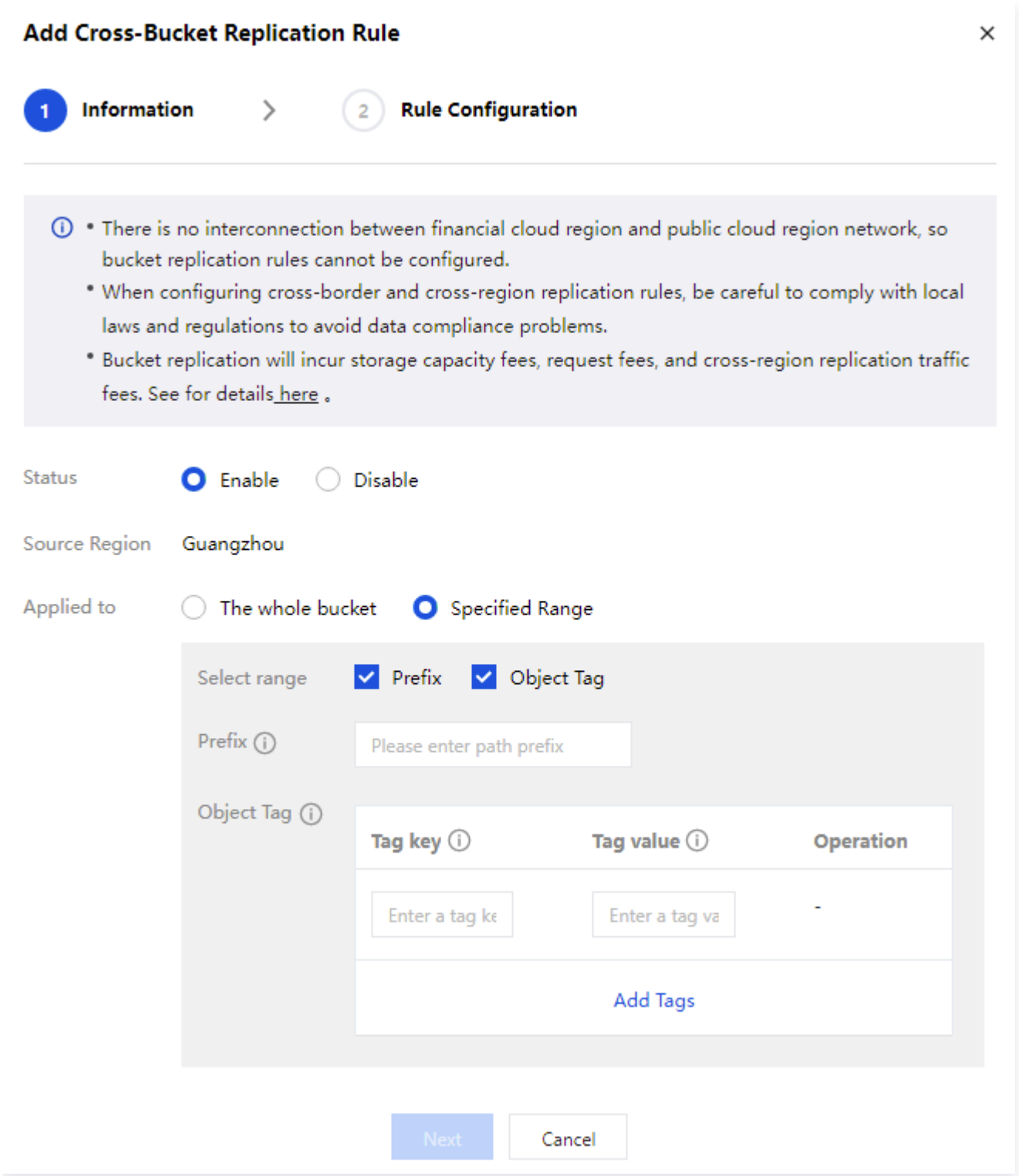
使用方法

COSコンソールの使用

バケットレプリケーションの有効化

1. COSコンソールにログインします。
2. 左側のナビゲーションメニューで、バケットリストをクリックします。
3. レプリケーションの対象となるソースバケットをクリックし、バケット詳細ページに移動します。
4. 左側のフォールトトレランスとディザスタリカバリ管理＞バケットレプリケーションを選択すると、このページが表示されます。
5. バケットレプリケーションエリアでルールを追加をクリックします。設定項目の説明は以下の通りです。

- 基本情報

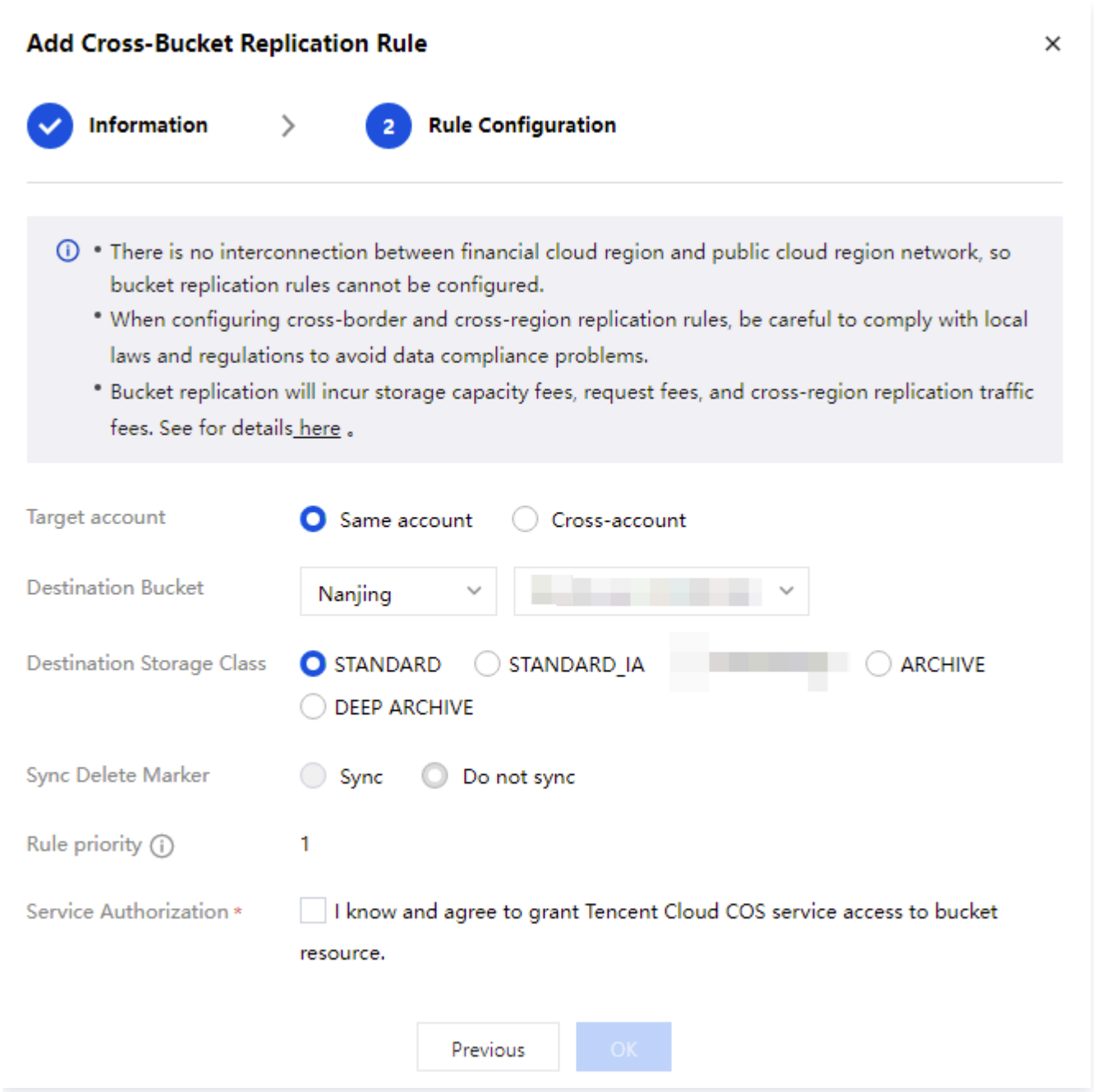


- ステータス：現在のルールの状態。
- ソースリージョン：ソースバケットが所属するリージョン。
- 適用範囲：レプリケーション対象となるソースバケット内のオブジェクト範囲を指します。バケット全体または指定範囲の選択がサポートされています。
  - バケット全体：ソースバケットのすべての増分オブジェクトをターゲットバケットにレプリケートすることを指します。

- 指定範囲：オブジェクトプレフィックス、オブジェクトタグに基づき、レプリケーション対象の増分オブジェクトをフィルタリングすることを指します。指定範囲を選択する場合は、以下のいずれかを少なくとも一つ設定してください。
  - オブジェクトプレフィックス：同一のオブジェクトキープレフィックスを持つオブジェクトに対し、レプリケーションルールを適用するよう指定できます。例：`prefix/`。正規表現はサポートしていません。
  - オブジェクトタグ：同一のタグが付与されたオブジェクトに対し、レプリケーションルールを適用するよう指定できます。最大10個のタグを指定でき、英字の大文字と小文字は区別されます。

> ⚠ 注意：
> オブジェクトプレフィックスとオブジェクトタグを同時に指定できます。オブジェクトプレフィックスとオブジェクトタグ、オブジェクトタグとオブジェクトタグの間は、すべて「AND」関係（つまり、すべての条件を同時に満たす必要がある）となります。例えば、バケットレプリケーションルールでオブジェクトプレフィックスをdoc、オブジェクトタグのキーと値のペアを`group = IT`と指定した場合、指定されたオブジェクトの範囲は、現在のバケット内でオブジェクトキーのプレフィックスがdocであり、かつオブジェクトタグがgroup = ITであるすべての増分オブジェクトです。

- ルール設定



- ターゲットバケット：オブジェクトがレプリケーションされた後に保存されるバケットを指します。ソースバケットと同一または異なるリージョンのバケットを選択できますが、現在のアカウント以外のバケットは選択できません。設定後、新規レプリケーションルールを追加する際、同じ対象リージョンに対しては、以前に指定したターゲットバケットのみ設定できます。

  説明：
  - ソースバケットとターゲットバケットの両方でバージョニング機能が有効になっていない場合、まずバージョニング機能を有効にしてから、バケットレプリケーションルールを設定してください。
  - 仮にルール1でターゲットバケットとして南京リージョンのバケットAを指定したとします。その後、ルール2を新規追加し、ターゲットバケットとして南京リージョンのバケットに設定す

る場合、選択できるのはバケットAのみで、それ以外を選択した場合はエラーが発生します。

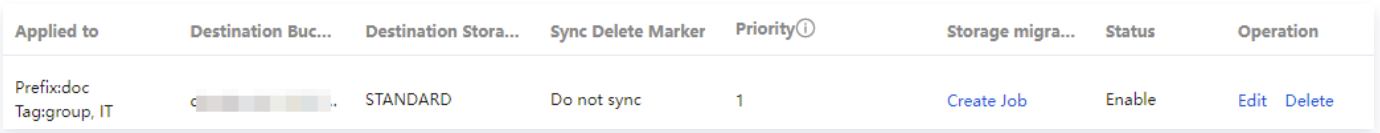
- ターゲットストレージタイプ：オブジェクトがターゲットバケットにレプリケーションされた後のストレージタイプを指します。ターゲットバケット内にレプリケートされたオブジェクトのストレージタイプを変更できます。一部のストレージタイプは特定のリージョンでのみ利用可能です。ストレージタイプの詳細と対応リージョンについては、ストレージタイプの概要をご参照ください。
- 削除マーカーの同期：同期する、または同期しないを選択できます。デフォルトは「同期しない」です。
  - 同期：削除マーカーの同期を選択した場合、バケットレプリケーション機能はそのマーカーをターゲットバケットにコピーするため、慎重に選択してください。ソースバケットのファイルが削除された場合、バケットレプリケーションの動作は以下の通りです。
    - バージョンIDを指定せずにファイルを削除する場合、COSはソースバケットに削除マーカーを追加します。バケットレプリケーション機能はこのマーカーをターゲットバケットにコピーするため、ターゲットバケットのファイルも同期して削除されます。
    - バージョンIDを指定してファイルを削除する場合、COSはソースバケット内の指定オブジェクトバージョンを削除しますが、この削除操作はターゲットバケットにはコピーされません。つまり、COSはターゲットバケット内の指定オブジェクトバージョンを削除しません。この動作により、悪意のあるデータ削除を防ぐことができます。
  - 同期しない：削除マーカーの同期を選択しない場合、ターゲットバケットに新しい削除マーカーは追加されません。ソースバケットのファイルが削除されても、バケットレプリケーション機能は削除マーカーをターゲットバケットにコピーしないため、ターゲットバケットのファイルは影響を受けません。

いずれの場合も、ターゲットバケットでは対応するファイルが物理的に削除されることなく、ユーザーはバージョンIDを指定することによりオブジェクトの過去のバージョンにアクセスできます。バージョニングと削除マーカーの詳細については、バージョニングの概要ドキュメントをご参照ください。

説明：
- ライフサイクルによって生成された削除マーカーや階層化操作などはターゲットバケットに同期されません。ターゲットバケットでも期限切れオブジェクトを削除したい場合は、ターゲットバケットに対し、ソースバケットと同一のライフサイクルルールを個別に設定する必要があります。詳細については、バケットレプリケーションの概要および レプリケーション動作の説明をご参照ください。
- 適用範囲で指定範囲を選択し、かつ選択範囲をオブジェクトタグで指定した場合、ルール設定ページで削除マーカーの同期を選択することはできません。

- ルール優先度：現在のバケットレプリケーションルールの優先度を指します。システムはデフォルトロジックに基づき各ルールに優先度を割り当てます。変更する場合は、ルール一覧ページで優先度の調整をクリックして操作してください。操作の詳細については、ルール優先度の調整をご参照ください。

6. 情報を確認の上、権限付与にチェックを入れた後、確定をクリックすると、バケットレプリケーションルールが表示されます。

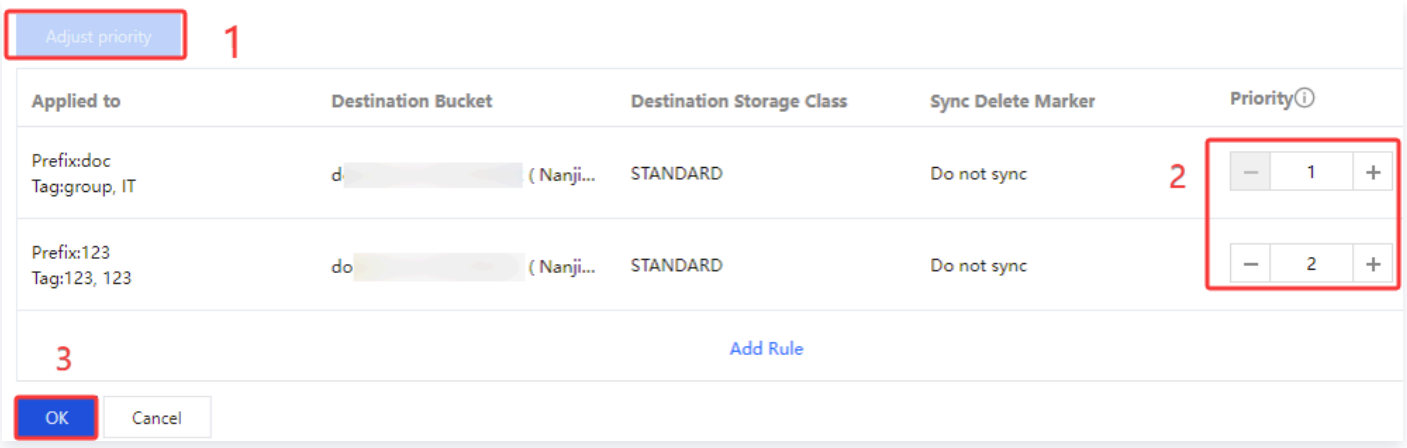


> ⚠ 注意：
> ルールを設定した後、ルールに対して管理操作を行うことができます。編集ボタンで現在のルール内容やステータスを変更でき、削除ボタンで現在のルールを削除できます。

バケットレプリケーションルールの優先度調整

> ⚠ 注意：
> - 優先度は1～1000で設定可能です。数字が小さいほど優先度が高くなります。
> - 異なるルールの優先度を重複させることはできません。
> - この機能のアップグレード前に「バケットレプリケーションルール」を設定済みの場合、これらのルールには一時的に優先度がなく、「-」と表示されます。詳細は以下の通りです。
>   - アップグレード後にルールを新規追加または編集すると、デフォルトロジックに基づき、既存ルールに優先度が割り当てられ、従来のルールを優先度フィールド付きの新しい形式に一括変換されます。ビジネス要件に応じて優先度を調整できます。
>   - アップグレード後にルールを新規追加または編集しない場合でも、バケットレプリケーションは既存のルールに基づいて継続されます。

以下の方法で、バケットレプリケーションルールの優先度を調整できます。操作手順は以下の通りです。

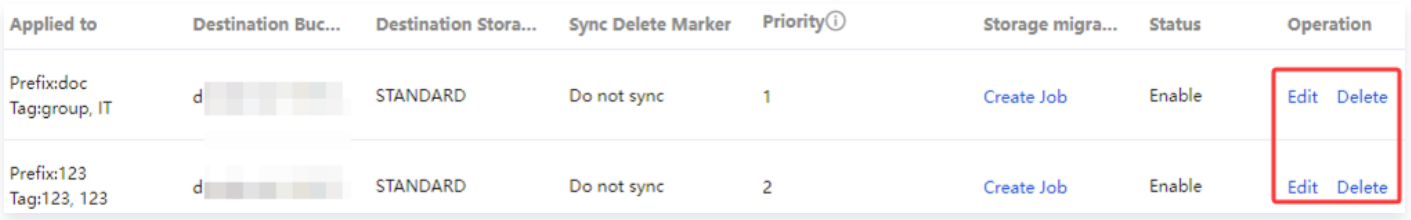


1. バケットレプリケーションのルール一覧ページで、優先度の調整をクリックします。
2. プラス/マイナスボタン、または直接入力で希望の優先度を設定します。
3. 確定をクリックして、バケットレプリケーションルールの優先度の調整を完了します。

バケットレプリケーションの無効化

> ⚠ 注意：
> - まだ完了していないレプリケーション操作は、機能を無効化した時点で中止され、続行できなくなります。
> - バケットレプリケーションを再度有効化した場合、有効化後に新たに追加されたオブジェクトのみがレプリケーション操作の対象となります。

バケットレプリケーション機能は、ルールの編集と削除の2つの方法で無効化できます。



Applied to	Destination Buc...	Destination Stora...	Sync Delete Marker	Priority	Storage migra...	Status	Operation
Prefix:doc Tag:group, IT	d	STANDARD	Do not sync	1	Create Job	Enable	Edit Delete
Prefix:123 Tag:123, 123	d	STANDARD	Do not sync	2	Create Job	Enable	Edit Delete

- ルールの編集：バケットレプリケーション管理項目内で追加済みのルールを編集し、ルール内のステータスボタンで一時的に無効化できます。これにより、現在のバケットレプリケーション機能は一時停止され、コピー済みのデータはターゲットバケットに保持されますが、ソースバケットの増分データはターゲットバケットにコピーされなくなります。再度使用するには、有効化ボタンをクリックしてください。
- ルールの削除：バケットレプリケーション管理項目内で追加済みのルールを削除します。削除後、設定済みのバケットレプリケーションルールは無効となります。コピー済みのデータはターゲットバケットに保持されますが、ソースバケットの増分データはレプリケートされなくなります。再度使用するには、新たにルールを追加する必要があります。

REST APIの使用

REST APIを直接使用して、バケットのバケットコピールールの設定と管理を行うことができます。具体的には次のAPIドキュメントをご参照ください。

- PUT Bucket replication
- GET Bucket replication
- DELETE Bucket replication

SDKの使用

SDKのバケットコピーメソッドを直接呼び出すことができます。詳細については、下記の各言語のSDKドキュメントをご参照ください。

- Android SDK
- C SDK
- C++ SDK
- .NET SDK

- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- ミニプログラム SDK

マルチAZ特性の概要

最終更新日：：2024-06-26 10:57:13

マルチAZ（Available Zone）とはTencent Cloud COSがリリースしたマルチAZストレージアーキテクチャです。このストレージアーキテクチャによってユーザーデータにデータセンターレベルの障害復旧機能を提供することが可能です。
お客様のデータは都市の複数の異なるデータセンターに分散して保存されます。あるデータセンターに自然災害、停電などの極端な状況による全面的な障害が発生した場合でも、マルチAZストレージアーキテクチャによって安定した信頼性の高いストレージサービスを引き続き提供することができます。
マルチAZ特性により、設計上のデータ信頼性は99.9999999999%（トゥエルブナイン）に、設計上のサービス可用性は99.995%に達します。データをCOSにアップロードする際にオブジェクトのストレージタイプを指定するだけで、オブジェクトをマルチAZのリージョンに保存することができます。

> 説明：
> - COSのマルチAZ特性は現時点では北京、広州、上海、中国香港、シンガポール。その他のパブリッククラウドリージョンでも順次サポート予定です。
> - COSのマルチAZ特性を使用する場合、ストレージ容量料金が相対的に高くなります。詳細については、製品価格をご参照ください。

マルチAZのメリット

データをマルチAZリージョンに保存すると、データがいくつかのパートに分割され、同時にイレイジャーコーディングアルゴリズムに従って対応するチェックコードパートが算出されます。オリジナルのデータパートとチェックコードパートは分割されてそのリージョンの異なるデータセンターに均等に保存され、同一都市内障害復旧が可能になります。あるデータセンターが利用できなくなった場合も、別のデータセンターのデータは正常に読み取りと書き込みができるため、お客様のデータが消失することなく永続的に保存され、お客様の業務におけるデータ連続性と高可用性が維持できます。COSのマルチAZを使用すると次のようなメリットがあります。

- 同一都市内障害復旧：異なるデータセンター間での障害復旧が可能です。マルチAZストレージアーキテクチャでは、オブジェクトデータは同一リージョンの異なるデータセンターの異なるデバイス内に保存されます。データセンターのうち1か所に障害が発生した際、冗長データセンターは引き続き利用できるため、ユーザーの業務は影響を受けず、データも消失しません。
- 安定性・永続性：イレイジャーコーディングによる冗長ストレージ方式を採用し、99.9999999999%の設計上のデータ信頼性を実現しています。また、データの分割ストレージ、読み取り/書き込み同時実行により、設計上のサービス可用性は99.995%に達しています。
- 使いやすさ：オブジェクトのストレージタイプによって、データをどのストレージアーキテクチャに保存するかを指定します。バケット内の任意のオブジェクトを指定してマルチAZアーキテクチャに保存することができ、より簡単に使用できます。

マルチAZストレージと非マルチAZストレージの仕様面での制限の比較は次のとおりです。

比較項目	マルチAZストレージ	非マルチAZストレージ
設計上のデータ永続性	99.9999999999%　（トゥエルブナイン）	99.999999999%（イレブンナイン）
設計上のサービス可用性	99.995%	99.99%
サポートするリージョン	ストレージタイプの概要のドキュメントをご参照ください	
サポートするストレージタイプ	標準ストレージ（マルチAZ）（MAZ_STANDARD）低頻度ストレージ（マルチAZ）（MAZ_STANDARD_IA）INTELLIGENT_TIERINGストレージ（マルチAZ）（MAZ_INTELLIGENT_TIERING）	標準ストレージ（STANDARD）低頻度ストレージ（STANDARD_IA）アーカイブストレージ（ARCHIVE）ディープアーカイブストレージ（DEEP_ARCHIVE）INTELLIGENT_TIERINGストレージ（INTELLIGENT_TIERING）

利用方法

ユーザーはバケットのマルチAZ設定を有効化することができます。マルチAZ設定を有効化したバケットにオブジェクトをアップロードすると、オブジェクトのストレージタイプをマルチAZに設定できます。ユーザーがオブジェクトをアップロードする際にオブジェクトのストレージタイプを指定すると、オブジェクトをマルチAZストレージアーキテクチャ内に保存することができます。
簡潔に言えば、次の2つの手順を実行するだけで、ファイルをマルチAZアーキテクチャ内に保存することができます。

1. バケットを作成し、作成の際にマルチAZ設定を有効化します。バケットの作成ガイドについては、バケットの作成のドキュメントをご参照ください。
2. ファイルをアップロードし、その際にファイルのストレージタイプを指定します。ファイルのアップロードガイドについては、オブジェクトのアップロードのドキュメントをご参照ください。

> 説明：
> - バケットのマルチAZ設定は有効化すると変更できませんので、慎重に設定してください。作成済みのバケットはデフォルトでマルチAZ設定が有効化されていません。有効化できるのは新しく作成したバケットのみとなります。
> - マルチAZ設定を有効化したバケットには、標準ストレージ（マルチAZ）、低頻度ストレージ（マルチAZ）、INTELLIGENT_TIERINGストレージ（マルチAZ）タイプをアップロードすることができます。このうち、INTELLIGENT_TIERINGストレージ（マルチAZ）タイプをアップロードする場合は、バケットで同時にマルチAZ設定およびINTELLIGENT_TIERING設定を有効化する必要があります。

- 既存のデータをマルチAZバケットに保存したい場合は、マルチAZ設定を有効化したバケットを新規作成し、同時にCOS Batchの一括コピー機能を使用し、既存のバケット内のファイルを新規バケットに一括コピーすることができます。COS Batchの使用ガイドについては、バッチ処理の操作ドキュメントをご参照ください。

使用制限

現在COSは、標準ストレージ（マルチAZ）、低頻度ストレージ（マルチAZ）、INTELLIGENT_TIERINGストレージ（マルチAZ）タイプのアップロードをサポートしています。このため、ストレージタイプの変更を伴う関連の機能には同様に制限が存在します。関連機能の制限についての説明は次のとおりです。

- ストレージタイプの制限：現在は標準ストレージ（マルチAZ）、低頻度ストレージ（マルチAZ）、INTELLIGENT_TIERINGストレージ（マルチAZ）タイプのアップロードのみをサポートしています。このうち、INTELLIGENT_TIERINGストレージ（マルチAZ）タイプをアップロードする場合は、バケットで同時にマルチAZ設定およびINTELLIGENT_TIERING設定を有効化する必要があります。
- 操作の制限：現在はオブジェクトのアップロード、ダウンロードおよび削除操作のみサポートしています。オブジェクトはマルチAZのバケットへのコピーのみをサポートしており、シングルAZのバケットへのコピーはサポートしていません。
- ライフサイクルの制限：現在は期限切れとなったオブジェクトの削除のみをサポートしています。マルチAZストレージタイプからシングルAZストレージタイプへの移行はサポートしていません。
- クロスリージョンレプリケーションの制限：マルチAZストレージタイプをシングルAZストレージタイプにコピーすることはサポートされていません。

データセキュリティ
サーバー側の暗号化の概要

最終更新日：：2025-11-13 17:37:05

概要

Cloud Object Storage（COS）はデータをデータセンター内のディスクに書き込む前に、オブジェクトレベルでデータの暗号化を適用する保護ポリシーをサポートしています。データはアクセス時に自動的に復号されます。暗号化と復号の操作プロセスはサーバー側で完了します。このサーバー側での暗号化機能によって静的データを有効に保護することができます。

⚠ 注意：
- この操作はアーカイブタイプのオブジェクトの暗号化の設定をサポートしていません。暗号化が必要な場合は、まず当該オブジェクトの復元操作を行い、復元完了後、ストレージタイプを標準または低頻度に変更してから、暗号化設定を行えます。
- 暗号化されたオブジェクトへのアクセスは、ユーザーがオブジェクトへのアクセス権限を持っている限り、暗号化されていないオブジェクトへのアクセスと使用感に違いはありません。
- サーバー暗号化はオブジェクトデータのみを暗号化し、オブジェクトのメタデータは暗号化されません。また、オブジェクトサイズにも影響を与えません。
- バケット内のオブジェクトを一覧表示する際、暗号化の有無に関わらず、すべてのオブジェクトがリストによって返されます。

ユースケース

- プライベートデータストレージのケース：プライベートデータのストレージについては、サーバー側での暗号化は、保存されているデータに対して行うことができます。ユーザーのプライバシーは保証され、ユーザーがアクセスした際は自動的に復号されます。
- プライベートデータ転送のケース：プライベートデータの転送については、COSはHTTPSを使用してデプロイしたSSL証明書を提供して暗号化機能を実現します。転送リンクレイヤー上に暗号化レイヤーを設け、データの転送過程でのハッキングや改ざんを確実に防止します。

暗号化方式

COSがサポートしているサーバー側の暗号化方式はSSE-COS、SSE-KMS、SSE-Cです。ユーザーはご自身に合った暗号化方式を選択し、COSに保存したデータの暗号化を行うことができます。

SSE-COSの暗号化

SSE-COS暗号化はCOSホスト鍵のサービス側暗号化である。テンセントクラウドCOSによってマスター鍵と管理データをホスティングする。ユーザーはCOSを介してデータを直接管理し、暗号化します。SSE-COSは多要素強暗号化を採用しており、一意の鍵を使用して各オブジェクトを暗号化し、定期的にローテーションされたマスター鍵を介して鍵自体を暗号化することを保証しています。
SSE-COSモードでは、AES256とSM4の2種類の暗号化アルゴリズムをサポートしており、デフォルトではAES256アルゴリズムが使用されます。オブジェクトをアップロードする際に、サーバー暗号化ヘッダー x-cos-server-side-encryption を使用して暗号化アルゴリズムを指定できます。

> ⚠ 注意:
> プリサインURLを使用してアップロードされたオブジェクトの場合、SSE-COS暗号化は使用できません。COSコントロールパネルまたはHTTPリクエストヘッダーを使用してサーバー暗号化を指定する必要があります。

SSE-KMSの暗号化

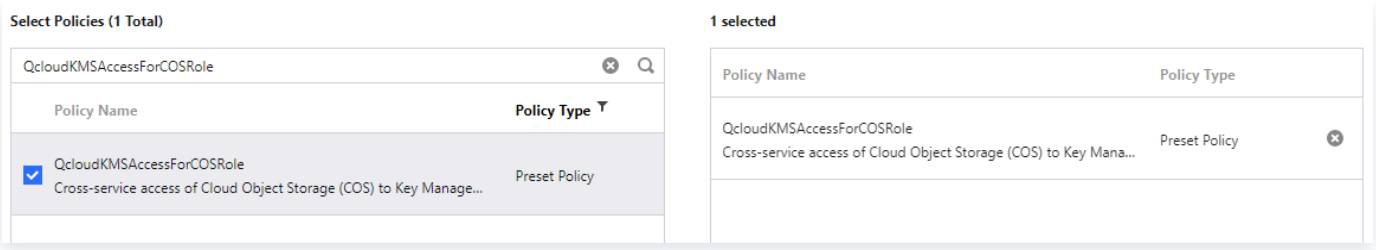
SSE-KMS暗号化は、KMSホストキーによるサーバー側の暗号化です。KMSはTencent Cloudが推進するセキュリティ管理系サービスの1つであり、サードパーティ認証を経たハードウェアセキュリティモジュールHSM（Hardware Security Module）を使用してキーの生成と保護を行うものです。ユーザーがキーの作成と管理を手軽に行えるよう支援し、複数のアプリケーションや複数の業務でのキー管理のニーズを満たすとともに、規制とコンプライアンスの要件にも適合します。
SSE-KMSモードでは、 AES256とSM4の2種類の暗号化アルゴリズムをサポートしており、デフォルトではAES256アルゴリズムが使用されます。オブジェクトをアップロードする際に、サーバー暗号化ヘッダー x-cos-server-side-encryption-cos-kms-algorithm を使用して暗号化アルゴリズムを指定できます。
SSE-KMS暗号化を初めて使用する際は、KMSサービスのアクティブ化 を行う必要があります。KMSサービスをアクティブ化すると、システムは自動的にデフォルトのマスターキー（CMK）1個を作成します。またはKMSコンソール で自らキーを作成し、キーのポリシーと使用方法を定義することもできます。KMSでは、ユーザーがキー素材のソースをKMSまたは外部から自ら選択することができます。その他の情報については、キーの作成 および外部キーのインポート をご参照ください。

> ⚠ 注意:
> - SSE-KMSはオブジェクトデータのみを暗号化し、オブジェクトメタデータは一切暗号化しません。
> - SSE-KMSは現在、北京、上海、中国香港、広州、上海金融、バンコクリージョンのみサポートしています。
> - SSE-KMS暗号化の使用には別途料金が発生し、KMSによって課金されます。詳細については、KMS課金概要 をご参照ください。
> - SSE-KMS暗号化を使用したオブジェクトには有効な署名を使用してアクセスする必要があり、匿名ユーザーはアクセスできなくなります。

注意事項

COSコンソールを使用してSSE-KMS暗号化を行ったことがなく、API方式のみを使用してSSE-KMS暗号化を行っている場合は、まずCAMロールを作成する必要があります。具体的な作成手順は次のとおりです。

1. CAMコンソールにログインし、ロールリストページに進みます。
2. ロールの新規作成をクリックし、Tencent Cloud製品サービスをロールエンティティとして選択します。
3. ロールをサポートするサービスはCOSを選択し、次のステップをクリックします。
4. ロールポリシーを設定します。QcloudKMSAccessForCOSRoleを検索してチェックを入れ、次のステップをクリックします。



5. ロールのタグキーとタグ値をマークし、次のステップをクリックします。
6. 指定のロール名：COS_QCSRoleを入力します。

> ⚠ 注意：
> アクセス管理CAMロールリスト内の既存のロール名「COS_QCSRole」は引き続き使用可能で、適用状態には影響しません。

7. 最後に完了をクリックすれば作成は終了です。

SSE-Cの暗号化

SSE-C暗号化はユーザーが提供するカスタムキーを使用したサーバー暗号化です。ユーザーがアップロードするオブジェクトに対して、COSはユーザーが提供した暗号化キーペアを使ってデータを暗号化します。SSE-Cモードでは、AES256およびSM4の2種類の暗号化アルゴリズムをサポートしており、デフォルトではAES256アルゴリズムが使用されます。オブジェクトをアップロードする際には、サーバー暗号化ヘッダー x-cos-server-side-encryption-customer-algorithm を使用して暗号化アルゴリズムを指定できます。

> ⚠ 注意：
> COSはユーザーが提供した暗号化キーを保存せず、暗号化キーにランダムデータを追加したHMAC値を保存します。このHMAC値はユーザーがオブジェクトにアクセスするリクエストを検証するために使用されます。COSはランダムデータのHMAC値を使って暗号化キーを導き出すことはできず、暗号化されたオブジェクトを復号化することもできません。したがって、ユーザーが暗号化キーを失った場合、オブジェクトを再取得することはできません。

使用方法

COSコントロールパネルの使用

⚠ 注意：
- SSE-C暗号化はコントロールパネル操作をサポートしておらず、API経由でのみ使用可能です。
- SSE-COSおよびSSE-KMS暗号化のSM4アルゴリズムは、現在コントロールパネルおよびSDKによる呼び出しをサポートしておらず、API経由でのみ使用可能です。

1. COSコンソールにログインします。
2. 左側のナビゲーションメニューで、バケットリストをクリックし、バケットリストのページに移動します。
3. オブジェクトが存在するバケットを見つけ、そのバケット名をクリックし、バケット管理ページに移動します。
4. 左側のナビゲーションメニューでファイルリストを選択し、ファイルリストページに移動します。
5. 暗号化を設定したいオブジェクトを見つけ、右側の操作列で詳細をクリックします。
6. 「サーバーサイド暗号化」の項目で、対応する暗号化方式を選択し、保存をクリックします。
7. 現在、以下の2つの暗号化方式をサポートしています。
   - SSE-COS：COSキーを使用したサーバーサイドの暗号化です。SSE-COSに関する詳細は、サーバーサイド暗号化概要：SSE-COSをご参照ください。
   - SSE-KMS：Tencent CloudのKey Management System（KMS）キーを使用したサーバーサイドの暗号化です。デフォルトキーまたは自身で作成したキーを使用できます。キー情報については、KMSキーの作成をご参照ください。SSE-KMSに関する詳細は、サーバーサイド暗号化概要：SSE-KMSをご参照ください。

   ⓘ 説明：
   - 初めてSSE-KMS暗号化を使用する場合、KMSサービスを有効にする必要があります。
   - SSE-COSで暗号化されたオブジェクトは匿名アクセスをサポートしますが、SSE-KMSで暗号化されたオブジェクトは有効な署名付きリクエストでのみアクセス可能であり、匿名ユーザーによるアクセスはできません。
   - 複数のオブジェクトに対して一括で暗号化を設定する必要がある場合は、複数のオブジェクトにチェックを入れ、上部のその他の操作＞暗号化方式の変更をクリックして設定します。

REST APIの使用

SSE-COSおよびSSE-KMS暗号化では、以下のAPIインターフェースがサポートされています。
- PUT Object
- Initiate Multipart Upload
- PUT Object - Copy
- POST Object

ⓘ 説明：

POSTリクエスト以外でオブジェクトをアップロードする場合、`x-cos-server-side-encryption-*` ヘッダーを提供してサーバー暗号化を適用できます。POSTリクエストを使用する場合、フォームフィールドに `x-cos-server-side-encryption-*` フィールドを提供する必要があります。詳細については、POST Objectをご参照ください。

SSE-C暗号化では、以下のAPIインターフェースがサポートされています。

- PUT Object
- Initiate Multipart Upload
- Upload Part
- POST Object
- PUT Object - Copy

説明:

- POSTリクエスト以外でオブジェクトをアップロードする場合、`x-cos-server-side-encryption-*` ヘッダーを提供してサーバー暗号化を適用できます。POSTリクエストを使用する場合、フォームフィールドに `x-cos-server-side-encryption-*` フィールドを提供する必要があります。詳細については、POST Objectをご参照ください。
- GET ObjectおよびHEAD Objectインターフェースを使用してSSE-C暗号化されたオブジェクトをリクエストする際には、`x-cos-server-side-encryption-*` ヘッダーを提供して指定されたオブジェクトを復号化する必要があります。詳細については、共通リクエストヘッダー - SSE-Cをご参照ください。

SDKのオブジェクトタグ管理メソッドを呼び出すことができます。詳細は、以下の各言語のSDKドキュメントをご参照ください。
Android、C、.NET(C#)、Go、iOS、Java、JavaScript、Node.js、PHP、Python、ミニプログラム。

バケット暗号化の概要

最終更新日：：2025-09-26 14:28:32

概要

バケット暗号化はバケットに対する設定の1つであり、バケット暗号化を設定することで、新たにバケットにアップロードされたすべてのオブジェクトに対し、デフォルトで、指定された暗号化方式で暗号化を行うことができます。
現在、バケットでサポートされている暗号化方式は次の通りです：

- SSE-COS暗号化：サービスがホストするキーによるクラウドオブジェクトストレージ（COS）のサーバーサイド暗号化。
- SSE-KMS暗号化：KMSがホストするキーを使用したサーバーサイド暗号化。

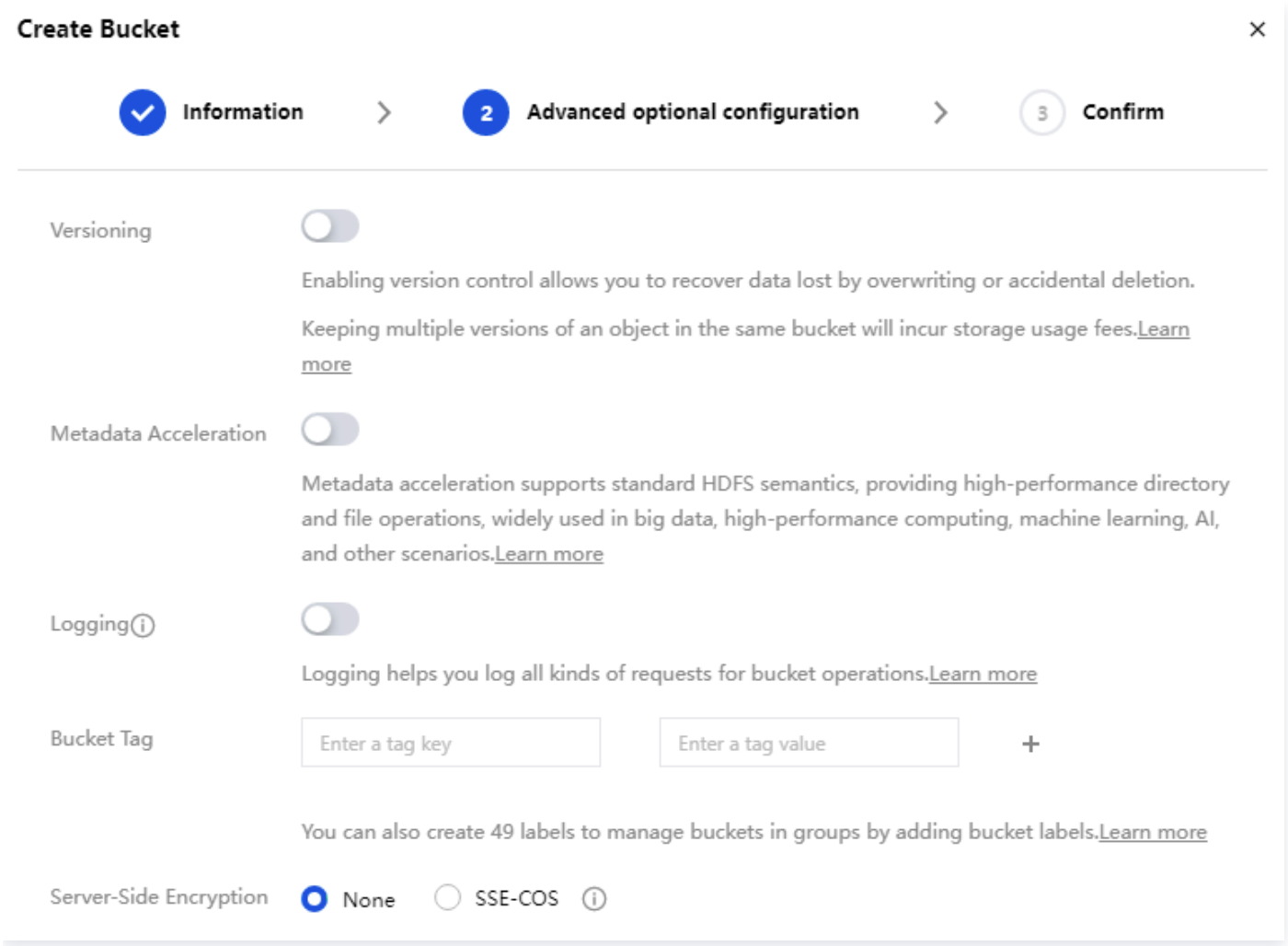
サーバー側の暗号化に関するその他の情報については、サーバー側の暗号化の概要をご参照ください。

使用方法

COSコンソールの使用

バケット新規作成時の暗号化設定

バケットの作成時に、以下の図のようにバケットの暗号化を追加できます。関連する設定項目の説明については、バケットの作成存儲桶をご参照ください。



作成済みバケットでの暗号化設定

バケットの作成時に暗号化を設定しなかった場合、以下の手順でバケットに暗号化を設定することができます。

1. バケットリストページで暗号化を設定したいバケットを見つけ、その名前をクリックし、バケットの設定ページに進みます。
2. 左側ナビゲーションバーで、セキュリティ管理 > サーバー側の暗号化をクリックします。
3. サーバー側の暗号化の設定項目で編集をクリックし、現在のステータスを「有効」に変更します。
4. 指定する暗号化方式を選択し、保存をクリックすると、バケットの暗号化設定が完了します。

   現在、バケットでサポートされている暗号化方式は次の通りです：
   - SSE-COS暗号化：サービスがホストするキーによるクラウドオブジェクトストレージ（COS）のサーバーサイド暗号化。
   - SSE-KMS暗号化：KMSがホストするキーを使用したサーバーサイド暗号化。

> 説明：
> サーバーサイド暗号化に関する紹介およびサポート地域については、サーバーサイド暗号化の概要を参照してください。

REST APIの使用

次のAPIによってバケットの暗号化を設定できます。

- PUT Bucket encryption
- GET Bucket encryption
- DELETE Bucket encryption

注意事項

暗号化されたバケットへのオブジェクトのアップロード

バケット暗号化機能を設定したいバケットについては、次の数点に注意が必要です。

- バケット暗号化では、バケット内にすでに存在するオブジェクトに対する暗号化操作は行いません。
- バケット暗号化を設定した後、このバケットにアップロードされるオブジェクトは次のようになります。
  - PUTリクエストに暗号化情報が含まれていない場合、アップロードされるオブジェクトはバケットの暗号化設定によって暗号化されます。
  - PUTリクエストに暗号化情報が含まれている場合、アップロードされるオブジェクトはPUTリクエスト内の暗号化情報によって暗号化されます。
- バケット暗号化を設定した後、このバケットに送信されるリストレポートは次のようになります。
  - リスト自体に暗号化が設定されていない場合、送信されるリストはバケットの暗号化設定によって暗号化されます。
  - リスト自体に暗号化が設定されている場合、送信されるリストはリストの暗号化設定によって暗号化されます。
- バケット暗号化を設定した後、このバケットにback-to-originされるデータは、デフォルトでバケットの暗号化設定によって暗号化されます。

クロスリージョンレプリケーションルールが設定されているバケットに対する暗号化

クロスリージョンレプリケーションルールが設定されているターゲットバケットに対し、さらにバケット暗号化を設定する場合は、次の数点に注意が必要です。

- ソースバケット内のオブジェクトが暗号化されていない場合、ターゲットバケット内のレプリカオブジェクトに対してはデフォルトで暗号化が設定されます。
- ソースバケット内のオブジェクトが暗号化されている場合、ターゲットバケット内のレプリカオブジェクトはソースバケットの暗号化を継承するため、バケット暗号化設定は実行されません。

盗用リスク検出

最終更新日：：2025-08-29 17:49:20

紹介

Cloud Object Storage (COS) はバケット盗用リスク検出をサポートしており、素早く設定することができます。この機能には、リスク検出と概要、検出詳細、使用統計の 3 つのモジュールが含まれています。盗用に関する防止対策の詳細については、 盗用防止ガイド を参照してください。

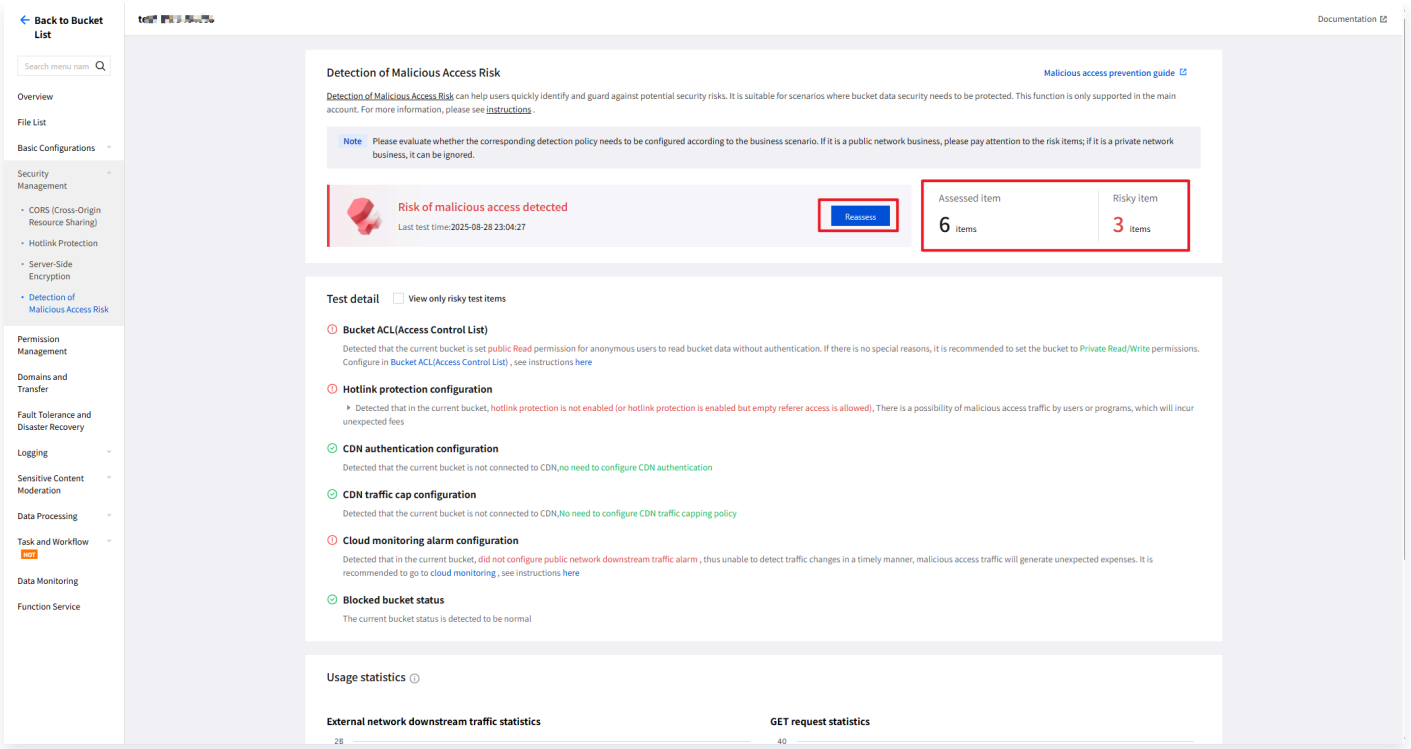
> 注意：
> - 現在、ルートアカウントに対する盗用リスク検出のみがサポートされています。
> - 検出結果が盗用リスクの疑いがある場合、ビジネスシーンに応じて関連する検出ポリシーを設定する必要があるかどうかを評価することができます。パブリックネットワークビジネスの場合は、リスク項目に注意してください。プライベートネットワークビジネスの場合は、無視することができます。

操作手順

リスク検出と概要

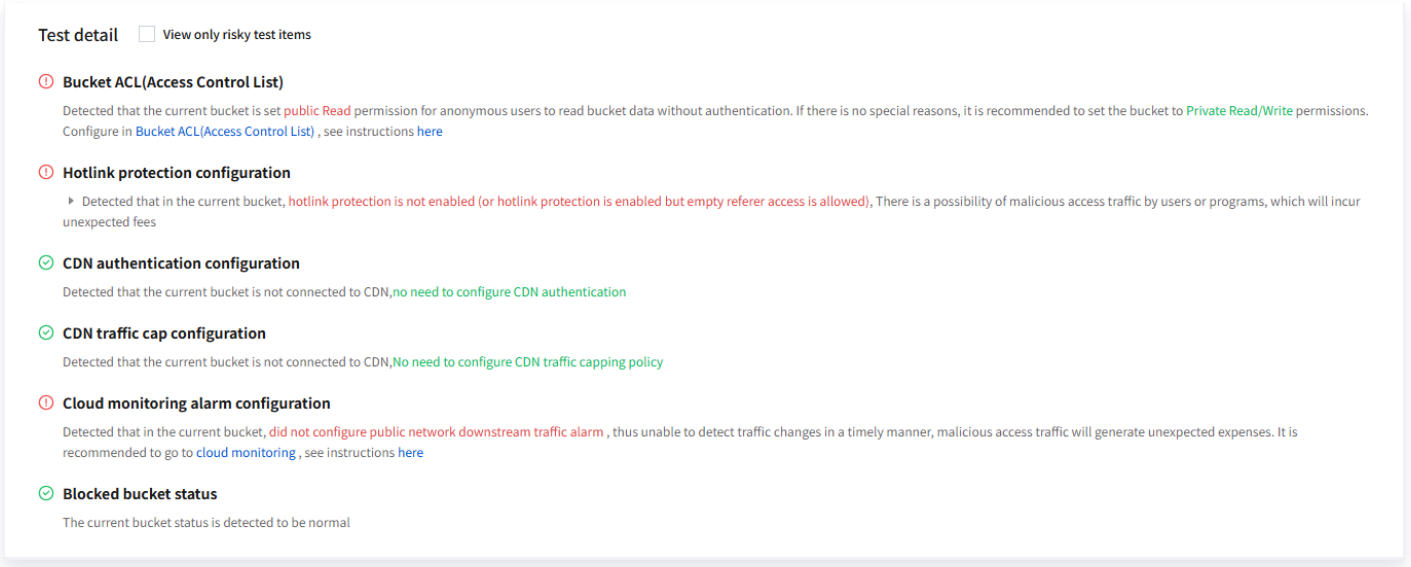
1. COS コンソール にログインします。
2. 左側のナビゲーションバーで、バケットリストをクリックします。
3. 盗用リスクを検出する必要があるバケットを見つけ、そのバケット 名をクリックします。
4. バケット構成ページで、セキュリティ管理 > 盗用リスク検出 をクリックします。盗用リスク検出ページに入ると、右側に全体の検出状況が表示されます。
5. 再検出をクリックすると、検出項目の盗用リスク検出が開始されます。

> 説明：
> このページに入ると、デフォルトで盗用リスク検出が開始されます。



検出詳細

検出詳細モジュールでは、各検出項目のリスク結果を表示し、リスクの高い検出項目をフィルタリングすることができます。さらに、各リスク項目に対して、システムは関連する最適化提案を提供しますので、お客様はそれを迅速に設定することができます。



盗用リスクの検出には、 バケットのアクセス権、防犯リンク設定、CDN 認証設定、CDN トラフィック上限設定、Cloud Monitor アラーム設定、バケットのブロック状態 の 6 項目が含まれています。

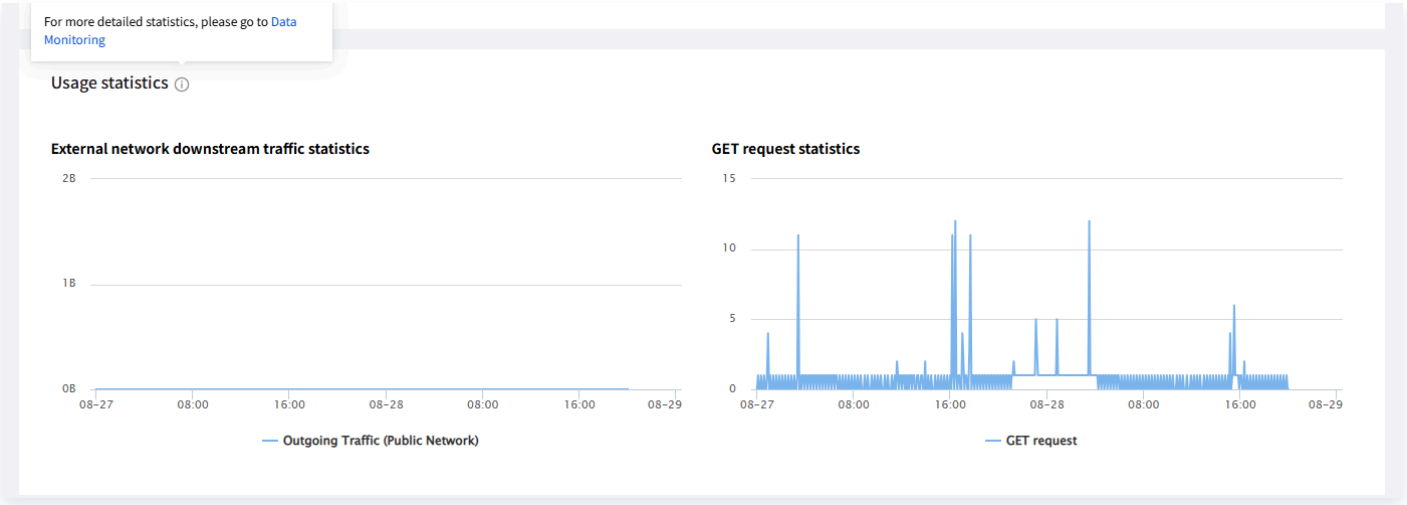
検出項目の説明は以下の通りです。

- バケットのアクセス権： COS バケットのアクセス権を確認します。

  - 現在のバケットのアクセス権がパブリックリード権限に設定されている場合、匿名ユーザーが認証なしでバケットデータを読み取ることができますが、セキュリティリスクが高いためこの構成は推奨されません。
  - 特別なビジネスシーンがない場合、バケットをプライベートリード・ライト権限に設定することが推奨されます。操作ガイドラインについては、盗用防止ガイド–バケットのアクセス権の変更を参照してください。
- 防犯リンクの設定： COS バケットの防犯リンクの設定を確認します。
  - 現在のバケットが防犯リンクを有効にしていない場合、または有効化後に空の Referer アクセスを許可している場合、悪意のあるユーザーやプログラムによってトラフィックを盗用され、予期外のコストが発生する可能性があります。
  - 特別なビジネスシーンがない場合、 防犯リンクを有効にし、空の Referer アクセスを拒否することが推奨されます。操作ガイドについては、 盗用防止ガイド–バケットの防犯リンクの有効化を参照してください。
- CDN 認証の設定： COS バケットのドメイン名の CDN 認証の設定を確認します。
  - 現在のバケットがプライベートバケットであり、CDN に接続されているが、CDN 認証が設定されていないドメイン名が存在する場合は、悪意のあるユーザが営利目的でコンテンツを盗用する可能性があります。
  - 特別なビジネスシーンがない場合、ドメイン名に CDN 認証を設定することが推奨されます。操作ガイドについては、 CDN 認証設定の説明を参照してください。
- CDN トラフィック上限の設定： COS バケットのドメイン名の CDN トラフィック上限の設定を確認します。
  - 現在のバケットにはCDN トラフィック上限設定ポリシーが設定されていないドメイン名が存在している場合、悪意のあるユーザーがトラフィックを盗用し、損失を引き起こす可能性があります。
  - 特別なビジネスシーンがない場合、ドメイン名に対して CDN トラフィック上限ポリシーを設定することが推奨されます。操作ガイドについては、 CDN トラフィック上限の設定を参照してください。
- Cloud Monitor アラーム設定： COS バケットの Cloud Monitor(Tencent Cloud Observability Platform)アラームポリシーを確認します。
  - パブリックネットワークの下りトラフィックに対するアラートポリシーを設定しない場合、トラフィックの変化をタイムリーに感知できず、盗用リスクがあり、予期外の料金が発生する可能性があります。
  - 特別なビジネスシーンがない場合、 パブリックネットワークの下りトラフィックアラームポリシーを設定することが推奨されます。操作ガイドについては、 盗用防止ガイド–Cloud Monitor アラームの設定を参照してください。
- バケットブロック状態 ： COS バケットのブロック状態を確認します。
  - 現在のバケット状態が異常な場合、悪意のあるユーザの盗用によりバケットがブロックされている可能性があります。トラフィック増加が予期とおりの場合、ブロック解除を申請するよう連絡してください。

使用統計

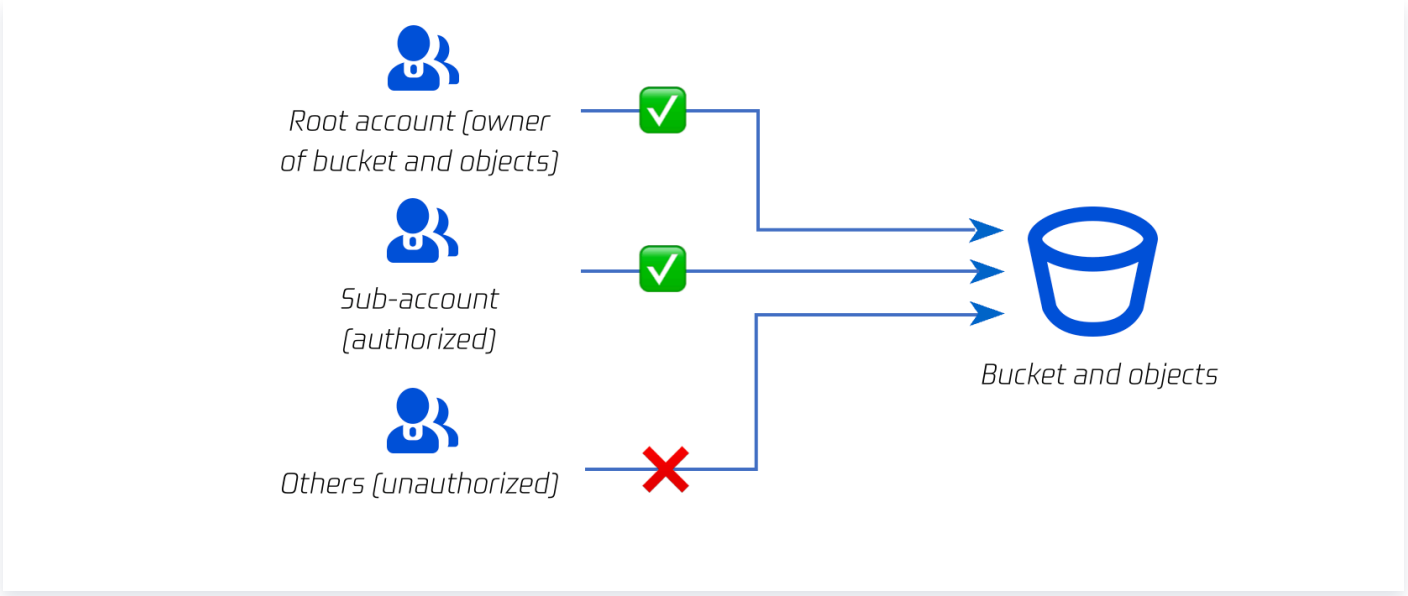
使用統計モジュールでは、現在のバケットの過去 2 日間のパブリックネットワークにおける下りトラフィックと GET リクエスト統計を表示することができます。詳細なデータを表示するには、 データモニタリングをクリックするか、バケット構成ページからデータモニタリングに進んでください。



クラウドアクセスマネジメント
アクセス権限設定の説明
アクセス管理の概要
アクセス制御の基本概念

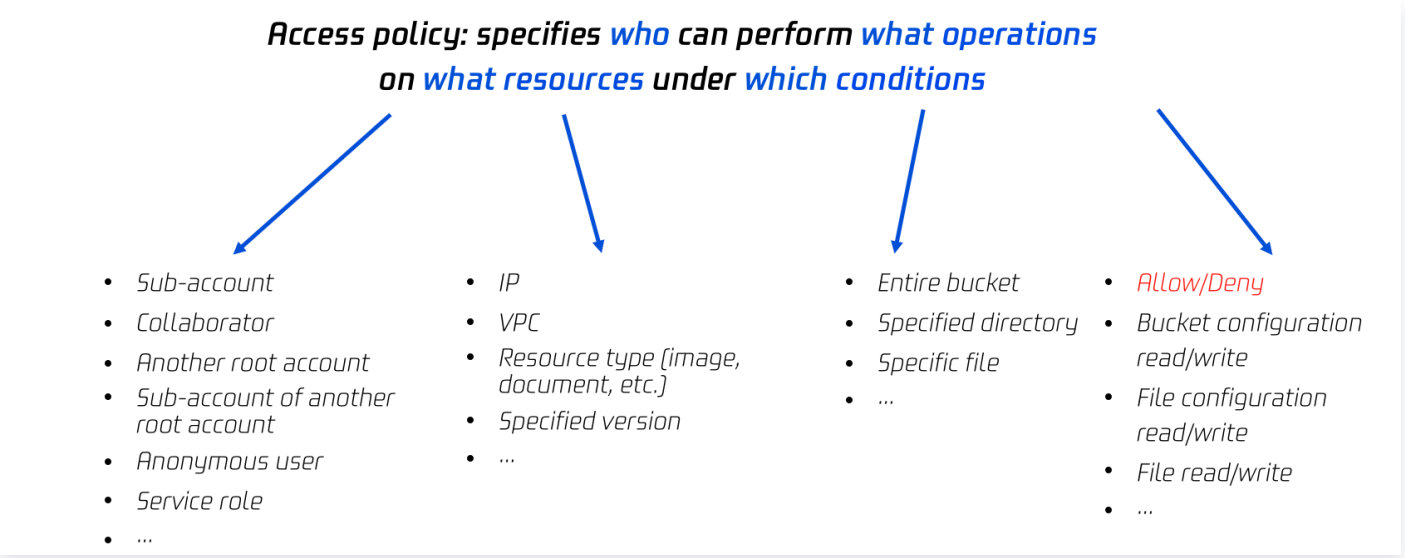
最終更新日：：2024-06-26 11:09:29

デフォルトでは、Cloud Object Storage（COS）のリソース（バケットおよびオブジェクト）はすべてプライベートです。Tencent Cloudルートアカウント（リソース所有者）でなければバケットおよびオブジェクトへのアクセス、変更を行うことはできず、他のユーザー（サブアカウント、匿名ユーザーなど）はいずれも権限承認がなければURLから直接オブジェクトにアクセスすることはできません。
Tencent Cloudサブアカウントを作成すると、アクセスポリシーによってサブアカウントに権限を付与することができます。非Tencent Cloudユーザーにリソースを開放したい場合は、リソース（バケット、オブジェクト、ディレクトリ）のパブリック権限（パブリック読み取り）を設定することで実現できます。



アクセス制御の要素

アクセス権限の付与とは、どの人が、どのような条件下で、どのリソースに対して、具体的な操作を行うかという制御機能の組み合わせをユーザーが決定できることを指します。このため、1つのアクセス権限行為の記述には、通常プリンシパル、リソース、操作、条件（オプション）の4つの要素が含まれます。



アクセス権限の要素

Tencent Cloudのプリンシパル（Principal）

ユーザーがTencent Cloudアカウントを申請する際、システムはTencent Cloudサービスへのログインに用いるルートアカウントのIDを作成します。Tencent Cloudルートアカウントはユーザー管理機能によって、異なる職責を持つユーザーを分類し管理します。ユーザーのタイプにはコラボレーター、メッセージ受信者、サブユーザーおよびロールなどがあります。具体的な定義についてはCAMのユーザーのタイプおよび用語集のドキュメントをご参照ください。

> 説明：
> 社内のある同僚への権限承認を行いたい場合は、まずCAMコンソールでサブユーザーを作成し、バケットポリシー、ACLまたはユーザーポリシーのいずれかまたは複数の手段を選択し、サブユーザーの具体的な権限を設定する必要があります。

COSのリソース（Resource）

BucketおよびObjectはCOSの基本リソースです。そのうちフォルダは特殊なオブジェクトであり、フォルダを通じてフォルダ下のオブジェクトの権限承認を行うことができます。詳細については、フォルダの権限の設定をご参照ください。
また、バケットとオブジェクトにはどちらもそれらに関連するサブリソースが存在します。
バケットのサブリソースには次のものが含まれます。

- aclおよびpolicy：バケットのアクセス制御情報です。
- website：バケットの静的ウェブサイトホスティング設定です。
- tagging：バケットのタグ情報です。
- cors：バケットのクロスドメイン設定情報です。
- lifecycle：バケットのライフサイクル設定情報です。

オブジェクトのサブリソースには次のものが含まれます。

- acl：オブジェクトのアクセス制御情報です。
- restore：アーカイブタイプのオブジェクトの復元設定です。

COSのアクション（Action）

COSではリソースに対する一連のAPI操作をご提供しています。詳細については、操作リストのドキュメントをご参照ください。

COSの条件（Condition、オプション）

オプションです。vpc、vipなどの権限発効の条件についての詳細は、CAMの発効条件をご参照ください。

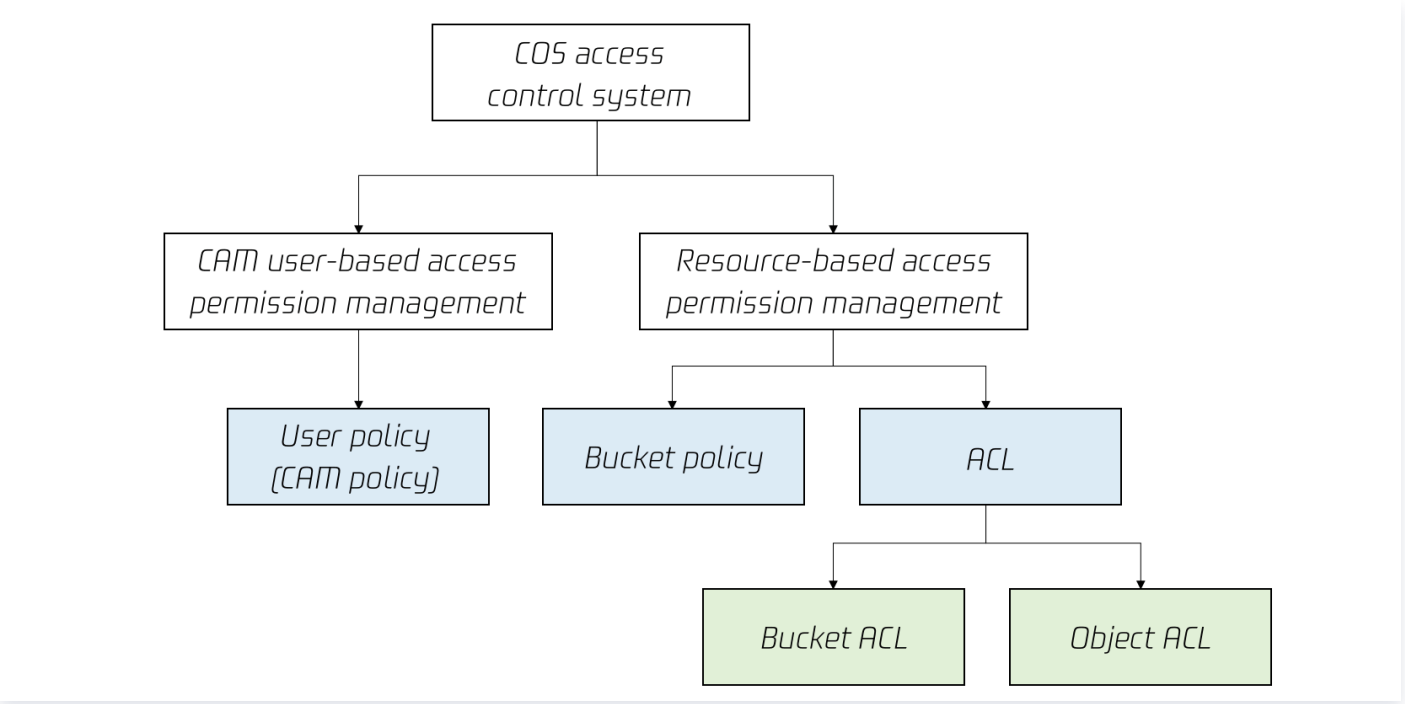
プライベートの原則

> 説明：
> デフォルトでは、Tencent Cloud COS内のリソースはすべてプライベートです。

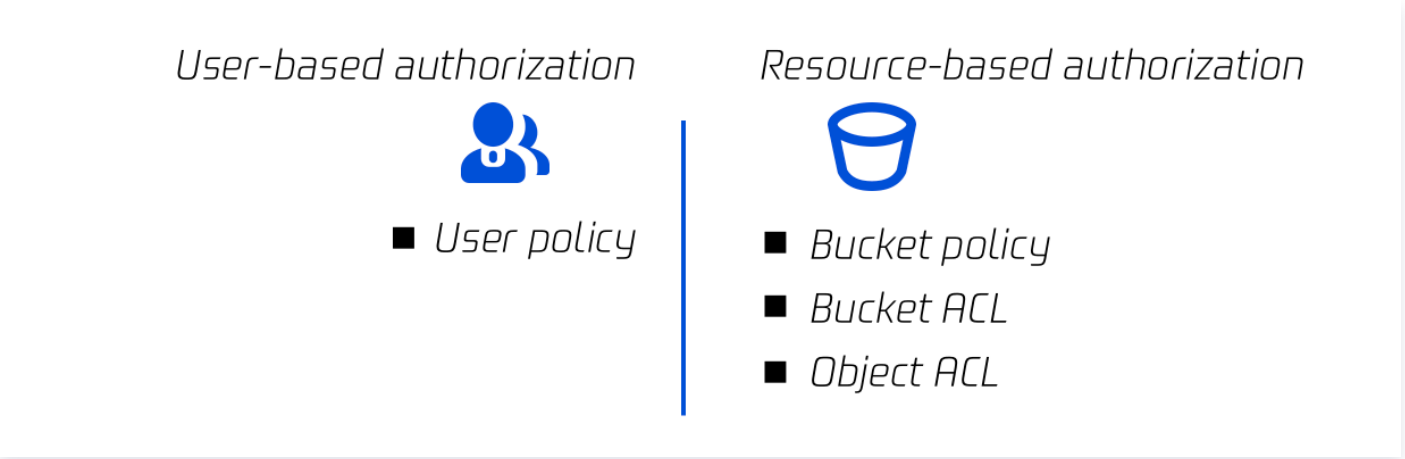
- リソース所有者（バケットのリソースを作成したTencent Cloudルートアカウント）はこのリソースに対する最高権限を有します。リソース所有者はアクセスポリシーを編集および変更することができ、第三者または匿名ユーザーに対しアクセス権限を与えることができます。
- Tencent CloudのCAM（Cloud Access Management）アカウントを使用してバケット作成またはオブジェクトのアップロードを行う場合、親アカウントにあたるルートアカウントがリソース所有者となります。
- バケット所有者のルートアカウントは他のTencent Cloudルートアカウントに対し、オブジェクトのアップロード権限を付与することができます（クロスアカウントアップロード）。この場合、オブジェクトの所有者は引き続きバケット所有者のルートアカウントとなります。

アクセス制御の複数の手段

COSは複数の権限設定方式をご提供してアクセス制御を実現しています。これにはバケットポリシー、ユーザーポリシー（CAMポリシー）、バケットACLおよびオブジェクトACLが含まれます。
これらはポリシー設定の出発点に基づき、リソースベースとユーザーベースの2種類の方式に分けられます。また権限承認の方式に基づき、ポリシーとACLの2種類の方式に分けられます。

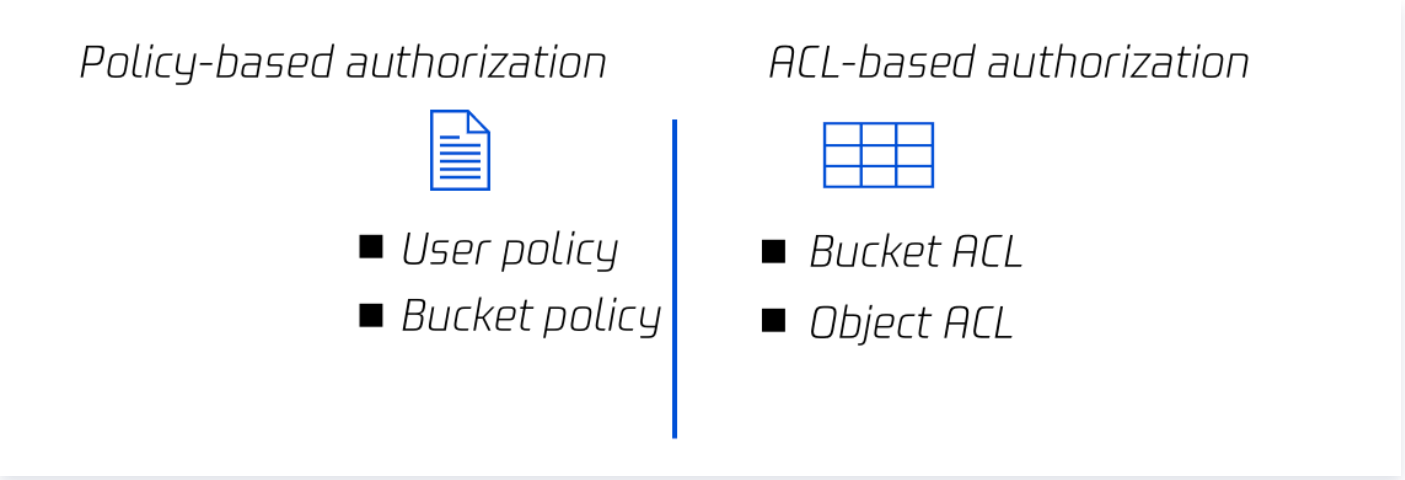


分類方法1：リソースベース vs ユーザーベース



- リソースを出発点とする場合：権限を具体的なリソースにバインドします。バケットポリシー、バケットACLおよびオブジェクトACLが含まれ、COSコンソールまたはCOS APIによって設定します。
- ユーザーを出発点とする場合：ユーザーポリシー（CAMポリシー）では権限をユーザーにバインドします。ポリシー作成時にユーザーを入力する必要はなく、リソース、アクション、条件などを指定し、CAMコンソールで設定します。

分類方法2：ポリシー vs ACL



- ポリシー：ユーザーポリシー（CAMポリシー）とバケットポリシーはどちらも完全なポリシー構文に基づいて権限承認を行うものです。権限承認の動作は各APIの対応する動作に細分化され、許可/ 拒否のエフェクトを指定することができます。
- ACL：バケットACLとオブジェクトACLはどちらもアクセス制御リスト（ACL）をベースにして実装されます。ACLはリソースにバインドされた、被付与者と付与される権限を指定したリストです。整理され抽象化された権限に対応し、許可のエフェクトのみを指定することができます。

リソースベースのポリシー

リソースベースのポリシーにはバケットポリシー、バケットACL、オブジェクトACLの3種類が含まれます。バケットとオブジェクトの次元でそれぞれアクセス制御を行うことができます。具体的には次の表をご参照ください。

次元	タイプ	記述方式	サポートするプリンシパル	サポートするリソース粒度	サポートするアクション	サポートするエフェクト
Bucket	アクセスポリシー言語（Policy）	JSON	サブアカウント、ロール、Tencent Cloudサービス、他のルートアカウント、匿名ユーザーなど	バケット、オブジェクト、プレフィックスなど	それぞれの具体的な操作	許可/明示的な拒否

Bucket	アクセス制御リスト（ACL）	XML	他のルートアカウント、サブアカウント、匿名ユーザー	バケット	整理された読み取り/書き込み権限	許可のみ
Object	アクセス制御リスト（ACL）	XML	他のルートアカウント、サブアカウント、匿名ユーザー	オブジェクト	整理された読み取り/書き込み権限	許可のみ

バケットポリシー（Bucket Policy）

バケットポリシー（Bucket Policy）はJSON言語を使用して記述され、匿名IDまたはTencent Cloudのあらゆる CAM アカウントに対し、バケット、バケット操作、オブジェクトまたはオブジェクト操作への権限付与をサポートしています。Tencent Cloud COSのバケットポリシーは、そのバケット内のほとんどすべての操作の管理に用いることができます。ACLでは記述できないアクセスポリシーを、バケットポリシーを使用して管理することをお勧めします。その他の内容については バケットポリシー のドキュメントをご参照ください。

> ⚠ 注意：
> Tencent Cloudのルートアカウントは、そのアカウント下のリソース（バケットを含む）に対する最大の権限を有しています。バケットポリシーではほとんどすべての操作を制限できますが、ルートアカウントは常にPUT Bucket Policy操作の権限を有しており、ルートアカウントがこの操作を呼び出す際、バケットポリシーのチェックは行われません。

以下は、匿名ユーザーに対し広州にあるバケットexamplebucket-1250000000内のすべてのオブジェクトへのアクセスを許可するポリシーです。署名のチェックは必要なく、そのままバケット内のすべてのオブジェクトをダウンロード（GetObject）できます。すなわち、URLを知っている匿名ユーザーは誰でもオブジェクトをダウンロードできます（パブリック読み取り方式に類似）。

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": ["cos:GetObject"],
```

```
      "Resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"]
    }
  ],
  "Version": "2.0"
}
```

アクセス制御リスト（ACL）

アクセス制御リスト（ACL）はXML言語を使用して記述する、リソースにバインドされた、被付与者と付与される権限を指定したリストです。各バケットとオブジェクトにはそれぞれに、これらとバインドされたACLがあり、匿名ユーザーまたはその他のTencent Cloudのルートアカウントに対し、基本的な読み取り/書き込み権限を付与することができます。その他の内容については、ACLのドキュメントをご参照ください。

> ⚠ 注意：
> 発行されたACLにその記述があるかどうかにかかわらず、リソースの所有者は常にリソースに対してFULL_CONTROL権限を有します。

以下はバケットACLの例です。バケット所有者（ユーザーUIN：100000000001）の完全制御権限を記述しています。

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RootAccount">
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

以下はオブジェクトACLの例です。オブジェクト所有者（ユーザーUIN：100000000001）の完全制御権限を記述し、すべての人が読み取り可能な（匿名ユーザーのパブリック読み取り）権限を付与しています。

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
        <URI>http://cam.qcloud.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

ユーザーポリシー

ユーザーはCAMで、ルートアカウント下の異なるタイプのユーザーに対し、異なる権限を付与することができます。
ユーザーポリシーとバケットポリシーの最大の違いは、ユーザーポリシーはエフェクト（Effect）、アクション（Action）、リソース（Resource）、条件（Condition、オプション）のみを記述し、プリンシパル（Principal）は記述しない点です。このため、ユーザーポリシーの作成完了後、サブユーザー、ユーザーグループまたはロールに対しバインド操作を実行する必要があります。またユーザーポリシーは、匿名ユーザーへのアクションおよびリソース権限の付与はサポートしていません。
プリセットポリシーを使用した権限のバインドを行うことも、ユーザーポリシーを自ら作成した後に指定のプリンシパルにバインドすることで、アカウント下のユーザーに対するアクセス管理を実現することもできます。その他の内容については、ユーザーポリシーのドキュメントをご参照ください。
以下は、広州にあるバケットexamplebucket-1250000000のすべてのCOS操作権限を付与するポリシーです。ポリシーを保存した後、さらにCAMのサブユーザー、ユーザーグループまたはロールにバインドすることで発効します。

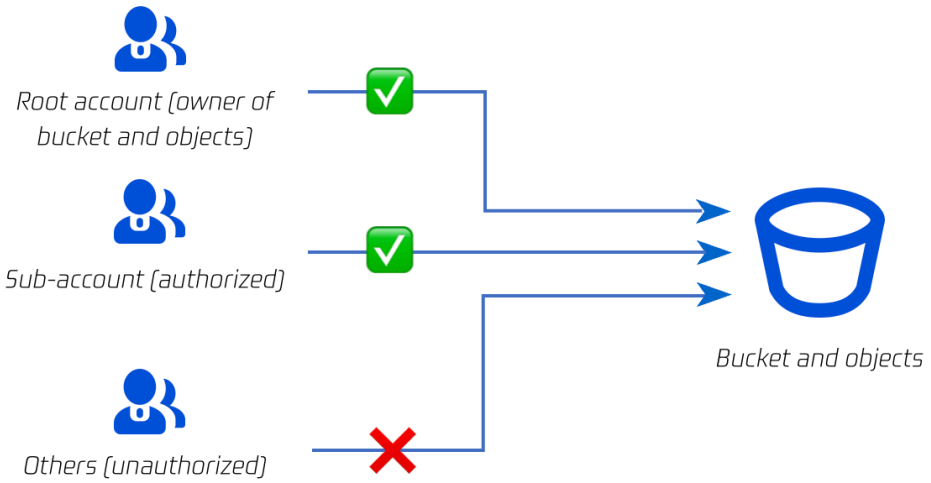
```
{
  "Statement": [
    {
```

```
        "Effect": "Allow",
        "Action": ["cos:*"],
        "Resource": [
            "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*",
            "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/"
        ]
      }
    ],
    "Version": "2.0"
}
```

COSの権限承認とID認証のフロー

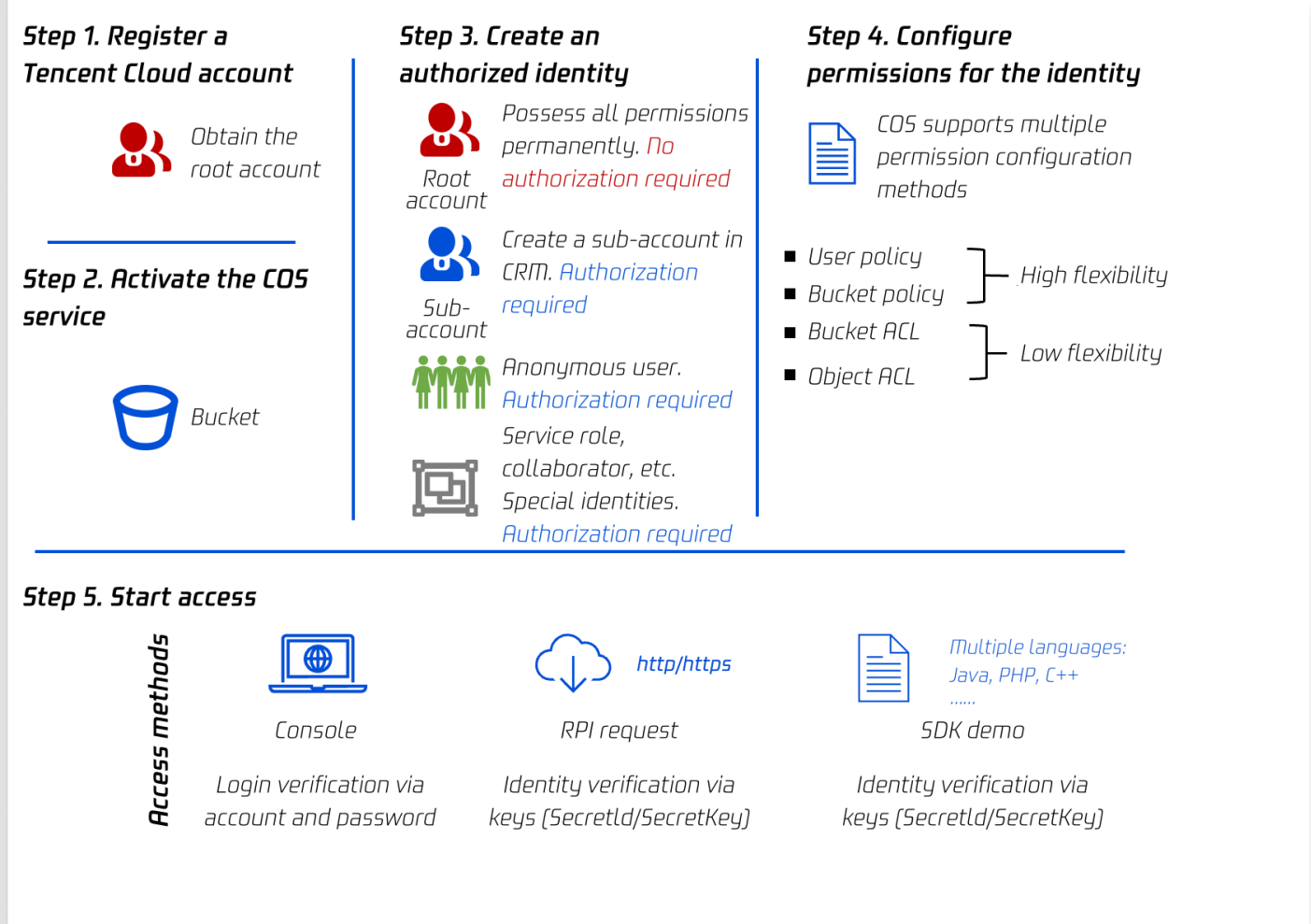
最終更新日：：2024-06-26 11:09:29

デフォルトでは、Cloud Object Storage（COS）のリソース（バケット、オブジェクト）はすべてプライベート読み取り/書き込みであり、オブジェクトのURLを取得しても、匿名ユーザーは署名がないため、URLからリソースの内容にアクセスすることはできません。



主な手順

COSの権限承認およびID認証のフローは、Tencent Cloudアカウントの登録から始まる5段階となっています。Tencent Cloudアカウントの登録、COSサービスの有効化、権限承認IDの作成、IDへの権限設定、アクセスとID認証の開始、の5つです。



ステップ1：Tencent Cloudアカウントの登録

Tencent Cloudアカウントの登録後、お客様のアカウントはルートアカウントとして存在します。ルートアカウントは最高権限を有します。

ステップ2：COSサービスのアクティブ化

COSサービスをアクティブ化すると、お客様の作成したすべてのバケットがルートアカウントの所有となります、ルートアカウントはすべてのリソースの最高使用権限を有するとともに、サブアカウントを作成してサブアカウントの権限承認を行う権限を有します。

ステップ3：権限承認IDの作成

> ⚠ 注意：
> バケットまたはオブジェクトをパブリック読み取りとして開放している場合を除き、COSへのアクセスには必ずID認証を経なければなりません。

ルートアカウントによって複数のIDを作成し、それぞれ異なるリソースの異なる使用権限を付与することができます。

- 例えば会社の同僚、特定部門のユーザーなどの指定したユーザーに権限承認を行いたい場合は、Cloud Access Management（CAM）コンソールでこれらのユーザーにサブアカウントを作成し、その後バケットポリシー、ユーザーポリシー（CAMポリシー）、バケットACL、オブジェクトACLなどの様々な権限承認方式によって、サブアカウントに指定したリソースの指定したアクセス権限を承認することができます。
- 例えば第三者が何のID認証も経ずに直接URLからオブジェクトをダウンロードすることを許可するなど、匿名ユーザーに権限承認を行いたい場合は、リソース権限をデフォルトのプライベート読み取りからパブリック読み取りに変更する必要があります。
- Tencent Cloudのその他のサービス（CDNなど）でCOSバケットを使用したい場合も、同様の権限承認のフローに従う必要があります。お客様の許可の下で、これらのサービスはサービスロールによって正しくCOSにアクセスすることができます。作成済みのサービスロールはCAMコンソールで確認できます。

異なるTencent Cloudアカウント間で権限承認を行う状況で、1つのCOSバケットにのみ権限承認を行いたい場合は、 バケットポリシーまたはバケットACLによって直接別のルートアカウントに権限承認を行うことができます。複数のCOSバケットまたは複数のTencent Cloudリソース権限承認が必要な場合は、CAMコンソールでコラボレーターIDを作成し、より広範囲の権限承認を行うことができます。

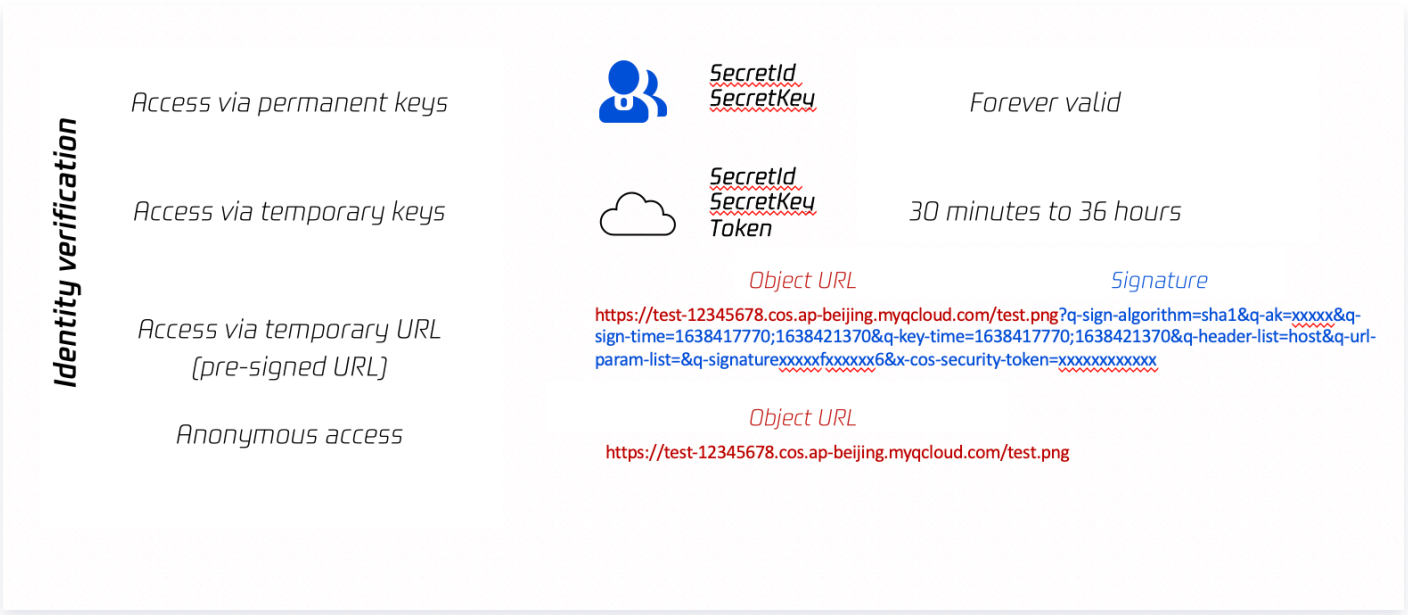
ステップ4：IDへの権限設定

COSは複数の権限設定方法をサポートしています。それにはバケットポリシー 、ユーザーポリシー（CAMポリシー） 、バケットACL およびオブジェクトACL があり、ご自身のユースケースに合った権限承認の方法を選択することができます。

ステップ5：アクセスとID認証の開始

COSにはコンソール、APIリクエスト、SDKなどの複数の手段でアクセスすることができます。セキュリティ上の観点から、バケットはデフォルトではプライベート読み取りとなっており、どの手段であってもID認証を経る必要があります。コンソールの場合は、アカウントパスワードを使用するとログインできます。APIリクエストおよびSDKの場合、すべてのユーザーはキー（SecretId/SecretKey）を使用してID認証を受ける必要があります。

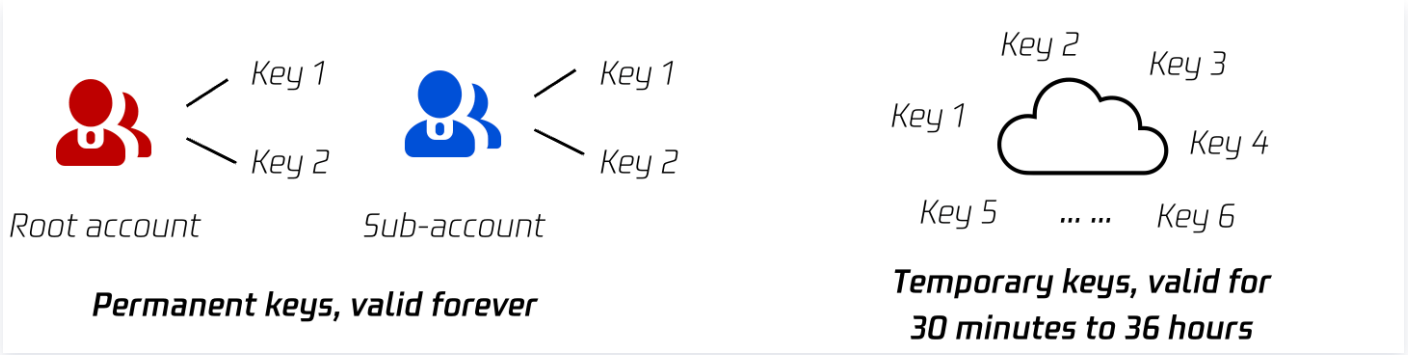
COSのID認証方式



デフォルトではCOSバケットはプライベートであり、キー （パーマネントキー、一時キー）によるCOSアクセスであっても署名付きURLを使用したアクセスであっても、ID認証の段階を経る必要があります。特別な必要性がある場合はバケットをパブリック読み取りとして開放することもできますが、これはリスクを伴う操作であり、あらゆるユーザーがID認証を経ることなく、オブジェクトURLから直接オブジェクトをダウンロードできてしまいます。

1. パーマネントキーを使用したアクセス

キー（SecretId/SecretKey）は、ユーザーがTencent Cloud APIキーにアクセスしてID認証を受ける際に必要なセキュリティ証明書であり、APIキー管理で取得することができます。各ルートアカウントおよびサブアカウントはいずれも複数のキーを作成することができます。



パーマネントキーにはSecretIdとSecretKeyが含まれます。各ルートアカウントおよびサブアカウントはいずれも2対のパーマネントキーを生成することができます。パーマネントキーはアカウントの永続的なIDを表すもので、削除しなければ長期的に有効です。詳細については、パーマネントキーを使用したCOSアクセスをご参照ください。

2. 一時キーを使用したアクセス

一時キーにはSecretId、SecretKey、Tokenが含まれます。各ルートアカウントおよびサブアカウントはいずれも複数の一時キーを生成することができます。パーマネントキーと比較して、一時キーには有効期間（デフォルトでは1800秒）があり、有効期間はルートアカウントでは最長7200秒まで、サブアカウントでは最長129600秒までそれぞれ設定可能です。詳細については、フェデレーションによる一時アクセス証明書の取得をご参照ください。
一時キーはフロントエンドの直接送信などの、一時的な権限承認のシーンに適します。信頼性の低いユーザーに対しては、パーマネントキーではなく一時キーを発行することで、安全性を高めることができます。詳細については、一時キーを使用したCOSアクセスをご参照ください。

3. 一時URLを使用したアクセス（署名付きURL）

詳しい内容および使用説明については署名付きURLを使用したCOSアクセスをご参照ください。

オブジェクトのダウンロード

第三者がバケットからオブジェクトをダウンロードできるようにし、なおかつ相手にCAMアカウントまたは一時キーなどの方法を使用させたくない場合は、署名付きURLを使用して署名を第三者に渡し、一時的なダウンロード操作を完了させることができます。有効な署名付きURLを受領した人は誰でもオブジェクトをダウンロードできます。

- コンソールまたはCOSBrowserから一時ダウンロードリンクを取得する（有効期間1～2時間）
  コンソールまたはCOSBrowserから、一時ダウンロードリンクを直接素早く取得し、ブラウザでこのリンクを直接入力することでオブジェクトをダウンロードできます。詳細については、一時リンクのスピーディーな取得のドキュメントをご参照ください。
- SDKを使用した署名付きURLの生成
  SDKを使用して、有効期間をカスタマイズした署名付きURLを一括取得することができます。詳細については、SDKを使用した署名付きURLの一括取得のドキュメントをご参照ください。
- 署名ツールを使用した署名付きURLの生成
  プログラミングに不慣れなユーザーに適しており、有効期間をカスタマイズした署名付きURLを取得することができます。詳細については、署名ツールの使用に関するドキュメントをご参照ください。
- ご自身での署名リンクの結合
  署名付きURLは実際にはオブジェクトのURLの後に署名を結合したものです。このため、SDK、署名生成ツールなどによってご自身で署名を生成し、URLと署名を結合して署名リンクとすることもできます。ただし、署名生成のアルゴリズムは複雑なため、一般的な状況ではこの方法の使用は推奨しません。

オブジェクトのアップロード

第三者がオブジェクトをバケットにアップロードできるようにし、なおかつ相手にCAMアカウントまたは一時キーなどの方法を使用させたくない場合は、署名付きURLを使用して署名を第三者に渡し、一時的なアップロード操作を完了させることができます。有効な署名付きURLを受領した人は誰でもオブジェクトをアップロードできます。

- 手段1：SDKを使用した署名付きURLの生成
  各言語のSDKがアップロード署名付きURLの生成メソッドを提供しています。生成メソッドについては、署名付きURLによるアップロード権限承認を参照し、使いやすい開発言語を選択してください。

- 手段2：ご自身での署名リンクの結合
  署名付きURLは実際にはオブジェクトのURLの後に署名を結合したものです。このため、SDK、署名生成ツールなどによってご自身で署名を生成し、URLと署名を結合して署名リンクとし、オブジェクトのアップロードに用いることもできます。署名生成のアルゴリズムは複雑なため、一般的な状況ではこの方法の使用は推奨されないことに注意が必要です。

4. 匿名アクセス

デフォルトではCOSバケットはプライベートであり、キー （パーマネントキー、一時キー）によるCOSアクセスであっても署名付きURLを使用したアクセスであっても、ID認証の段階を経る必要があります。
特別な必要性がある場合はバケットまたはオブジェクトをパブリック読み取りとして開放することもできますが、あらゆるユーザーがいかなるID認証も経ることなく、オブジェクトURLから直接オブジェクトをダウンロードできてしまいます。

> ⚠ 注意：
> リソースをパブリック読み取りとして開放することにはセキュリティ上のリスクが伴います。リソースのリンクが一度漏洩すると、誰でもアクセス可能となり、悪意あるユーザーによってトラフィックが不正使用されるおそれがあります。

バケットをパブリック読み取りとして開放する

コンソール上でバケット全体をパブリック読み取りに設定することができます。この場合、バケット内の各オブジェクトはすべてオブジェクトURLから直接ダウンロード可能となります。設定方法については、バケットアクセス権限の設定 をご参照ください。

オブジェクトをパブリック読み取りとして開放する

コンソール上で単一のオブジェクトをパブリック読み取りに設定することができます。この場合、そのオブジェクトのみ、URLから直接ダウンロード可能となり、その他のオブジェクトは影響を受けません。設定方法については、オブジェクトのアクセス権限の設定 をご参照ください。

フォルダをパブリック読み取りとして開放する

コンソール上でフォルダをパブリック読み取りに設定することができます。この場合、そのフォルダ下のすべてのオブジェクトがURLから直接ダウンロード可能となり、フォルダ外のオブジェクトは影響を受けません。設定方法については、フォルダの権限の設定 をご参照ください。

最小権限の原則説明

最終更新日：：2024-06-26 11:09:29

概要

Cloud Object Storage（COS）を使用する際、一時キーを使用して対応するリソースの操作権限をユーザーに与えるか、またはご自身のサブユーザーまたはコラボレーターに対し、対応するユーザーポリシーを付与することで、それらのアカウントがCOS上のリソース操作を補助できるよう許可するか、あるいはバケットに対応するバケットポリシーを追加し、指定のユーザーがバケット内で指定された操作を行い、指定されたリソースを操作することを許可するかの、いずれかが必要となる可能性があります。これらの権限設定を行う際は、データアセットのセキュリティを保障するため、必ず最小権限の原則に従う必要があります。
最小権限の原則とは、権限承認を行う際、必ず権限の範囲を明確化し、指定するユーザーに、どのような条件の下で、どのような操作を実行でき、どのようなリソースにアクセスできる権限を承認するかを明確にする必要があることを指します。

注意事項

権限承認の際は最小権限の原則を厳守し、ユーザーに対し限られた操作（例えば `action:GetObject` の権限承認など）の実行、限られたリソース（例えば `resource:examplebucket-1250000000/exampleobject.txt` の権限承認など）へのアクセスのみを許可することをお勧めします。
大きすぎる権限を与えることで、意図しない越権操作が行われ、データセキュリティリスクが生じることを避けるため、ユーザーにすべてのリソースへのアクセス（例えば `resource:*` など）、またはすべての操作（例えば `action:*` など）の権限を承認することは可能な限り避けるよう強く推奨します。
潜在的なデータセキュリティリスクの例を次に挙げます。

- データ漏洩：ユーザーに指定のリソースをダウンロードする権限、例えば `examplebucket-1250000000/data/config.json` および `examplebucket-1250000000/video/` のダウンロード権限を与えても、権限ポリシーに `examplebucket-1250000000/*` を設定していた場合、このバケット下のすべてのオブジェクトのダウンロードが許可されることになり、越権操作行為があった場合は意図しないデータ漏洩が発生します。
- データが上書きされる：ユーザーにリソースをアップロードする権限、例えば `examplebucket-1250000000/data/config.json` および `examplebucket-1250000000/video/` のアップロード権限を与えても、権限ポリシーに `examplebucket-1250000000/*` を設定していた場合、このバケット下のすべてのオブジェクトのアップロードが許可されることになり、越権操作行為があった場合は意図しないオブジェクトの上書きが発生します。このようなリスクを防止するためには、最小権限の原則に従うほか、バージョン管理によってデータのすべての過去バージョンを保持し、追跡に役立てることも可能です。
- 権限の漏洩：ユーザーにバケットのオブジェクトリスト `cos:GetBucket` の出力を許可しても、権限ポリシーに `cos:*` を設定していた場合、バケット権限の再承認、オブジェクトの削除、バケットの削除を含む、このバケットのすべての操作が許可されることになり、データに極めて大きなリスクをもたらします。

使用ガイド

最小権限の原則の下では、ポリシーに次の情報を明確に指定する必要があります。

- プリンシパル（principal）：どのサブユーザー（ユーザーIDの入力が必要）、コラボレーター（ユーザーIDの入力が必要）、匿名ユーザーまたはユーザーグループに権限を付与したいかを明確に指定する必要があります。一時キーを使用してアクセスする場合、この項の指定は不要です。
- ステートメント（statement）：次のいくつかのパラメータに、対応するパラメータを明確に入力します。
  - エフェクト（effect）：このポリシーが「許可」であるか「明示的な拒否」であるかを明確に述べる必要があります。allowとdenyの2種類が含まれます。
  - アクション（action）：このポリシーが許可または拒否するアクションを明確に述べる必要があります。アクションは単一のAPIのアクションまたは複数のAPIアクションのセットとすることができます。
  - リソース（resource）：このポリシーが権限を承認する具体的なリソースを明確に述べる必要があります。リソースは6つの部分で記述されます。リソース範囲の指定は、`exampleobject.jpg` などの指定されたファイル、または `examplePrefix/*` などの指定されたディレクトリとすることができます。業務上の必要がない限り、すべてのリソースにアクセスする権限（ワイルドカード `*` ）をユーザーにみだりに付与しないでください。
  - 条件（condition）：ポリシー発効の制約条件を記述します。条件はオペレーター、アクションキーとアクション値から構成されています。条件値には時間、IPアドレスなどの情報を含めることができます。

一時キーの最小権限ガイド

一時キー申請の過程で、権限ポリシーのPolicyフィールドを設定することで、操作およびリソースを制限し、権限を指定された範囲内に限定することができます。一時キー生成に関する説明については、一時キーの生成および使用ガイドのドキュメントをご参照ください。

権限承認の例

Java SDKを使用して、あるオブジェクトへのアクセス権限を承認

Java SDKを使用して、バケット `examplebucket-1250000000` 内のオブジェクト `exampleObject.txt` のダウンロード権限を承認したい場合、それに応じて設定する必要があるコードは次のようになります。

```
// githubが提供するmaven統合メソッドによってjava sts sdkをインポートします
import java.util.*;
import org.json.JSONObject;
import com.tencent.cloud.CosStsClient;

public class Demo {
    public static void main(String[] args) {
        TreeMap<String, Object> config = new TreeMap<String, Object>();
```

```
        try {
            String secretId = System.getenv("secretId");//ユーザーのSecretIdです。サブアカウントのキーを使用し、権限承認は最小権限ガイドに従って行い、使用上のリスクを低減させることをお勧めします。サブアカウントキーの取得については、https://cloud.tencent.com/document/product/598/37140をご参照ください
            String secretKey = System.getenv("secretKey");//ユーザーのSecretKeyです。サブアカウントのキーを使用し、権限承認は最小権限ガイドに従って行い、使用上のリスクを低減させることをお勧めします。サブアカウントキーの取得については、https://cloud.tencent.com/document/product/598/37140をご参照ください
            // ご自身のSecretIdに置き換えます
            config.put("SecretId", secretId);
            // ご自身のSecretKeyに置き換えます
            config.put("SecretKey", secretKey);

            // 一時キーの有効期間の単位は秒であり、デフォルトでは1800秒、最長7200秒まで設定可能です
            config.put("durationSeconds", 1800);

            // ご自身のbucketに置き換えます
            config.put("bucket", "examplebucket-1250000000");
            // bucketの所在リージョンに置き換えます
            config.put("region", "ap-guangzhou");

            // ここを許可するパスのプレフィックスに変更します。ご自身のウェブサイトのユーザーログインセッションによって、a.jpg または a/* または * などのように、アップロードを許可する具体的なパスを判断することができます。
            // 「*」を入力すると、ユーザーにすべてのリソースへのアクセスを許可することになります。業務上の必要がない限り、最小権限の原則に従って、対応する範囲のアクセス権限をユーザーに与えてください。
            config.put("allowPrefix", "exampleObject.txt");

            // キーの権限リストです。シンプルアップロード、フォームアップロード、マルチパートアップロードには次の権限が必要です。その他の権限リストについては https://cloud.tencent.com/document/product/436/31923をご覧ください
            String[] allowActions = new String[] {
                    // データをダウンロードします
                    "name/cos:GetObject"
            };
            config.put("allowActions", allowActions);
```

```
            JSONObject credential = CosStsClient.getCredential(config);
            //成功すると一時キー情報が返されます。次のようにキー情報を印刷します
            System.out.println(credential);
        } catch (Exception e) {
            //失敗するとエラーがスローされます
            throw new IllegalArgumentException("no valid secret !");
        }
    }
}
```

APIを使用して、あるオブジェクトへのアクセス権限を承認

APIを使用して、バケット `examplebucket-1250000000` 内のオブジェクト `exampleObject.txt` およびディレクトリ `examplePrefix` 下のすべてのオブジェクトのダウンロード権限を承認したい場合、そのために書き込む必要があるPolicyは次のようになります。

```
{
  "version": "2.0",
  "statement":[
    {
      "action":[
        "name/cos:GetObject"
      ],
      "effect": "allow",
      "resource":[
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-
1250000000/exampleObject.txt",
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-
1250000000/examplePrefix/*"
      ]
    }
  ]
}
```

署名付きURLの最小権限ガイド

署名付きURL方式によって一時的なアップロードおよびダウンロード操作を実現できます。また、署名付きURLは誰にでも送信することができ、有効な署名付きURLを受領した人は誰でもオブジェクトをアップロードまたはダウンロードできます。

> ⚠ 注意：
> 一時キーとパーマネントキーはどちらも署名付きURLの生成に用いることができますが、最小権限の原則に従って一時キーの生成を行い、一時キーを使用して署名の計算を行うことを強く推奨します。セキュリティ上のリスクを防止するため、権限が大きすぎるパーマネントキーは可能な限り使用しないでください。

権限承認の例

ユーザーに対し、署名付きURLを使用してオブジェクトをダウンロードする権限を承認

一時キーを使用して署名付きのダウンロードリンクを生成し、返す必要のあるいくつかのパブリックヘッダー（例えばcontent-type、content-languageなど）の上書きを設定します。Javaのサンプルコードは次のとおりです。

```
// 取得した一時キー（tmpSecretId、tmpSecretKey、sessionToken）を渡します
String tmpSecretId = "SECRETID";
String tmpSecretKey = "SECRETKEY";
String sessionToken = "TOKEN";
COSCredentials cred = new BasicSessionCredentials(tmpSecretId,
tmpSecretKey, sessionToken);
// bucketのリージョンを設定します。COSリージョンの略称については
https://cloud.tencent.com/document/product/436/6224をご参照ください
// clientConfigにはregion、https（デフォルトではhttp）、タイムアウト、プロキシなど
を設定するsetメソッドが含まれます。使用にあたってはソースコードまたはよくあるご質問の
Java SDKのパートを参照できます
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// httpsプロトコルを使用したURLを生成したい場合はこの行を設定します。設定することをお
勧めします。
// clientConfig.setHttpProtocol(HttpProtocol.https);
// cosクライアントを生成します
COSClient cosClient = new COSClient(cred, clientConfig);
// バケットの命名形式はBucketName-APPIDです
String bucketName = "examplebucket-1250000000";
// ここでのkeyはオブジェクトキーです。オブジェクトキーはオブジェクトのバケット内での固
有識別子です
String key = "exampleobject";
GeneratePresignedUrlRequest req =
        new GeneratePresignedUrlRequest(bucketName, key,
HttpMethodName.GET);
// ダウンロード時に返されるhttpヘッダーを設定します
```

```
ResponseHeaderOverrides responseHeaders = new ResponseHeaderOverrides();
String responseContentType = "image/x-icon";
String responseContentLanguage = "zh-CN";
// レスポンスヘッダーに含まれるファイル名の情報を設定します
String responseContentDispositon = "filename=\"exampleobject\"";
String responseCacheControl = "no-cache";
String cacheExpireStr =
        DateUtils.formatRFC822Date(new Date(System.currentTimeMillis() +
24L * 3600L * 1000L));
responseHeaders.setContentType(responseContentType);
responseHeaders.setContentLanguage(responseContentLanguage);
responseHeaders.setContentDisposition(responseContentDispositon);
responseHeaders.setCacheControl(responseCacheControl);
responseHeaders.setExpires(cacheExpireStr);
req.setResponseHeaders(responseHeaders);
// 署名の有効期限を設定します(オプション)。設定しない場合は、デフォルトで
ClientConfigの署名有効期限(1時間)を使用します
// ここでは署名が30分後に期限切れとなるよう設定します
Date expirationDate = new Date(System.currentTimeMillis() + 30L * 60L *
1000L);
req.setExpiration(expirationDate);
URL url = cosClient.generatePresignedUrl(req);
System.out.println(url.toString());
cosClient.shutdown();
```

ユーザーポリシーの最小権限ガイド

ユーザーポリシーとはCAMコンソールで追加するユーザー権限ポリシーを指し、ユーザーにCOSリソースへのアクセス権限を与えるために用いられます。ユーザーアクセスポリシー概要の設定説明については、 アクセスポリシーの言語概要のドキュメントをご参照ください。

権限承認の例

アカウントに対し、あるオブジェクトへのアクセス権限を承認

アカウントUIN `100000000001` に対し、バケット `examplebucket-1250000000` 内のオブジェクト `examleObject.txt` のダウンロード権限を承認したい場合、それに応じたアクセスポリシーは次のようになります。

```
{
    "version": "2.0",
```

```
    "principal":{
       "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000001"
       ]
    },
     "statement":[
         {
             "action":[
                 "name/cos:GetObject"
             ],
             "effect": "allow",
             "resource":[
                 "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000.ap-guangzhou.myqcloud.com/exampleObject.txt"
             ]
         }
     ]
}
```

サブアカウントに対し、あるディレクトリへのアクセス権限を承認

サブアカウントUIN `100000000011` （ルートアカウントのUINは `100000000001` ）に対し、バケット `examplebucket-1250000000` 内のディレクトリ `examplePrefix` 下のオブジェクトのダウンロード権限を承認したい場合、それに応じたアクセスポリシーは次のようになります。

```
{
    "version": "2.0",
    "principal":{
       "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000011"
       ]
    },
     "statement":[
         {
             "action":[
                 "name/cos:GetObject"
             ],
             "effect": "allow",
             "resource":[
```

```
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000.ap-guangzhou.myqcloud.com/examplePrefix/*"
            ]
        }
    ]
}
```

バケットポリシーの最小権限ガイド

バケットポリシーとはバケット内で設定するアクセスポリシーを指し、指定のユーザーがバケットおよびバケット内のリソースに対し指定された操作を行うことを許可します。バケットポリシーの設定については、バケットポリシーの追加のドキュメントをご参照ください。

権限承認の例

サブアカウントに対し特定のオブジェクトへのアクセス権限を承認

サブアカウントUIN `100000000011`（ルートアカウントのUINは `100000000001`）に対し、バケット `examplebucket-1250000000` 内のオブジェクト `exampleObject.txt` およびディレクトリ `examplePrefix` 下のすべてのオブジェクトのダウンロード権限を承認したい場合、それに応じたアクセスポリシーは次のようになります。

```
{
  "Statement": [
    {
      "Action": [
        "name/cos:GetObject"
      ],
      "Effect": "allow",
      "Principal": {
        "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000011"
        ]
      },
      "Resource": [
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-
1250000000/exampleObject.txt",
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-
1250000000/examplePrefix/*"
      ]
    }
```

```
    ],
    "version": "2.0"
}
```

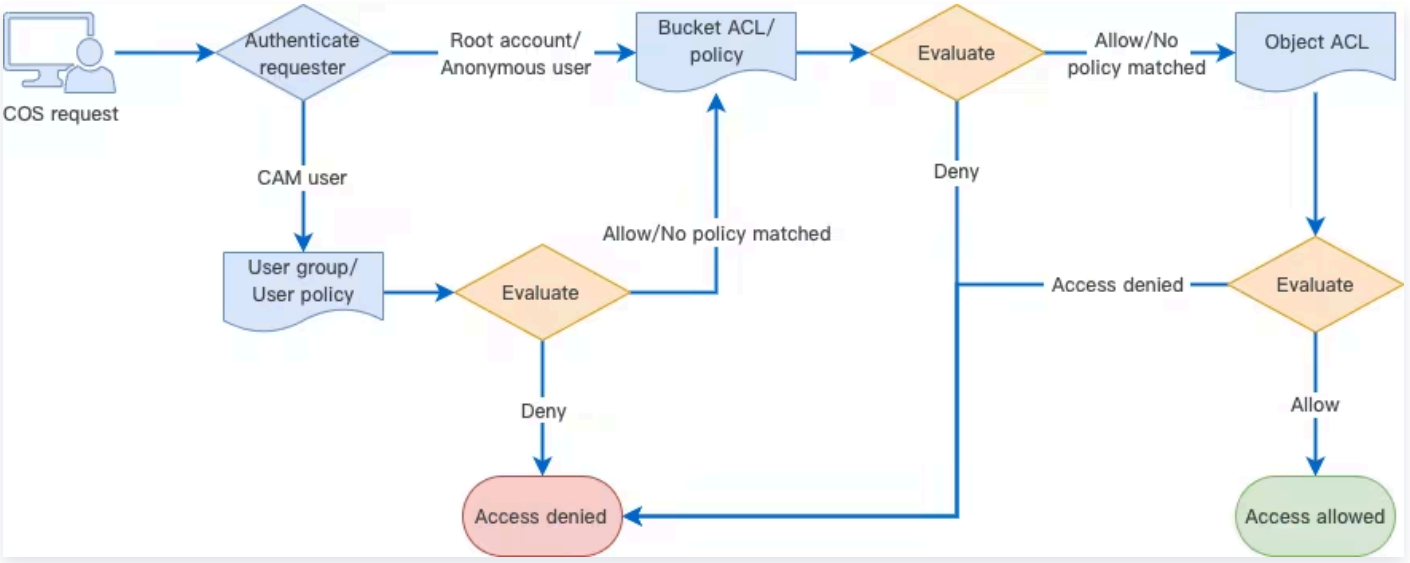
アクセスポリシーの評価フロー

最終更新日：：2024-06-26 11:09:29

COSバケットおよびバケット内のリソースにアクセスする際は権限承認を経てからアクセスする必要があります。Tencent Cloudの権限システムでは、リソースが所属するルートアカウントはデフォルトでバケットおよびバケット内のリソースに対しすべての管理権限を有します。CAMユーザー（その他のルートアカウント、コラボレーター、サブアカウント）および匿名ユーザーなどのその他のタイプのユーザーは、ルートアカウントによる権限承認を経てからでなければアクセスできません。
アカウントのアクセスポリシーには、ユーザーグループポリシー、ユーザーポリシー、バケットアクセス制御リスト（ACL）、バケットポリシー（Policy）などの異なるポリシータイプがあります。アクセスポリシーの評価においては、次のいくつかの重要な要素があります。

1. ユーザーのID認証：ユーザーがCOS上のリソースにアクセスする場合、次の2つの状況があります。
   - リクエスト署名がある場合、COSはリクエスト署名からユーザーのアカウント情報を解析した後、リクエストをCAMに転送してID認証を行います。
   - 署名のないリクエストの場合は、匿名ユーザーと認識され、認証の次の段階に進みます。
2. アクセスポリシーの種類：アクセスポリシーにはユーザーグループ、ユーザー、バケットなどの複数のタイプが含まれます。アクセスポリシーの順序はアクセスポリシーの種類によって決定されます。
3. ポリシーのコンテキスト情報：ユーザーのリソースアクセスリクエストを処理する際は、ユーザーグループポリシー、ユーザーポリシー、バケットポリシーなどの複数のポリシーに記録された権限の詳細に基づいて総合的に判断し、リクエストを許可するかどうかを最終的に決定します。

アクセスポリシーの評価フロー



Tencent Cloud COSがリクエストを受信すると、最初にリクエスト送信者のIDを確認し、リクエスト送信者が関連の権限を有しているかどうかを検証します。検証のプロセスには、ユーザーポリシー、バケットアクセスポリシー、リソースベースのアクセス制御リストのチェックが含まれ、リクエストに対する認証を行います。

Tencent Cloud COSはリクエストを受信した際、最初にID認証を行い、ID認証の結果に基づいてリクエスト送信者のIDを分類します。IDのカテゴリーに応じて異なるアクションがとられる可能性があります。

1. 検証済みのTencent Cloudルートアカウント：ルートアカウントはその所属リソースに対しすべての操作権限を有します。一方、アカウントに属さないリソースについては、リソース権限の評価が必要であり、認証に合格するとリソースへのアクセスが許可されます。
2. 検証済みのCAMユーザー（サブアカウントまたはコラボレーター）：ユーザーポリシーの評価 —— CAMユーザーは必ず親アカウントにあたるルートアカウントの権限承認を受けていなければ、関連のアクセスが許可されません。CAMユーザーが他のルートアカウントに属するリソースにアクセスしたい場合は、CAMユーザーが属するルートアカウントのリソース権限の評価が必要であり、認証に合格するとリソースへのアクセスが許可されます。
3. ID特性を持たない匿名ユーザー：リソース権限の評価 —— バケットアクセスポリシーまたはバケット、オブジェクトのアクセス制御リストの権限を評価し、認証に合格するとリソースへのアクセスが許可されます。
4. 上記のユーザーカテゴリー以外のリクエスト送信者：アクセスが拒否されます。

アクセスポリシーの評価根拠

Tencent Cloudの権限システムでは、アクセスポリシーの評価フローにおいて、全プロセスでポリシーのコンテキスト情報に基づいて権限の評価を行います。また同時に次のいくつかの基本原則があります。

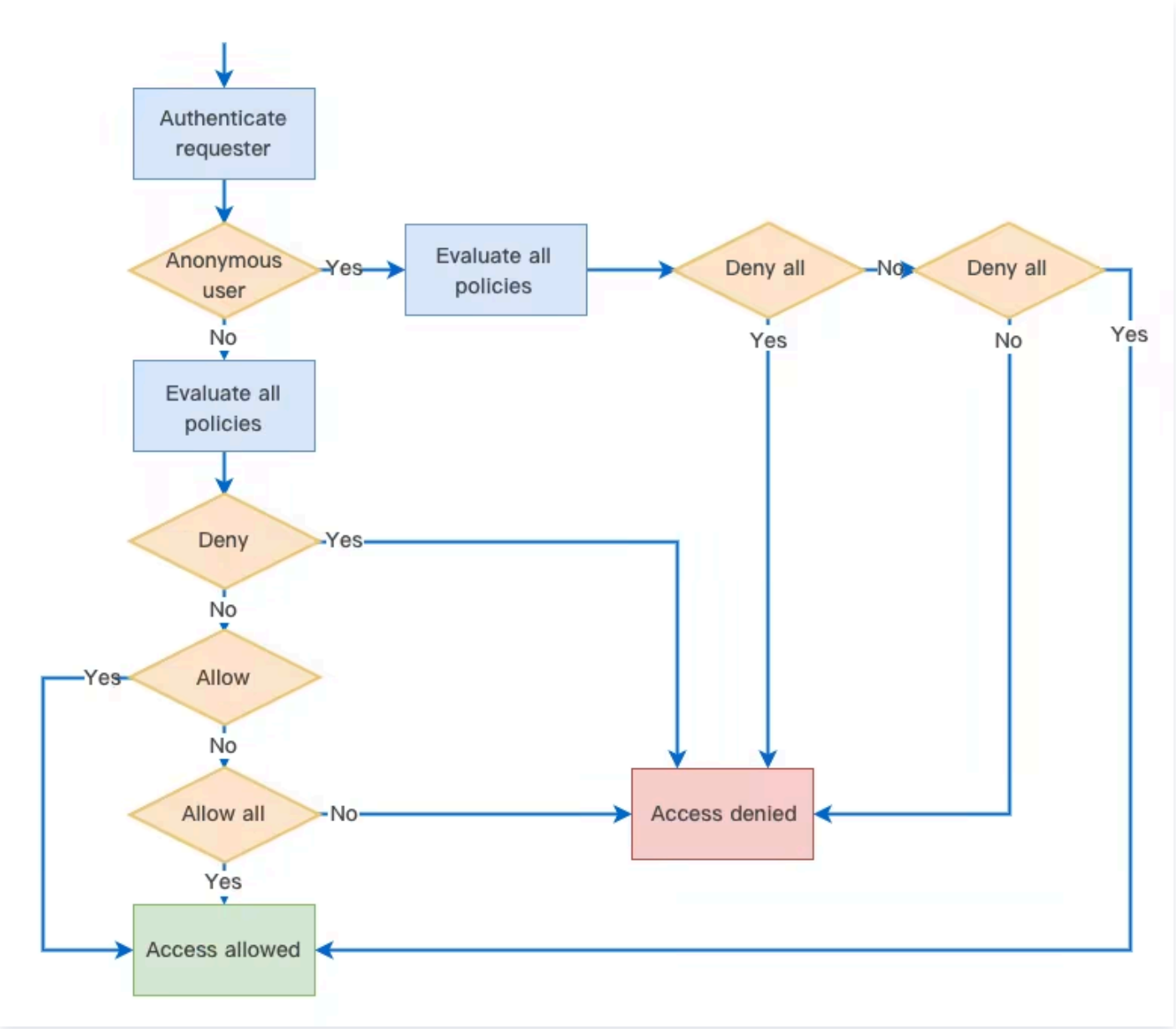
1. デフォルトでは、すべてのリクエストが暗黙的に拒否（deny）されます。ルートアカウントはデフォルトでアカウント下のすべてのリソースのアクセス権限を有します。
2. ユーザーグループポリシー、ユーザーポリシー、バケットポリシーまたはバケット/オブジェクトのアクセス制御リストに明示的な許可が存在する場合は、このデフォルト値を上書きします。
3. いずれかのポリシーの中に明示的な拒否がある場合は、あらゆる許可を上書きします。
4. 有効な権限の範囲はIDベースのポリシー（ユーザーグループポリシー、ユーザーポリシー）とリソースベースのポリシー（バケットポリシーまたはバケット/オブジェクトのアクセス制御リスト）を合わせたものとなります。

説明：

- 明示的な拒否：ユーザーポリシー、ユーザーグループポリシー、バケットPolicyの中で特定のユーザーに対して明確なDenyポリシーが存在する場合。例えば、ルートアカウントがユーザーポリシーの中で、サブユーザーUIN 100000000011が `GET Object` 操作を行うことを明確にDenyと設定している場合、そのサブユーザーはそのルートアカウント下のオブジェクトリソースをダウンロードすることはできません。
- 明示的な許可：ユーザーポリシー、ユーザーグループポリシー、バケットPolicy、バケットACLの中で、 `grant-\*` によって特定のユーザーを明確に指定し、特定のユーザーに対し明確なAllowポリシーを設けます。
- すべての人を拒否：バケットPolicyの中で、 `Deny anyone` と明確に指定します。すべての人を拒否すると、任意の署名なしリクエストが拒否されます。署名付きリクエストに対してはIDベースのポリシーによる認証が行われます。

- すべての人を許可：バケットPolicyの中で、`Allow anyone`と明確に指定するか、またはバケットACLの中で`public-\*`と明確に指定します。
- 有効な権限の範囲はIDベースのポリシーとリソースベースのポリシーを合わせたものとなります。1回の完全な認証において、COSは最初にユーザーのIDを解析し、ユーザーのIDに基づいて、そのユーザーがアクセス権限を持つリソースの権限チェックを行います。同時にリソースベースのポリシーに基づき、ユーザーを匿名ユーザーとみなして権限チェックを行います。2回のチェックのうち1回が成功するとアクセスが許可されます。

アクセスポリシーの評価根拠は次の図のとおりです。最初にリクエストの署名の有無に基づき、ユーザーが匿名ユーザーかどうかを評価します。ユーザーが匿名ユーザーであれば、「すべての人を拒否」または「すべての人を許可」のポリシーがないかどうかを評価し、これに基づいてアクセス許可またはアクセス拒否の判断を行います。ユーザーが正当なCAMユーザーまたはリソースを所有するルートアカウントの場合は、明示的な拒否、明示的な許可または「すべての人を許可」のポリシーがないかどうかを評価し、これに基づいてアクセス許可またはアクセス拒否の判断を行います。



ポリシーのコンテキスト情報

ポリシーのコンテキスト情報とは、ポリシーに記録される権限の詳細を指します。最小権限の原則の下で、ユーザーはポリシーの中で次の情報を明確に指定する必要があります。

- プリンシパル（principal）：どのサブユーザー（ユーザーIDの入力が必要）、コラボレーター（ユーザーIDの入力が必要）、匿名ユーザーまたはユーザーグループに権限を付与したいかを明確に指定する必要があります。一時キーを使用してアクセスする場合、この項の指定は不要です。
- ステートメント（statement）：次のいくつかのパラメータに、対応するパラメータを明確に入力します。
- エフェクト（effect）：このポリシーが「許可」であるか「明示的な拒否」であるかを明確に述べる必要があります。allowとdenyの2種類が含まれます。
- アクション（action）：このポリシーが許可または拒否するアクションを明確に述べる必要があります。アクションは単一のAPIのアクションまたは複数のAPIアクションのセットとすることができます。

- リソース（resource）：このポリシーが権限を承認する具体的なリソースを明確に述べる必要があります。リソースは6セグメント式で記述します。リソース範囲の指定は、exampleobject.jpgなどの指定されたファイル、またはexamplePrefix/*などの指定されたディレクトリとすることができます。業務上の必要がない限り、すべてのリソースにアクセスする権限（ワイルドカード*）をユーザーにみだりに付与しないでください。
- 条件（condition）：ポリシー発効の制約条件を記述します。条件はオペレーター、アクションキーとアクション値から構成されています。条件値には時間、IPアドレスなどの情報を含めることができます。

ポリシーの作成は一定のポリシー構文に従って行う必要があります。アクセスポリシーの言語概要を参照し、業務上の必要性に応じて作成することができます。ユーザーポリシーおよびバケットポリシーの作成例については、ユーザーポリシー構文の構造およびバケットポリシーの例をそれぞれご参照ください。

アクセスポリシーの評価の例

ルートアカウントUIN 100000000001がサブアカウントUIN 100000000011にユーザープリセットポリシーをバインドし、このサブアカウントにルートアカウント下のリソースを読み取り専用として許可したとします。このユーザーポリシーの詳細は次のとおりです。このユーザーポリシーはこのサブアカウントに対し、すべての `List` 、 `Get` 、 `Head` 操作および `OptionsObject` 操作の実行を許可しています。

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "cos:List*",
                "cos:Get*",
                "cos:Head*",
                "cos:OptionsObject"
            ],
            "resource": "*",
            "effect": "allow"
        }
    ]
}
```

また、ルートアカウントはあるプライベート読み取り/書き込みバケット `examplebucket-1250000000` に次のバケットポリシーを追加しました。

```
{
  "Statement": [
    {
```

```
      "Principal": {
        "qcs": [
          "qcs::cam::anyone:anyone"
        ]
      },
      "Effect": "Deny",
      "Action": [
        "name/cos:GetObject"
      ],
      "Resource": [
        "qcs::cos:ap-guangzhou:uid/100000000011:examplebucket-
1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

このバケットポリシーはすべてのユーザーのオブジェクトダウンロード実行（ `GetObject` ）の操作を明示的に拒否しています。このため、アクセスポリシー評価のフローにおいては、次が行われます。

1. このサブアカウントが署名パラメータによって `GetObject` をリクエストした場合、このリクエストが表すユーザーのIDが対応するユーザーポリシーとマッチした場合、アクセスポリシーの評価検証は合格となります。
2. このサブアカウントが署名のないパラメータによって `GetObject` をリクエストした場合、システムによって匿名のリクエストと判断され、バケットポリシーによってアクセスが拒否されます。

権限制御方法の紹介
バケットポリシー

最終更新日：：2025-11-24 16:49:47

バケットポリシーは設定したバケットおよびバケット内のオブジェクトに対して機能します。バケットポリシーによってCAMサブアカウント、その他のルートアカウント、また匿名のユーザーに対しても、バケットおよびバケット内のオブジェクトの操作権限を付与することができます。

概要

> ⚠ 注意：
> Tencent Cloudのルートアカウントは、そのアカウント下のリソース（バケットを含む）に対する最大の権限を有しています。バケットポリシーではほとんどすべての操作を制限できますが、ルートアカウントは常にPUT Bucket Policy操作の権限を有しており、ルートアカウントがこの操作を呼び出す際、バケットポリシーのチェックは行われません。

バケットポリシー（Bucket Policy）はJSON言語を使用して記述し、匿名IDまたはTencent Cloudのあらゆる CAM アカウントに対し、バケット、バケット操作、オブジェクトまたはオブジェクト操作の権限を付与できます。Tencent Cloud COSのバケットポリシーは、そのバケット内のほとんどすべての操作の管理に用いることができます。ACLでは記述できないアクセスポリシーを、バケットポリシーを使用して管理することをお勧めします。

ユースケース

> ⚠ 注意：
> バケット作成およびバケットリスト取得という2つのサービスレベルの操作権限は、CAMコンソール で設定する必要があります。

そのCOSバケットに誰がアクセス可能かをお知りになりたい場合は、バケットポリシーの使用をお勧めします。バケットを検索し、バケットポリシーをチェックすることで、アクセス可能な人が誰かを知ることができます。次のようなケースで推奨されます。

- 特定のバケットについて権限を付与する
- バケットポリシーがACLに比べてより柔軟である
- ユーザーポリシーと異なり、バケットポリシーがクロスアカウント権限付与および匿名ユーザーへの権限付与をサポートしている

バケットポリシーの構成

バケットポリシーはJSON言語を使用して記述します。構文は、プリンシパル（principal）、エフェクト（effect）、アクション（action）、リソース（resource）、条件（condition）などの基本要素を含めたアクセスポリシー言語の統一ルールに従います。詳細については、アクセスポリシーの言語概要をご参照ください。このうち、バケットポリシーのリソース範囲はそのバケット内に限定されますが、バケット全体、指定されたディレクトリ、指定されたオブジェクトに対する権限を付与することができます。

> ⚠ 注意：
> バケットポリシーを追加する際は、必ず業務の必要性に応じて、最小権限の原則に従って権限を付与してください。他のユーザーに対し、すべてのリソース `(resource:*)` またはすべてのアクション `(action:*)` の権限を直接与えてしまうと、権限が大きすぎるためにデータセキュリティ上のリスクが生じる場合があります。

コンソール設定の例

> ⓘ 説明：
> - COSコンソールを使用してバケットポリシーを設定する場合、ユーザーが持つバケットに関連する権限を付与する必要があります。例えば、バケットタグの取得やバケットリストアップの権限などです。
> - バケットポリシーのサイズ制限は20KBです。

例：サブアカウントが持つバケットの特定のディレクトリの全権限を付与します。設定情報は次のとおりです。

設定項目	設定値
エフェクト	許可
プリンシパル	サブアカウント、サブアカウントのUIN。このサブアカウントは現在のルートアカウント下のサブアカウントである必要があります。例：100000000011
リソース	特定のディレクトリのプレフィックス。例：`folder/sub-folder/*`
アクション	すべての操作
条件	なし

コンソールは、グラフィカル設定とポリシー設定という2種類の方式でのバケットポリシーの追加と管理をサポートしています。

グラフィカル設定

ターゲットバケットの権限管理に進み、Policy権限設定 > グラフィック設定を選択し、ポリシーの追加をクリックして、ポップアップウィンドウでポリシーの設定を行います。

ステップ1：テンプレートの選択（オプション）

COSは複数のポリシーテンプレートをご提供しており、様々な被付与者、リソース範囲の組み合わせを選択することで、お客様がバケットポリシーの設定を迅速に行えるようサポートします。テンプレートがニーズに合わない場合はこのステップをスキップするか、またはステップ2：ポリシーの設定で権限付与操作を追加または削除することができます。

被付与者

- すべてのユーザー（匿名アクセス可）：匿名ユーザーに操作権限を開放したい場合は、この項目を選択します。次のステップでポリシーを設定する際、すべてのユーザーが自動的に追加され、「*」と表示されます。
  - オブジェクトリストの取得（GetBucket）、バケット設定に関する権限などを匿名ユーザーに公開することはリスクが高いため、この項目を選択した場合、COSは対応するテンプレートを提供しません。必要な場合は、後続の「ポリシー設定」ステップでご自身で追加してください。
  - 許可されたユーザーがすべてのユーザー（匿名アクセス可）として指定されている場合、オブジェクトへのリクエスト時に署名を付与する必要はなく、すべてのユーザーがリンクを通じてオブジェクトに直接アクセス可能となります。データ漏洩のリスクがあるため、慎重に設定してください。
  - すべてのユーザーを拒否を選択しても、匿名ユーザーのみを拒否し、CAMユーザーは拒否されません。サブアカウントを拒否したい場合は、バケットポリシーで指定ユーザーを選択し、該当するサブアカウントのUINを指定してください。
- 指定ユーザー：指定するサブアカウント、ルートアカウントまたはクラウドサービスに操作権限を開放したい場合は、この項目を選択します。次の手順でポリシーを設定する際に、具体的なアカウントUINを指定する必要があります。

> 説明：
> 権限を付与されたユーザーが指定のアカウントである場合、オブジェクトをリクエストする際には認証のために署名を付与する必要があります。署名の詳細については、リクエスト署名をご参照ください。

リソース範囲

- バケット全体：バケットの設定項目に関連する権限を付与したい場合、またはリソース範囲としてバケット全体を指定したい場合は、この項目を選択します。次の手順でポリシーを設定する際に、バケット全体がリソースとして自動的に追加されます。
- 指定ディレクトリ：リソース範囲を指定のフォルダに限定したい場合は、この項目を選択します。次の手順でポリシーを設定する際に、具体的なディレクトリを指定する必要があります。この項目を選択した場合、COSはバケット設定に関連するポリシーテンプレートを提供しません。この種の権限はリソースとしてバケット全体を指定する必要があるためです。

テンプレート

権限を付与したいアクションのグループです。COSはお客様が選択した被付与者およびリソース範囲に基づき、推奨するポリシーテンプレートをご提供します。テンプレートがニーズに合わない場合はこのス

テップをスキップするか、または次のステップの「ポリシーの設定」で権限付与操作を追加または削除することができます。

- カスタムポリシー（プリセット設定はご提供していません）：テンプレートをご利用にならない場合はこの項目を選択し、次のステップの「ポリシーの設定」で、必要に応じてご自身でポリシーを追加できます。
- その他のテンプレート：COSはお客様が選択した被付与者およびリソース範囲の組み合わせに基づき、それぞれに推奨するテンプレートをご提供します。必要なテンプレートにチェックを入れると、次のステップのポリシー設定で、COSが自動的に必要な操作を追加します。

テンプレートの説明は次の表をご参照ください。

<table>
<tr><th>被付与者</th><th>リソース範囲</th><th>ポリシーテンプレート</th><th>説明</th></tr>
<tr><td colspan="2">すべての組み合わせ</td><td>カスタムポリシー</td><td>任意の被付与者、リソース範囲の組み合わせの場合、このテンプレートを選択するとプリセットポリシーのご提供は行われません。次のステップのポリシー設定で、ご自身でポリシーを直接追加することができます。</td></tr>
<tr><td rowspan="4">すべてのユーザー（匿名アクセス可能）</td><td rowspan="2">バケット全体</td><td>読み取り専用オブジェクト（オブジェクトリストのリストアップを含まない）</td><td rowspan="4">匿名ユーザーの場合、COSはファイル読み取り（ダウンロードなど）、ファイル書き込み（アップロード、変更など）の推奨テンプレートをご提供します。COSの推奨テンプレートには、バケット内の全オブジェクトのリストアップ、読み取り/書き込み権限、バケット設定などのその他の機密性の高い権限は含まれず、他の余分な権限を開放しないようにすることで、データの安全性を高めています。必要に応じて、後のステップでご自身で動作権限を追加または削除することができます。</td></tr>
<tr><td>読み取り/書き込みオブジェクト（オブジェクトリストのリストアップを含まない）</td></tr>
<tr><td rowspan="2">指定ディレクトリ</td><td>読み取り専用オブジェクト（オブジェクトリストのリストアップを含まない）</td></tr>
<tr><td>読み取り/書き込みオブジェクト（オブジェクトリストのリストアップを含まない）</td></tr>
</table>

指定ユーザー	バケット全体	読み取り専用オブジェクト（オブジェクトリストのリストアップを含まない）	指定ユーザーとバケット全体の組み合わせの場合、COSは最も多くの推奨テンプレートをご提供します。ファイル読み取り、書き込みおよびファイルリストアップのほか、次のような機密性の高い権限のテンプレートも含まれ、これらは信頼できるユーザーへの使用に適しています。 • 読み取り/書き込みバケットおよびオブジェクトACL：バケット取得、変更ACL、オブジェクトACL。GetObjectACL、PutObjectACL、GetBucketACL、PutBucketACLを含む • バケット一般設定項目：バケットタグ、クロスドメイン、back-to-originなどの機密性が高くない権限。 • バケットの機密性の高い設定項目：バケットポリシー、バケットACL、バケット削除などにかかわる機密性の高い権限であり、慎重に使用する必要があります。
		読み取り専用オブジェクト（オブジェクトリストのリストアップを含む）	
		読み取り/書き込みオブジェクト（オブジェクトリストのリストアップを含まない）	
		読み取り/書き込みオブジェクト（オブジェクトリストのリストアップを含む）	
		バケットおよびオブジェクトの読み取り/書き込みACL	
		バケット一般設定項目	
		バケットの機密性の高い設定項目	
	指定ディレクトリ	読み取り専用オブジェクト（オブジェクトリストのリストアップを含まない）	指定ユーザーと指定ディレクトリの組み合わせの場合、COSはファイル読み取り（ダウンロードなど）、ファイル書き込み（アップロード、変更など）以外にも、オブジェクトリストのリストアップ権限を含む推奨テンプレートをご提供します。指定したユーザー向けに指定したフォルダのファイルの読み取り、書き込み、リスト
		読み取り専用オブジェクト（オブジェクトリス	

		トのリストアップを含む)	アップ権限を開放したい場合は、この組み合わせを選択することをお勧めします。必要に応じて、後のステップでご自身で動作権限を追加または削除することができます。
		読み取り/書き込みオブジェクト（オブジェクトリストのリストアップを含まない)	
		読み取り/書き込みオブジェクト（オブジェクトリストのリストアップを含む)	

ステップ2：ポリシーの設定

ステップ1で選択した被付与者、指定ディレクトリおよびテンプレートの組み合わせに対し、COSはポリシー設定において対応する操作、被付与者、リソースなどを自動的に追加します。お客様が指定ユーザー、指定ディレクトリを選択された場合は、ポリシー設定の際に具体的なユーザーUINとディレクトリを指定する必要があります。

> 説明：
> ディレクトリに対する権限付与の場合、入力するリソースパスの後に `/*` を付ける必要があることにご注意ください。例えば、ディレクトリのtest権限の場合、 `test/*` と入力する必要があります。

COSがご提供する推奨テンプレートがニーズに合わない場合は、お客様ご自身でポリシーの内容を調整し、被付与者、リソースおよびアクションを追加または削除することもできます。下図に示します。
設定項目の説明は次のとおりです。

- エフェクト：許可または拒否を選択できます。ポリシー構文の「allow」と「deny」に対応します。
- ユーザー：すべてのユーザー(`*`)、ルートアカウント、サブアカウント、クラウドサービスを含む被付与者を追加、削除できます。

> 注意：
> - 条件内の `IP` と `VPC ID` は、同一のポリシー内に同時に設定することはできません。ポリシーを分けて追加してください。
> - エフェクトが拒否の場合、すべてのユーザーの適用範囲は匿名ユーザーのみに限定されます。匿名ユーザーはバケットへアクセスできなくなりますので、操作は、慎重に行ってください。

- リソース：バケット全体または指定ディレクトリのリソースを追加できます。
- アクション：権限付与が必要な操作を追加、削除します。
- 条件：ユーザーのアクセスIPの制限など、権限付与の際の指定条件です。その他の条件に関する詳細については、条件キーの説明と使用例をご参照ください。

> ⚠ 注意：
> - 条件内の `IP` と `VPC ID` は、同一のポリシー内に同時に設定することはできません。ポリシーを分けて追加してください。
> - ご利用のCVMとCOSバケットが異なるリージョンにある場合、またはCVM以外からアクセスする場合、条件として `IP` を選択し、パブリックIPアドレスを入力してください。ご利用のCVMとCOSバケットが同一リージョンにある場合は、条件として `VPC ID` を選択し、VPC IDを入力してください。

ポリシー文法

グラフィカル設定を使用する場合を除き、直接JSON言語を使用してポリシーを作成することができます。
バケットポリシーを作成したら、保存をクリックします。下図の通りです。また、API または SDK を通じてバケットポリシーを追加することもできます。

JSONポリシーの例

次のポリシーの記述例は、ルートアカウントID 100000000001（APPIDは1250000000）下のサブアカウントID 100000000011に対し、北京リージョンのバケットのexamplebucket-bj下のディレクトリ `folder/sub-folder` 内のオブジェクトについて、すべての操作の権限を許可するものです。

```
{
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000011"
        ]
      },
      "effect": "allow",
      "action": [
        "name/cos:*"
      ],
      "resource": [
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-bj-
1250000000/folder/sub-folder/*"
```

```
        ]
      }
    ],
    "version": "2.0"
}
```

操作方法

COSではコンソール、API、SDKなどの複数の方式でバケットポリシーを追加することができます。コンソールはグラフィカル設定と一般的な権限付与テンプレートをサポートしており、ポリシー言語に不慣れな場合でもすぐにポリシーを追加できるようになっています。

操作方法		説明
コンソール		直感的で使いやすいWebページ
API		RESTful APIで直接COSをリクエスト
SDK	Android	豊富なSDKデモを提供しており、様々な開発言語に対応しています。
	Node.js	
	小程序	
	.NET(C#)	
	Go	
	Java	
	JavaScript	
	Node.js	
	Python	
	ミニプログラム	

その他のバケットポリシーの例

- バケットポリシー（Policy）による権限付与の例
- 他のルートアカウント下のサブアカウント操作名のバケットへの権限付与

> ⚠ 注意：

> バケットポリシーを追加する際は、必ず業務の必要性に応じて、最小権限の原則に従って権限を付与してください。他のユーザーに対し、すべてのリソース `(resource:*)` またはすべてのアクション `(action:*)` の権限を直接与えてしまうと、権限が大きすぎるためにデータセキュリティ上のリスクが生じる場合があります。

次に、サブネット、プリンシパル、VPC IDを制限するバケットポリシーの例についてご説明します。

事例1

匿名ユーザー（ `qcs::cam::anyone:anyone` ）からの、サブネット10.1.1.0/24およびvpcidがaqp5jrc1のリクエストを拒否する例は以下の通りです。

> ⚠ 注意：
> この拒否ポリシーは匿名ユーザーのみに対して有効です。CAMユーザー（サブアカウント、メインアカウント）を制限したい場合は、principalフィールドに該当アカウントのArnを明示的に指定する必要があります。

```
{
  "statement": [
    {
      "action": [
        "name/cos:*"
      ],
      "condition": {
        "ip_equal": {
          "qcs:ip": [
            "10.1.1.0/24"
          ]
        },
        "string_equal": {
          "vpc:requester_vpc": [
            "vpc-aqp5jrc1"
          ]
        }
      },
      "effect": "deny",
      "principal": {
        "qcs": [
          "qcs::cam::anyone:anyone"
        ]
```

```
      },
      "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

事例2

サブアカウント（ `qcs::cam::uin/100000000001:uin/100000000002` ）がアクセス元vpcidがaqp5jrc1からアクセスする際に、特定のバケットへのアクセスを許可するポリシーの記述例は以下の通りです：

```
{
  "statement": [
    {
      "action": [
        "name/cos:*"
      ],
      "condition": {
        "string_equal": {
          "vpc:requester_vpc": [
            "vpc-aqp5jrc1"
          ]
        }
      },
      "effect": "allow",
      "principal": {
        "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000002"
        ]
      },
      "resource": [
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/*"
      ]
    }
  ],
  "version": "2.0"
```

```
}
```

ユーザーポリシー

最終更新日：：2025-11-25 16:02:41

Tencent Cloudルートアカウントは Cloud Access Management（CAM） コンソールでCAMユーザーを作成し、ポリシーをバインドすることで、CAMユーザーに対しTencent Cloudのリソースを使用する権限を与えることができます。

概要

ユーザーは CAM で、ルートアカウント下の異なるタイプのユーザーに対し、異なる権限を付与することができます。これらの権限はアクセスポリシー言語で記述し、ユーザーを出発点として権限承認を行うため、ユーザーポリシーと呼ばれます。

ユーザーポリシーとバケットポリシーとの違い

ユーザーポリシーとバケットポリシーの最大の違いは、ユーザーポリシーはエフェクト（Effect）、アクション（Action）、リソース（Resource）、条件（Condition、オプション）のみを記述し、プリンシパル（Principal）は記述しない点です。このため、ユーザーポリシーの使用方法は次のようになります。

- ユーザーポリシーは作成完了後、さらにサブユーザー、ユーザーグループまたはロールに対しバインド操作を実行する必要があります。
- ユーザーポリシーはアクションおよびリソース権限の匿名ユーザーへの付与をサポートしていません。

プリセットポリシーとカスタムポリシー

ユーザーポリシーには、プリセットポリシーとカスタムポリシー の2種類が含まれます。プリセットポリシーを使用した権限のバインド 、または ユーザーポリシーを自ら作成 して権限のバインドを行うことができます。詳細については、CAMの 権限承認ガイド をご参照ください。

ユースケース

ユーザーが行えること、および推奨されるユーザーポリシーについてお知りになりたい場合は、CAMユーザーを検索し、その所属するユーザーグループの権限を確認することで、ユーザーが何を行うことができるかを知ることができます。次のようなケースで推奨されます。

- バケット作成（PutBucket）、バケットリストアップ（GetService）などの、Cloud Object Storage（COS）サービスレベルの権限を設定したい場合。
- ルートアカウント下のすべてのCOSバケットおよびオブジェクトを使用したい場合。
- ルートアカウント下の大量のCAMユーザーに同等の権限を付与したい場合。

ユーザーポリシー構文

ポリシー文法

バケットポリシーと同様に、ユーザーポリシーもJSON言語を使用して記述し、アクセスポリシー言語の統一ルール（プリンシパル、エフェクト、アクション、リソース、条件など）に従います。ただし、ユーザーポリシーはユーザー/ユーザーグループに直接バインドされるため、ユーザーポリシーではプリンシパル（Principal）の入力は不要です。
次の表はユーザーポリシーとバケットポリシーとの違いを比較したものです。

要素	ユーザーポリシー	バケットポリシー
プリンシパル	入力不要	入力必須
エフェクト	入力必須	入力必須
アクション	入力必須	入力必須
リソース	入力必須	このバケット内のリソース
条件	オプション	オプション

バケットポリシーの例

以下は、典型的なユーザーポリシーの例です。このポリシーは、広州にあるバケットexamplebucket-1250000000のすべてのCOS操作権限を付与するポリシーです。ポリシーを保存した後、さらにCAMのサブユーザー、ユーザーグループまたはロールにバインドすることで発効します。

```
{
  "statement": [
    {
      "effect": "Allow",
      "action": ["cos:*"],
      "resource": [
          "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

ユーザーポリシーによるサブアカウントへのCOSアクセス権限承認

前提条件

CAMサブアカウントを作成済みであることが必要です。作成方法については サブアカウントの作成 をご参照ください。

設定手順

CAMでは プリセットポリシーとカスタムポリシー をご提供しています。プリセットポリシーはCAMの提供するシステムプリセットポリシーであり、COS関連ポリシーについては プリセットポリシー をご覧ください。カスタムポリシーはユーザーがリソース、アクションなどの要素を自ら定義することができ、よりフレキシブルです。カスタムポリシーを新規作成し、サブアカウントへの権限承認を行う方法については以下でご説明します。

1. CAMコンソール にログインします。
2. ポリシー > カスタムポリシーの新規作成 > ポリシー構文で作成を選択し、ポリシー作成ページに進みます。
3. 実際のニーズに応じて空白テンプレートを選択して権限承認ポリシーをカスタマイズするか、またはCOSにバインドされたシステムテンプレートを選択します。ここでは空白テンプレートを例にとります。
4. 空白テンプレートを選択し、次へをクリックして、ポリシーを入力してください。以下の基本要素を含める必要があります。
   - resource：権限を承認するリソース。
     - すべてのリソース（`"*"`）
     - 指定するバケット(`"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"`)
     - バケット内の指定するディレクトリまたはオブジェクト(`"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/test/*"`)
   - action：権限を承認するアクション。
   - effect：エフェクト。`"allow"`（許可）または`"deny"`（拒否）を選択します。
   - condition：発効条件。オプションです。

COSはユーザーポリシーの例をご提供しています。次のドキュメントをご参照の上、ポリシーの内容をコピーしてポリシー内容のエディタボックス内に貼り付け、入力に間違いがないことを確認してから完了をクリックします。

- サブアカウントのCOSアクセス権限の承認
- COS API権限承認ポリシー使用ガイド

5. 作成完了後、CAMコンソール のポリシー > カスタムポリシーで、作成したカスタムポリシーを確認し、ポリシーをサブアカウントにバインドすることができます。
6. サブアカウントにチェックを入れ、OKをクリックして権限を承認すると、限定されたCOSリソースにサブアカウントを使用してアクセスできるようになります。

プリセットポリシー

1. CAMではいくつかのプリセットポリシーをご提供しています。CAMコンソール のポリシー > プリセットポリシーで確認し、「COS」を検索してフィルタリングすることができます。

2. ポリシー名をクリックし、ポリシー構文 > JSONに進み、具体的なポリシーの内容を確認します。プリセットポリシーのリソース（ `resource` ）はCOSのすべてのリソース(`"*"`)に設定されており、変更はサポートされていません。COSバケット、オブジェクトの一部について権限を承認したい場合は、JSONのプリセットポリシーをコピーし、カスタムポリシー を作成することができます。

表1と表2に、CAMがご提供するCOS関連のプリセットポリシーとその説明について列記します。
表1：COSプリセットポリシー

プリセットポリシー	説明	JSONポリシー

QcloudCOSBucketConfigRead	この権限を有するユーザーはCOSバケット設定を読み取ることができます	<pre>{
		"version": "2.0",
		"statement": [
		{
		"effect": "allow",
		"action": [
		"cos:PutBucket*"
		],
		"resource": "*"
		}
		]
		}</pre>

QcloudCOSBucketConfigWrite	この権限を有するユーザーはCOSバケット設定を変更することができます	<pre>{
		"version": "2.0",
		"statement": [
		{
		"effect": "allow",
		"action": [
		"cos:PutBucket*"</pre>

		<pre>],
		"resource": "*"
		}
		]
		}</pre>

QcloudCOSDataFullControl	COSバケット内のデータの読み取り、書き込み、削除、リストアップを含むアクセス権限	<pre>{
		"version": "2.0",
		"statement": [
		{
		"effect": "allow",
		"action": [
		"cos:GetService",
		"cos:GetBucket",
		"cos:ListMultipartUploads",
		"cos:GetObject*",
		"cos:HeadObject",
		"cos:GetBucketObjectVersions",
		"cos:OptionsObject",
		"cos:ListParts",
		"cos:DeleteObject",
		"cos:PostObject",</pre>

```
"cos:PostObjectRestore",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload",
"cos:DeleteMultipleObjects"
            ],
            "resource": "*"
        }
    ]
}
```

表2：COSの操作とプリセットポリシーとの関係

説明	操作	QcloudCOSBucketConfigRead	QcloudCOSBucketConfigWrite	QcloudCOSDataFullControl	QcloudCOSDataReadOnly	QcloudCOSDataWriteOnly
バケットリストアップ	GetService	✕	✕	✓	✕	✕
バケット作成	PutBucket	✕	✓	✕	✕	✕

バケット削除	DeleteBucket	✕	✕	✕	✕	✕
バケット基本情報取得	HeadBucket	✓	✕	✕	✕	✕
バケット設定項目取得	GetBucket*	✓	✕	✕	✕	✕
バケット設定項目変更	PutBucket*	✕	✓	✕	✕	✕
バケットアクセス権限取得	GetBucketAcl	✓	✕	✕	✕	✕
バケットアクセス権限変更	PutBucketAcl	✕	✓	✕	✕	✕
バケット内オブジェクトリストアップ	GetBucket	✓	✕	✓	✕	✕
バケット内オブジェクトの全バージョンリストアップ	GetBucketObjectVersions	✓	✕	✓	✕	✕
オブジェクトアップロード	PutObject	✕	✕	✓	✕	✓
マルチパートアップロード	ListParts InitiateMultipartUpload UploadPart	✕	✕	✓	✕	✓

	UploadPartCopy CompleteMultipartUpload AbortMultipartUpload ListMultipartUploads					
オブジェクトダウンロード	GetObject	✕	✕	✓	✓	✕
オブジェクトメタデータ確認	HeadObject	✕	✕	✓	✓	✕
クロスドメイン（CORS）事前チェック	OptionsObject	✕	✕	✓	✓	✕

その他のユーザーポリシーの例

- サブアカウントのCOSアクセス権限の承認
- COS API権限承認ポリシー使用ガイド

ACL

最終更新日：：2025-09-11 16:16:27

基本概念

アクセス制御リスト（ACL）はXML言語を使用して記述する、リソースにバインドされた、被付与者と付与される権限を指定したリストです。各バケットとオブジェクトにはそれぞれに、これらとバインドされたACLがあり、匿名ユーザーまたはその他のTencent Cloudのルートアカウントに対し、基本的な読み取り/書き込み権限を付与することができます。

> ⚠ 注意：
> リソースにバインドされたACLを使用した管理にはいくつかの制限があります。
> - リソースの所有者は常にリソースに対してFULL_CONTROL権限を有し、その取り消しまたは変更はできません。
> - 匿名ユーザーはリソースの所有者にはなれません。この場合、オブジェクトリソースの所有者はバケットの作成者（Tencent Cloudルートアカウント）となります。
> - 権限はCloud Access Management（CAM）のルートアカウントまたは事前定義済みのユーザーグループにのみ付与することができます。カスタマイズしたユーザーグループに権限を付与することはできず、サブユーザーへの権限付与も推奨されません。
> - 権限の付加条件はサポートしていません。
> - 明示的な拒否の権限はサポートしていません。
> - 1つのリソースにつき最大100のACLポリシーを持つことができます。

ユースケース

> ⚠ 注意：
> 匿名ユーザーにアクセスを開放すること（パブリック読み取り）はハイリスクな操作であり、トラフィックが不正使用される危険性があります。やむを得ずパブリック読み取りを使用する場合は、リンク不正アクセス防止の設定によってセキュリティ保護を実行できます。

バケットとオブジェクトに簡単なアクセス権限のみを設定したい場合、または匿名ユーザーにアクセスを開放したい場合はACLを選択できます。ただし、より多くの状況下では、バケットポリシーまたはユーザーポリシーの方が柔軟性が高いため、これらを優先的に使用することをお勧めします。ACLの適用ケースには次のものがあります。

- 簡単なアクセス権限のみを設定する場合。
- コンソールでアクセス権限をすばやく設定する場合。
- あるオブジェクト、ディレクトリまたはバケットへのアクセスをすべての匿名インターネットユーザーに開放したい場合は、ACLによる操作の方が便利です。

ACLの要素

被付与者 Grantee

サポートする被付与者のIDは、何らかのCAMルートアカウントまたは何らかの事前定義済みのCAMユーザーグループです。

> ⚠ 注意：
> - 他のTencent Cloudルートアカウントにアクセス権限を付与する際、この権限を付与されたルートアカウントはそのアカウント下のサブユーザー、ユーザーグループまたはロールにアクセス権限を与えることができます。
> - Cloud Object Storage（COS）は、匿名のユーザーまたはCAMユーザーグループにWRITE、WRITE_ACPまたはFULL_CONTROL権限を与えることを強く非推奨とします。一度権限を許可すると、ユーザーグループはお客様のリソースのアップロード、ダウンロード、削除などを行うことができるようになり、このことはお客様のデータの消失、料金引き落としなどのリスクをもたらす場合があります。

バケットまたはオブジェクトのACLにおいてサポートされるIDには次のものがあります。

- クロスアカウント：ルートアカウントのIDを使用してください。アカウントIDはアカウントセンターの アカウント情報 で取得します（例：100000000001）。
- 事前定義済みのユーザーグループ：URIタグを使用して事前定義済みのタグ付けをしたユーザーグループを使用してください。次のユーザーグループをサポートしています。
  - 匿名ユーザーグループ – `http://cam.qcloud.com/groups/global/AllUsers` このグループは、リクエストが署名済みかどうかにかかわらず、誰でも権限なしにリソースにアクセスできることを表します。
  - 認証ユーザーグループ – `http://cam.qcloud.com/groups/global/AuthenticatedUsers` このグループは、Tencent Cloud CAMアカウント認証を経たすべてのユーザーがリソースにアクセスできることを表します。

操作 Permission

Tencent Cloud COSがリソースのACLにおいてサポートする操作は、実際には一連の操作の集合であり、バケットおよびオブジェクトACLにはそれぞれ異なる意味があります。

バケットの操作

次の表に、バケットACLで設定可能な操作のリストを列記しています。

操作セット	説明	許可される行為
READ	オブジェクトのリ	HeadBucket、GetBucketObjectVersions、

	ストアップ	ListMultipartUploads
WRITE	オブジェクトのアップロード、上書き、削除	PutObject、PutObjectCopy、PostObject、InitiateMultipartUpload、UploadPart、UploadPartCopy、CompleteMultipartUpload、 DeleteObject
READ_ACP	バケットのACLの読み取り	GetBucketACL
WRITE_ACP	バケットのACLの書き込み	PutBucketACL
FULL_CONTROL	上記4種類の権限のセット	上記のすべての行為のセット

> ⚠ 注意：
> バケットのWRITE、WRITE_ACPまたはFULL_CONTROL権限の付与は慎重に行ってください。バケットのWRITE権限の付与は、被付与者に既存のあらゆるオブジェクトの上書きまたは削除を許可するものです。

オブジェクトの操作

次の表に、オブジェクトACLで設定可能な操作のリストを列記しています。

操作セット	説明	許可される行為
READ	オブジェクトの読み取り	GetObject、GetObjectVersion、HeadObject
READ_ACP	オブジェクトのACLの読み取り	GetObjectACL、GetObjectVersionACL
WRITE_ACP	オブジェクトのACLの書き込み	PutObjectACL、PutObjectVersionACL
FULL_CONTROL	上記3種類の権限のセット	上記のすべての行為のセット

> ⓘ 説明：
> オブジェクトについてはWRITE操作セットの権限はサポートしていません。

既定ACL

COSでは一連の既定ACLによる権限承認がサポートされており、権限を簡単に記述できるようになっています。既定ACLを使用して記述する場合、PUT Bucket/ObjectまたはPUT Bucket/Object aclにx-cos-aclヘッダーを含め、必要な権限を記述する必要があります。同時にリクエスト本文にもXMLの記述内容が含まれる場合は、ヘッダー内の記述が優先的に選択され、リクエスト本文のXMLの記述は無視されます。

バケットの既定ACL

規定名	説明
private	作成者（ルートアカウント）はFULL_CONTROL権限を有し、その他の人は権限を持ちません（デフォルト）
public-read	作成者はFULL_CONTROL権限を有し、匿名ユーザーグループはREAD権限を有します
public-read-write	作成者と匿名ユーザーグループの両方がFULL_CONTROL権限を有します。通常はこの権限の付与は非推奨です
authenticated-read	作成者はFULL_CONTROL権限を有し、認証ユーザーグループはREAD権限を有します

オブジェクトの既定ACL

規定名	説明
default	空の記述であり、この場合は各レベルのディレクトリに基づく明示的な設定およびバケットの設定によって、リクエストを許可するかどうかを決定します（デフォルト）
private	作成者（ルートアカウント）はFULL_CONTROL権限を有し、その他の人は権限を持ちません
public-read	作成者はFULL_CONTROL権限を有し、匿名ユーザーグループはREAD権限を有します
authenticated-read	作成者はFULL_CONTROL権限を有し、認証ユーザーグループはREAD権限を有します
bucket-owner-read	作成者はFULL_CONTROL権限を有し、バケット所有者はREAD権限を有します
bucket-owner-full-control	作成者とバケット所有者の両方がFULL_CONTROL権限を有します

説明：
オブジェクトについてはpublic-read-write権限はサポートしていません。

事例

バケットのACL

バケットを作成する際、COSはデフォルトのACLを作成し、リソース所有者に対しリソースの完全制御権限（FULL_CONTROL）を与えます。その例を次に示します。

```
<AccessControlPolicy>
  <Owner>
    <ID>Owner-Cononical-CAM-User-Id</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>Owner-Cononical-CAM-User-Id</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

オブジェクトのACL

オブジェクトを作成する際、COSはデフォルトではACLを作成しません。この場合はオブジェクトの所有者がバケット所有者となります。オブジェクトはバケットの権限を継承し、それはバケットのアクセス権限と一致します。オブジェクトにはデフォルトのACLがないため、Bucket Policyの中のアクセス者およびその行為に対する定義に従って、リクエストが許可されるかどうかを判断します。詳細については、アクセスポリシーの言語概要のドキュメントをご参照ください。
オブジェクトについてその他のアクセス権限を付与したい場合は、これを基本としてさらにACLを追加し、オブジェクトのアクセス権限を記述することができます。例えば匿名ユーザーに対し、単一オブジェクトの読み取り専用の権限を付与する場合の例は次のとおりです。

```
<AccessControlPolicy>
  <Owner>
    <ID>Owner-Cononical-CAM-User-Id</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>Owner-Cononical-CAM-User-Id</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
```

```
      <Grantee>
        <URI>http://cam.qcloud.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

利用方法

COSコンソールの使用

アクセス権限の設定

COSコンソールから、バケットへのアクセス権限の設定または変更を行うことができます。COSは、以下2種類の権限タイプの設定をサポートしています。

- パブリック権限：プライベート読み取り/書き込み、パブリック読み取りプライベート書き込み、およびパブリック読み取り/書き込みです。パブリック権限の説明については、バケット概要の 権限のタイプ をご参照ください。
- ユーザー権限：ルートアカウントは、デフォルトではバケットのすべての権限を所有しています（完全制御）。またCOSは、サブアカウントに対するデータ読み取り、データ書き込み、権限読み取り、権限書き込み、さらには完全制御の最高権限の付与もサポートしています。

> 説明：
> - バケットがプライベート読み取り/書き込みの場合、または指定のアカウントに対してユーザー権限を承認している場合は、オブジェクトのリクエストの際に、ID認証用の署名を含める必要があります。署名に関する説明については、リクエスト署名 をご参照ください。
> - バケットがパブリック読み取り/プライベート書き込みまたはパブリック読み取り/書き込みの場合、オブジェクトのリクエストの際に署名を含める必要はなく、匿名のユーザーが直接リンクを通じてオブジェクトにアクセスできます。データ漏洩のリスクがありますので、慎重に設定してください。

単一承認

1. COSコンソール にログインします。
2. 左側ナビゲーションバーで、バケットリストをクリックします。
3. アクセス権限を設定または変更したいバケットを見つけ、そのバケット名をクリックします。
4. バケット設定ページで、権限管理>バケットのアクセス権限をクリックし、バケットのパブリック権限とユーザー権限（例えばサブアカウントの追加があります。サブアカウントIDはCAM コンソールで確認できます）を設定します。

> ⚠ 注意：
> デフォルトのアラームが設定されていない場合、パブリックリード/プライベートライトまたはパブリックリード/ライト権限を使用する必要がある場合は、アラームポリシーを有効にすることをお勧めします。すでに設定されている場合は、追加の設定は不要で、通知も表示されません。

5. 保存をクリックすると、バケットアクセス権限の設定が完了します。

一括承認

1. COSコンソールにログインします。
2. 左側ナビゲーションバーで、バケットリストをクリックします。
3. リストの上にある承認管理をクリックします。
4. ポップアップウィンドウで、承認したいバケットを選択します。次に、バケットのパブリック権限とユーザー権限（例えばサブアカウントの追加、サブアカウントIDはCAM コンソールで確認できます）を設定します。

> ⚠ 注意：
> 権限管理は、公共の権限とユーザーの権限を同時に変更します。ユーザーの権限のみを変更する必要がある場合は、以前のバケットの公共の権限と現在の設定が一致しているか必ず確認してください。そうしないと、以前の権限が上書きされます。

5. 設定の完了後にOKをクリックすると、複数のバケットのアクセス権限の設定が完了します。

オブジェクトのアクセス権限の設定

Cloud Object Storage（COS）は、オブジェクトディメンションに基づいたアクセス権限の設定を提供します。またこの権限は、バケットのアクセス権限よりも優先度が高いです。

> ⓘ 説明：
> - オブジェクトのアクセス権限は、ユーザーがデフォルトのドメイン名でアクセスする場合にのみ有効です。CDNアクセラレーションドメイン名やカスタムドメイン名でアクセスする場合、バケットのアクセス権限が優先されます。
> - アクセスポリシールールには数の制限があります。詳細については、仕様と制限をご参照ください。

操作手順

1. COSコンソールにログインします。
2. 左側ナビゲーションバーでバケットリストをクリックし、バケットリストページに進みます。
3. オブジェクトがあるバケットを見つけ、そのバケット名をクリックし、バケット管理ページに進みます。
4. 左側ナビゲーションバーでファイルリストを選択し、ファイルリストページに進みます。

5. アクセス権限を設定したいオブジェクトを見つけ、右側の詳細をクリックし、ファイルの詳細ページに進みます（フォルダの場合は、右側の権限の設定をクリックします）。
6. オブジェクトのアクセス権限バーで、実際の必要性に応じてアクセス権限を設定します。

   COSは、オブジェクトに対して、以下2種類の権限タイプの設定をサポートしています。
   - パブリック権限：権限の継承、プライベート読み取り/書き込みおよびパブリック読み取りプライベート書き込みが含まれます。パブリック権限の説明については、オブジェクト概要の 権限のタイプ をご参照ください。
   - ユーザー権限：ルートアカウントはデフォルトではオブジェクトのすべての権限を所有しています（完全制御）。
7. 保存をクリックすると、オブジェクトのアクセス権限を設定できます。

   複数のオブジェクトに対して一括でアクセス権限の設定または変更を行う必要がある場合は、複数のオブジェクトにチェックを入れ、上部にあるその他の操作 > アクセス権限の変更をクリックすれば設定できます。

タグに基づくプロジェクトリソースの管理

最終更新日：：2024-06-26 11:09:29

概要

> 説明：
> このドキュメントでは主に、タグシステムの下でのタグおよびタグ認証によるプロジェクトリソース管理の利用方法についてご説明します。過去のバージョンのコンソール上でプロジェクトを使用したことがあり、かつプロジェクト認証方式でサブアカウントのアクセス権限を付与したことがある、一部の古くからのユーザーに適用します。

プロジェクト管理はプロジェクトの次元でリソースに対する集中管理を行うものです。プロジェクトの機能をサポートしているクラウド製品リソースをプロジェクトに追加し、Cloud Access Management（CAM）コンソールのポリシー>カスタムポリシーの新規作成>製品機能またはプロジェクトによる権限の作成によってプロジェクトポリシーを生成することができます。プロジェクトポリシーをプロジェクトに関連するユーザーまたはユーザーグループにバインドすることで、ユーザーまたはユーザーグループにプロジェクトリソースの操作権限を許可することができます。
Cloud Object Storage（COS）は過去のバージョンのコンソール上で、プロジェクトベースでユーザーに関連する権限管理操作を提供していますが、プロジェクトポリシーにはプロジェクトに追加された全製品下の全リソースの完全なアクセス権限が含まれ、多次元的なタグ付けおよびカテゴライズのニーズを満たすことができないだけでなく、精密な権限管理を行うこともできませんでした。新バージョンのCOSコンソールでは、COSはタグ方式によるプロジェクトリソースの権限管理のみサポートしています。
COSはタグサービスを利用して旧プロジェクト機能との互換性を実現しています。タグサービスのシステムでは、プロジェクトは特殊なタグであり、そのタグキーは `project` となります。プロジェクトコンソールでプロジェクトを新規作成し、そのプロジェクト下にバケットを作成することも引き続き可能です。COSはお客様のバケット作成時にバケットのプロジェクトの帰属関係を自動的にタグにダブルライトすることで、コンソール上で表示できるようにします。

> 説明：
> - バケットのカテゴリー管理のニーズがおありの場合は、プロジェクトの手段ではなく、直接タグによってバケットを管理することで、権限制御や請求書分割などのタスクを実現することをお勧めします。コンソール上でのタグの追加方法に関しては、バケットタグの設定をご参照ください。
> - プロジェクトについてお知りになりたい場合は、CAMのプロジェクトとタグをご参照ください。タグサービスについてお知りになりたい場合は、タグの製品ドキュメントをご参照ください。
> - タグのメリットについてお知りになりたい場合は、タグ使用のメリットをご参照ください。

サブアカウントへのプロジェクトアクセス権限の付与

サブアカウントに対してプロジェクトへのアクセス権限を付与するには、次の手順に従って操作を行ってください。

1. プロジェクト管理コンソールにログインし、プロジェクトを新規作成し、プロジェクト名をカスタマイズして送信します。その後、そのプロジェクト下に作成したバケットまたはCVMなどのリソースを選択します。
すでにプロジェクトがあり、かつすでに帰属するストレージまたはコンピューティングリソースがある場合は、この手順をスキップできます。
2. プロジェクトの作成が完了し、対応するリソースをバインドした後、ポリシー管理ページに進み、カスタムポリシーの新規作成>タグによる権限承認をクリックし、追加するCOSサービスおよび承認する操作権限を選択します。対応するプロジェクトのタグを選択し、サブアカウントに対し、このプロジェクトタグ下のすべてのリソースへのアクセスを承認します。
3. 次のステップをクリックし、この権限を承認するユーザー/ユーザーグループ/ロールを選択し、最後に完了をクリックします。

> 説明：
> - デフォルトのポリシー内容は、サブアカウントに対しこのプロジェクトタグ下のすべてのリソースへのアクセスを付与するものです。ユーザーがタグ下の一部のリソースについてのみ、指定した操作を行えるようにしたい場合は、構文の構造ドキュメントを参照し、ポリシー構文の中の `action`（指定した操作の設定）および `resource`（操作可能なリソースの設定）を変更することができます。
> - サブアカウントでバケットを作成できるようにしたい場合は、サブアカウントに対しさらに `PUT Bucket` の操作権限を付与する必要があります。ポリシー管理ページで、カスタムポリシーの新規作成>ポリシージェネレーターで作成をクリックすると、サブアカウントに対し対応する権限を付与することができます。

権限承認方式の選択方法

最終更新日：：2024-06-26 11:09:29

バケットポリシー、ユーザーポリシー、バケットACL、オブジェクトACLの違い

ユーザーポリシー、バケットポリシー、バケットACL、オブジェクトACLの間の違いについては表1のとおりです。

- ユーザーポリシーはユーザーベースの権限承認方式であり、バケットポリシー、バケットACL、オブジェクトACLはリソースベースの権限承認方式です。
- ユーザーポリシーとバケットポリシーの権限承認はアクセスポリシー言語に基づいて行われ、権限制御の柔軟性の程度がより高くなっています。権限はそれぞれのアクション（action）について具体的に付与され、なおかつ権限のエフェクト（effect）をdenyまたはallowとすることができます。バケットACLとオブジェクトACLはどちらもアクセス制御リスト（ACL）をベースにした権限承認であり、権限制御の柔軟性の程度は相対的に低く、より簡単に使用することができますが、サポートされる権限は基本的な読み取り/書き込み権限のみです。
- ユーザーポリシーは匿名ユーザーへの権限承認をサポートしていません。バケットポリシー、バケットACL、オブジェクトACLは匿名ユーザーへの権限承認をサポートしています。
- ユーザーポリシーはCloud Access Management（CAM）コンソールで設定し、バケットポリシー、バケットACL、オブジェクトACLはCloud Object Storage（COS）コンソールで設定します。

表1 権限承認方式の違いの比較

比較項目		ユーザーポリシー	バケットポリシー	バケットACL	オブジェクトACL
分類		ユーザーベースの権限承認	リソースベースの権限承認	リソースベースの権限承認	リソースベースの権限承認
権限制御の柔軟性の程度		柔軟性が高い	柔軟性が高い	柔軟性が低い	柔軟性が低い
コンソール設定		CAMコンソール	COSコンソール	COSコンソール	COSコンソール
アクセス制御要素	ユーザー	このアカウントが管理するすべてのCAM ID（サブアカウント、ロールなど）	サブアカウント、ルートアカウント、匿名ユーザー	サブアカウント、他のルートアカウント、匿名ユーザー	サブアカウント、他のルートアカウント、匿名ユーザー
			サブアカウントのTencent Cloudサービ		

			ス、ロールなど		
		クロスアカウント権限承認を行う場合は先にコラボレーターとしての追加が必要	クロスアカウント権限承認を直接サポート		
	エフェクト	許可+拒否	許可+拒否	許可のみ	許可のみ
	リソース	すべてのリソース、COSのすべてのバケット、指定のバケット（プレフィックス、オブジェクトなど）	指定のバケット（プレフィックス、オブジェクトなど）	バケット全体	指定のオブジェクト
	アクション	具体的な各アクション	具体的な各アクション（バケット作成、バケットリストアップを除く）	簡略化された読み取り/書き込み権限	簡略化された読み取り/書き込み権限
	条件	サポートしています	サポートしています	サポートしていません	サポートしていません

適切な権限承認方式を選択するにはどうすればよいですか。

表1に列記した違いから、それぞれの権限承認方式の優劣を判断し、お客様ご自身の必要性に応じて適切な権限承認方式をご選択いただくことができます。
ただし、どの方式を選択する場合でも、できる限り統一性を保つことをお勧めします。バケットポリシー、ユーザーポリシー、ACLの数が増えるにつれて、権限の維持管理が難しくなります。

ACLを選択するのはどのような場合ですか。

> ⚠ 注意：
> 匿名ユーザーにアクセスを開放すること（パブリック読み取り）はハイリスクな操作であり、トラフィックが不正使用される危険性があります。やむを得ずパブリック読み取りを使用する場合は、リンク

> 不正アクセス防止の設定によってセキュリティ保護を実行できます。

バケットとオブジェクトに簡単なアクセス権限のみを設定したい場合、または匿名ユーザーにアクセスを開放したい場合はACLを選択できます。ただし、より多くの状況下では、バケットポリシーまたはユーザーポリシーの方が柔軟性が高いため、これらを優先的に使用することをお勧めします。

- 簡単なアクセス権限のみを設定する場合。
- コンソールでアクセス権限をすばやく設定する場合。
- あるオブジェクト、ディレクトリまたはバケットへのアクセスをすべての匿名インターネットユーザーに開放したい場合は、ACLによる操作の方が便利です。
- 各アカウント下に設定できるACLの数は1000個までです。この上限を超える場合はバケットポリシーまたはユーザーポリシーを代わりに選択してください。

ユーザーポリシーを選択するのはどのような場合ですか。

ユーザーが行えることについてお知りになりたい場合は、ユーザーポリシーを推奨します。CAMユーザーを検索し、その所属するユーザーグループの権限を確認することで、ユーザーが何を行うことができるかを知ることができます。次のようなケースで推奨されます。

- バケット作成、バケットリストアップなどの、COSサービスレベルの権限を設定したい場合。
- ルートアカウント下のすべてのCOSバケットおよびオブジェクトを使用したい場合。
- ルートアカウント下の大量のCAMユーザーに同等の権限を付与したい場合。

バケットポリシーを選択するのはどのような場合ですか。

そのCOSバケットに誰がアクセス可能かをお知りになりたい場合は、バケットポリシーの使用をお勧めします。バケットを検索し、バケットポリシーをチェックすることで、アクセス可能な人が誰かを知ることができます。次のようなケースで推奨されます。

- 特定のバケットについて単独で権限を付与する場合。
- ACLと異なり、権限は特定のアクション（action）について具体的に付与され、権限のエフェクト（effect）をdenyまたはallowとする必要があります。
- ユーザーポリシーと異なり、バケットポリシーはクロスアカウント権限承認および匿名ユーザーへの権限承認をサポートしています。

アクセスポリシー言語
アクセスポリシーの言語概要

最終更新日：：2024-06-26 11:09:29

> ⚠ 注意：
> サブユーザーまたはコラボレーターにアクセスポリシーを追加する際は、必ず業務の必要性に応じて、最小権限の原則に従って権限を付与してください。サブユーザーまたはコラボレーターに対し、すべてのリソース `(resource:*)` またはすべてのアクション `(action:*)` の権限を直接与えてしまうと、権限の範囲が大きすぎるためにデータセキュリティ上のリスクが生じる場合があります。

概要

アクセスポリシーはCloud Object Storage（COS）リソースへのアクセス権限を付与するために用いられます。アクセスポリシーにはJSONをベースにしたアクセスポリシー言語を使用します。アクセスポリシー言語の権限によって、指定するプリンシパル（principal）に指定のCOSリソースに対する指定のアクションを実行させることができます。
アクセスポリシー言語はバケットポリシー（Bucket Policy）の基本要素および使用法を記述するものです。ポリシー言語の説明に関してはCAMポリシー管理をご参照ください。

アクセスポリシーの要素

アクセスポリシー言語には次の基本的な意味を持つ要素が含まれます。

- プリンシパル（principal）：ポリシーが権限を付与するエンティティを記述します。例えばユーザー（ルートアカウント、サブアカウント、匿名ユーザー）、ユーザーグループなどです。この要素はバケットアクセスポリシーに対して有効ですが、ユーザーアクセスポリシーには追加すべきではありません。
- ステートメント（statement）：1つまたは複数の権限の詳細情報を記述します。この要素にはエフェクト、アクション、リソース、条件などのいくつかの他の要素の権限または権限セットが含まれます。1つのポリシーには1つのステートメント要素しかありません。
- エフェクト（effect）：ステートメントによる結果が「許可」であるか「明示的な拒否」であるかを記述します。allowとdenyの2種類が含まれます。この要素は入力必須項目です。
- アクションaction：許可する、または拒否するアクションを記述します。アクションはAPI（nameプレフィックスで記述）または機能セット（permidプレフィックスが付いた特定のAPIセット）とすることができます。この要素は入力必須項目です。
- リソース（resource） は権限が適用されるリソースについて記述します。リソースは6セグメント式で説明されます。リソース定義の詳細は各製品によって異なります。リソースの指定方法については、作成したリソースステートメントに対応する製品ドキュメントをご参照ください。この要素は必須項目です。

- 条件（condition）：ポリシー発効の制約条件を記述します。条件はオペレーター、アクションキーとアクション値から構成されています。条件値には時間、IPアドレスなどの情報を含めることができます。一部のサービスは、条件に対しほかの値を指定することを認めています。この要素は入力必須項目ではありません。

要素の使用法

プリンシパルの指定

プリンシパル（principal）の要素はリソースへのアクセスを許可される、または拒否されるユーザー、アカウント、サービスまたはその他のエンティティを指定するために用いられます。principalの要素はバケット内でのみ作用します。ユーザーポリシーでは指定する必要がありませんが、これはユーザーポリシーが特定のユーザーに直接付加されるものだからです。principalの指定の例は次のとおりです。

```
"principal":{
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
]
}
```

匿名ユーザーに権限を付与：

```
"principal":{
"qcs": [
"qcs::cam::anonymous:anonymous"
]
}
```

ルートアカウントUIN 100000000001に権限を付与：

```
"principal":{
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
]
}
```

サブアカウントUIN 100000000011（ルートアカウントのUINは100000000001）に権限を付与：

> ⚠ 注意：

> 操作を行う前に、サブアカウントがルートアカウントのサブアカウントリストに追加されていることを確認する必要があります。

```
"principal":{
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000011"
]
}
```

エフェクトの指定

リソースへのアクセス権限を明示的に付与（許可）していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否（deny）することもでき、そうした場合はユーザーがそのリソースに確実にアクセスできなくなります。これは他のポリシーがアクセス権限を付与している場合であっても同様です。許可のエフェクトを指定する例を次に挙げます。

```
"effect" : "allow"
```

アクションの指定

COSはポリシーの中で、ある特定のCOSアクションを指定することができると定義しています。指定するアクションは発行されるAPIのリクエスト操作 と完全に同じです。一部のバケット操作およびオブジェクト操作を次に列記します。その他の具体的な操作については、API操作リスト のドキュメントをご参照ください。

バケットの操作

説明	対応するAPIインターフェース
name/cos:GetService	GET Service
name/cos:GetBucket	GET Bucket (List Objects)
name/cos:PutBucket	PUT Bucket
name/cos:DeleteBucket	DELETE Bucket

オブジェクト操作

説明	対応するAPIインターフェース
name/cos:GetObject	GET Object

name/cos:PutObject	PUT Object
name/cos:HeadObject	HEAD Object
name/cos:DeleteObject	DELETE Object

アクション許可の指定の例を次に示します。

```
"action":[
"name/cos:GetObject",
"name/cos:HeadObject"
]
```

リソースの指定

リソース（resource）要素は単一または複数の操作オブジェクト（COSバケットまたはオブジェクトなど）を記述します。すべてのリソースには下記の6セグメント式の記述方法を使用することができます。

```
qcs:project_id:service_type:region:account:resource
```

パラメータの説明は次のとおりです。

パラメータ	説明	入力必須かどうか
qcs	qcloud serviceの略称であり、Tencent Cloudのクラウドサービスであることを表します。	はい
project_id	プロジェクト情報を記述します。CAMのレガシーロジックとの互換性のためにのみ使用されます。	オプション
service_type	「COS」のような、製品の略称を記述します。	はい
region	リージョンの情報を説明します。Tencent Cloud COSがサポートするアベイラビリティリージョンをご参照ください。	はい
account	リソース所有者のルートアカウント情報を記述します。現在、リソース所有者の記述方式は2種類をサポートしています。1つはuin方式、すなわちルートアカウントのUINアカウントであり、`uin/${OwnerUin}`で表され、uin/100000000001のようになります。もう1つはuid方式、すなわちルートアカウントのAPPIDであり、`uid/${appid}`で表され、uid/1250000000のようになります。現在、COSのリソース所有者はすべて	はい

	uid方式を使用して記述され、ルートアカウントの開発者APPIDとなります。	
resource	具体的なリソースの詳細を記述します。COSサービスではバケットのXML APIアクセスドメイン名を使用して記述します。	はい

バケットexamplebucket-1250000000を指定する例を次に挙げます。

```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"]
```

バケットexamplebucket-1250000000内の/folder/フォルダ下の全オブジェクトを指定する例を次に挙げます。

```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/folder/*"]
```

バケットexamplebucket-1250000000内のオブジェクト、/folder/exampleobjectを指定する例を次に挙げます。

```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/folder/exampleobject"]
```

条件の指定

アクセスポリシー言語は権限承認を受ける際の条件を指定することができます。例えばユーザーのアクセス元の制限、権限承認時間の制限などです。現在サポートされている条件オペレーターのリストおよび一般的な条件キーとその例などの情報を次に列記します。

条件オペレーター	意味	条件名	例
ip_equal	IP一致	qcs:ip	{"ip_equal":{"qcs:ip ":"10.121.2.0/24"}}
ip_not_equal	IP不一致	qcs:ip	{"ip_not_equal":{"qcs:ip ":["10.121.1.0/24", "10.121.2.0/24"]}}

以下は、アクセスIPが10.121.2.0/24のネットワークセグメント内という条件を満たす例です。

```
"ip_equal":{"qcs:ip ":"10.121.2.0/24"}
```

以下は、アクセスIPが101.226.100.185および101.226.100.186であるという条件を満たす例です。

```
"ip_equal":{
    "qcs: ip": [
        "101.226.100.185",
        "101.226.100.186"
    ]
}
```

実際の例

ルートアカウントが匿名ユーザーを許可し、アクセス元のIPが101.226.100.185または101.226.100.186であった場合、華南リージョンのバケットexamplebucket-1250000000内のオブジェクトに対し、GET（ダウンロード）およびHEAD操作を実行し、認証は必要ありませんでした。その他の事例については、権限設定の関連事例 をご参照ください。

```
{
    "version": "2.0",
    "principal":{
        "qcs": [
            "qcs: : cam: : anonymous: anonymous"
        ]
    },
    "statement":[
        {
            "action":[
                "name/cos: GetObject",
                "name/cos: HeadObject"
            ],
            "condition":{
                "ip_equal":{
                    "qcs: ip": [
                        "101.226.100.185",
                        "101.226.100.186"
                    ]
                }
            },
            "effect": "allow",
            "resource":[
                "qcs: : cos: ap-guangzhou: uid/1250000000:
examplebucket-1250000000.ap-guangzhou.myqcloud.com/*"
```

```
                ]
            }
        ]
    }
```

発効条件

最終更新日：：2025-07-14 10:13:41

概要

発効条件とはアクセスポリシー言語の一部であり、完全な発効条件には次の要素が含まれます。

- 条件キー：発効条件の具体的な種類を表します。例えば、ユーザーのアクセス元のIP、権限承認時間などです。
- 条件オペレーター：発効条件の判断方法を表します。
- 条件値：条件キーの値です。

詳細については CAM発効条件 をご参照ください。

> 説明：
> - 条件キーを使用してポリシーを作成する際は、必ず最小権限の原則を遵守し、適用可能なリクエスト（action）にのみ該当する条件キーを追加してください。アクション（action）を指定する際にワイルドカード「*」を使用すると、リクエストが失敗しますので避けてください。
> - Cloud Access Management（CAM）コンソールを使用してポリシーを作成する際は、構文形式にご注意ください。version、principal、statement、effect、action、resource、conditionの構文要素は全て小文字で記述してください。

発効条件の例

次のバケットポリシーの例における発効条件（condition）は、ユーザーが10.217.182.3/24または111.21.33.72/24ネットワークセグメントに属している場合にのみ、`cos:PutObject` アクションの権限付与が完了することを表します。このうち、

- 条件キーは `qcs:ip` であり、発効条件の種類がIPであることを表します。
- 条件オペレーターは `ip_equal` であり、発効条件の判断方法がIPアドレスの同一性であることを表します。
- 条件値は配列 `["10.217.182.3/24","111.21.33.72/24"]` であり、発効条件判断の規定値を表します。ユーザーが配列の中の任意のIPがあるネットワークセグメントに属している場合、条件判断はすべてtrueとなります。

```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
```

```
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "ip_equal":{
                    "qcs:ip":[
                        "10.217.182.3/24",
                        "111.21.33.72/24"
                    ]
                }
            }
        }
    ]
}
```

COSのサポートする条件キー

Cloud Object Storage（COS）のサポートする条件キーには2種類あり、1つはIP、VPC、HTTPSを含むすべてのリクエストに適用可能なもので、もう1つはリクエストヘッダーおよびリクエストパラメータによる条件キーであり、一般的にこのリクエストヘッダーまたはリクエストパラメータを持つリクエストにのみ適用できます。これらの条件キーに関する説明および実際の使用例については、条件キーの説明およびユースケースのドキュメントをご参照ください。

> 説明：
> 発効条件、条件キーなどの概念はすべてユーザーリクエストを対象としてCAMが実行するものです。ライフサイクル、バケットコピールールが有効になっている場合の削除、コピーなどの動作はCOSが行うものであり、ユーザーによるリクエストではないため、条件キーがライフサイクル、バケットコピールールに対して起こす動作は無効となります。

ほとんどのリクエストに適用される条件キー

第一類、ほとんどのリクエストに適用される条件キーは、以下の通りです。

条件キー	適用リクエスト	意味	タイプ
cos:secure-transport	すべてのリクエスト	リクエストにHTTPSプロトコルが適用されているかどうか確認	Boolean
qcs:ip	すべてのリクエスト	リクエスト元のIPネットワークセグメント	IP
cos:tls-version	すべてのhttpsリクエスト	httpsリクエストに使用しているTLSバージョン	Numeric
cos:host	すべてのリクエスト	リクエストのHostヘッダー	String

リクエストヘッダーおよびリクエストパラメータによる条件キー

2種類目は、リクエストヘッダー（Header）およびリクエストパラメータ（Param）による条件キーです。リクエストごとにリクエストヘッダーおよびリクエストパラメータが異なるため、これらの条件キーは一般的にこの種類のヘッダーまたはリクエストパラメータが含まれるリクエストにのみ適します。
例えば、条件キー `cos:content-type` は、リクエストヘッダー `Content-Type` を使用する必要があるアップロードクラスのリクエスト（PutObjectなど）に適用できます。条件キー `cos:response-content-type` はGetObjectオブジェクトにのみ適用できます。このリクエストのみがリクエストパラメータ `response-content-type` をサポートしているためです。
COSが現在サポートしている、リクエストヘッダーおよびリクエストパラメータによる条件キーと、それらの条件キーが適用可能なリクエストは下表のとおりです。

条件キー	適用リクエスト	リクエストヘッダー/リクエストパラメータの確認	タイプ
cos:x-cos-storage-class	PutObject PostObject InitiateMultipartUpload	リクエストヘッダー：x-cos-storage-class	String
cos:versionid	GetObject DeleteObject PostObjectRestore PutObjectTagging GetObjectTagging DeleteObjectTagging HeadObject	リクエストパラメータ：versionid	String
cos:prefix	GetBucket GetBucketObjectVersions ListMultipartUploads ListLiveChannels	リクエストパラメータ：prefix	String

cos:x-cos-acl	PutObject PutObject-Copy PostObject PutObjectACL PutBucket PutBucketACL InitiateMultipartUpload	リクエストヘッダー：x-cos-acl	String
cos:content-length	このリクエストは適用範囲が広いため、リクエストボディ付きのリクエストなどの代表的なリクエストに注目	リクエストヘッダー：Content-Length	Numeric
cos:content-type	このリクエストは適用範囲が広いため、リクエストボディ付きのリクエストなどの代表的なリクエストに注目	リクエストヘッダー：Content-Type	String
cos:response-content-type	GetObject	リクエストパラメータ：response-content-type	String
qcs:request_tag	PutBucket PutBucketTagging	リクエストヘッダー：x-cos-tagging リクエストパラメータ：tagging	String
cos:x-cos-forbid-overwrite	PutObject PutObject-Copy InitiateMultipartUpload CompleteMultipartUpload	リクエストヘッダー：x-cos-forbid-overwrite	String
cos:object-lock-mode	PutObject PutObject-Copy InitiateMultipartUpload PutObjectRetention	リクエストヘッダー：x-cos-object-lock-mode リクエストボディのフィールド：PutObjectRetentionリクエストボディのRetention.Mode	String
cos:object-lock-remaining-retention-days	PutObject PutObject-Copy InitiateMultipartUpload PutObjectRetention	リクエストヘッダー：x-cos-object-lock-remaining-retention-days リクエストボディのフィールド：PutObjectRetentionリクエストボディのRetention.RetainUntilDate	Timestamp

cos:object-lock-retain-until-date	PutObject PutObject-Copy InitiateMultipartUpload PutObjectRetention	リクエストヘッダー：x-cos-object-lock-remaining-retention-days リクエストボディのフィールド：リクエストボディのフィールド：PutObjectRetentionリクエストボディのRetention.RetainUntilDate	Timestamp
x-cos-grant-read	PutObject PutObject-Copy PostObject PutObjectACL PutBucket PutBucketACL InitiateMultipartUpload	リクエストヘッダー：x-cos-grant-read	String
x-cos-grant-read-acp	PutObject PutObject-Copy PostObject PutObjectACL PutBucket PutBucketACL InitiateMultipartUpload	リクエストヘッダー：x-cos-grant-read-acp	String
x-cos-grant-write	PutObject PutObject-Copy PostObject PutObjectACL PutBucket PutBucketACL InitiateMultipartUpload	リクエストヘッダー：x-cos-grant-write	String
x-cos-grant-write-acp	PutObject PutObject-Copy PostObject PutObjectACL PutBucket PutBucketACL InitiateMultipartUpload	リクエストヘッダー：x-cos-grant-write-acp	String
x-cos-grant-full-control	PutObject PutObject-Copy PostObject PutObjectACL	Request header：x-cos-grant-full-control	String

	PutBucket PutBucketACL InitiateMultipartUpload		

条件オペレーター

COSの条件キーは次の条件オペレーターをサポートしており、文字列（String）、数値型（Numeric）、ブール型（Boolean）およびIPなどの様々なタイプの条件キーに適用できます。

条件オペレーター	意味	タイプ
string_equal	文字列一致（大文字と小文字を区別）	String
string_not_equal	文字列不一致（大文字と小文字を区別）	String
string_like	文字列が類似（大文字と小文字を区別）。現在は文字列の前後へのワイルドカード `*` の追加をサポートしています（例：`image/*`）	String
ip_equal	IP一致	IP
ip_not_equal	IP不一致	IP
numeric_equal	数値一致	Numeric
numeric_not_equal	数値不一致	Numeric
numeric_greater_than	数値が大きい	Numeric
numeric_greater_than_equal	数値が同じか大きい	Numeric
numeric_less_than	数値が小さい	Numeric
numeric_less_than_equal	数値が同じか小さい	Numeric

_if_existの意味

上記のすべての条件オペレーターは、その後に `_if_exist` を追加することで単一条件オペレーターとすることができます。例えば、`string_equal_if_exist` などです。条件オペレーターに `_if_exist` が含まれるかどうかで異なる点は、リクエストに、条件キーに対応するリクエストヘッダーまたはリクエストパラメータが含まれない場合にどのように処理されるかの違いです。

- `_if_exist` を含まない条件演算子（例： `string_equal` ）を使用した場合、該当するリクエストヘッダーまたパラメータがリクエストに含まれていないと、デフォルトで条件に一致せず、 `False` となります。
- 条件オペレーターに `_if_exist` が含まれる場合、例えば `string_equal_if_exist` の場合は、リクエストに対応するリクエストヘッダー/リクエストパラメータが含まれないとき、デフォルトで条件にヒットし、 `True` となります。

事例

事例1：指定されたオブジェクトバージョンのダウンロードを許可

例えば、次のバケットポリシーについて、エフェクトがallowの場合は、リクエストパラメータversionidによる“MTg0NDUxNTc1NjIzMTQ1MDAwODg”のGetObjectリクエストの承認が許可されることを表します。条件にヒットした場合（True）、allowの権限付与ポリシーに基づいてリクエストは承認されます。条件にヒットしなかった場合（False）、allowの権限付与ポリシーに基づいて、リクエストは権限を得られず、リクエストは失敗します。

```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:GetObject"
            ],
            "condition":{
                "string_equal":{
                    "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ]
        }
```

```
    ]
}
```

条件オペレーターが `string_equal` または `string_equal_if_exist` の場合、conditionのヒット状況およびリクエストが承認されるかどうかは下表のとおりとなります。

条件オペレーター	リクエスト	conditionにヒットするか	リクエストが承認されるか
string_equal	versionidなし	FALSE	不承認
string_equal_if_exist	versionidなし	TRUE	承認
string_equal	versionid付き、指定のもの	TRUE	承認
string_equal_if_exist	versionid付き、指定のもの	TRUE	承認
string_equal	versionid付き、指定のもの以外	FALSE	不承認
string_equal_if_exist	versionid付き、指定のもの以外	FALSE	不承認

事例2：指定されたオブジェクトバージョンのダウンロードを拒否

次のバケットポリシーの例では、エフェクトがdenyであり、リクエストパラメータversionidによる「MTg0NDUxNTc1NjIzMTQ1MDAwODg」のGetObjectリクエストが拒否されることを表します。条件にヒットした場合（True）、denyの権限付与ポリシーに基づいてリクエストは失敗します。条件にヒットしなかった場合（False）、denyの権限付与ポリシーに基づいて、リクエストは拒否されません。

```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"deny",
            "action":[
```

```
                "name/cos:GetObject"
            ],
            "condition":{
                "string_equal":{
                    "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ]
        }
    ]
}
```

条件オペレーターが `string_equal` または `string_equal_if_exist` の場合、conditionのヒット状況およびリクエストが拒否されるかどうかは下表のとおりとなります。

条件オペレーター	リクエスト	conditionにヒットするか	リクエストが拒否される/拒否されない
string_equal	versionidなし	FALSE	拒否されない
string_equal_if_exist	versionidなし	TRUE	拒否される
string_equal	versionid付き、指定のもの	TRUE	拒否される
string_equal_if_exist	versionid付き、指定のもの	TRUE	拒否される
string_equal	versionid付き、指定のもの以外	FALSE	拒否されない
string_equal_if_exist	versionid付き、指定のもの以外	FALSE	拒否されない

関連説明

特殊文字はurlencodeでの処理が必要

リクエストパラメータ内の特殊文字はurlencodeでの処理が必要です。このため、バケットポリシーの中で、リクエストパラメータによる条件キーを使用する場合は、先にurlencode処理をしておく必要があります。例え

ば、 `cos:response-content-type` 条件キーを使用する際、条件値が"image/jpeg"であれば、必ずurlencodeで"image%2Fjpeg"に変換してからバケットポリシーに入力する必要があります。

最小権限の原則に従い、*の使用を避ける

条件キーを使用する際は最小権限の原則に従い、権限の設定が必要なactionのみを追加し、ワイルドカード「*」の使用は避けてください。ワイルドカード「*」を乱用すると、一部のリクエストが失敗する場合があります。例えば次の例のように、GetObject以外の他のリクエストはいずれもリクエストパラメータresponse-content-typeの使用をサポートしていません。
deny + string_equal_if_exist条件オペレーターはリクエスト内にこの条件キーがない場合、デフォルトでtrueとして処理します。このため、PutObject、PutBucketなどのリクエストを発行した際に、このdeny statementにヒットし、リクエストが拒否されます。
allow + string_equalはリクエストにこの条件キーがない場合、デフォルトでfalseとして処理します。このため、PutObject、PutBucketなどのリクエストを発行した際に、このallow statementにヒットすることができず、リクエストが許可されません。

```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"allow",
            "action":[
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        },
        {
```

```
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"deny",
            "action":[
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        }
    ]
}
```

別の方法として、allow+string_equal_if_existおよびdeny + string_not_equalを使用すると、response-content-typeリクエストパラメータを含まないリクエストが許可されます。
deny + string_equal条件オペレーターはリクエスト内にこの条件キーがない場合、デフォルトでfalseとして処理します。このため、PutObject、PutBucketなどのリクエストを発行した際に、このdeny statementにヒットせず、リクエストは拒否されません。
allow + string_equal_if_existはリクエスト内にこの条件キーがない場合、デフォルトでtrueとして処理します。このため、PutObject、PutBucketなどのリクエストを発行した際に、allow statementにヒットすることができ、リクエストは権限を取得します。
ただし、このように条件オペレーターを使用すると、GetObjectにresponse-content-typeを含めるかどうかの制限が行えなくなります。GetObjectにresponse-content-typeリクエストパラメータが含まれない場合は、他のリクエストと同様にデフォルトで許可されます。GetObjectにresponse-content-typeリクエストパラメータが含まれる場合のみ、ご自身で定めた条件に従って、リクエストパラメータの内容が意図するものと一致しているかを確認することができ、それによって条件付きの権限付与を実現できます。

```
{
    "version":"2.0",
    "statement":[
```

```
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"allow",
            "action":[
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "string_equal_if_exist":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"deny",
            "action":[
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "string_not_equal":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        }
```

```
    ]
}
```

このため、より安全な方法は、最小権限の原則に従い、ワイルドカード「*」を使用せず、actionをGetObjectに限定することとなります。
次の例では、ポリシーの発効条件は、「GetObjectリクエストに必ずresponse-content-typeが含まれ、かつリクエストパラメータの値が必ずimage%2Fjpeg"である場合にのみ権限を得られる」と厳格に限定されています。
その他のリクエストは次の例におけるポリシーの影響を受けないため、最小権限の原則に従って、追加で単独の権限を付与することができます。

```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:GetObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/1250000000:uin/1250000001"
                ]
            },
```

```
            "effect":"deny",
            "action":[
                "name/cos:GetObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-
1250000000/*"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        }
    ]
}
```

リクエスト方法の紹介
パーマネントキーを使用したCOSアクセス

最終更新日：：2025-11-24 16:58:21

背景の説明

RESTful APIによって、Cloud Object Storage（COS）に対しHTTP匿名リクエストまたはHTTP署名リクエストを送信することができます。匿名リクエストは一般的に、静的ウェブサイトのホスティングなどの、パブリックアクセスを必要とするケースに用いられます。そのほかの大多数のケースでは署名リクエストが必要となります。署名リクエストは匿名リクエストより、署名値を1つ多く持っています。署名はキー（SecretId/SecretKey）およびリクエスト情報を暗号化して生成した文字列をベースとしたものです。SDKは署名を自動計算しますので、お客様はユーザー情報の初期化の際にキーを設定しさえすれば、署名の計算について気にする必要はありません。RESTful APIを通じて送信するリクエストには、署名アルゴリズムに基づいて計算した署名を追加する必要があります。

パーマネントキーの取得

パーマネントキーはCAMコンソールのAPIキー管理ページで取得できます。パーマネントキーにはSecretIdとSecretKeyが含まれ、アカウントの永続的なIDを表します。有効期限はありません。

- SecretId：APIを呼び出した人のID識別に用いられます。
- SecretKey：署名文字列の暗号化およびサーバーによる署名文字列の検証に用いられるキーです。

パーマネントキーを使用したCOSアクセス

APIリクエストによるCOSアクセス

APIリクエストを使用する場合、プライベートバケットの場合は必ず署名リクエストを使用しなければなりません。パーマネントキーで署名を生成し、Authorizationヘッダーに追加することで、署名リクエストを作成できます。リクエストをCOSに送信すると、COSは署名とリクエストが一致しているかを検証します。

> 説明：
> 署名生成のアルゴリズムは複雑なため、SDKを直接使用してリクエストを送信することで、このプロセスを省略することをお勧めします。

1. パーマネントキーによる署名生成
   署名アルゴリズムの説明については、リクエスト署名のドキュメントをご参照ください。COSは署名生成ツールもご提供しており、SDKで署名を生成することもできます。SDKの署名実装をご参照ください。ご自身でプログラムを作成して署名を生成することも可能ですが、署名アルゴリズムは複雑なため、通常その方法は推奨されません。

2. Authorizationヘッダーの入力
   APIリクエストを送信する際、署名を標準Http Authorizationヘッダーに入力します。次はGetObjectリクエストの例です。

```
GET /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: q-sign-algorithm=sha1&q-ak=SecretId&q-sign-
time=KeyTime&q-key-time=KeyTime&q-header-list=HeaderList&q-url-param-
list=UrlParamList&q-signature=Signature
```

SDKツールによるCOSアクセス

1. パーマネントキーによるID情報の初期化
   SDKツールのインストールが完了した後、最初にユーザーのID情報の初期化が必要です。ルートアカウントまたはサブアカウントのパーマネントキー（SecretIdおよびSecretKey）を入力します。
2. SDKを直接使用するCOSリクエスト
   初期化すると、SDKツールを使用してアップロード・ダウンロードなどの基本操作を直接行うことができるようになります。SDKツールがお客様の代わりにキーによって署名を生成し、COSにリクエストを送信するため、APIリクエストのようにご自身で署名を生成する必要はありません。

例えば、Java SDKのコードは次のようになります。その他の言語のdemoについては、SDKの概要 のクイックスタートドキュメントをご参照ください。

```
// 1 ユーザーID情報（secretId、secretKey）を初期化します。
// SECRETIDおよびSECRETKEYについてはCAMコンソール
https://console.intl.cloud.tencent.com/cam/capiで確認および管理を行ってくださ
い
String secretId = "SECRETID";
String secretKey = "SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
// 2 bucketのリージョンを設定します。COSリージョンの略称については
https://cloud.tencent.com/document/product/436/6224をご参照ください
// clientConfigにはregion、https(デフォルトではhttp)、タイムアウト、プロキシなど
を設定するsetメソッドが含まれます。使用にあたってはソースコードまたはよくあるご質問の
Java SDKのパートを参照できます。
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// ここではhttpsプロトコルの設定と使用を推奨します
// バージョン5.6.54からは、デフォルトでhttpsを使用します
```

```
clientConfig.setHttpProtocol(HttpProtocol.https);
// 3 cosクライアントを生成します。
COSClient cosClient = new COSClient(cred, clientConfig);
```

一時キーを使用したCOSアクセス

最終更新日：：2025-11-24 17:02:54

一時キーの説明

一時キーは、Security Token Service（STS）が提供する一時アクセスのための証明書です。一時キーはTmpSecretId、TmpSecretKey、Tokenの3つの部分から成ります。パーマネントキーと異なり、一時キーには次のような特徴があります。

- 有効期間が短く（30分～36時間）、パーマネントキーを開示する必要がないため、アカウント漏洩のリスクが低くなります。
- 一時キーを取得する際、policyパラメータを渡して一時的な権限を設定することで、使用者の権限の範囲をさらに限定することができます。

このため、一時キーはフロントエンドの直接転送などの、一時的な権限承認のシーンに適します。信頼性の低いユーザーに対しては、パーマネントキーではなく一時キーを発行することで、安全性を高めることができます。

一時キーの取得

一時キーは、当社のご提供するCOS STS SDK方式で取得するか、またはSTSのクラウドAPI方式によって直接取得することができます。

詳細については、一時キーの生成および使用ガイド をご参照ください。

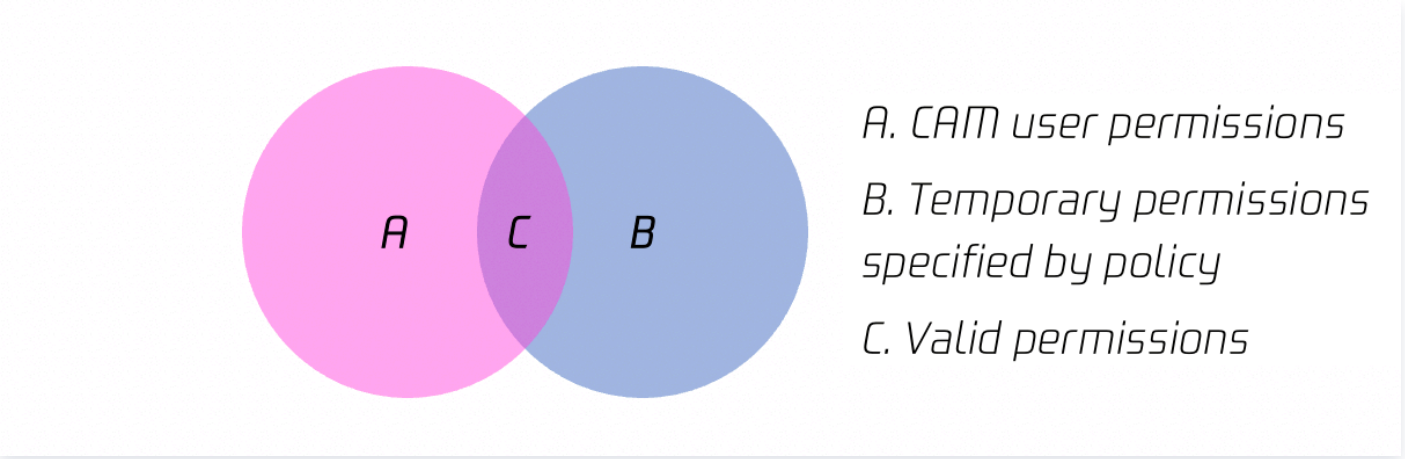
一時キーの権限

一時キーを申請する前に、Cloud Access Management（CAM）ユーザー（Tencent Cloudルートアカウントまたはサブアカウント）を取得しておく必要があります。Policyパラメータを設定することで、一時キーに一時ポリシーを追加して使用者の権限を制限することができます。

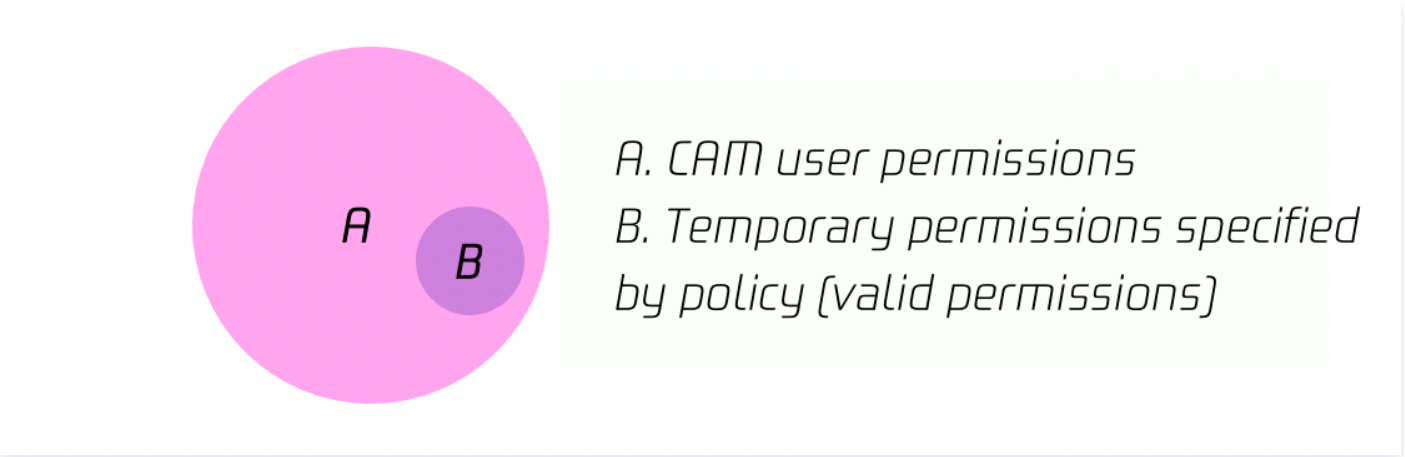
- policyパラメータを設定しない場合、取得した一時キーはCAMユーザーと同等の権限を有します。
- policyパラメータを設定した場合、取得した一時キーの権限は、CAMユーザーの権限をベースにして、さらにpolicyで設定した範囲内に制限されます。

例えば、「A」がCAMユーザーの従来の権限を表し、「B」がpolicyパラメータによって一時キーに設定した権限を表すとした場合、「A」と「B」の共通部分が一時キーの最終的に有効な権限となります。

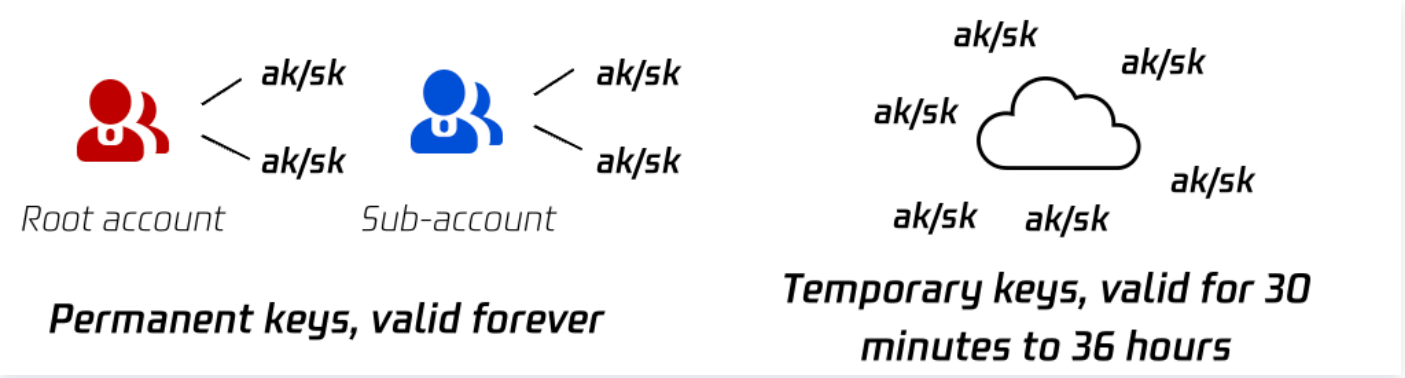
下図のように、CAMユーザー権限とpolicyの一時権限の共通部分が有効な権限です。



下図のように、policyがCAMユーザー権限の範囲内にある場合は、policyが有効な権限となります。



一時キーを使用したCOSアクセス



一時キーにはSecretId、SecretKey、Tokenが含まれます。各ルートアカウントおよびサブアカウントはいずれも複数の一時キーを生成することができます。パーマネントキーと異なり、一時キーの有効期間は30分から36時間しかありません。一時キーはフロントエンドの直接転送などの、一時的な権限承認のシーンに適します。信頼性の低いユーザーに対しては、パーマネントキーではなく一時キーを発行することで、安全性を高めることができます。詳細については、一時キーの生成および使用ガイドおよびフロントエンドの直接転送に用いる一時キーの使用ガイドをご参照ください。

- APIリクエストの送信
  パーマネントキーと同じように、一時キーによっても署名を生成することができます。リクエストヘッダーにAuthorizationを入力することで、署名リクエストを作成します。COSはリクエストを受信すると、署名が有効か、および一時キーが有効期間内かどうかをチェックします。
  署名アルゴリズムの説明については、リクエスト署名をご参照ください。COSは署名生成ツールもご提供しており、SDKで署名を生成することもできます。SDKの署名実装をご参照ください。
- SDKツールの使用
  SDKツールをインストールすると、一時キーを使用してユーザーのID情報を初期化できるほか、一時キー（SecretId、SecretKey、Token）を使用してCOSClientを初期化し、署名を生成せずに、SDKを使用して直接アップロード、ダウンロードなどの操作を行うことができるようになります。一時キーの生成については一時キーの生成および使用ガイドをご参照ください。

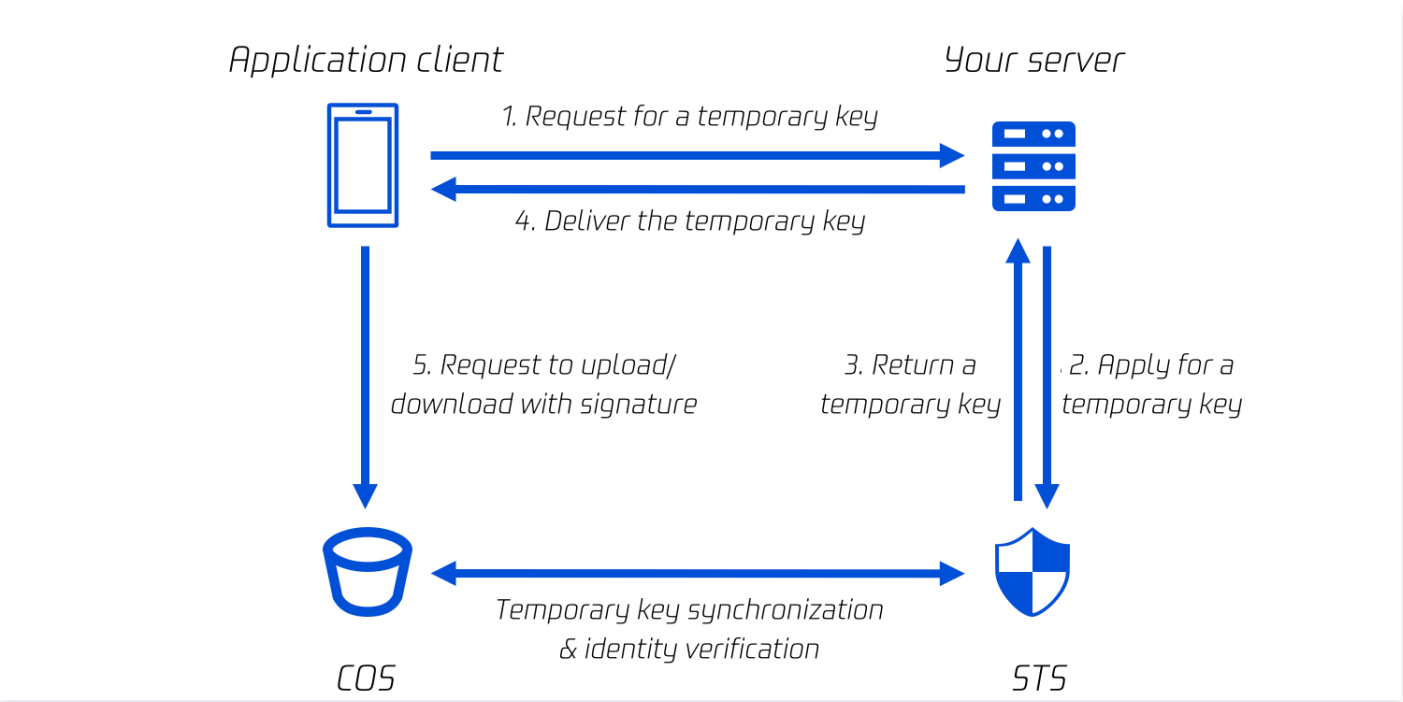
Java SDKについては次の例をご参照ください。その他の言語のdemoについては、SDKの概要をご参照ください。

```
// 1 取得した一時キー (tmpSecretId、tmpSecretKey、sessionToken)を渡します
String tmpSecretId = "SECRETID";
String tmpSecretKey = "SECRETKEY";
String sessionToken = "TOKEN";
BasicSessionCredentials cred = new BasicSessionCredentials(tmpSecretId,
tmpSecretKey, sessionToken);
// 2 bucketのリージョンを設定します。COSリージョンの略称については
https://cloud.tencent.com/document/product/436/6224をご参照ください
// clientConfigにはregion、https(デフォルトではhttp)、タイムアウト、プロキシなど
を設定するsetメソッドが含まれます。ご利用にあたっては、ソースコードまたはよくあるご質
問のJava SDKのパートをご参照ください
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// 3 cosクライアントを生成します
COSClient cosClient = new COSClient(cred, clientConfig);
```

一時キーのユースケース

一時キーは主に第三者に対しCOSへの一時的なアクセス権限を承認するために用いられます。例えば、ユーザーがクライアントAppを開発し、データをCOSバケットに保存したとします。この場合、パーマネントキーを直接Appクライアント上に置くことはセキュリティ上問題がありますが、クライアントに対してはアップロード・ダウ

ンロードの権限を承認する必要があります。このようなケースでは一時キーを使用することができます。



上の図のように、ユーザーがAppクライアントを開発し、ユーザーのサーバー上にパーマネントキーを保存し、一時キーを使用してフロントエンド直接転送を行うには次のいくつかの手順が必要です。

1. Appクライアントからユーザーサーバーに、データアップロード・ダウンロード用の一時キーをリクエストします。
2. ユーザーサーバーはパーマネントキーのIDを使用して、STSサーバーに一時キーを申請します。
3. STSサーバーはユーザーサーバーに一時キーを返します。
4. ユーザーサーバーは一時キーをAppクライアントに送信します。
5. Appクライアントは一時キーを使用して署名リクエストを生成し、COSに対しデータのアップロード・ダウンロードをリクエストします。

一時キーはフロントエンドデータの直接転送のユースケースに適しています。次のベストプラクティスをご参照の上、一時キーをご利用ください。

- Web端末直接転送の実践
- ミニプログラム直接転送の実践
- モバイルアプリケーション直接転送の実践

署名付きURLを使用したCOSアクセス

最終更新日：：2025-11-13 16:35:30

Cloud Object Storage（COS）は署名付きURLを使用したオブジェクトのアップロード、ダウンロードをサポートしています。その原理は、URLに署名を埋め込んで署名付きリンクを生成するものです。署名の有効期限によって、署名付きURLの有効期間を管理することができます。
署名付きURLを使用してダウンロードを行い、一時URLを取得してファイル、フォルダの一時的な共有に用いることができます。あるいは長い署名有効期間を設定することで、長期間有効なURLを取得し、ファイルの長期的な共有に用いることもできます。詳細については、ファイルの共有 をご参照ください。
また、署名付きURLを使用してアップロードを行うこともできます。詳細については、ファイルのアップロード をご参照ください。

ファイルの共有（ファイルのダウンロード）

COSはオブジェクトの共有をサポートしています。署名付きURLを使用することで、ファイルやフォルダを他のユーザーと期限付きで共有することができます。署名付きURLの原理は、署名をオブジェクトURLに埋め込んで結合するものです。その後の署名生成アルゴリズムについては、リクエスト署名 をご参照ください。
バケットはデフォルトではプライベート読み取りであり、オブジェクトのURLから直接ダウンロードしようとするとアクセスエラーが表示されます。オブジェクトURLの後に有効な署名を結合すると、署名付きURLを取得できます。署名にはID情報が含まれるため、署名付きURLはオブジェクトのダウンロードに用いることができます。

> **説明：**
> やむを得ずパーマネントキーを使用して署名付きURLを生成する場合は、リスク回避のため、パーマネントキーの権限の範囲をアップロードまたはダウンロード操作のみに限定することをお勧めします。また、生成した署名の有効期間を、今回のアップロードまたはダウンロード操作に必要な最短の期限までに設定し、指定した署名付きURLの有効期限が過ぎるとリクエストが中断するようにします。失敗したリクエストは新しい署名を申請後に再度実行する必要があります。中断からの再開はサポートしていません。

```
// オブジェクトURL
https://test-12345678.cos.ap-beijing.myqcloud.com/test.png

// 署名付きURL（署名値を結合したオブジェクトURL）
https://test-12345678.cos.ap-beijing.myqcloud.com/test.png?q-sign-algorithm=sha1&q-ak=xxxxx&q-sign-time=1638417770;1638421370&q-key-time=1638417770;1638421370&q-header-list=host&q-url-param-list=&q-signature=xxxxxfxxxxxx6&x-cos-security-token=xxxxxxxxxxxx
```

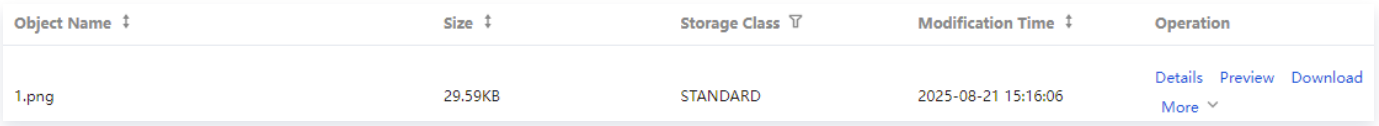
次にファイル共有の方法をいくつかご紹介します。これらの方法は本質的にはすべて署名を自動生成し、オブジェクトURLの後ろに結合することで、ダウンロードやプレビューに直接用いることができる一時リンクを生成するものです。

一時リンクのクイック取得（有効期間1～2時間）

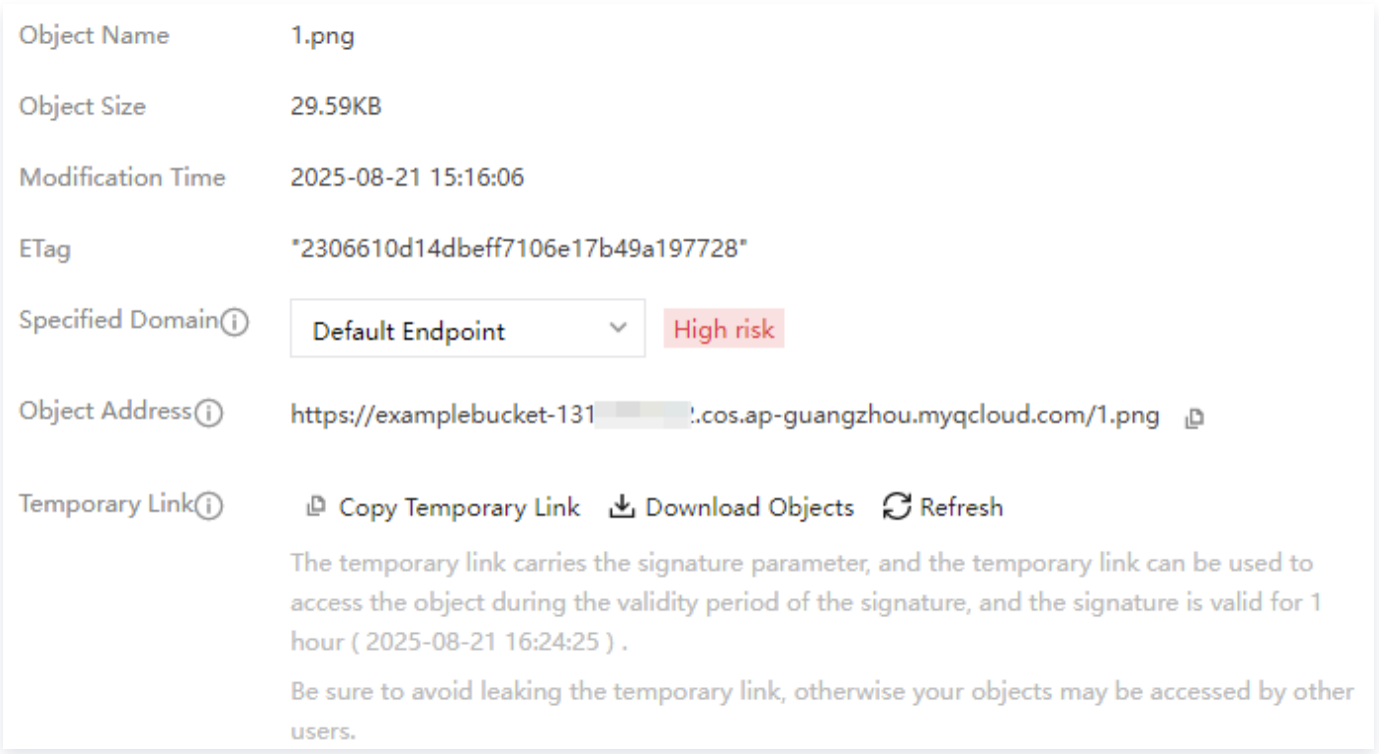
コンソールまたはCOSBrowserツールによってオブジェクトの一時リンクをクイック取得することができます。

コンソール（Webページ）

1. COSコンソールにログインし、バケット名をクリックし、「ファイルリスト」に進み、オブジェクトの詳細をクリックします。



2. オブジェクトの詳細ページに進み、一時リンクをコピーします。有効期間は1時間です。



COSBrowser（クライアント）

ドキュメントファイルリンクの発行を参照し、ルートアカウントキーを使用して最長2時間の一時リンクを取得することができます。サブアカウントキーを使用する場合は、最長1.5日間の一時リンクを取得することができます。

時間をカスタマイズした一時リンクの取得

SDKを使用した署名付きURLの一括取得

適するケース：一時リンクを一括取得したい場合、プログラミングの基礎を習得したユーザー
コンソールおよびCOSBrowserから取得する一時リンクは有効期間が短いため、より長時間の一時リンクが必要な場合は、SDKを使用して署名付きURLを生成し、署名の有効期間の管理を実現することもできます。生成メソッドについては署名付きURLによるダウンロード権限承認を参照し、使いやすい開発言語を選択してください。
署名付きURLの生成には一時キーまたはパーマネントキーを使用することができます。両者の違いは、一時キーの最長有効期間は36時間以内であり、パーマネントキーには有効期限がないという点です。このことは署名付きURLの有効期間に間接的な影響を与えます。
パーマネントキーを使用した署名付きURLの生成（任意の期間）
パーマネントキーには有効期限がないため、署名付きURLの有効期間は設定した署名の有効期間によって決まります。SDKの署名付きURL生成メソッドを直接呼び出すことができます。操作手順は次のとおりです。

1. secret_id、secret_key、regionなどを入力してclientを初期化します。
2. バケット名、オブジェクト名、署名の有効期間を入力し、時間をカスタマイズした署名付きURLを生成します。詳細については下記の各言語のSDKドキュメントをご参照ください。

Android SDK	C SDK	C++ SDK	.NET SDK
Go SDK	iOS SDK	Java SDK	JavaScript SDK
Node.js SDK	PHP SDK	Python SDK	ミニプログラムSDK

一時キーを使用した署名付きURLの生成（36時間以内）
フロントエンドデータの直接転送のケースでは、一時キーの使用が必要な場合が多くあります。一時キーの説明と生成ガイドについては次をご参照ください。

- 一時キーを使用したCOSアクセス
- 一時キーの生成および使用ガイド
- COSへのフロントエンド直接転送に用いる一時キーのセキュリティガイド

一時キーは最長36時間までであり、署名付きURLの有効期間は設定した署名の有効期間と一時キーの有効期間の最小値から決定されます。設定した署名の有効期間をX、一時キーの有効期間をY、リンクの実際の有効期間をTとします。

```
T=min(X,Y)。X<=36のため、T<=36です。
```

一時キーを使用して署名付きURLを生成するには、次の2つの手順が必要です。

1. 一時キーの取得を行います。
2. 一時キーを取得すると、パーマネントキーに類似した関数を使用して署名付きURLを生成できるようになります。一時キーを使用してclientを初期化する場合、SecretId、SecretKeyの入力のほかにtokenも入力し、パラメータ `x-cos-security-token` も含める必要があることに注意が必要です。詳細については、下記の各言語のSDKドキュメントをご参照ください。

Android SDK	C SDK	C++ SDK	.NET SDK

Go SDK	iOS SDK	Java SDK	JavaScript SDK
Node.js SDK	PHP SDK	Python SDK	ミニプログラムSDK

フォルダの共有

フォルダは一種の特殊なオブジェクトであり、コンソールまたはCOSBrowser ツールを使用してフォルダを共有することができます。詳細については、フォルダの共有 をご参照ください。

ファイルのアップロード

第三者がオブジェクトをバケットにアップロードできるようにし、なおかつ相手にCAMアカウントまたは一時キーなどの方法を使用させたくない場合は、署名付きURLを使用して署名を第三者に渡し、一時的なアップロード操作を完了させることができます。有効な署名付きURLを受領した人は誰でもオブジェクトをアップロードできます。

> 説明:
> やむを得ずパーマネントキーを使用して署名付きURLを生成する場合は、リスク回避のため、パーマネントキーの権限の範囲をアップロードまたはダウンロード操作のみに限定することをお勧めします。また、生成した署名の有効期間を、今回のアップロードまたはダウンロード操作に必要な最短の期限までに設定し、指定した署名付きURLの有効期限が過ぎるとリクエストが中断するようにします。失敗したリクエストは新しい署名を申請後に再度実行する必要があります。中断からの再開はサポートしていません。

- 手段1：SDKを使用した署名付きURLの生成
  各言語のSDKがアップロード署名付きURLの生成メソッドを提供しています。生成メソッドについては、署名付きURLによるアップロード権限承認 を参照し、使いやすい開発言語を選択してください。
- 手段2：ご自身での署名リンクの結合
  署名付きURLは実際にはオブジェクトURLの後に署名を結合したものです。このため、SDK、署名生成ツールなどによってご自身で署名を生成し、URLと署名を結合して署名リンクとし、オブジェクトのアップロードに用いることもできます。ただし、署名生成のアルゴリズムは複雑なため、一般的な状況ではこの方法の使用は推奨されません。

匿名でのCOSアクセス

最終更新日：：2024-06-26 11:09:29

Cloud Object Storage（COS）バケットはデフォルトではプライベートであり、COSへのアクセスはID認証を経る必要があり、オブジェクトのURLによるCOSアクセスには署名が必要です。ただし、リソース（バケット、オブジェクト、フォルダ）がパブリック読み取りに開放された場合は匿名アクセスが許可され、オブジェクトのURLを通じてリソースを直接ダウンロードすることが可能になります。
COSは権限開放の範囲に基づき、バケットレベル、オブジェクトレベル、フォルダレベルでのパブリック読み取り設定をサポートしています。

バケットをパブリック読み取りとして開放する

バケットをパブリック読み取り・プライベート書き込みとして開放すると、バケット内のすべてのオブジェクトに匿名でのアクセスが可能になります。設定方法については、バケットアクセス権限の設定をご参照ください。

オブジェクトをパブリック読み取りとして開放する

指定のオブジェクトをパブリック読み取り・プライベート書き込みとして開放すると、そのオブジェクトにオブジェクトのURLから直接アクセスすることが可能になります。設定方法については、オブジェクトのアクセス権限の設定をご参照ください。

フォルダをパブリック読み取りとして開放する

フォルダをパブリック読み取り・プライベート書き込みとして開放すると、そのフォルダ内のすべてのファイルに匿名でのアクセスが可能になります。設定方法については、フォルダの権限の設定をご参照ください。

パブリック読み取り権限の評価の仕組み

COS権限の評価の仕組みについては、アクセスポリシーの評価フローをご参照ください。バケット、フォルダ、オブジェクトレベルでのパブリック読み取り権限設定に競合が発生した場合、優先順位は次のようになります。
あるオブジェクトのACLに対しては、オブジェクトACLの優先度が最も高くなります。オブジェクトACLが継承権限の場合は、フォルダACLに準じます。フォルダACLが継承権限の場合は、バケットACLに準じます。

CDNアクセラレーションを使用したアクセス
CDNアクセラレーションの概要

最終更新日：：2024-06-26 11:09:29

Content Delivery Network（CDN）を使用してCloud Object Storage（COS）のアクセラレーションを行うことで、バケット内のコンテンツを広範囲にダウンロード、配信することができます。特に、同一のコンテンツを繰り返しダウンロードするユースケースに適しています。back-to-origin認証機能を使用すると、CDNを使用したプライベート読み取りバケット内のコンテンツのアクセラレーションを実現できます。CDN認証機能を使用すると、コンテンツを正当なユーザーにのみダウンロード可能とすることができ、ダウンロードを開放することに伴うデータセキュリティおよびトラフィックコストなどの問題を防ぐことができます。

> 説明：
> CDNアクセラレーションドメイン名を有効にした場合、CDNアクセラレーションドメイン名を使用してデータのダウンロードやアクセスを行うと、CDN back-to-originラフィックとCDNトラフィックが発生します。詳しくは、COSをCDNオリジンサーバーとした場合発生するトラフィック をご参照ください。

Content Delivery Network

CDNの定義

CDNは既存のInternet上に追加された新しいネットワークアーキテクチャのレイヤーであり、世界各国に分布する高性能なアクセラレーションノードで構成されます。これらの高性能なサービスノードは一定のキャッシュポリシーに従ってお客様の業務コンテンツを保存しており、お客様のユーザーがある業務コンテンツをリクエストすると、リクエストがユーザーから最も近いサービスノードにスケジューリングされ、サービスノードから直接迅速にレスポンスすることで、ユーザーのアクセス遅延を効果的に低減し、可用性を向上させます。
CDNはキャッシュおよびback-to-origin行為を行います。すなわち、ユーザーがあるURLにアクセスしたとき、エッジノードに解決されてもレスポンスを必要とするキャッシュコンテンツにヒットしなかった場合、またはキャッシュが期限切れの場合は、オリジンサーバーに戻ってレスポンスを必要とするコンテンツを取得するものです。

適用ケース

- レスポンス遅延およびダウンロード速度に対する要件が比較的厳しいケース。
- 地域、国、大陸を越えて数GBから数TBのデータを転送する必要があるケース。
- 同一のコンテンツを集中的に繰り返しダウンロードする必要があるケース。

セキュリティのタイプ

- back-to-origin認証：ユーザーのリクエストしたデータがエッジノードでキャッシュにヒットしなかった場合、CDNはback-to-originを行ってデータ内容を取得する必要があります。COSをオリジンサーバーとして使用し、back-to-origin認証を有効にすると、CDNエッジノードは特殊なサービスIDを使用してCOSオリジンサーバーにアクセスし、プライベートアクセスバケット内のデータの取得とキャッシュを実現することができます。
- CDNサービス権限承認：CDNエッジノードはCDNサービス権限承認を追加することにより、特殊なサービスIDを使用してCOSオリジンサーバーにアクセスすることができるようになります。CDNサービス権限承認を追加してからでなければ、back-to-origin認証は有効化できません。
- CDN認証設定：ユーザーがエッジノードにアクセスしてキャッシュデータを取得する際、エッジノードは認証設定ルールに基づいて、URLアクセスにおけるID認証フィールドを検証することで、権限のないアクセスを防止し、リンク不正アクセス防止を実現し、エッジノードのキャッシュデータの安全性と信頼性を高めます。

COSのアクセスノード

アクセスノードの定義

アクセスノードとはユーザーがバケットを作成する際、バケットのリージョンおよび名称に基づいて、システムが自動的にバケットに割り当てるアクセスドメイン名であり、このドメイン名によってバケット内のデータにアクセスすることができます。
静的ウェブサイト機能を有効にすると、静的ウェブサイトのアクセスノードを1つ追加で取得でき、それはデフォルトのアクセスノードとは性質の異なる、特殊な設定のレスポンス内容を表示するために用いられます。

アクセスノード

- アクセスノード：ユーザーがバケットを作成すると、COSはバケットに対し自動的に1つのXMLアクセスノードを割り当てます。 このアクセスノードの形式は<BucketName-APPID>.cos.<Region>.myqcloud.comであり、RESTful APIによるアクセスに適用されます。ユーザーはアクセスドメイン名によって、またAPIドキュメントの実行ルールを確認してバケットの設定を行ったり、オブジェクトのアップロード、ダウンロード操作を行ったりすることができます。
- 静的ウェブサイトノード：コンソール上のバケット基本設定画面で静的ウェブサイトのホスティング機能を有効にすることができ、有効にすると1つのアクセスノードが提供されます。アクセスノードの形式は<BucketName-APPID>.cos-website.<Region>.myqcloud.comです。静的ウェブサイトは特殊なインデックスページ（IndexPage）、エラーページ（ErrorPage）およびリダイレクトなどをサポートしており、サポートはオブジェクトのダウンロード系の操作のみとなります。ユーザーは静的ウェブサイトのドメイン名によってコンテンツを取得できます。

アクセス権限

- パブリック読み取り：バケットをパブリック読み取りに設定すると、バケットのアクセスドメイン名によって誰でもそのバケットにアクセスすることができます。ユーザーがパブリック読み取りバケットをオリジンサーバーとしてback-to-originを行う場合は、CDNアクセラレーションを直接有効化でき、CDN認証およびback-to-origin認証を使用する必要はありません。

- プライベート読み取り：バケットをプライベート読み取りに設定すると、ユーザーはアクセスポリシーを作成することによってアクセス者を管理でき、これにはCDNサービス権限承認の管理なども含まれます。ユーザーがプライベート読み取りバケットをオリジンサーバーとしてback-to-originを行う場合、back-to-origin認証を有効にしていてもCDN認証を有効にしていなければ、権限範囲外の人がCDNによって直接バケットにアクセスすることが可能になってしまいます。このため、プライベート読み取りバケットについては、CDN認証とback-to-origin認証を同時に有効にすることでデータの安全性を保障するよう強く推奨します。

CDNアクセラレーションを使用したCOSアクセス

ユーザーはカスタムCDNアクセラレーションドメイン名によって、COSへのアクセラレーションアクセスを行うことができます。ユーザーは初めにICP登録済みのカスタムドメイン名をご自身で準備し、オリジンサーバーをCOSバケットに指定することで、カスタムCDNアクセラレーションドメイン名を使用したバケット内のオブジェクトに対するアクセラレーションアクセスを実現できます。

> 説明：
> Tencent Cloud CDNのアクセラレーションドメイン名はデフォルトではIPアドレスを提供しません。ドメイン名の解決状況を確認したい場合は、digコマンドを使用して照会することができます。

パブリック読み取りバケット

バケットをパブリックアクセス許可に設定すると、CDN back-to-originをCOSアクセスノードに設定した場合、back-to-origin認証を有効にしなくても、CDNエッジノードがバケット内のオブジェクトデータを取得してキャッシュすることが可能です。
CDNコンソールで 認証設定 を有効にして、バケット内のデータを限定的に保護することも引き続き可能です。CDNのこの機能を有効にしているかどうかにかかわらず、バケットのアクセスドメイン名を知っているユーザーはバケット内のすべてのオブジェクトにアクセス可能なためです。CDN認証設定の違いによる、ドメイン名のパブリック読み取りバケットに対するアクセス機能の違いについては次の表をご参照ください。

CDN認証設定	CDNアクセラレーションドメイン名アクセス	COSドメイン名アクセス	一般的なケース
無効（デフォルト）	アクセス可	アクセス可	全サーバーでパブリックアクセス許可、CDNとオリジンサーバーどちらからのアクセスも可
有効	URLを使用した認証が必要	アクセス可	CDNアクセスに対してはリンク不正アクセス防止が有効、ただしオリジンサーバーアクセスは保護されないため非推奨

プライベート読み取りバケット

バケットがデフォルトのプライベート読み取りである場合、CDN back-to-originをCOSアクセスノードに設定すると、CDNエッジノードはデータの取得とキャッシュが一切できなくなります。このため、CDNサービスIDをバケットアクセスポリシー（Bucket Policy）に追加するとともに、そのIDによる次の操作の実行を許可する必要があります。

- GET Object：オブジェクトのダウンロード
- HEAD Object：オブジェクトメタデータの照会
- OPTIONS Object：クロスドメイン設定のプリフライトリクエスト

CDNコンソールとCOSコンソールはいずれもワンクリックでの権限承認機能をご提供しています。CDNサービス権限承認の追加をクリックすれば完了です。この操作の完了後、back-to-origin認証オプションを有効にする必要があります。これでCDNエッジがこのサービスIDを使用してCOS内のデータにアクセスできるようになります。

> ⚠ 注意：
> - バケットがプライベート読み取りに設定されている場合は、必ず権限承認を追加し、back-to-origin認証を有効にしてください。これを行わなければCOSはアクセスを拒否します。
> - CDNエッジは各ルートアカウントにつき、1つのサービスアカウントを生成します。このため、アカウントの権限承認はアクセラレーションドメイン名が所属するルートアカウントに対してのみ有効であり、異なるアカウント間でバインドしたアクセラレーションドメイン名はアクセスが拒否されます。

CDNサービス権限承認を追加し、back-to-origin認証を有効にすると、CDNエッジノードはデータを直接取得およびキャッシュできるようになります。このため、プライベートデータの保護が必要な場合は、認証設定を有効にして、バケット内のデータを保護することを強く推奨します。CDN認証設定の違いによる、ドメイン名のプライベート読み取りバケットに対するアクセス機能の違いについては次の表をご覧ください。

CDN認証設定	CDNアクセラレーションドメイン名アクセス	COSドメイン名アクセス	一般的なケース
無効（デフォルト）	アクセス可	COSを使用した認証が必要	CDNドメイン名に直接アクセス可能、オリジンサーバーデータ保護
有効	URLを使用した認証が必要	COSを使用した認証が必要	全リンクのアクセスを保護、CDN認証リンク不正アクセス防止をサポート

CDNアクセラレーションの設定

最終更新日：：2024-06-26 11:09:29

ユースケース

- レスポンス遅延およびダウンロード速度に対する要件が比較的厳しいケース。
- 地域、国、大陸を越えて数GBから数TBのデータを転送する必要があるケース。
- 同一のコンテンツを集中的に繰り返しダウンロードする必要があるケース。

> 注意：
> ダウンロードリクエストがTencent Cloud VPC内（例えばTencent Cloud CVMを使用したバケットへのアクセスなど）の場合は、COSの標準ドメイン名をそのまま使用することをお勧めします。カスタムCDNアクセラレーションドメイン名を使用すると、ユーザーはパブリックネットワークを経由してCDNノードにアクセスしたことになり、CDN back-to-originトラフィック料金、CDNトラフィック料金などが追加で発生します。

関連説明

ドメイン名の定義、CDN back-to-origin認証およびCDN認証設定に関しては、CDNアクセラレーションの概要のドキュメントでご紹介しているため、ここには記載しません。
CDN back-to-origin認証、CDN認証設定はカスタムCDNアクセラレーションドメイン名およびCOSドメイン名のオリジンサーバーバケットへのアクセス方式に影響する場合があります。具体的には下表のとおりです。

バケットへのアクセス権限	CDN back-to-origin認証を有効にしているか	CDN認証設定を有効にしているか	カスタムCDNアクセラレーションドメイン名によってオリジンサーバーにアクセス可能か	COSオリジンサーバードメイン名によってオリジンサーバーにアクセス可能か	適用ケース
パブリック読み取り	無効	無効	アクセス可	アクセス可	全サーバーパブリックアクセス
パブリック読み取り	有効	無効	アクセス可	アクセス可	非推奨
パブリック読み取り	無効	有効	URLを使用した認証が必要	アクセス可	非推奨

パブリック読み取り	有効	有効	URLを使用した認証が必要	アクセス可	非推奨
プライベート読み取り+CDNサービス権限承認	有効	有効	URLを使用した認証が必要	COSを使用した認証が必要	全リンク保護
プライベート読み取り+CDNサービス権限承認	無効	有効	URLを使用した認証が必要	COSを使用した認証が必要	非推奨
プライベート読み取り+CDNサービス権限承認	有効	無効	アクセス可	COSを使用した認証が必要	オリジンサーバー保護
プライベート読み取り+CDNサービス権限承認	無効	無効	アクセス不可	COSを使用した認証が必要	非推奨
プライベート読み取り	無効	有効または無効	アクセス不可	COSを使用した認証が必要	CDN利用不可

⚠ 注意：
上記の表中のオリジンサーバー保護のケースでは、CDN認証設定を有効にしていない場合、CDNエッジノードにキャッシュしたデータが悪意をもってプルされるおそれがあるため、同時にCDN認証設定を有効にすることでデータの安全性を保障するよう、強く推奨します。

CDNアクセラレーションの設定

ユーザーはCOSコンソール上でバケットにカスタムドメイン名をバインドした後、カスタムドメイン名でCDNアクセラレーションを有効にし、最後にカスタムドメイン名によってバケットへのアクセスをアクセラレーションすることができます。カスタムドメイン名をバインドする際、ユーザーはご自身でカスタムドメイン名のサービスプロバイダにCNAME解決の追加を依頼する必要があります。

⚠ 注意：
現在COSでカスタムアクセラレーションドメイン名を使用している場合はCDNを有効にする必要があります。ご自身の状況に応じて判断してください。

1. ドメイン名で国内のCDNにアクセスするには、ICP登録が必要です。ただしICP登録はTencent Cloudを通じて行わなくてもよく、アクセスするドメイン名が確実にICP登録されていればアクセスできます。
2. ドメイン名で海外のCDNにアクセスする場合は、ICP登録の必要はありません。ただし、Tencent Cloud上で保存するデータおよび操作行為については、関係国の法律法令、ならびに『Tencent Cloudサービス契約』を遵守する必要があることに必ずご注意ください。

前提条件

1. ドメイン名を登録していること。Tencent Cloudのドメイン名登録またはその他のサービスプロバイダを通じてドメイン名を登録できます。
2. ドメイン名のICP登録を行っていること。ドメイン名で国内のCDNにアクセスするには、ドメイン名のICP登録を完了していることが必要です。

操作手順

> ⚠ 注意：
> カスタムドメイン名の追加およびCDNアクセラレーションの有効化は、COSコンソールとCDNコンソールのどちらでも行うことができます。CDNコンソールからカスタムドメイン名を追加したい場合は、ドメイン名へのアクセスをご参照ください。

1. バケットの選択

COSコンソールにログインし、左側ナビゲーションバーのバケットリストをクリックし、CDNアクセラレーションを有効化したいバケットをクリックして、バケットに進みます。

2. カスタムCDNアクセラレーションドメイン名の追加

ドメイン名と伝送管理>カスタムCDNアクセラレーションドメイン名をクリックし、カスタムCDNアクセラレーションドメイン名の設定項目でドメイン名の追加をクリックし、設定可能な状態にします。

- ドメイン名：バインド対象のカスタムドメイン名（例えば `www.example.com`）を入力します。入力するドメイン名はICP登録済みであること、およびDNSサービスプロバイダが対応するCNAMEを設定していることが必要です。詳細については、CNAME設定をご参照ください。アクセスするカスタムCDNアクセラレーションドメイン名が次に該当する場合は、ドメイン名所有権の検証を行う必要があります。詳細については、ドメイン名所有権の検証のドキュメントをご確認ください。
  - このドメイン名を初めて接続する
  - このドメイン名が他のユーザーによって接続されている
  - 接続するドメイン名が汎用ドメイン名である
- アクセラレーションリージョン：中国本土、中国本土以外およびグローバルアクセラレーションをサポートしています。グローバルアクセラレーションとは、すべてのリージョン間でのバケットアクセラレーションをサポートすることを指します。

- オリジンサーバーのタイプ：デフォルトではデフォルトオリジンサーバーです。オリジンサーバーにしているバケットで静的ウェブサイトを有効にし、かつ静的ウェブサイトのアクセラレーションを行いたい場合は、オリジンサーバーのタイプを静的ウェブサイトのオリジンサーバーに設定することができます。具体的には、CDNアクセラレーションの概要をご参照ください。
- 認証：back-to-origin認証をオンにします。プライベート読み取りバケットに対しては、back-to-origin認証を有効化してオリジンサーバーを保護してください。

> ⚠ 注意：
>
> プライベート読み取りバケットに対して、back-to-origin認証とCDNサービス権限承認を同時に有効化した場合、CDNがオリジンサーバーへアクセスする際に署名を必要としないため、CDNキャッシュリソースがパブリックネットワーク配信を行い、データセキュリティに影響します。CDN認証を有効化することをお勧めします。

- back-to-origin認証の有効化
  back-to-origin認証は、CDNエッジノードのサービスIDを検証し、不正アクセスを阻止するために用いられます。具体的には次のとおりです。
  - パブリック読み取りバケット：CDNエッジノードに何の権限承認を行わなくても、バケットに直接アクセスできるため、back-to-origin認証を有効にする必要はありません。
  - プライベート読み取りバケット：CDNエッジノードはback-to-origin認証によるサービスIDの検証を受ける必要があり、検証に合格したCDNエッジノードのみがバケット内のオブジェクトにアクセスできます。back-to-origin認証の有効化を選択します。

back-to-origin認証を有効化した後、右側の保存をクリックし、約5分待つと、カスタムドメイン名の追加とCDNアクセラレーションのデプロイが完了します。

- CDN認証の有効化

> ⚠ 注意：
>
> カスタムドメイン名でCDNアクセラレーションを有効化すると、このドメイン名によって誰でもオリジンサーバーに直接アクセスできるようになります。データに一定のプライバシーが存在する場合は、必ずCDN認証設定を有効にして、オリジンサーバーデータを保護してください。

  カスタムドメイン名のデプロイが完了すると、CDN認証欄にCDN認証機能設定リンクが表示されます。設定をクリックすると、CDNコンソールに直接進んでCDN認証設定を行うことができます。具体的な操作方法については、認証設定をご参照ください。
- CDNキャッシュの自動更新：有効化すると、COSバケットのファイル更新ルールをトリガーした際にCDNキャッシュを自動更新します。COSの関数の計算で設定できます。操作ガイドはCDNキャッシュ更新の設定をご参照ください。

- HTTPS証明書：カスタムCDNアクセラレーションドメイン名に HTTPS証明書を追加する必要がある場合は、CDNコンソールで設定できます。

3. ドメイン名の解決

カスタムドメイン名でCDNにアクセスすると、システムから自動的にCNAMEドメイン名（`.cdn.dnsv1.com`を拡張子とするもの）が割り当てられます。お客様はドメイン名サービスプロバイダにCNAME設定を完了するよう依頼する必要があります。具体的にはCNAME設定をご参照ください。

> ⚠ 注意：
> CNAMEドメイン名には直接アクセスできません。

4. 機能の無効化

上記の手順が完了すると、 カスタムドメイン名によってバケット内のリソースにアクセスできるようになります。カスタムCDNアクセラレーションを無効化したい場合は、次の方法で行うことができます。
カスタムCDNアクセラレーションドメイン名管理画面で、編集をクリックするとドメイン名のステータスを変更できます。ドメイン名のステータスをオンラインからオフラインに変更し、保存をクリックしてデプロイされるまで待ちます。約5分で機能が無効化されます。CDNコンソール上では対応するカスタムアクセラレーションドメイン名のステータスが有効化済みから無効化済みに変わります。

> ⚠ 注意：
> カスタムドメイン名を削除したい場合、カスタムドメイン名のステータスがオンラインの場合は、ドメイン名を直接削除することはできません。ステータスを必ずオフラインにしてからドメイン名を削除してください。無効化または削除の操作はCDNコンソールで行うことができます。詳細については、ドメイン名の操作をご参照ください。

単一リンクの速度制限

最終更新日：：2024-06-26 11:09:29

単一リンクの速度制限

COSではファイルのアップロード、ダウンロードの際のトラフィックを管理し、他のアプリケーションのためのネットワーク帯域幅を確保することができます。PutObject、PostObject、GetObject、UploadPartリクエストの際にパラメータx-cos-traffic-limitを付加し、速度制限値を設定すると、COSは設定された速度制限値に基づいて、そのリクエストのネットワーク帯域幅を制御します。

利用説明

- ユーザーはPUT Object、POST Object、GET Object、Upload Partリクエストの際にx-cos-traffic-limit リクエストヘッダー（POST Objectリクエストについてはリクエストのフォームフィールド）を付加して、そのリクエストの速度制限値を指定します。このパラメータはheader、リクエストパラメータ内、またはフォームアップロードインターフェースを使用する場合はフォームドメイン内に設定することができます。
- x-cos-traffic-limitパラメータ値は数字でなければならず、単位はデフォルトではbit/sです。
- 速度制限値の範囲は819200～838860800、すなわち100KB/s～100MB/sです。この範囲を超過するとエラーコード400が返されます。

> 説明：
> 単位換算公式：1MByte=1024KByte=1048576Byte=8388608bitです。

APIの使用例

シンプルアップロードのAPIの例を次に示します。速度制限値は1048576 bit/s、すなわち128KB/sです。

```
PUT /exampleobject HTTP/1.1
Host: examplebucket-1250000000.cos.ap-beijing.myqcloud.com
Content-Length: 13
Authorization: q-sign-algorithm=sha1&q-
ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-
time=1561109068;1561116268&q-key-time=1561109068;1561116268&q-header-
list=content-length;content-md5;content-type;date;host&q-url-param-
list=&q-signature=998bfc8836fc205d09e455c14e3d7e623bd2****
x-cos-traffic-limit: 1048576
```