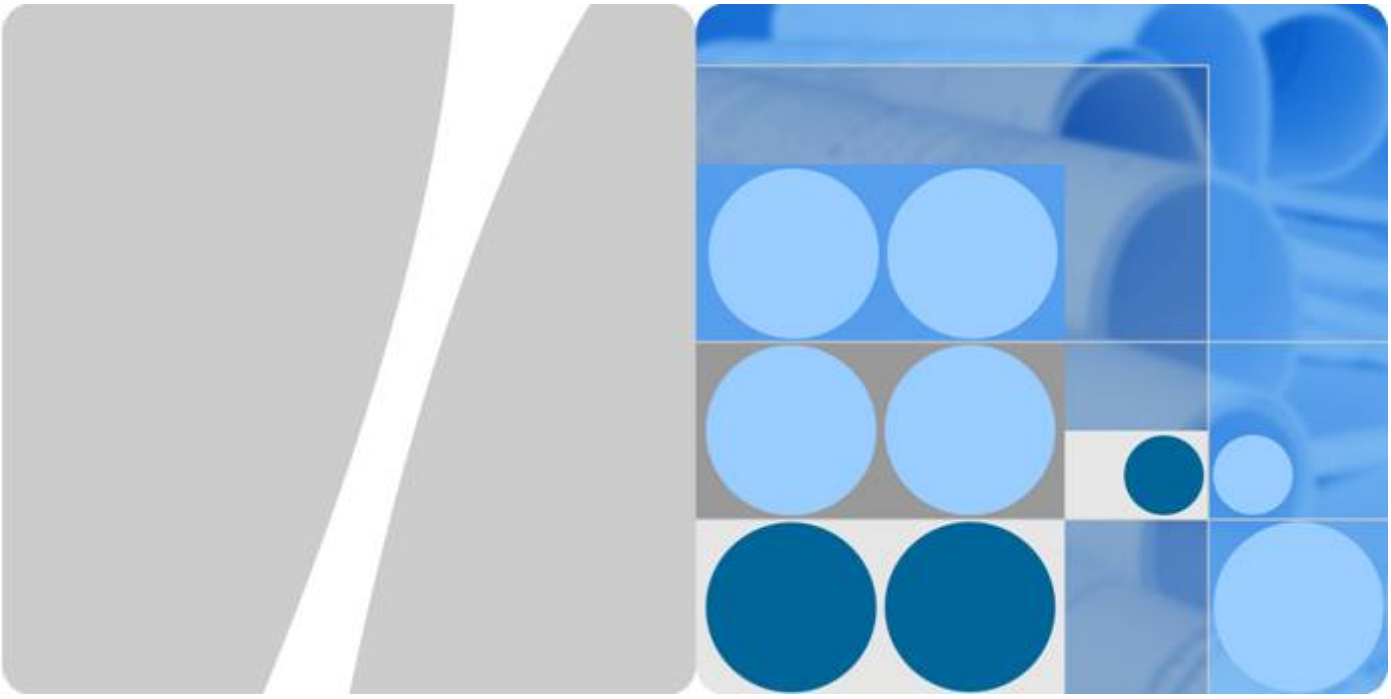


Object Storage Service

Image Processing Feature Guide

Date **2024-01-29**

Contents

1 Introduction

1.1 What Is Image Processing?

Introduction

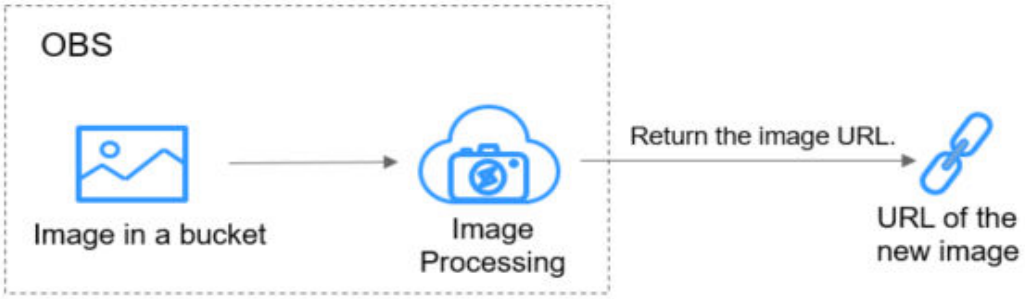
Image processing is a feature integrated in Object Storage Service (OBS). It provides stable, secure, efficient, and inexpensive image processing services. By using this feature, you can slim, crop, resize, and watermark images, as well as convert the formats of images.

You can access this feature via OBS Console and REST APIs, to process images stored in OBS anytime and anywhere and obtain the processed images right away.

Architecture

You can upload your images to OBS using OBS Console, OBS clients, REST APIs, or third-party clients. Before downloading and using an image, you can create an image style or input image processing parameters to process it, such as cropping and compressing. You can obtain the new URL of that image after processing it with styles or parameters. **Figure 1-1** illustrates the architecture of the image processing function.

Figure 1-1 Image processing architecture



1.2 Application Scenarios

Image processing enables you to resize, crop, or compress images on cloud. You do not have to download space-consuming software to your local computers.

For example, you can add effects to or resize images in your cloud album anytime and anywhere, and quickly share the images with your friends online.

1.3 Constraints

Operations

- All image processing operations will not change original images.
- Cold storage does not support image processing.
- OBS buckets that use SSE-KMS do not support image processing.
- Currently, only buckets of version 3.0 support image processing. To check a bucket's version, go to the **Basic Information** area on the bucket's **Overview** page.
- If an image is set to be accessible by anonymous users, the image can be accessed directly through a web browser and signature is not required in image processing requests. Example:

  https://bucketname.obs.region.example.com/example.jpg?x-image-process=style/stylename

Images

- Supported original formats: JPG, JPEG, PNG, BMP, WebP, GIF, and TIFF.
- Supported target formats: JPG, PNG, BMP, and WebP.
- The supported maximum size of an image is 25 MB, with maximum width of 4096 px and height of 5000 px after processing.
- An animated image (like a GIF or WebP image) will be returned without processing if it is greater than 2 MB in size or has over 50 frames.
- Currently, processing images in CMYK may change their color.

Commands

Commands are not allowed to have more than 512 characters, and the maximum number of commands is 20.

1.4 Common Concepts

Style

A style is an aggregation of parameters or image processing operations. When performing the same operations on multiple images, you can create an image style as a template to avoid repetitive operations. Each bucket supports a maximum of 100 styles.

Exif Information

Exchangeable Image File (Exif) information exists in images shot by camera or cellphone. Exif information is embedded into images in JPEG or TIFF format. It includes the shooting parameters, such as the camera type, shooting time, and shooting mode, as well as the thumbnail, and other property information of the image.

1.5 Methods to Access Image Processing

You can access image processing in the following ways:

- Log in to OBS Console to preview image effects in different style templates.

  On the image processing page of OBS Console, you can create a style template by configuring parameters on the GUI or by coding. You can view the effect of the style template in the preview area. After creating an image style template, you can copy the link to obtain the new image URL.

  For details about how to quickly get started on OBS Console, see **Procedure**.

- Use applications to call REST APIs to access the image processing.

  OBS provides REST APIs. In the REST architectural architecture, resources on a network are identified by Uniform Resource Identifiers (URIs). Applications on clients locate resources using Uniform Resource Locators (URLs). The URL is in the **https://***Endpoint***/***uri* format. You can obtain the processed image simply by putting a URL that complies with the command rules of image processing in a browser. For more API access information, see the *Object Storage Service API Reference*.

  For details about how to quickly get started through the API, see **Procedure**.

1.6 Image Processing Function Overview

Table 1-1 lists the functions provided by the image processing feature in OBS.

Table 1-1 Image processing functions

Function	Description	Use
Obtaining Image Information	Obtains the basic information of an image, including: format, size, and average color value.	Calling APIs
Brightness	Enhances image effects, including brightness, contrast ratio, sharpness, and blur.	Console GUI Coding Calling APIs
Resizing Images	Resizes images based on specified width and height.	Coding Calling APIs

Function	Description	Use
Public Parameters	Adds watermarks to images at specified positions. Watermarks can be texts, pictures, and their combinations. The color, font, and size of the text on watermarks are adjustable, and you can also resize, rotate, and crop watermarks.	Console GUI Coding Calling APIs
Converting Formats	Converts images to various formats, and supports interlaced rendered images after conversion.	Console GUI Coding Calling APIs
Rotation Settings	Rotates images clockwise, and supports automatic rotation according to the rotation configuration of cameras and mobile phones.	Coding Calling APIs
Common Cropping	Crops images according to the specified width, height, circle radius, index mode, and rounded rectangle.	Coding Calling APIs
Changing Quality	Compresses JPG images based on the relative quality and absolute quality. After compression, the image quality is reduced but occupies less space. In scenarios that do not require high image quality, this function helps you save traffic and have faster image loading.	Coding Calling APIs
Slimming Images	Reduces the image size without compromising its quality. This function helps to slim images while maintaining the original quality, accelerating loading and saving traffic.	Coding Calling APIs
Image Persistency	Persistency indicates that images are asynchronously stored in the specified OBS bucket, so that you can access the processed images directly, improving user experience.	Coding Calling APIs
Command Access Method	Orchestrates multiple process commands in sequence. With this function, you can add multiple process commands to the URL of the image that you want to process, and separate each command using the designated delimiter. Then the commands are executed one by one from left to right.	Coding Calling APIs

Function	Description	Use
Creating Image Styles	Customizes image styles. Each image style specifies a set of process operations. For images require the same process operations, you can create an image style to batch process them.	Console GUI Coding

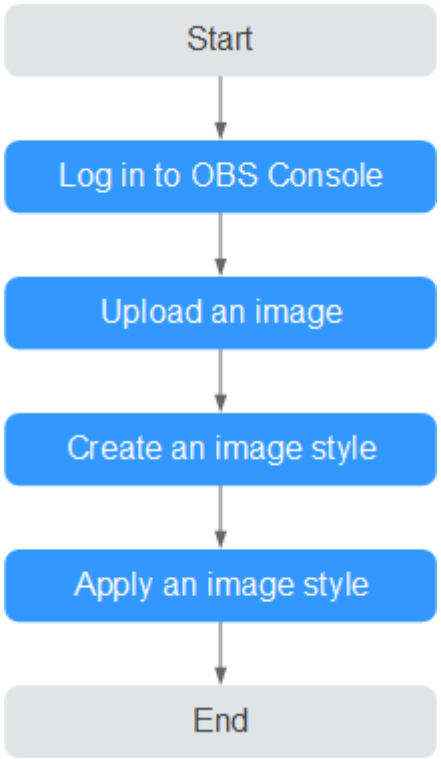
2 Start to Process (Using OBS Console)

2.1 Procedure

On the image processing page of OBS Console, you can create a style template by configuring parameters on the GUI or by coding. You can view the effect of the style template in the preview area. After creating an image style template, you can copy the link to obtain the new image URL.

Figure 2-1 shows the procedure of accessing and using Image Processing on OBS Console.

Figure 2-1 Flowchart of using OBS Console



2.2 Uploading Images

You can upload images using OBS Console, OBS Browser+, and REST APIs.

This section describes how to upload images on OBS Console. Skip this section if the image to be processed has existed in the bucket.

NOTE

For details about the restrictions on the format and size of an uploaded image, see **Constraints**.

Prerequisites

At least one bucket has been created.

Procedure

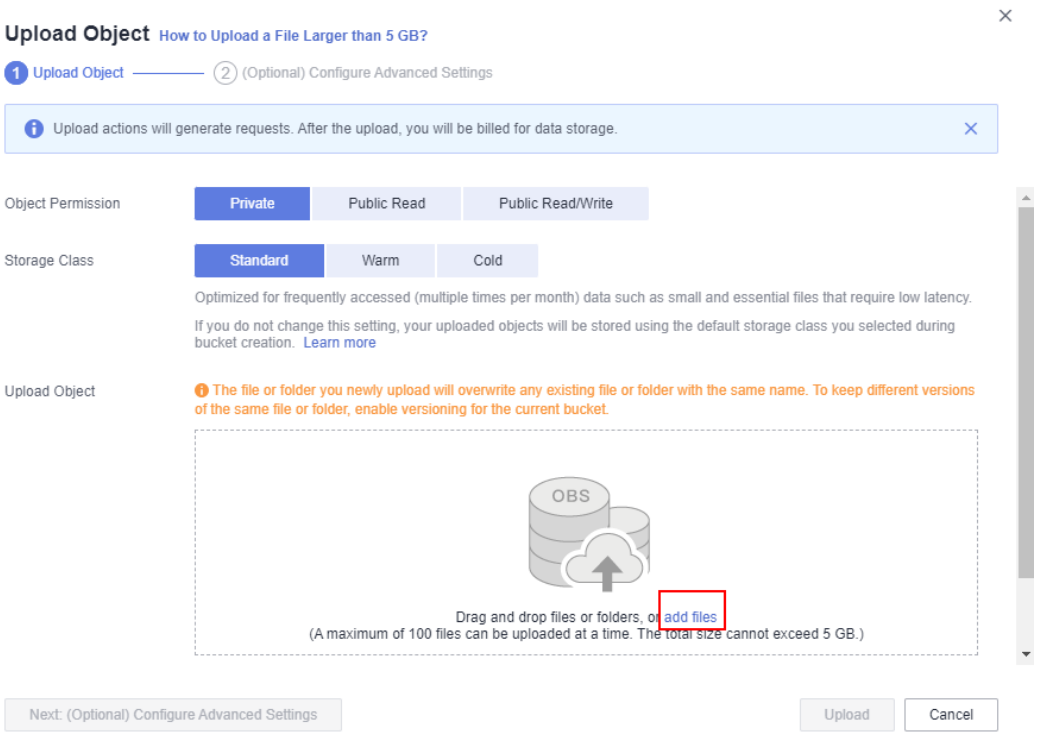
Step 1 On the console homepage, click **Service List** in the upper left corner and choose **Storage** > **Object Storage Service**.

Step 2 In the bucket list, click the bucket you want to go to the **Objects** page.

Step 3 Click **Upload Object**. The **Upload Object** page is displayed.

Step 4 Click **add file** marked by red box in **Figure 2-2** to open the local file browser.

Figure 2-2 Uploading an image



Step 5 Select the image that you want to upload and click **Open**.

Step 6 Click **Upload**.

----End

2.3 Creating Image Styles

Context

By creating image styles, you can process the image such as cropping, compressing, and watermarking. Each image style specifies a set of process operations. For images require the same process operations, you can create an image style to batch process them. Once a style is successfully created, it can be used by multiple images in the bucket.

When creating styles, you can view the style effects of the sample image on the right.

When using REST APIs to access image processing, you can call the style name in the URL to avoid entering complex commands. For details about the domain name rules for API access, see **Style Access Method**.

You can create a maximum of 100 styles for one bucket at one time.

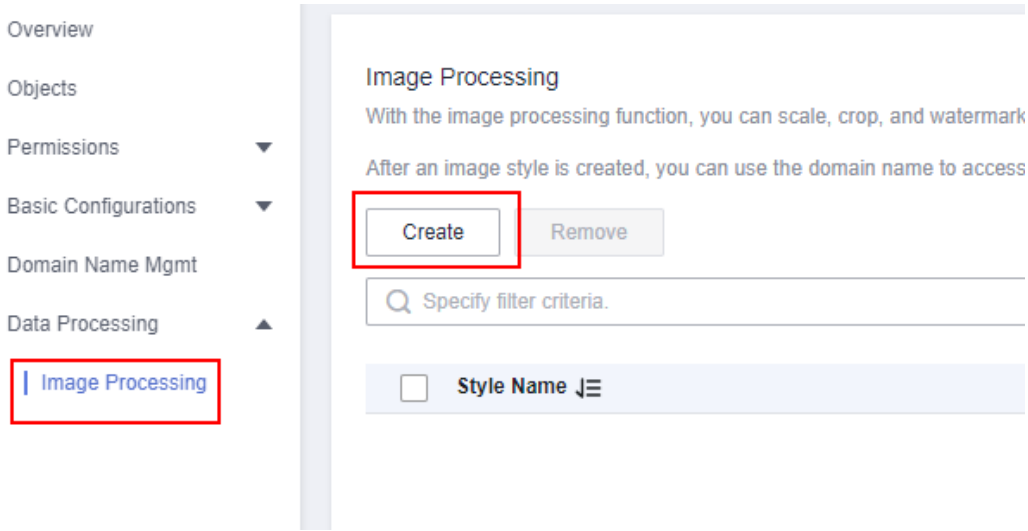
You can create image processing styles or replicate existing image processing styles from another bucket.

Creating an Image Processing Style

Step 1 On the console homepage, click **Service List** in the upper left corner and choose **Storage** > **Object Storage Service**.

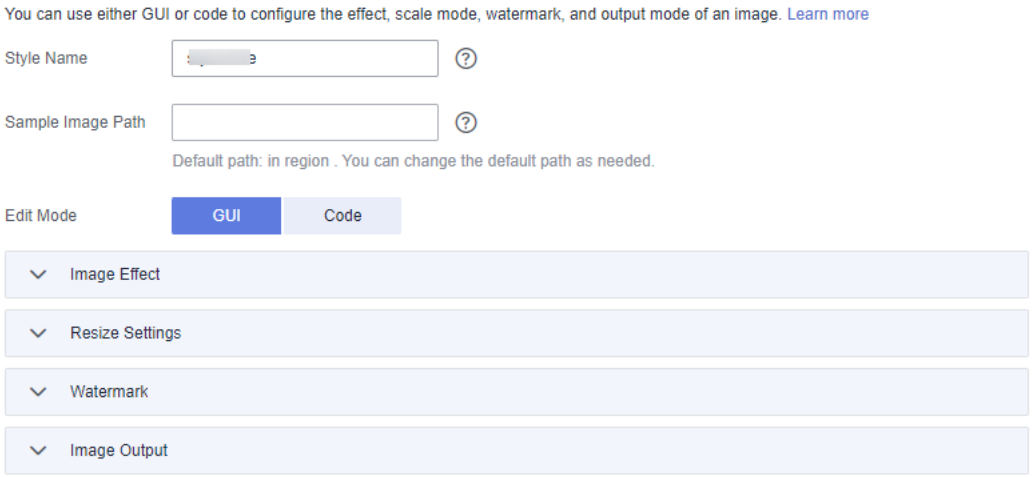
Step 2 Click a desired bucket name. In the navigation pane, choose **Data Processing** > **Image Processing**.

Figure 2-3 Image processing



Step 3 Click **Create**. The style editing page is displayed. For details, see **Figure 2-4**.

Figure 2-4 Creating a style



Step 4 On the editing page, you can edit the style name and basic properties. You can also set the resize mode, as well as perform operations like rotation/cropping, watermarking, and image output.

- Style Name

  Input an easy-to-remember style name. Only letters (uppercase and lowercase), digits, periods (.), underlines (_), and hyphens (-) are allowed. The style name contains 1 to 256 characters, for example, **rotate_0001**.

- Edit Mode

  You can either choose GUI mode for visible editing, or choose Code mode.

  An example code is as follows:

  ```
  image/sharpen,100/blur,r_1,s_1/resize,m_lfit,h_400,w_400,limit_1
  ```

- Parameter settings

  You can set image effects, resizing, watermarks, and output parameter values.

Step 5 After finishing editing the image style, click **OK** to save the style. The new style will be displayed in the style list.

----End

2.4 Applying Image Styles

When a created image style exists in a bucket, use either of the following methods to apply the image style:

- **Copying Link**: On OBS console, obtain the image URL when previewing the image on the details page. Enter the URL in the address bar of a browser, and you can obtain the processed image. **Copying Links** shows the procedure in detail.
- **Concatenating domain names**: Concatenate a domain name by referring to the following rule, enter it in the address bar of a browser, and obtain the processed image.

```
<Image URL>?x-image-process=style/<Style name>
```

The image URL can be obtained from the object details page. For details, see **Editing Domain Names**. The style name is the one defined when the style was created. A style name can contain only uppercase or lowercase letters, digits, periods (.), underscores (_), and hyphens (-) and is 1 to 256 characters long, for example, **rotate_0001**.

Copying Links

Step 1 On the console homepage, click **Service List** in the upper left corner and choose **Storage** > **Object Storage Service**.

Step 2 Click the name of the bucket that houses the desired style. The **Objects** page is displayed.

Step 3 Click the name of an existing image or of a newly uploaded image. The details page is displayed.

Step 4 Click the **Preview Image** tab to preview the effect of the current style.

Step 5 Click **Copy Link**. After prompted with **Copied successfully**, you can obtain the address of the image that uses styles and then access the image in the browser.

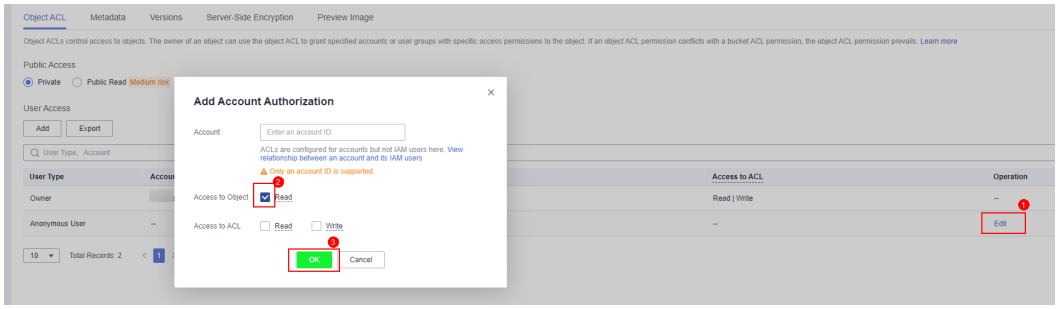
----End

Editing Domain Names

Step 1 On the OBS object list page, click an image to be processed. The image details page is displayed.

Step 2 Choose **Object ACL** > **User Access** > **Anonymous User** and click **Edit**. In the displayed dialog box, grant the object read permission to anonymous users and click **OK**.

Figure 2-5 Granting the object read permission to anonymous users



Step 3 Click the icon next to the link to copy the image URL.

Step 4 Add **?x-image-process=style/<*Style name*>** behind the copied URL. Enter it in the address bar of a browser, and then you can access the processed image.

Example:

https://bucketname.obs.region.example.com/example.jpg?x-image-process=style/stylename

NOTE

The preceding image links and styles are examples for reference only. Change accordingly in practice.

----End

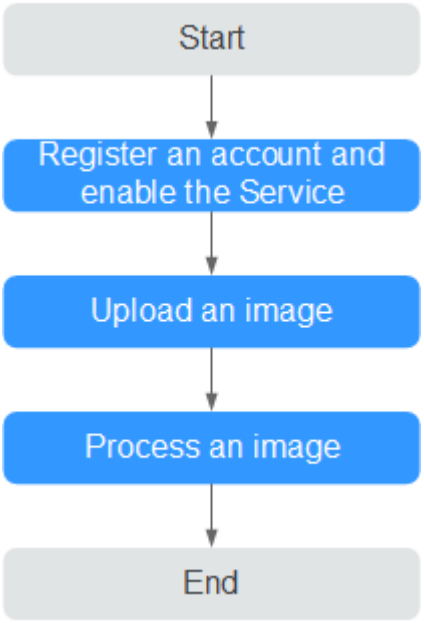
3 Start to Process (Using APIs)

3.1 Procedure

OBS provides REST APIs. Under the RESTful architecture, a resource on a network is identified by a Uniform Resource Identifier (URI). Applications on clients can locate resources using Uniform Resource Locators (URLs). The URL is in the following format: **https://***Endpoint*/*uri*. You can obtain the processed image simply by inputting a URL that complies with the command rules of Image Processing in the address box of the browser.

Figure 3-1 shows the procedure of accessing and using Image Processing with the REST API.

Figure 3-1 Flowchart of using the REST API



3.2 Uploading Images

You can upload images using OBS Console, OBS Browser+, and REST APIs.

Skip this section if the image to be processed has existed in the bucket.

NOTE

If you want to use a user-defined domain name for image processing, you need to grant the read permission of the image to anonymous users. For details about how to configure the read permission, see **Configuring an Object ACL**.

3.3 Processing Images

This section describes the URL constitution to process images using REST APIs. Once you have enabled OBS successfully, you can call REST APIs to process images simply by putting a URL that complies with the command rules of image processing in a browser.

URL Constitution

A URL consists of the OBS domain name, bucket name, the original image name, and processing command or style name.

Command Access Method

URL format: https://*bucketName.endpoint*/*objectName*?x-image-process=image/*commands*

- *endpoint* is the endpoint address of the region where the bucket resides. You can obtain the endpoint address from the bucket's basic information or from the **Regions and Endpoints** page.
- *bucketName* is the name of the bucket that accommodates the image to be processed on OBS.
- *objectName* is the name of the original image stored in the *bucketName* bucket on OBS. The suffix of the image name must be supported by image processing.
- *commands* are the processing commands. Three types of delimiters are used between commands or command parameters. See **Delimiters**. If no commands are entered, the original image will be returned.

Example: https://image-demo.obs.region.example.com/example.jpg?x-image-process=image/crop,x_100,y_50

- **Delimiters**

Delimiters are separation identifiers used in URLs to distinguish one field from another in the command. For details, see **Table 3-1**.

Table 3-1 Delimiters

Name	Character	Sequence	Description
Parameter delimiter	_	Fixed	Delimiter between the command parameter and its value.
Command delimiter	,	Irrelevant	Delimiter between multiple command parameters.
Pipe delimiter	/	Relevant	Delimiter between two processing commands. See **Pipes**.

- **Pipes**

If an image is to be processed by multiple operations, such as cropping and resizing, the operation commands need to be connected to each other by the pipe delimiter "/". The processing operations are executed from left to right according to the designated sequence of pipes.

For example, https://image-demo.obs.region.example.com/example.jpg?x-image-process=image/resize,w_100,h_100/quality,q_80 has two pipes. The pipes will be executed from left to right in sequence and the command output of the previous pipe will be used as the input of the next pipe.

Style Access Method

URL format: https://*bucketName.endpoint*/*objectName*?x-image-process=style/*stylename*.

- *endpoint* is the endpoint address of the region where the bucket resides. You can obtain the endpoint address from the bucket's basic information or from the **Regions and Endpoints** page.
- *bucketName* is the bucket name on OBS.
- *objectName* is the name of the original image stored in the *bucketName* bucket on OBS. The suffix of the image name must be supported by image processing.
- *stylename* is the style name that has been created in the *bucketName* bucket on OBS Console. Currently, you cannot perform other operations related to styles by calling REST APIs, such as creating, changing, and deleting styles.

Example: https://image-demo.obs.region.example.com/example.jpg?x-image-process=style/stylename

4 Typical Cases

4.1 Graphical User Interface (GUI) Mode

This section introduces an example of image processing in GUI mode. In this example, a style of a FZShuSong text watermark is created on the top left of the original image.

Procedure

Step 1 On the console homepage, click **Service List** in the upper left corner and choose **Storage** > **Object Storage Service**.

Step 2 In the bucket list, click the bucket you want to go to the **Objects** page.

Step 3 Click **Upload Object**. The **Upload Object** page is displayed.

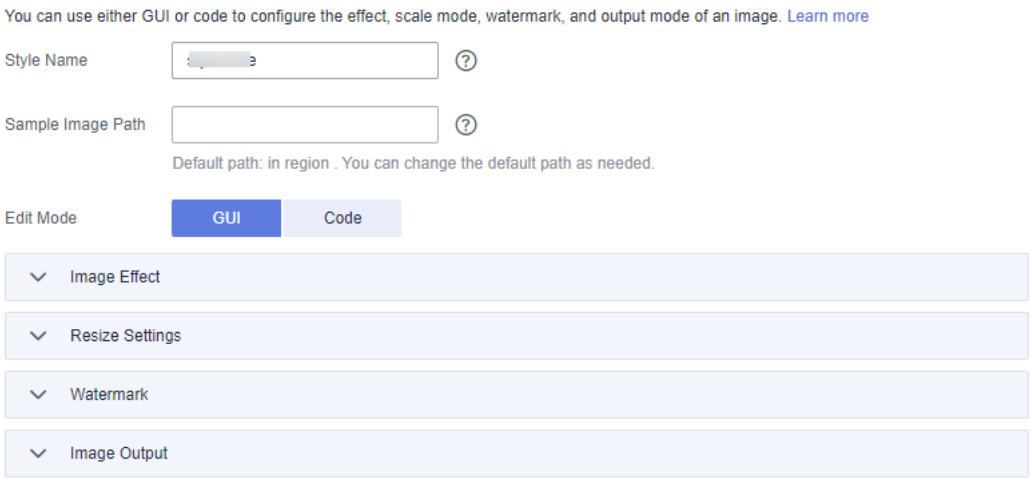
Step 4 Select the image that you want to upload and click **Open**.

Step 5 Click **Upload** to upload the image. The uploaded image is displayed in the object list.

Step 6 In the navigation pane, choose **Data Processing** > **Image Processing**.

Step 7 Click **Create**. The style editing page is displayed. For details, see **Figure 4-1**.

Figure 4-1 Creating a style



Step 8 Input the style name **obs-111**. On the editing page, select **GUI**.

Step 9 Click **Watermark**.

1. In the drop-down list of watermark types, select **Text watermark**.
2. In the input box, input **Hello**.
3. Select **DroidSansFallback**.
4. Input **600** for text size.
5. Keep the default value **100** for the watermark transparency.
6. Select the top left sign for the watermark location  .
7. Keep the disabled state for text shadow  .
8. Keep the default value **10** for both the vertical margin and horizontal margin.

The style effect will be displayed on the right in real time. **Figure 4-2** shows the final style effects.

Figure 4-2 Style obs-111

Step 10 After finishing editing the image style, click **OK** to save the style. The new style **obs-111** will be displayed in the style list.

Step 11 In the navigation pane, choose **Objects**. Click **mountain.jpg** in the object list to go to the file details page.

Step 12 Click the **Preview Image** tab to preview the effect of the current style.

Step 13 Click **Copy Link**. After prompted with **Copied successfully**, you can obtain the access address of the image file.

----End

4.2 Code Mode

This section shows an example of how to create a resizing style in code mode on OBS Console.

Procedure

Step 1 On the console homepage, click **Service List** in the upper left corner and choose **Storage** > **Object Storage Service**.

Step 2 In the bucket list, click the bucket you want to go to the **Objects** page.

Step 3 Click **Upload Object**. The **Upload Object** page is displayed.

Step 4 Select the image that you want to upload and click **Open**.

Step 5 Click **Upload** to upload the image. The uploaded image is displayed in the object list.

Step 6 In the navigation pane, choose **Data Processing** > **Image Processing**.

Step 7 Click **Create**. The style editing page is displayed.

Step 8 Input the style name **style002**. On the editing page, select **Code**.

Step 9 In the code input box, input the following command and parameters for resizing.

Specify a rectangle, whose w and h equal 100. Lock the aspect ratio, and obtain the smallest image in the extended area of the 100 x 100 rectangle.

```
image/resize,m_mfit,h_100,w_100
```

The style effect will be displayed on the right in real time. **Figure 4-3** shows the final style effects.

Figure 4-3 Style style002

Step 10 After finishing editing the image style, click **OK** to save the style. The new style **style002** will be displayed in the style list.

Step 11 In the navigation pane, choose **Objects**. Click **mountain.jpg** in the object list to go to the file details page.

Step 12 Click the **Preview Image** tab to preview the effect of the current style.

Step 13 Click **Copy Link**. After prompted with **Copied successfully**, you can obtain the access address of the image file.

----End

5 Obtaining Image Information

You can obtain the information of an image by making an API call only.

Image information includes some basic information about the image: width, height, and the file size and format of the image. If there is **Exif Information**, the Exif information will be returned in JSON format.

Operation name: info

Example

Query **example.jpg** information.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/info

```
{
    "format":"JPEG",
    "height":2000,
    "size":1046583,
    "width":2668
}
```

6 Obtaining Average RGB Value of an Image

You can obtain the average RGB value of an image by making an API call only.

This operation enables you to obtain the average RGB value of an image, which is returned as a hexadecimal value in JSON format.

Operation name: average-hue

Example

You can access the following address through a web browser and obtain the average RGB value of the **example.jpg** image:

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/average-hue

If information similar to the following is displayed, the average RGB value of the image is obtained successfully:

```
{"RGB":"#28577D"}
```

The original image of **example.jpg** is as follows:

The obtained average RGB value (#28577D) is:

7 Setting Image Effects

7.1 Brightness

You can use the GUI, code, or APIs to configure the brightness of an image.

Table 7-1 lists the parameters in detail.

Operation name: bright

Table 7-1 Brightness description

Parameter	Value Description	Code Example
value	Brightness of images, ranging [-100, 100]. The original brightness value is **0**. The image becomes brighter as the value increases from -100 to 100.	image/bright,50

Example

- Set the brightness to **50**.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/bright,50

- Set the brightness to **-50**.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/bright,-50

7.2 Contrast

You can use the GUI, code, or APIs to configure the contrast of an image.

Table 7-2 lists the parameters in detail.

Operation name: contrast

Table 7-2 Contrast description

Parameter	Value Description	Code Example
value	Contrast of images, ranging [-100, 100]. The original contrast value is **0**. The contrast becomes stronger as the value increases from -100 to 100.	image/contrast,-50

Example

- Set the contrast to **-50**.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/contrast,-50

7.3 Sharpening

You can use the GUI, code, or APIs to configure the sharpness of an image.

Table 7-3 lists the parameters in detail.

Operation name: sharpen

Table 7-3 Sharpen description

Parameter	Value Description	Code Example
value	The extent of sharpening, ranging [50 to 399] **50** leads to the weakest sharpening effect. The recommended value is **100** for optimized effect. The image becomes clearer as the value increases. However, the image may look unreal if you set the value too high.	image/sharpen,100

Example

- Set the sharpening value to **100**.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/sharpen,100

7.4 Blur

You can use the GUI, code, or APIs to configure the blur of an image.

Table 7-4 lists the parameters in detail.

Operation name: blur

Table 7-4 Blur description

Parameter	Value Description	Code Example
r	Radius of blur, ranging [1, 50]. A larger radius leads to a larger blurred area.	image/blur,r_3,s_2
s	Standard deviation of normal distribution, ranging [1, 50]. The image becomes more blurred as the value increases.	

NOTE

In GUI mode, the parameter **r** and **s** increase or decrease simultaneously.

Example

- Set **r** to **3** and **s** to **2**.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/blur,r_3,s_2

7.5 Grayscale

You can edit code on OBS Console or make an API call to convert an image to grayscale.

Table 7-5 lists the parameters in detail.

Operation name: colorspace

Table 7-5 Grayscale description

Parameter	Value Description	Code Example
value	Image color mode. You can set the parameter to **gray** to convert an image to grayscale.	image/colorspace,gray

Example

- Convert an image to grayscale.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/colorspace,gray

8 Resizing Images

You can use the GUI, code, or APIs to resize images. Images can be resized based on a specific rule or based on a fixed width, height, or percentage.

NOTE

- A long side refers to the side with a larger ratio of its original size to its target size, and a short side refers to the side with a smaller ratio. Assume that the original size of an image is 400 × 200 pixels and it is resized to 100 × 100 pixels. The ratio of 400 pixels to 100 pixels is 4, that of 200 pixels to 100 pixels is 2, and 4 is larger than 2, so the long side is 400 pixels and the short side is 200 pixels.
- For a target image after resizing, its long side cannot exceed 9,999 pixels, and the product of its width and height cannot exceed 24,999,999 pixels.
- If you only specify the height or width for resizing, the target image keeps the same aspect ratio as the original image and is returned in the original image's format.
- By default, the resize operation is not allowed to scale up an image. If you want an image to become larger after resizing, you need to set **limit** to **0** to obtain the enlarged image, or the original image will be returned. To do this, use the following format:

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,w_500,limit_0

Table 8-1 describes the parameters in the resize operation.

Table 8-1 Resize settings

Parameter	Value Description	Code Example
m	Type of resizing. The value can be **lfit** (the default value), **mfit**, **fill**, **pad**, or **fixed**. • **lfit**: Specify a rectangle with a given width (indicated by **w**) and height (indicated by **h**), lock the aspect ratio, and obtain the largest image in the rectangle. • **mfit**: Specify a rectangle with a given width (indicated by **w**) and height (indicated by **h**), lock the aspect ratio, and obtain the smallest image in the rectangle's extended area. • **fill**: Specify a rectangle with a given width (indicated by **w**) and height (indicated by **h**) and lock the aspect ratio. Obtain the smallest image in the rectangle's extended area, and center and crop the image. **fill**-based resizing actually centers and crops a target image resized with **mfit**. • **pad**: Specify a rectangle with a given width (indicated by **w**) and height (indicated by **h**) and lock the aspect ratio. Obtain the largest image in the rectangle and fill the blank area with color. **pad**-based resizing actually fills the blank area of a target image resized with **lfit**. • **fixed**: Resize an image based on a fixed width and height. • **ratio**: Specify an aspect ratio (a ratio of **w** to **h**), in the range of **1** to **1000**, and	image/ resize,m_lfit,h_100,w_100

Parameter	Value Description	Code Example
	obtain the largest image that meets the specified ratio.	
p	Percentage of the aspect ratio, in the range of **1** to **1000**. If the value is: ● < 100: The image is scaled down. ● = 100: The image is kept unchanged in size. ● > 100: The image is scaled up.	image/resize,p_50
h	Height of the target image, in the range of **1** to **9999**. The product of the target image's width and height cannot exceed 24,999,999.	image/resize,m_lfit,h_100
w	Width of the target image, in the range of **1** to **9999**. The product of the target image's width and height cannot exceed 24,999,999.	image/resize,m_fixed,h_100,w_100
l	The long side of the target image, in the range of **1** to **4096**. The product of the target image's width and height cannot exceed 24,999,999. The long side has a specified value, and the short side is scaled based on the ratio.	image/resize,l_100
s	The short side of the target image, in the range of **1** to **4096**. The product of the target image's width and height cannot exceed 24,999,999. The short side has a specified value, and the long side is scaled based on the ratio.	image/resize,s_100

Parameter	Value Description	Code Example
color	Color for filling the blank area after resizing. **color** can be used when you set **m** to **pad**. The value is a hexadecimal code, from **000000** to **FFFFFF** (which represents white and is the default value).	image/resize,m_pad,h_100,w_100,color_FF0000
limit	Whether to limit the size of the target image when the target image is larger than the original one. The value can be **0** or **1** (default value). • **0**: The size is not limited. • **1**: The size is limited.	image/resize,p_150,limit_0

CAUTION

If a resized image is aliased, you can add **/marker,u_plus** to the end of the image processing URL for optimization.

For example, by adding **/marker,u_plus**, the processing URL **https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_fixed,w_2668,h_1999,limit_0** becomes **https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_fixed,w_2668,h_1999,limit_0/marker,u_plus**. The latter displays an image with better quality.

Example

- Set **h** to **100** and **m** to **lfit** (the default value) to process the width proportionally.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_lfit,h_100

- Lock the aspect ratio and specify the short side to resize the image into 100 x 100 pixels.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_lfit,h_100,w_100

- Set the long side size to **100** and scale the short side based on the ratio.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,l_100

- Fix the width and height, center and crop the image, resize the image into 100 x 100 pixels.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_fill,h_100,w_100

- Fix both the width and height to **100**.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_fixed,h_100,w_100

- Fix the width and height. Resize the image into 100 x 100 pixels by specifying the short side and fill the blank area with white.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_pad,h_100,w_100

- Fix the width and height. Resize the image into 100 x 100 pixels by specifying the short side and fill the blank area with red.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,m_pad,h_100,w_100,color_FF0000

- Scale up the image to 150% of its original size and set **limit** to **0** to obtain the enlarged image.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,p_150,limit_0

- Set **p** to **30** to scale down the image to 30% of its original size.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,p_30

- Set the image's aspect ratio to 3:2.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,p_30

9 Rotating Images

9.1 Rotation Settings

You can edit code on OBS Console or make an API call to rotate images.

This operation rotates images clockwise. For details, see **Table 9-1**.

Operation name: rotate

Table 9-1 Rotation

Parameter	Value Description	Code Example
value	Rotation angle, clockwise [0, 360]. The default value is **0**, which means the image is not rotated. A larger value means the image is clockwise rotated by a larger angle.	image/rotate,90

NOTE

- After rotation, dimensions of an image may increase.

Example

Set the width to **100** and the rotation angle to **90**.

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,w_100/rotate,90

Set the width to **100** and the rotation angle to **220**.

https://obs.example.region.com/image-demo/example.jpg?x-image-process=image/resize,w_100/rotate,220

9.2 Adaptive Orientation

You can use the GUI, code, or APIs to use adaptive orientation.

Images shot by camera or cellphone may contain **Exif Information**, such as orientation parameters like **Orientation**. When shooting, the rotation information is recorded in the orientation parameters. The browser can auto-orient the image to the right position.

When you set the adaptive orientation, images that contain orientation parameters will be oriented automatically according to these parameters. For details, see **Table 9-2**.

Operation name: auto-orient

Table 9-2 Adaptive orientation description

Parameter	Value Description	Code Example
value	The value can be **0** or **1**. It is set to **1** by default. **0**: Not set the adaptive orientation. The image will not rotate automatically and will keep the default orientation. **1**: Set the adaptive orientation. The image will rotate automatically before resizing.	image/resize,w_100/auto-orient,0

NOTE

- The auto-orient operation is set only when both the height and width are shorter than 4096.
- If the Exif information does not contain rotation parameters, or if there is even no Exif information, then the auto-orient parameter is invalid and will cause no effect to the image.

Example

- Set the width of image to **100**, and do not set the adaptive orientation.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,w_100/auto-orient,0

9.3 Flipping

You can edit code on OBS Console or make an API call to flip images.

This operation flips images horizontally or vertically. **Table 9-3** lists the parameters in detail.

Operation name: flip

Table 9-3 Flip description

Parameter	Value Description	Code Example
value	If this parameter is set to **vertical**, an image is flipped vertically. If this parameter is set to **horizontal**, an image is flipped horizontally.	image/flip,vertical

NOTE

- After flipping, dimensions of an image may increase.

Example

Flip an image horizontally.

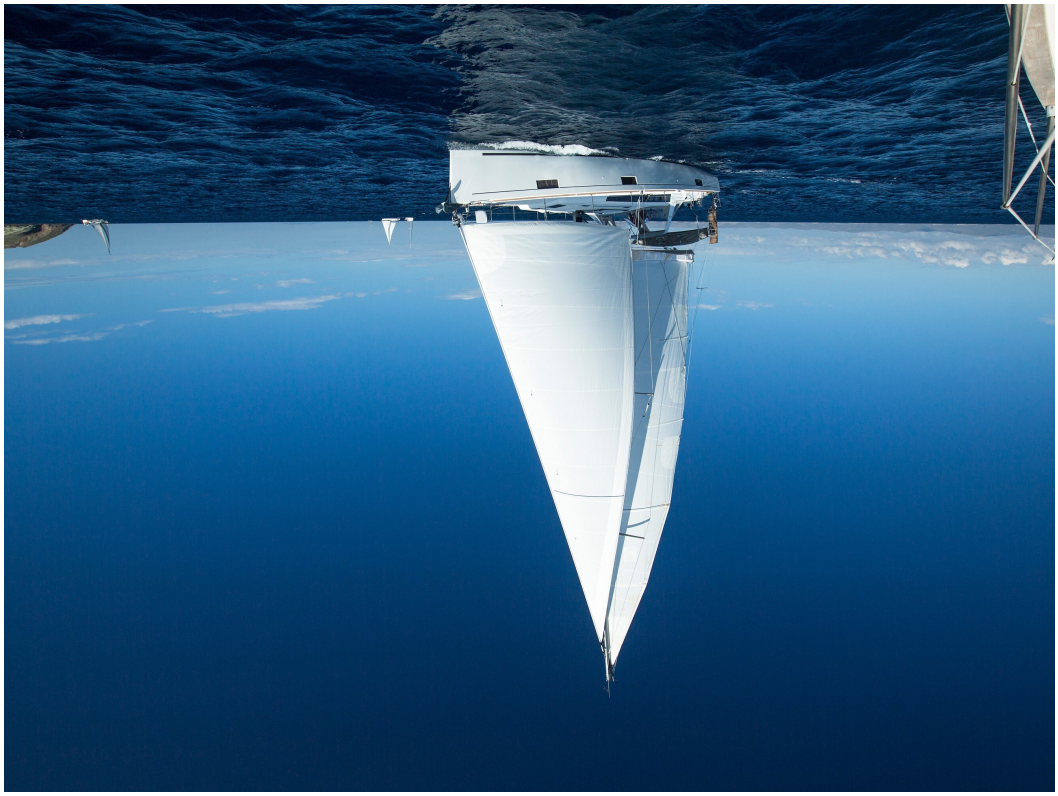
https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/flip,horizontal

Figure 9-1 Flipping an image horizontally

Flip an image vertically.

https://obs.example.region.com/image-demo/example.jpg?x-image-process=image/flip,vertical

Figure 9-2 Flipping an image vertically



10 Cropping Images

10.1 Common Cropping

You can edit code on OBS Console or make an API call to crop images.

You can start at any point on an image and crop the image into a rectangle with specified width and height. For details, see **Table 10-1**.

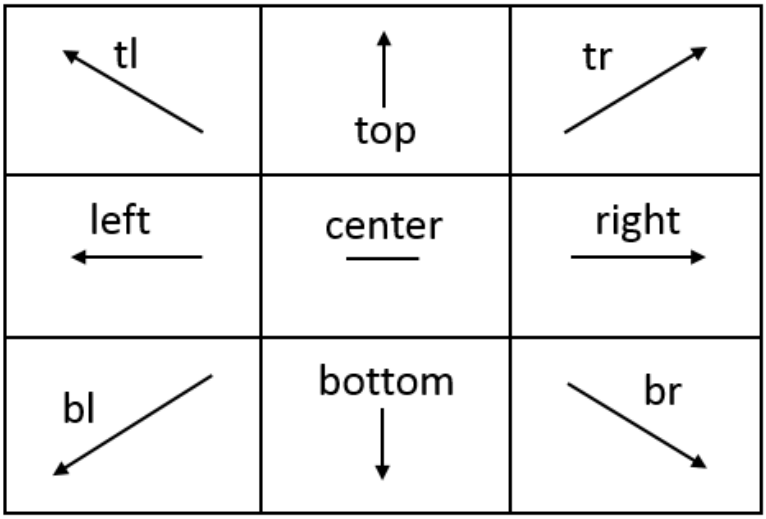
Parameter: crop

Table 10-1 Common cropping

Parameter	Value Description	Code Example
g	The location where cropping starts. **g** can be set to **tl**, **top**, **tr**, **left**, **center**, **right**, **bl**, **bottom**, and **br**. **Figure 10-1** is the 3 x 3 grid illustrating these values. Each value locates at the top left corner of a grid.	image/crop,x_10,y_10,w_200,h_200,g_br
h	Height of the cropped image, ranging [0, original height].	
w	Width of the cropped image, ranging [0, original width].	
x	x-coordinate of the start point. The top left corner is the default origin. x ranges [0, original width of the image].	
y	y-coordinate of the start point. The top left corner is the default origin. y ranges [0, original height of the image].	

The origins of cropping are shown as **Figure 10-1**.

Figure 10-1 3 x 3 grid of cropping origins



NOTE

- If x is larger than the origin width, or y is larger than the origin height, the cropping cannot be executed and a fault will be returned.
- If h is larger than the origin height and w is larger than the original width, the image will be cropped to the boundaries.

Example

- Set the start point of cropping to **(1000, 500)**, and set the width and height of cropping to **1000** both.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/crop,x_1000,y_500,w_1000,h_1000

- The cropping starts from **(10, 10)** in the bottom right (br) grid. The width and height are both set to **200**.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/crop,x_10,y_10,w_200,h_200,g_br

10.2 Inscribed Circle

You can edit code on OBS Console or make an API call to get an inscribed circle of a circle.

Choose the image center as the center of a circle, and crop the image according to the specified radius. The image then is cropped into a circle. For details, see **Table 10-2**.

Operation name: circle

Table 10-2 Description for inscribed circle

Parameter	Value Description	Code Example
r	Circular radius of the cropped image, ranging [0, half of the shorter side of the image].	image/circle,r_100

NOTE

- If the image is output in JPG format, the area outside the inscribed circle is white. If the image is output in vector format such as PNG, WebP, and BMP, the area outside the inscribed circle is transparent. It is advisable to output the image in PNG format.
- If **r** is larger than half of the shorter side, the inscribed circle output is still the largest inscribed circle of the image (with a radius equal to half of the shorter length).

Example

Set the radius of the cropped image to **100** and output the image in JPG format. Set the area outside the inscribed circle to white.

https://obs.example.region.com/image-demo/example.jpg?x-image-process=image/circle,r_100

10.3 Indexcropping

You can edit code on OBS Console or make an API call to crop images based on indexes.

Set the top left corner of the image as the starting point. Set x-axis overlapping with the width, and y-axis overlapping with the height. Crop the image into several consecutive partitions horizontally or vertically, each having equal width or height. Get the partition you want according to the index. For details, see **Table 10-3**.

Operation name: indexcrop

Table 10-3 Indexcrop description

Parameter	Value Description	Code Example
x	Width of each partition after horizontal cropping, ranging [1, original width of the image]. **x** and **y** cannot be chosen at the same time.	image/ indexcrop,x_1000,i_0
y	Height of each partition after vertical cropping, ranging [1, original height of the image]. **x** and **y** cannot be chosen at the same time.	
i	If there are n partitions in total, **i** ranges [0, n-1]. When i = 0, you obtain the first partition. If you set a value larger than **n-1**, the original image will be returned.	

Example

- Indexcrop the image horizontally. The width of each partition is 1000. Choose the first partition.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/indexcrop,x_1000,i_0

- Indexcrop the image horizontally. The width of each partition is 600. Choose the first partition.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/indexcrop,x_600,i_0

10.4 Rounded Corner Cropping

You can edit code on OBS Console or make an API call to get images with rounded corners.

This feature allows you to crop an image to a rounded rectangle by specifying the rounded corner size. The size can be configured by specifying the border radius and horizontal/vertical radius of each cropped rounded corner. For details, see **Table 10-4**.

Operation name: rounded-corners

Table 10-4 Description for rounded corner cropping

Parameter	Value Description	Code Example
r	Crop an image to a rounded corner rectangle, and specify the border radius of each corner. In this scenario, the horizontal radius and vertical radius are the same. Pixels (for example, 200) or percentage (for example, 25p) can be used as the unit. The value of pixels ranges from 1 to 4096. When the value is greater than half of the pixel of the original image's shorter side, set this parameter to half of the pixel of the shorter side. The value of percentage ranges from 1p to 50p. This parameter cannot be used together with **rx** and **ry**.	image/rounded-corners,r_100
rx	Indicates the horizontal radius of a rounded corner. Pixels (for example, 200) or percentage (for example, 25p) can be used the unit. The value of pixels ranges from 1 to 4096. When the value is greater than half of the pixel of the original image's shorter side, set this parameter to half of the pixel of the shorter side. The value of percentage ranges from 1p to 50p. This parameter must be used together with **ry**.	image/rounded-corners,rx_100,ry_200

Parameter	Value Description	Code Example
ry	Indicates the vertical radius of a rounded corner. Pixels (for example, 200) or percentage (for example, 25p) can be used the unit. The value of pixels ranges from 1 to 4096. When the value is greater than half of the pixel of the original image's shorter side, set this parameter to half of the pixel of the shorter side. The value of percentage ranges from 1p to 50p. This parameter must be used together with **rx**.	

NOTE

If the output image format is JPG, the cut-out corner area is white. If the output image format is PNG, WebP, or BMP, the cut-out corner area is transparent. You are advised to save the cropped images in PNG format.

Example

- Set the border radius of the example JPG image to **100**, and save it as PNG.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/rounded-corners,r_100/format,png

- Set the horizontal radius of each rounded corner of the example JPG image to **100**, and its vertical radius to **200**.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/rounded-corners,rx_100,ry_200

11 Watermarking Images

11.1 Public Parameters

You can use the GUI, code, or APIs to configure public parameters.

You can add a text or image watermark to the original image.

The Base64 code for URL transmission applies to paths of content and fonts of text watermarks, or paths of image watermarks. It is not advisable to put standard Base64 code directly into the URL for transmission. In Base64 encoding for URL transmission, contents are coded into character strings by standard Base64 code. After verifying these strings, replace the plus sign (+) with hyphen (-), and slash (/) with underline (_). For details about encoding, see those specified in RFC4648.

Operation name: watermark

Public parameters are applicable to setting image watermarks and text watermarks. You can add text watermarks and image watermarks to the same image. **Table 11-1** lists the basic parameters in detail.

Table 11-1 Public Parameters

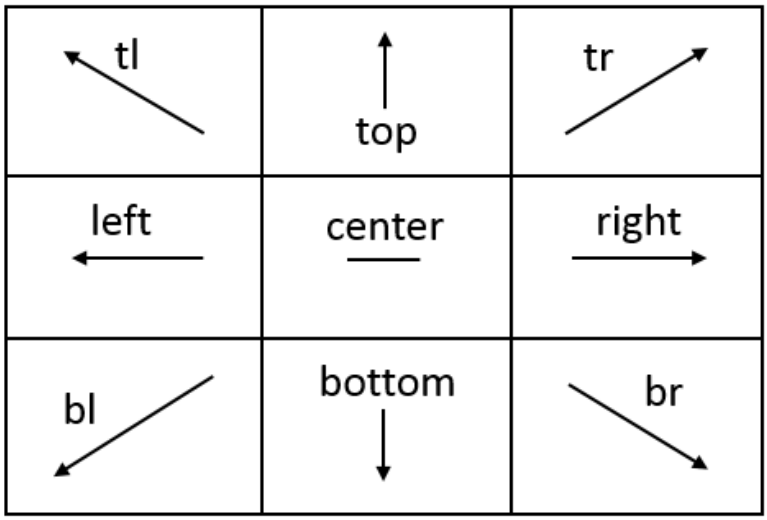
Parameter	Value Description	Code Example
g	Optional, represents the location of the watermark: **tl**, **top**, **tr**, **left**, **center**, **right**, **bl**, **bottom**, and **br**. The default value is **tl**. **Figure 11-1** is the 3 x 3 grid illustrating these values. Each value locates at the top left corner of a grid.	image/watermark,image_aW1hZ2UtZGVtby9sb2dvLnBuZw==,g_br,t_90,x_10,y_10

Parameter	Value Description	Code Example
x	Optional parameter, representing the horizontal distance from the image edge. By default, the origin is at the top left corner. **x** ranges [0, 4096]. It is set to **10** by default, with the unit pixel (px).	
y	Optional parameter, representing the vertical distance from the image edge. By default, the origin is at the top left corner. **x** ranges [0, 4096]. It is set to **10** by default, with the unit pixel (px).	
voffset	Optional parameter, representing the vertical offset from the horizontal centerline of the image. This parameter offsets the watermark up or down from the horizontal centerline of the image. **voffset** ranges [-1000, 1000]. It is set to **0** by default, with the unit pixel (px). **voffset** is meaningful only when **g** is set to **left**, **center**, or **right**. That is to say, the watermark locates in the left, center, or right grid.	
align	Optional parameter, representing the align mode of the text watermark and the image watermark. The value can be **0**, **1**, or **2**. It is set to **0** by default. ● **0**: The top edges of the text watermark and the image watermark are aligned. ● **1**: The centerlines of the text watermark and the image watermark are aligned. ● **2**: The bottom edges of the text watermark and the image watermark are aligned.	
order	Optional parameter, representing the sequence of the text watermark and the image watermark. The value can be **0** (default value) or **1**. ● **0**: Image in front, text behind. ● **1**: Text in front, image behind.	

Parameter	Value Description	Code Example
t	Optional parameter, representing the extent of transparency of the text or image watermark. **t** ranges [0, 100]. The default value is **100**, indicating the watermark is not transparent at all.	
interval	Optional parameter, representing the distance between the text watermark and image watermark. The interval ranges [0, 1000].	

Figure 11-1 is the 3 x 3 grid illustrating the location of the watermark.

Figure 11-1 3 x 3 grid of watermark location



NOTE

If both a text watermark and an image watermark are added to the original image, the location of the watermark can be adjusted by the horizontal distance **x**, vertical distance **y**, and the vertical offset from the horizontal centerline **voffset**. You can adjust the layout of the two watermarks as well.

11.2 Image Watermarks

Image watermark parameters are the parameters used when adding image watermarks.

You can pre-process the image watermark before adding it to the original image. These pre-processing operations include **Resizing Images**, **Rotating Images**, and **Cropping Images**, but does not include cropping it into an inscribed circle. In addition, you can scale the watermark based on the original image when resizing for pre-processing.

Table 11-2 lists the descriptions for image watermark parameters in detail.

Table 11-2 Image watermark parameters

Parameter	Value Description	Code Example
image	Watermark image path. This parameter is mandatory when you add a watermark. The image watermark address is: *bucketName*/*objectName*(required code) or *bucketName*/*objectName*?x-image-process=*image*/*command*(required code). **NOTICE** The content must be base64 code of URL. encodedObject = url_safe_base64_encode(object). For example, object **panda.png** will be encoded as **cGFuZGEucG5n**.	image/resize,w_400/watermark,image_aW1hZ2UtZGVtby9sb2dvLnBuZz94LWltYWdlLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMzA=,t_90,g_br,x_10,y_10
P	Watermark image size. The watermark image is scaled according to the percentage P of the original image (image to which the watermark is added). The value ranges from 1 to 100. **NOTICE** The preceding resize operation supports only the uppercase **P** parameter. To adjust the watermark image size, see **Resizing Images** (except the **p** parameter).	image-demo/logo.png?x-image-process=image/resize,P_50

API Call Examples

- The watermark file is **logo.png** (logo address: e-share/image-demo/logo.png). Make the watermark located at the bottom right corner, with a transparency of **90** and default horizontal and vertical margins of **10**.

  Parameters are as follows:

  Watermark image: **e-share/image-demo/logo.png**

  Corresponding Base64 code: **ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5n**

  Watermark position (bottom right): **g_br**

  Transparency: **t_90**

  Horizontal and vertical margins: **x_10,y_10**

  The URL request is as follows:

  https://obs.region.example.com/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5n,g_br,t_90,x_10,y_10

Figure 11-2 Example 1

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5n,g_br,t_90,x_10,y_10

- The watermark file is **logo.png** (logo address: e-share/image-demo/logo.png). Resize the watermark image by setting its width to **50**. Other parameters are the same as those in the previous example.

  Parameters are as follows:

  Watermark image: **e-share/image-demo/logo.png?x-image-process=image/resize,w_50**

  Corresponding Base64 code: **ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5nP3gtaW1hZ2UtcHJvY2Vzcz1pbWFnZS9yZXNpemUsd181MA**

  Watermark position (bottom right): **g_br**

  Transparency: **t_90**

  Horizontal and vertical margins: **x_10,y_10**

  The URL request is as follows:

  https://obs.region.example.com/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5nP3gtaW1hZ2UtcHJvY2Vzcz1pbWFnZS9yZXNpemUsd181MA,g_br,t_90,x_10,y_10

Figure 11-3 Example 2

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5nP3gtaW1hZ2UtcHJvY2Vzcz1pbWFnZS9yZXNpemUsd181MA,g_br,t_90,x_10,y_10

- The watermark file is **logo.png** (logo address: **e-share/image-demo/logo.png**). Make the watermark image 50 percent of its original size. Other parameters are the same as those in the previous example.

  Parameters are as follows:

  Watermark image: **e-share/image-demo/logo.png?x-image-process=image/resize,P_50**

  Corresponding Base64 code: **ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5nP3gtaW1hZ2UtcHJvY2Vzcz1pbWFnZS9yZXNpemUsUF81MA**

  Watermark position (bottom right): **g_br**

  Transparency: **t_90**

  Horizontal and vertical margins: **x_10,y_10**

  The URL request is as follows:

  https://obs.region.example.com/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5nP3gtaW1hZ2UtcHJvY2Vzcz1pbWFnZS9yZXNpemUsUF81MA,g_br,t_90,x_10,y_10

Figure 11-4 Example 3

https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5nP3gtaW1hZ2UtcHJvY2Vzcz1pbWFnZS9yZXNpemUsUF81MA,g_br,t_90,x_10,y_10

GUI Example

You can configure image watermarks by editing image style on OBS Console. The watermark file is **logo.png** (logo address: **e-share/image-demo/logo.png**). The watermark locates at the bottom right corner, with a transparency of **90** and default horizontal and vertical margins of **10**. Perform the following steps on the console:

Step 1 Log in to OBS Console, in the navigation tree on the left, click the bucket name and choose **Data Processing** > **Image Processing**.

Step 2 Click **Create**. The style editing page is displayed.

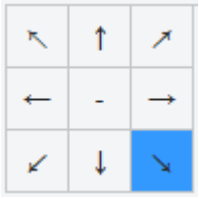
Step 3 On the editing page, input a style name, set the **Edit Mode** to **GUI**, and select **Watermark**.

NOTE

A style name consists of letters (uppercase and lowercase), digits, periods (.), underlines (_), and hyphens (-), and contains 1 to 256 characters, for example, **rotate_0001**.

Step 4 Configure the following watermark parameters.

- **Watermark Type**: Choose **Image Watermark**.
- **Watermark Image Path**: Enter **e-share/image-demo/logo.png**.

- **Image Size (%)**: Set this parameter based on whether the watermark image is zoomed out.
- **Brightness**: Retain the default value **0**.
- **Contrast**: Retain the default value **0**.
- **Transparency**: Set this parameter to **90**.
- **Watermark Position**: Select the arrow  at the bottom right.
- **Vertical Margin**: Retain the default value **10**.
- **Horizontal Margin**: Retain the default value **10**.

Step 5 After finishing editing the image style, click **OK** to save the style. The new style will be displayed in the style list.

You can use the new watermark style to process images by referring to **Applying Image Styles**.

----End

11.3 Text Watermarks

Text watermark parameters are the parameters used when adding text watermarks. These parameters include the font size, type, and color of texts. **Table 11-3** lists the descriptions for text watermark parameters in detail.

Table 11-3 Text watermark parameters

Parameter	Value Description	Code Example
text	Required parameter when adding text watermarks. **NOTICE** The parameter value must be encoded using URL-safe Base64, for example, **encodeText = url_safe_base64_encode(fontText)**, with a maximum length of 64 characters.	image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ,size_60,color_FF0000,type_ZmFuZ3poZW5nc2h1c29uZw==,g_center,rotate_30
size	Optional parameter that represents the font size of watermarks. It ranges from 0 to 1000, and it is set to **40** by default.	

Parameter	Value Description	Code Example
type	Optional parameter that represents the font type of watermarks. Table 11-4 shows the values in detail. The default value is wqy-zenhei (the value after encoding is d3F5LXplbmhlaQ). **NOTICE** - The parameter value must be encoded using URL-safe Base64, for example, encodeText = url_safe_base64_encode(fontType). - Line breaks are currently not supported.	
color	Optional parameter that represents the font color of watermarks. The value is a six-digit hexadecimal code, from **000000** to **FFFFFF** (which represents black and is the default value).	
shadow	Optional parameter that represents the extent of transparency of text watermarks. It ranges from 0 to 100.	
fill	Optional parameter, representing the overspread effect of watermarks. The value can be **0** or **1**. ● **0**: No effect. ● **1**: Overspread.	
rotate	Optional parameter, representing the clockwise angle of text watermarks. The angle is larger than 0 and smaller than 360 degrees.	

Table 11-4 Cross reference for font type encoding

Parameter	Value After URL base64 Encoding	Value Description	Remarks
droidsansfallback	ZHJvaWRzYW5zZmFsbGJhY2s=	DroidSansFallback	According to Request For Comments (RFC), the fuller "=" can be omitted, and the value becomes **ZHJvaWRzYW5zZmFsbGJhY2s**.

Parameter	Value After URL base64 Encoding	Value Description	Remarks
fangzhengfangsong	ZmFuZ3poZW5nZmFuZ3Nvbmc=	FZFongSong	According to RFC, the fuller "=" can be omitted, and the value becomes **ZmFuZ3poZW5nZmFuZ3Nvbmc**.
fangzhengheiti	ZmFuZ3poZW5naGVpdGk=	FZSimHei	According to RFC, the fuller "=" can be omitted, and the value becomes **ZmFuZ3poZW5naGVpdGk**.
fangzhengkaiti	ZmFuZ3poZW5na2FpdGk=	FZKaiTi	According to RFC, the fuller "=" can be omitted, and the value becomes **ZmFuZ3poZW5na2FpdGk**.
fangzhengshusong	ZmFuZ3poZW5nc2h1c29uZw==	FZShuSong	According to RFC, the fuller "=" can be omitted, and the value becomes **ZmFuZ3poZW5nc2h1c29uZw**.
wqy-microhei	d3F5LW1pY3JvaGVp	WenQuanYi Micro Hei	-
wqy-zenhei	d3F5LXplbmhlaQ==	WenQuanYi Zen Hei	According to RFC, the fuller "=" can be omitted, and the value becomes **d3F5LXplbmhlaQ**.

API Call Examples

- Add a text watermark **Hello** to the original image. Set text size to **60**, color to red, and font to FZShuSong.

  Parameters are as follows:

  URL Base64 code: **SGVsbG8g5Zu**, namely **text_SGVsbG8g5Zu**

  Font size: size_60

  Font color: color_FF0000

  Font type: type_ZmFuZ3poZW5nc2h1c29uZw==

  The URL request is as follows:

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/

watermark,text_SGVsbG8g5Zu,size_60,color_FF0000,type_ZmFuZ3poZW5nc2h1c29uZw==

- Add the text watermark to the original image and center the watermark. Rotate the text 30 degrees clockwise. Other parameters are set in a way similar to the previous example.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/watermark,text_SGVsbG8g5Zu,size_60,color_FF0000,type_ZmFuZ3poZW5nc2h1c29uZw==,g_center,rotate_30

- Insert image and text watermarks at the same time. Put the text watermark **Hello** at the bottom right, with font size set to **60**, shadow to **50**, and color to red.

  Use the image watermark **logo.png**, with both horizontal and vertical margins set to **10**.

  Set the transparency of this mixed watermark to **50** and put the image in front and the text behind, in the bottom alignment.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5n,text_SGVsbG8g5Zu,size_60,color_FF0000,shadow_50,type_ZmFuZ3poZW5nc2h1c29uZw==,g_br,x_10,y_10,align_2,order_0

- Insert an image watermark and a text watermark separately. Put the text watermark **Hello** at the bottom right, with font size set to **60** and color to red.

  Use the image watermark **logo.png**, with the horizontal margin set to **40** and vertical margin to **10**.

  Set the transparency of the image watermark to **90**. The image watermark is inserted first, and then the text watermark.

  https://obs.region.example.com/example.jpg?x-image-process=image/watermark,image_ZS1zaGFyZS9pbWFnZS1kZW1vL2xvZ28ucG5n,g_br,t_90,x_45,y_10/watermark,text_SGVsbG8g5Zu,size_60,color_FF0000,type_ZmFuZ3poZW5nc2h1c29uZw==,g_br,x_0,y_330


Hello
LOGO

12 Converting Formats

12.1 Converting Formats

You can use the GUI, code, or APIs to convert image formats. The original image can be converted into supported formats. For details, see **Table 12-1**.

- Supported original formats: JPG, JPEG, PNG, BMP, WebP, GIF, and TIFF.
- Supported target formats: JPG, PNG, BMP, and WebP.

Operation name: format

Table 12-1 Format conversion

Parameter	Value Description	Code Example
jpg	The image is saved in JPG format. If the original image is in vector formats such as WebP, BMP, and PNG, the transparent part will be padded to white.	image/format,jpg
webp	The image is saved in WebP format.	image/format,webp
bmp	The image is saved in BMP format.	image/format,bmp
png	The image is saved in PNG format.	image/format,png

Example

- Save the original image of JPG format into PNG format.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/format,png

12.2 Interlaced Image Loading

You can edit code on OBS Console or make an API call to interlace images.

With format conversion function, the image is output in Baseline JPEG format. If you want to output an image in Progressive JPEG, use the **interlace** parameter. Table 12-2 lists the parameters in detail.

- Presentation mode of Baseline JPG images: top to down.
- Presentation mode of Progressive JPEG images: blurred to clear.

Operation name: interlace

Table 12-2 Interlace description

Parameter	Value Description	Code Example
value	The value can be **0** or **1**. **0**: The output is a JPG image that is presented from top to down. **1**: The output is a JPEG image that is presented progressively.	image/format,jpg/interlace,1

Example

- Output a JPG image that is presented progressively.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/format,jpg/interlace,1

- Output a JPG image presented from top down.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/format,jpg/interlace,0

13 Changing Quality

You can edit code on OBS Console or make an API call to change the quality of an image.

To save space, you can compress images that are output in JPG format. **Table 13-1** lists the parameters in detail.

Operation name: quality

Table 13-1 Compression description

Parameter	Value Description	Code Example
q	Relative quality of the image. The image is compressed to q% of the original. **q** ranges from 1 to 100. Formula for compression: Target quality = Original quality x q% For example, if the original quality of the image is 100% and the relative quality is 80%, then the target quality of the image is 80%. If the original quality of the image is 80% and the relative quality is 80%, then the target quality of the image is 64%.	image/ resize,w_100,h_100/ quality,q_80

Parameter	Value Description	Code Example
Q	Absolute quality of the image. The image is compressed into Q%. Q is irrelevant to and does not depend on the original image. **Q** ranges from 1 to 100. Formula for compression: • When Original quality > Q%, Target quality = Q%. • When Original quality = Q%, Target quality = Original quality = Q%. • When Original quality < Q%, Target quality = Original quality. For example, if the original quality of the image is 100% and the absolute quality is 80%, then the target quality of the image is 80%. If the original quality of the image is 70% and the absolute quality is 80%, then the target quality of the image is 70%.	

NOTE

- **q** is valid only for JPG images.
- If both **q** and **Q** are used, the output is based on **Q**.
- **q** and **Q** are only valid for JPG images. For images in other formats, **q** and **Q** bring no effect and cause no impact.

Example

- Resize the image by setting the height and width both to **100**, and output a jpg image with relative quality of 80%.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,w_100,h_100/quality,q_80

- Resize the image by setting the height and width both to **100**, and output a jpg image with absolute quality of 80%.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,w_100,h_100/quality,Q_80

14 Slimming Images

You can edit code on OBS Console or make an API call to slim images.

Downsizing is a simplified image compression function. The output quality is 75% of the absolute quality. You do not need to configure any parameter. You can slim images just by running a command.

Operation name: imageslim

NOTE

- Only images in the JPG format support this function.
- It is recommended that you perform this operation following the completion of other processing operations.

Example

- If you want to resize an image to the width and height at 100 respectively, resize it first, and then slim it.

  https://obs.region.example.com/image-demo/example.jpg?x-image-process=image/resize,w_100,h_100/imageslim

15 Image Persistency

With image persistency, images are asynchronously stored in the specified OBS bucket, so that you can access the processed images directly for a better experience.

Currently, this function can be used only by coding or calling the API. In the image processing request interface, the image processing persistency request is sent in the format of **parameter name = parameter value**. Table 15-1 describes the parameters.

Table 15-1 Persistency

Parameter	Value	Description
x-image-save-object	*objectName*	This parameter is mandatory. Specifies the name of the target object, that is, the name of the processed image that will be stored in the bucket. Object naming requirements are as follows: • The value cannot contain the following special characters: **\:*?"<>|** • The value ranges from 1 to 1023 characters.
x-image-save-bucket	*bucketName*	This parameter is optional. Specifies the target bucket. The processed images are stored in the bucket. If this parameter is not specified, the images are saved to the current bucket by default. The bucket name ranges from 1 to 64 characters and must an existing bucket in OBS.

Java Sample Code

```
ObsClient obsClient = null;
String endPoint = "obs-endpoint";  // Current region
// Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and store them in the configuration file or environment variables.
// In this example, the AK and SK are stored in environment variables for identity authentication. Before
```

```
running this example, configure environment variables ACCESS_KEY_ID and SECRET_ACCESS_KEY.
// Obtain an AK and SK pair on the management console.
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY");
try {
    ObsConfiguration config = new ObsConfiguration();
    config.setEndPoint(endPoint);
    obsClient = new ObsClient(ak,sk ,config);
    TemporarySignatureRequest request = new TemporarySignatureRequest();
    request.setObjectKey("test.jpeg");  // Original object name before processing
    Map<String, Object> queryParams = new HashMap<>();
    queryParams.put("x-image-process", "image/resize,w_100");
    String objectName = "your saves objectName";  // Name of the processed object
    //Optional parameters
    String bucketName = "your saves Bucket";  // Bucket that stores the processed object
    queryParams.put("x-image-save-object", ServiceUtils.toBase64(objectName.getBytes("UTF-8")));
    queryParams.put("x-image-save-bucket", ServiceUtils.toBase64(bucketName.getBytes("UTF-8")));
    request.setQueryParams(queryParams);
    request.setBucketName("your bucket");  // Bucket that stores the original object
    TemporarySignatureResponse response = obsClient.createTemporarySignature(request);
    //URL to access
    response.getSignedUrl();
 } catch (Exception e) {
...//Handle exceptions.
 } finally {
     if (obsClient != null) {
         obsClient.close();
      }
 }
```

Python Sample Code

```
from obs import ObsClient
import os
import traceback
import requests

# Obtain an AK and SK pair using environment variables or import the AK and SK pair in other ways. Using hard coding may result in leakage.
# Obtain an AK and SK pair on the management console.
ak = os.getenv("AccessKeyID")
sk = os.getenv("SecretAccessKey")
# (Optional) If you use a temporary AK and SK pair and a security token to access OBS, obtain them from environment variables.
security_token = os.getenv("SecurityToken")
# Set server to the endpoint corresponding to the bucket. region is used here as an example. Replace it with the one in use.
server = "https://obs.region.com"
# Create an obsClient instance.
# If you use a temporary AK and SK pair and a security token to access OBS, you must specify security_token when creating an instance.
obsClient = ObsClient(access_key_id=ak, secret_access_key=sk, server=server)
try:
    # Generate a signed URL for image persistency.
    # Name of the bucket that stores the original object
    bucketName = 'originBucketName';
    # Original object name
    objectKey = 'test.png';

    # Name of the object after processing
    targetObjectName ="save.png"
    # (Optional) Name of the bucket that stores the new object
    targetBucketName ="saveBucketName"
    queryParams={}
    queryParams["x-image-process"]="image/resize,w_100"
    queryParams["x-image-save-object"]=base64.b64encode(targetObjectName .encode("utf-8")).decode()
    # Optional parameter
    queryParams["x-image-save-bucket"]=base64.b64encode(targetBucketName .encode("utf-8")).decode()

    res = obsClient.createSignedUrl(method='GET', bucketName=bucketName, objectKey=objectKey,
```

```
queryParams=queryParams, expires=3600)
    print('signedUrl:', res.signedUrl)
    print('actualSignedRequestHeaders:', res.actualSignedRequestHeaders)
    // Make a GET request for image persistency.
    r = requests.get(resp.signedUrl)
    print(r)
except:
    print(traceback.format_exc())
```

Node.js Sample Code

```
// Import the OBS library.
const ObsClient = require('esdk-obs-nodejs');
const https = require('https');
const http = require('http');
const urlLib = require('url');

// Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and store them in the configuration file or environment variables.
// In this example, the AK and SK are stored in environment variables for identity authentication. Before running this example, configure environment variables ACCESS_KEY_ID and SECRET_ACCESS_KEY.
// Obtain an AK and SK pair on the management console.
const ak = process.env.ACCESS_KEY_ID;
const sk = process.env.SECRET_ACCESS_KEY;
const server = "obs-endpoint";  // Current region

// Create an ObsClient instance.
const obsClient = new ObsClient({
    access_key_id: ak,
    secret_access_key: sk,
    server: server
});

// Name of the bucket that stores the original object
const bucketName = 'originBucketName';
// Original object name
const objectKey = 'test.png';
const method = 'GET';

// Name of the object after processing
const targetObjectName = "save.png";
// (Optional) Name of the bucket that stores the new object
const targetBucketName = 'saveBucketName';

const queryParams = {
    "x-image-process": "image/resize,w_100",
    "x-image-save-object": Buffer.from(targetObjectName, 'utf8').toString('base64'),
    // Optional parameter
    "x-image-save-bucket": Buffer.from(targetBucketName, 'utf8').toString('base64')
}

const res = obsClient.createSignedUrlSync({
    Method: method,
    Bucket: bucketName,
    Key: objectKey,
    QueryParams: queryParams
});

// Make a GET request for image persistency.
const url = urlLib.parse(res.SignedUrl);
const request = server.startsWith('http://') ? http : https;
const req = request.request({
    method: method,
    host: url.hostname,
    port: url.port,
    path: url.path,
    rejectUnauthorized: false,
    headers: res.ActualSignedRequestHeaders || {}
});
```

NOTE

- The object name and bucket name must be Base64 encoded and URL safe. The format is encodedObject = url_safe_base64_encode(name). For example, object **panda.png** will be encoded as **cGFuZGEucG5n**. After Base64 encoding, if the name contains plus signs (+) and slashes (/), replace them with hyphens (-) and underscores (_), respectively.
- If a signature matching error is reported, check whether the AK and SK pair is correct and whether the accessed URL is the same as the signature URL generated by the code.
- Currently, image persistency with the range header is not supported.

16 FAQ

16.1 What Is Image Processing?

Image processing is a feature integrated in Object Storage Service (OBS). It provides stable, secure, efficient, and inexpensive image processing services. By using this feature, you can slim, crop, resize, and watermark images, as well as convert the formats of images.

You can access this feature via OBS Console and REST APIs, to process images stored in OBS anytime and anywhere and obtain the processed images right away.

16.2 How to Access Image Processing?

- Log in to the management console to preview the effects in different styles on OBS.
- Call RESTful APIs to access Image Processing using applications.

16.3 How Many Styles Are Allowed To Be Created for Each Bucket?

Each bucket supports a maximum of 100 styles.

OBS provides two methods to create image styles:

- **Creating Image Styles**
- **Processing Images**

16.4 What Formats Are Supported by Image Processing?

Supported original formats: JPG, JPEG, PNG, BMP, WebP, GIF, and TIFF. Supported target formats: JPG, PNG, BMP, and WebP.

16.5 How Do I Access Image Processing with a URL?

Accessing Images Not Publicly Readable

To access images that cannot be read by the public, add image processing parameters during signature calculation to create a signed temporary URL.

Accessing Images Publicly Readable

To access images that can be read by the public, add image processing parameters to the URL request.

A Change History

Release Date	What's New
2024-01-29	This is the first official release.