Tencent Cloud EdgeOne

Edge Functions

Product Documentation

©2013-2025 Tencent Cloud International Pte. Ltd.

Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

- Edge Functions
  - Overview
  - Getting Started
  - Operation Guide
    - Function Management
    - Web Debugging
    - Function Trigger
    - Environment Variable
  - Runtime APIs
    - addEventListener
    - Cache
    - Cookies
    - Encoding
    - Fetch
    - FetchEvent
    - Headers
    - Request
    - Response
    - Streams
      - ReadableStream
      - ReadableStreamBYOBReader
      - ReadableStreamDefaultReader
      - TransformStream
      - WritableStream
      - WritableStreamDefaultWriter
    - Web Crypto
    - Web standards
    - Images
      - ImageProperties
  - Sample Functions
    - Example Overview
    - 301 Redirect
    - Obtaining Client URL Information
    - Customization Based on Client Geo Location
    - Obtaining Client Geo Location Information

©2013-2025 Tencent Cloud International Pte. Ltd.

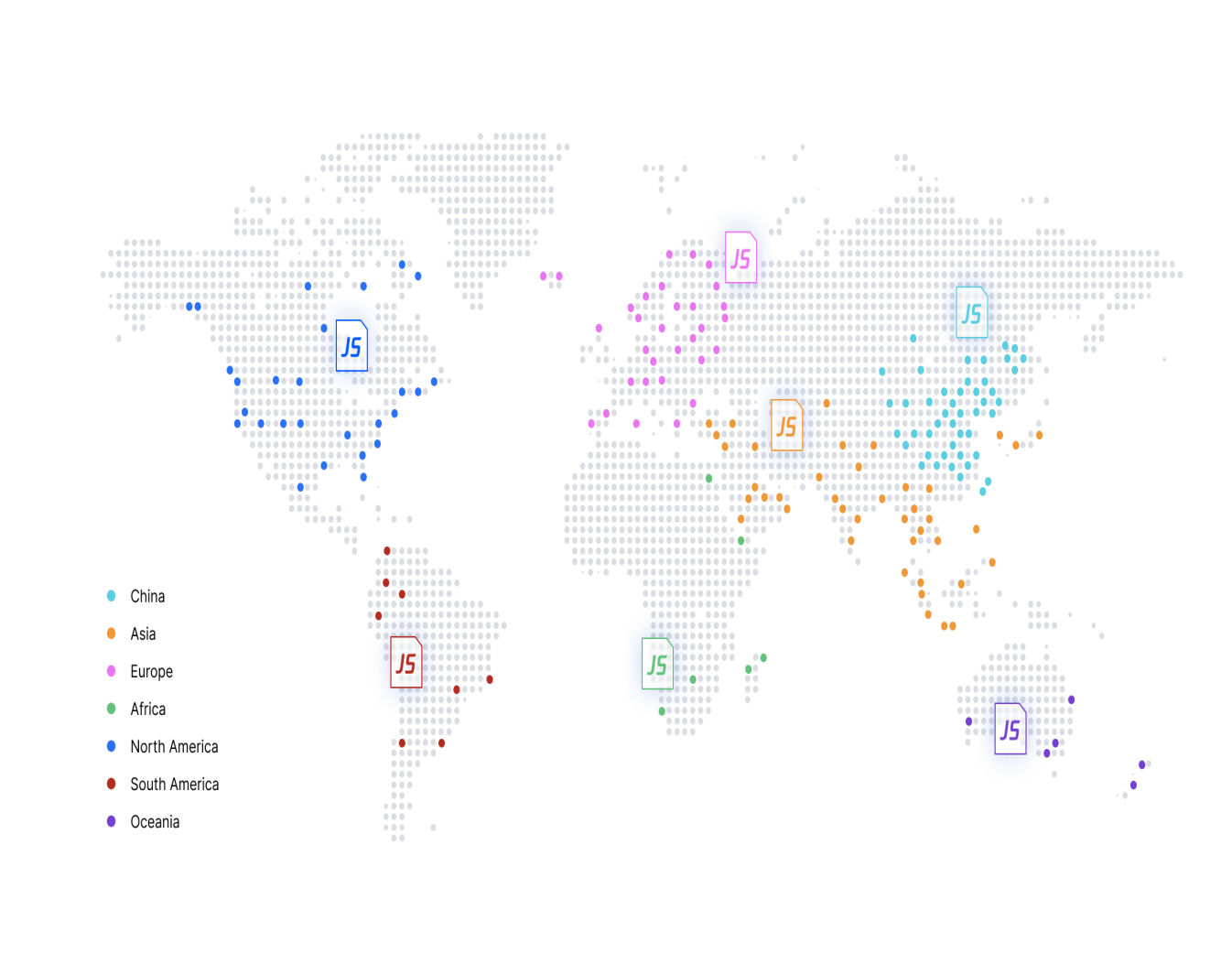
Batch Redirect
URL rewrite based on regular expressions
Returning an HTML Page
Returning a JSON Object
Fetch Remote Resources
Authenticating a Request Header
Modifying a Response Header
Performing an A/B Test
Setting Cookies
Performing Redirect Based on the Request Location
Using the Cache API
Caching POST Requests
Responding in Streaming Mode
Merging Resources and Responding in Streaming Mode
Protecting Data from Tampering
Rewriting a m3u8 File and Configuring Authentication
Adaptive Image Resize
Image Adaptive WebP
Customize Referer restriction rules
Remote Authentication
HMAC Digital Signature
Naming a Downloaded File
Obtaining Client IP Address
Complex origin-pull URL rewriting
Best Practices
Overview
Adaptive Image Format Conversion via Edge Functions

©2013-2025 Tencent Cloud International Pte. Ltd.

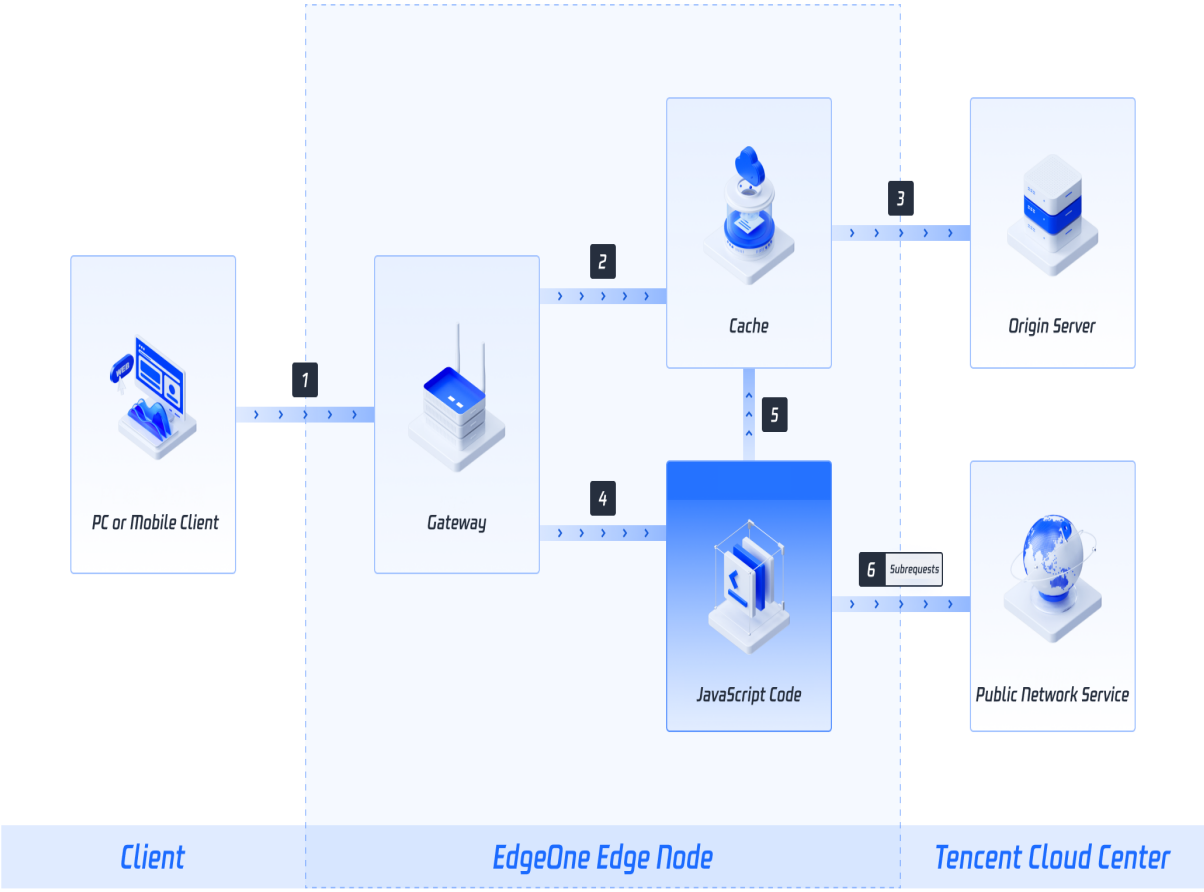
Edge Functions
Overview

Last updated：2024-07-16 10:17:02

Tencent Cloud Edge Functions provides a serverless code execution environment for the edge nodes of Tencent Cloud EdgeOne. This way, you can focus on writing business function code and configuring triggering rules, without the need to configure or manage infrastructure such as servers. The written code can be elastically and securely executed on the edge nodes that are closest to users.



How It Works



You can develop JavaScript functions and deploy them to the edge nodes of Tencent Cloud EdgeOne.

1. If a client request does not hit the configured function triggering rule, the request is handled in the following process: `(1)` The client request is sent to the gateway of an edge node of Tencent Cloud EdgeOne. > `(2)` The cache of the node responds if the requested content already exists in the cache. > `(3)` The origin server responds if the requested content does not exist in the cache.

2. If a client request hits the configured function triggering rule, the request is handled in one of the following processes:

`(1)` The client request is sent to the gateway of an edge node of Tencent Cloud EdgeOne. > `(4)` Edge Functions receives and executes the JavaScript code. > `(5)` Subrequests access the cache. > `(3)` The origin server responds if the requested content does not exist in the cache.

`(1)` The client request is sent to the gateway of an edge node of Tencent Cloud EdgeOne. > `(4)` Edge Functions receives and executes the JavaScript code. > `(6)` Subrequests access the public network service.

Benefits

Distributed Deployment

Tencent Cloud EdgeOne supports more than 3,200 edge nodes. Edge Functions is deployed on edge nodes in distributed mode.

Ultra-low Latency

Client requests are automatically scheduled to the edge nodes that are closest to users. If triggering rules are hit, edge functions are triggered to process the requests and return results to the client. This helps significantly reduce the client access latency.

Elastic Scaling

Edge Functions schedules requests to edge nodes that are allocated sufficient computing resources based on the proximity of the user when spikes occur in client requests.

Serverless Architecture

The serverless architecture of Edge Functions eliminates the need to focus on the maintenance of the memory, CPU, and network of servers and other infrastructure resources. You can focus on the development of business code.

Use Cases

©2013-2025 Tencent Cloud International Pte. Ltd.

Use Limits

Content	Limit	Note
Number of Functions per Site	100 functions	A single site supports up to 100 edge functions.
Number of Trigger Rules per Site	200 rules	A single site supports up to 200 trigger rules for functions.
Function Name Length	30 characters	2 to 30 characters, up to 30 characters.
Code Package Size	5 MB	The maximum size for a single function code package is 5 MB.
Request Body Size	1 MB	Client requests can carry a body up to 1 MB.
CPU Time	200 ms	CPU time slice allocated for single function execution, excluding I/O waiting time.

©2013-2025 Tencent Cloud International Pte. Ltd.

Programming Language	JavaScript	Currently only JavaScript is supported.
Number of Console Invocations	20 times	Each function is allowed to call the console method up to 20 times. Beyond 20 times, print operations will no longer be executed.
Number of Loop Executions	100,000 times	In the function, the `for` , `for in` , `for of` , `while` , `do while` loops are limited to execute no more than 100,000 times.

©2013-2025 Tencent Cloud International Pte. Ltd.

Getting Started

Last updated：2024-08-26 14:16:27

This document describes how to create a simple edge function and use the function to redirect a request to another URL and return the custom response header.

Overview

The site `example.com` is currently connected to EdgeOne service. Under this site, a custom HTML activity page needs to be provided to the users through the custom domain name `www.example.com`. This page can be deployed to the edge nodes of EdgeOne's global AZs through the edge function for users to access nearby.

Note:

1. How to connect to the site, please refer to Quick Start.
2. How to add an acceleration domain, please refer to Adding a Domain Name for Acceleration.

Directions

Step 1: Creating and Deploying a Function

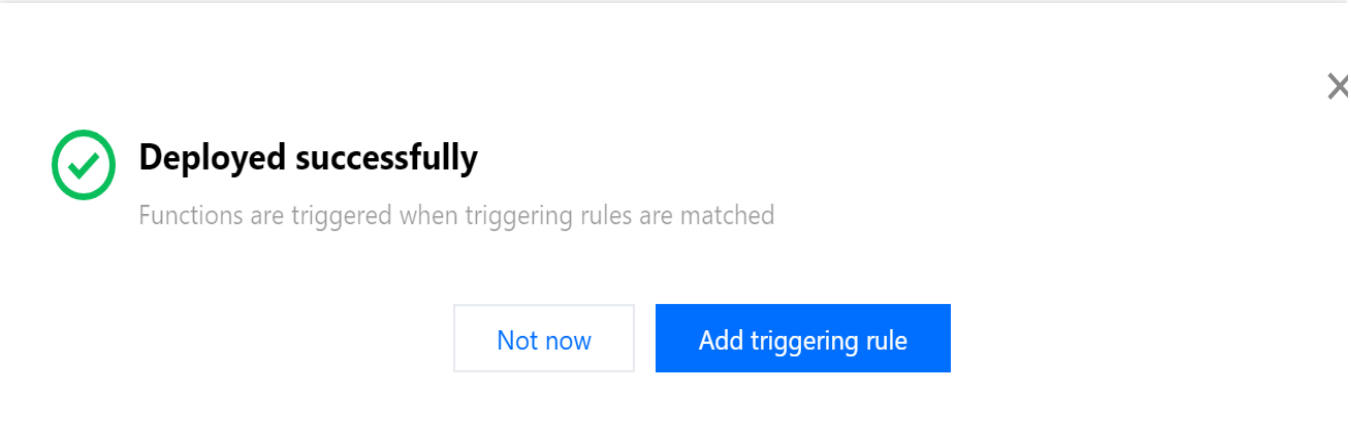
1. Log in to the EdgeOne Console and click **Site List** in the left sidebar. In the site list, click the **Site** to be configured.
2. On the site details page, click **Edge Function** > **Function Management**.
3. On the edge function management page, click **Create function** and select to create a function using a template. At this step, you can use a template to create a function according to actual business needs. Taking the current scenario as an example, you can select the "Create Hello World" template to create a new one. After selecting the template, click **Next**.
4. On the new Edge Functions page, configure the relevant parameters. The explanations for these parameters are as follows:

Function: Required, it can only contain letters, numbers, hyphens. It should start with a letter and end with a number or letter, 2-30 characters; and it cannot be modified after creation. For example: test-edgefunctions.

Description: Optional, it supports up to 60 characters. For Example: Custom HTML page and response header.

Code: the code corresponding to the template.

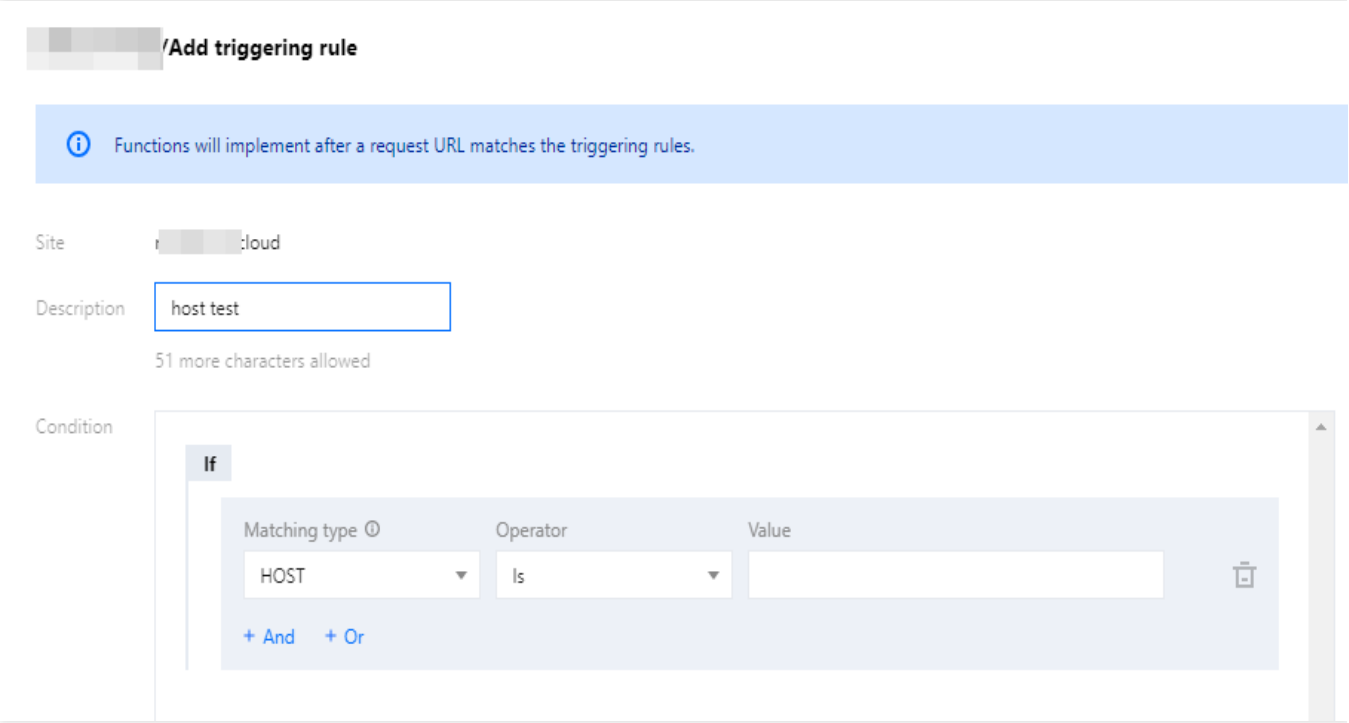
5. Click **Create and deploy**. If you see the following pop-up dialog box, the deployment is successful.



Step 2. Configuring a Triggering Rule

If you want to trigger function execution by setting HOST, URL Path, or file extensions of the matching sites, you could proceed in the following two steps:

1. After the function is deployed successfully, click **Add triggering rule**.
2. On the add triggering rule page, configure the matching conditions. Taking the current scenario as an example, you can select the Matching type as "HOST", the Operator as "Is", and the value as the added subdomain `www.example.com` . Click **OK** to create the trigger rule.



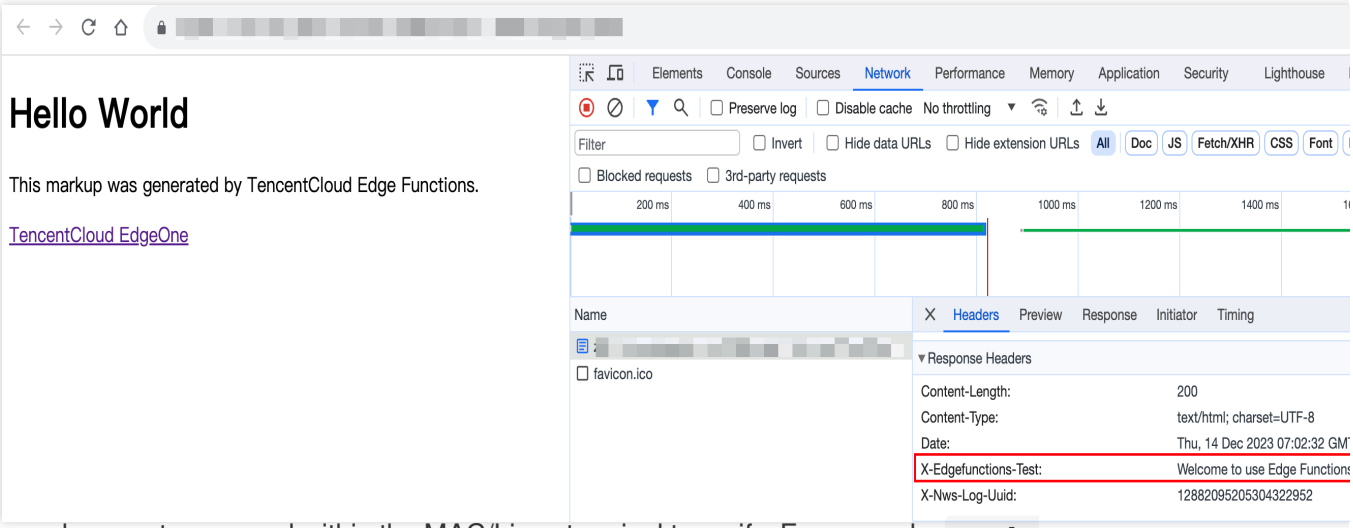
Step 3: Verifying the Edge Function

To check whether the function is working as expected, you can initiate a request in a browser or by using Curl.

Browser Verification

Curl Verification

Enter the URL in the browser, for instance: `https://www.example.com/test-edgefunctions` . This URL can match the set trigger conditions to trigger the execution of the function and view the response page information.



Run the curl request command within the MAC/Linux terminal to verify. For example: `curl https://www.example.com/test-edgefunctions` . The response can be viewed as follows:

```
➜  ~ curl -i https://[redacted]ions
HTTP/2 200
x-edgefunctions-test: Welcome to use Edge Functions.
content-type: text/html; charset=UTF-8
content-length: 238

<!DOCTYPE html>
<body>
  <h1>Hello World</h1>
  <p>This markup was generated by a TencentCloud Edge Functions.</p>
  <a href="https://cloud.tencent.com/product/teo">边缘安全加速平台 (TencentCloud EdgeOne)</a>
</body>
➜  ~
```

Operation Guide
Function Management

Last updated：2024-08-26 16:24:48

Overview

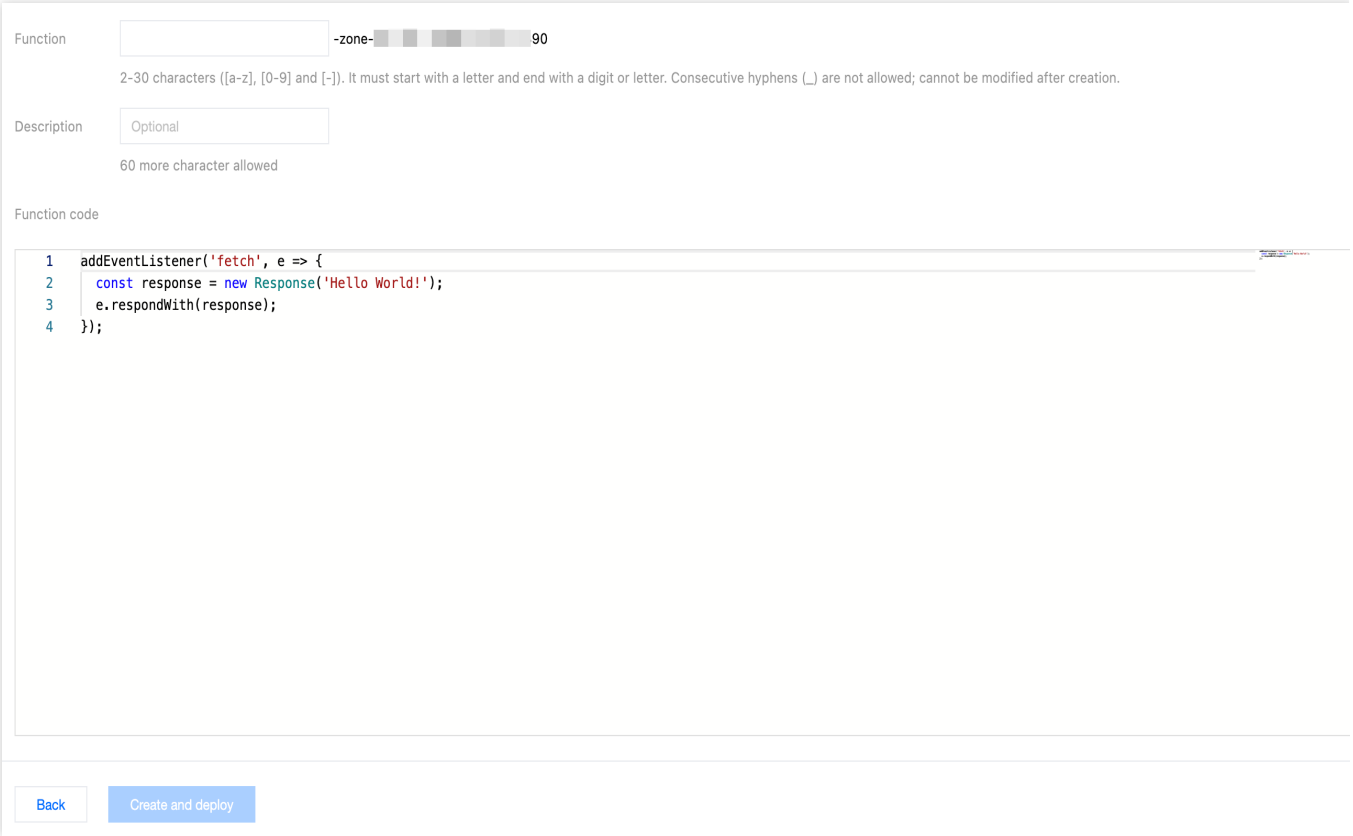
This document describes how to create, edit, and delete an edge function, and how to configure the rules that trigger the function.

Creating and Deploying a Function

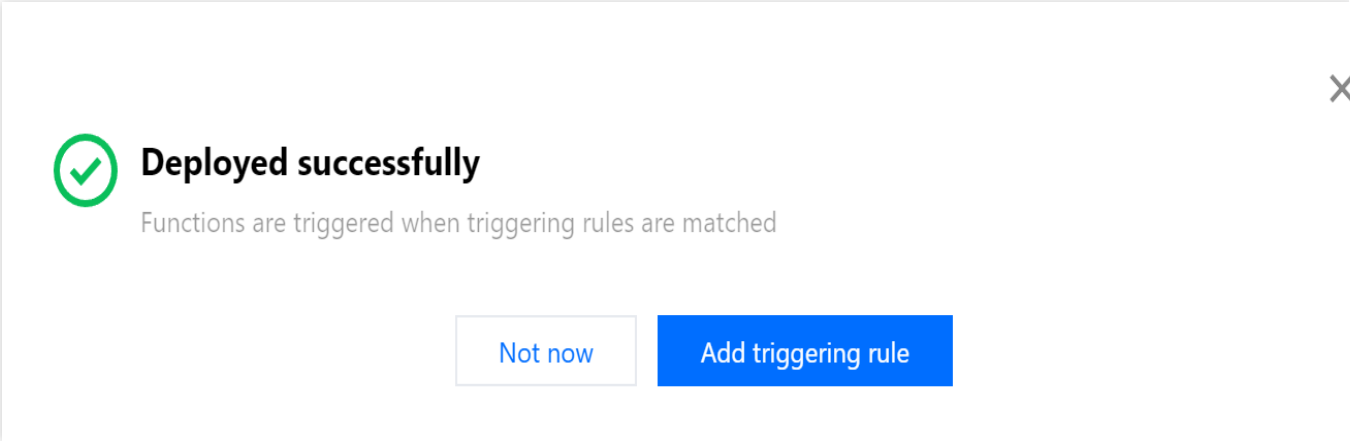
1. Log in to the EdgeOne console, click **Site List** in the left sidebar, and then click the **site** to be configured in the site list.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the edge function management page, click **Create function** and select using a template to create a function. At this step, you can use a template to create a function based on actual business needs. Taking the current scenario as an example, you can select the "Create Hello World" template to create a function. After selecting the template, click **Next**.
4. On the create new Edge Function page, configure the relevant parameters as follows:

Function: required, it can only include letters, numbers, hyphens. It must start with a letter and end with a digit or letter, 2-30 characters; it cannot be modified after creation. Example: test-edgefunctions.
Description: optional, it supports up to 60 characters. For instance, custom HTML page and response headers.
Code: the code corresponding to the template.

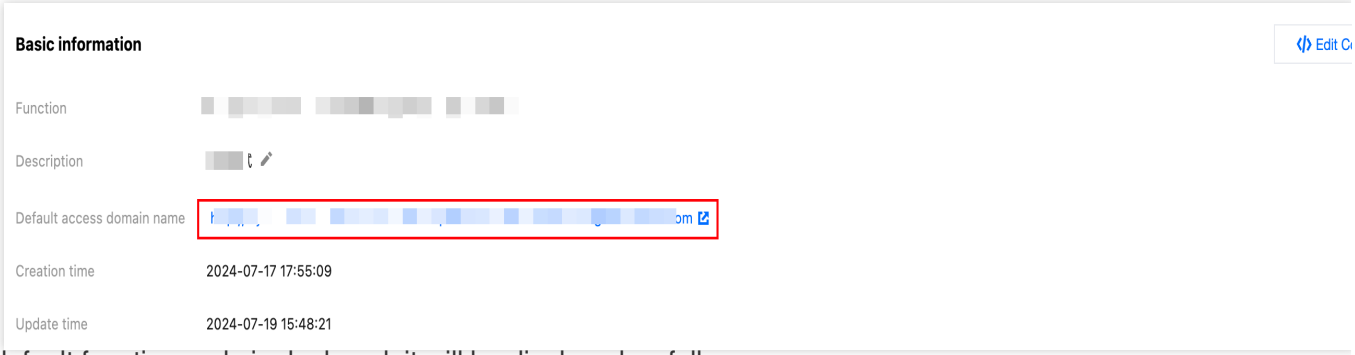
©2013-2025 Tencent Cloud International Pte. Ltd.



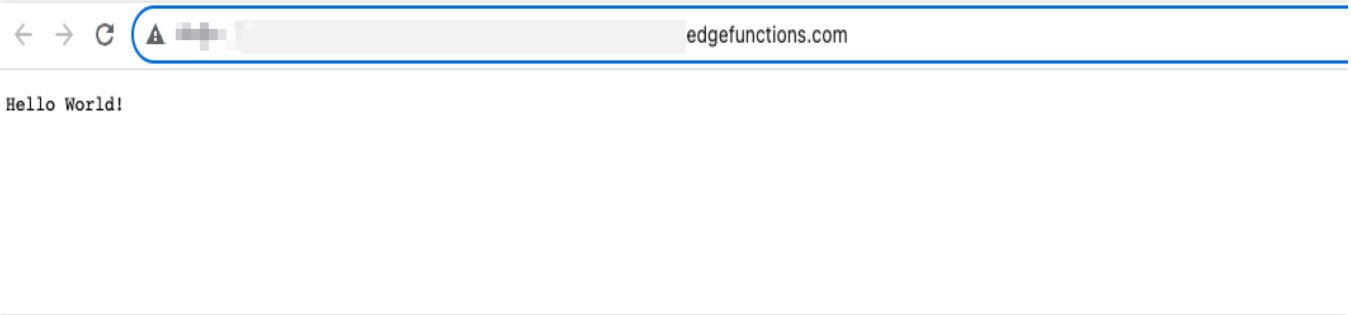
5. Click **Create and deploy**. If a dialog box appears as shown, it signifies the successful deployment.



After the deployment is successful, you can click **Default access domain name** assigned by the platform to verify its effectiveness by triggering the function execution.



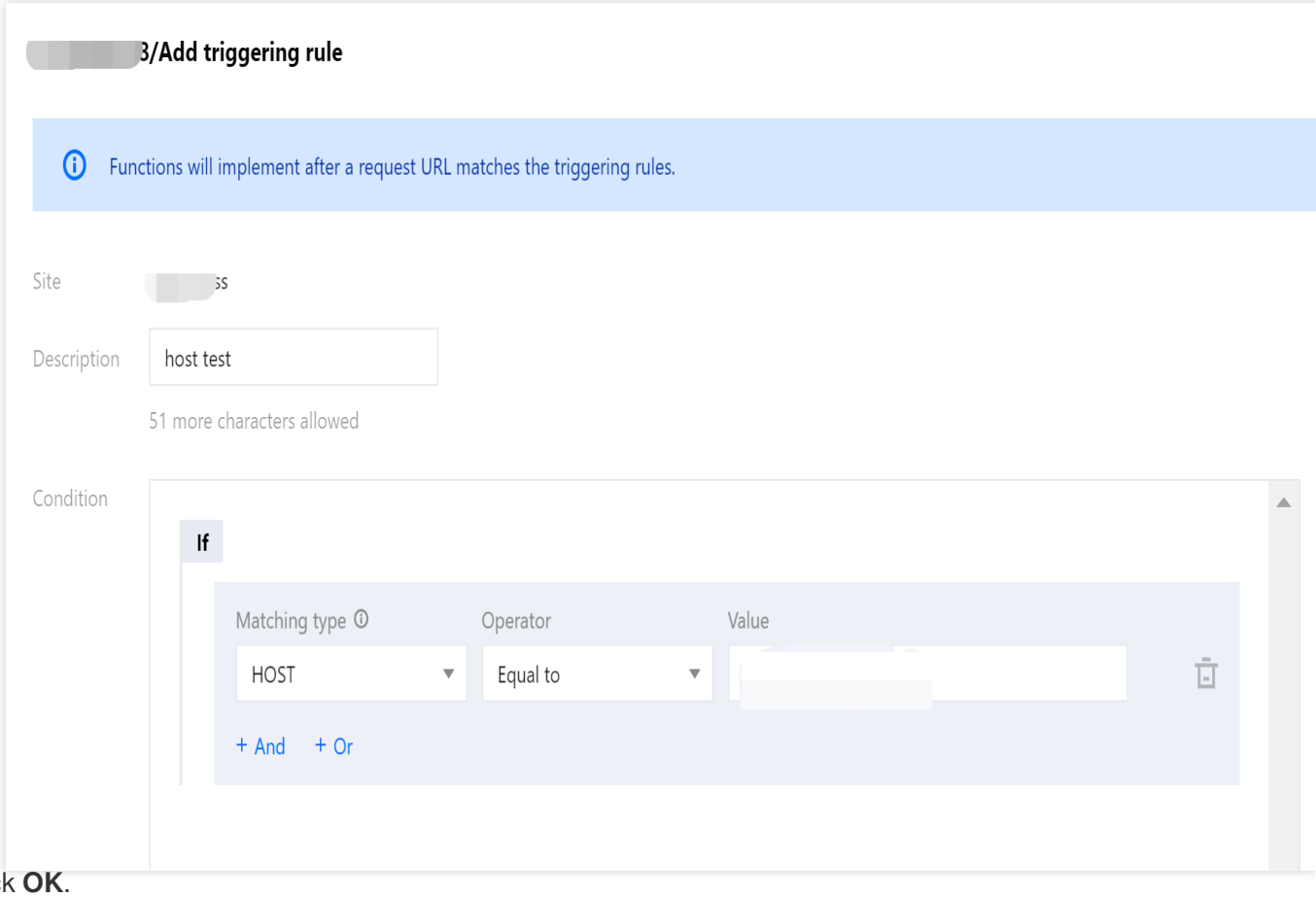
If the default function code is deployed, it will be displayed as follows:



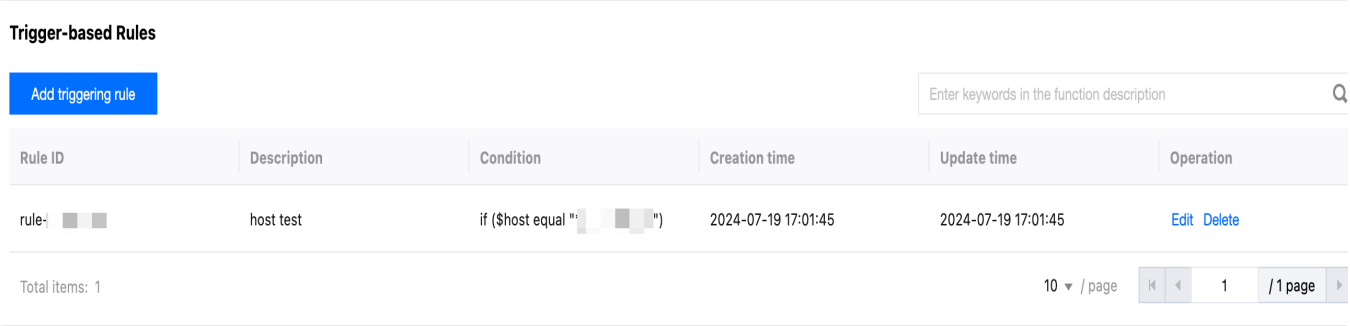
Configuring a Triggering Rule

If you want to trigger function execution by setting HOST, URL Path or file suffix of the matching site, there are two steps you can follow:

1. After the function is deployed successfully, click **Add triggering rule**.
2. On the **Add triggering rule** page, specify the matching type, operator, and value as needed.

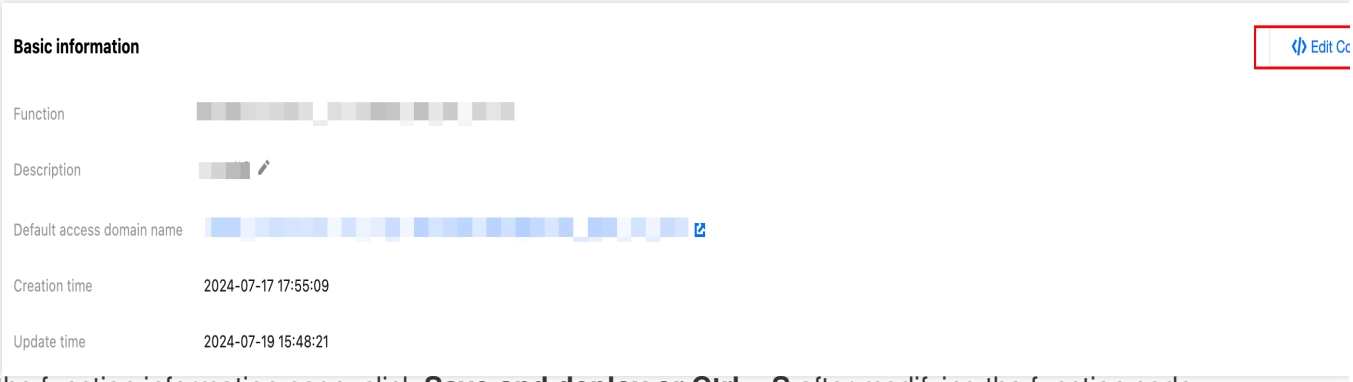


3. Click **OK**.

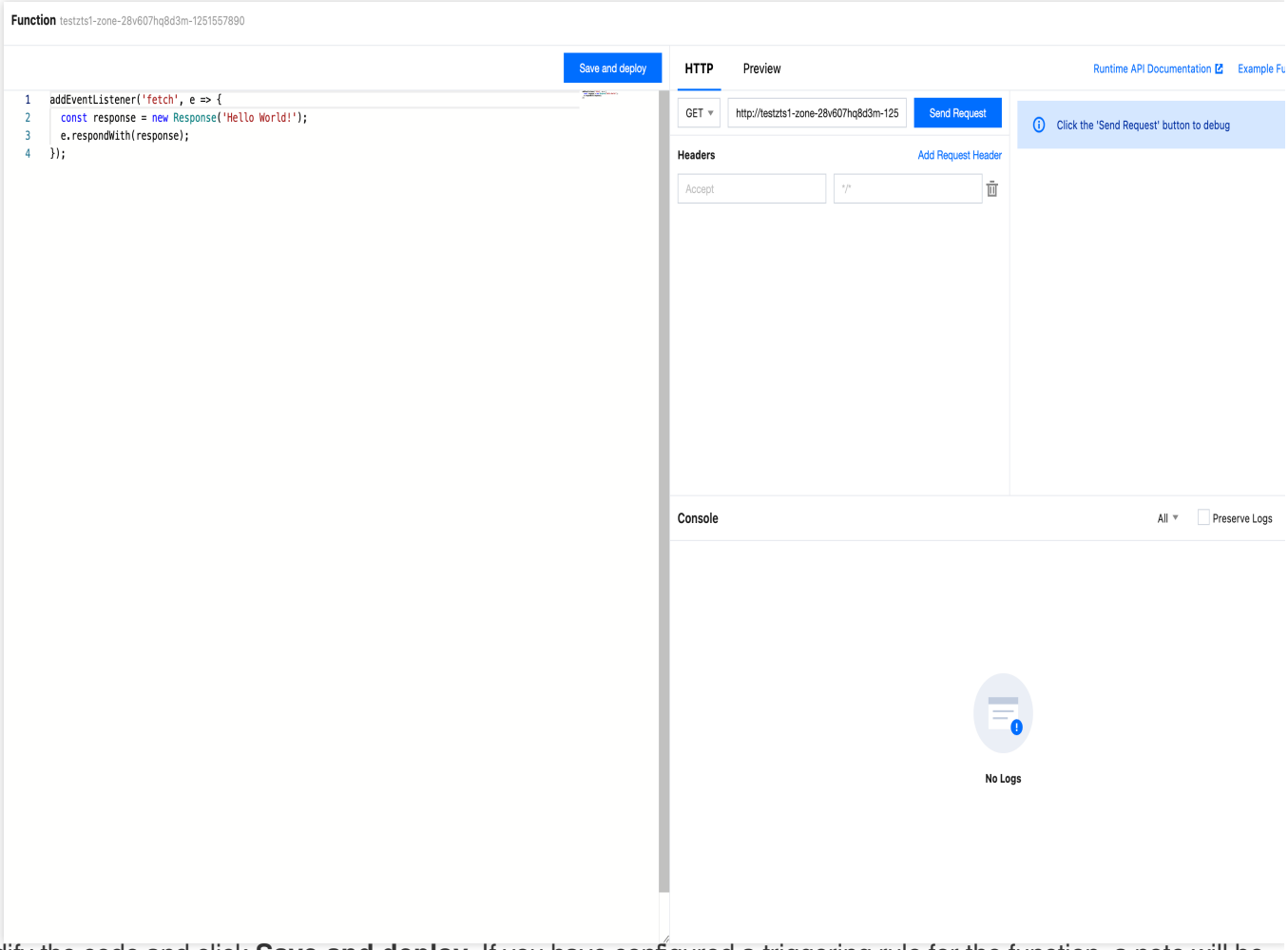


Editing the Function

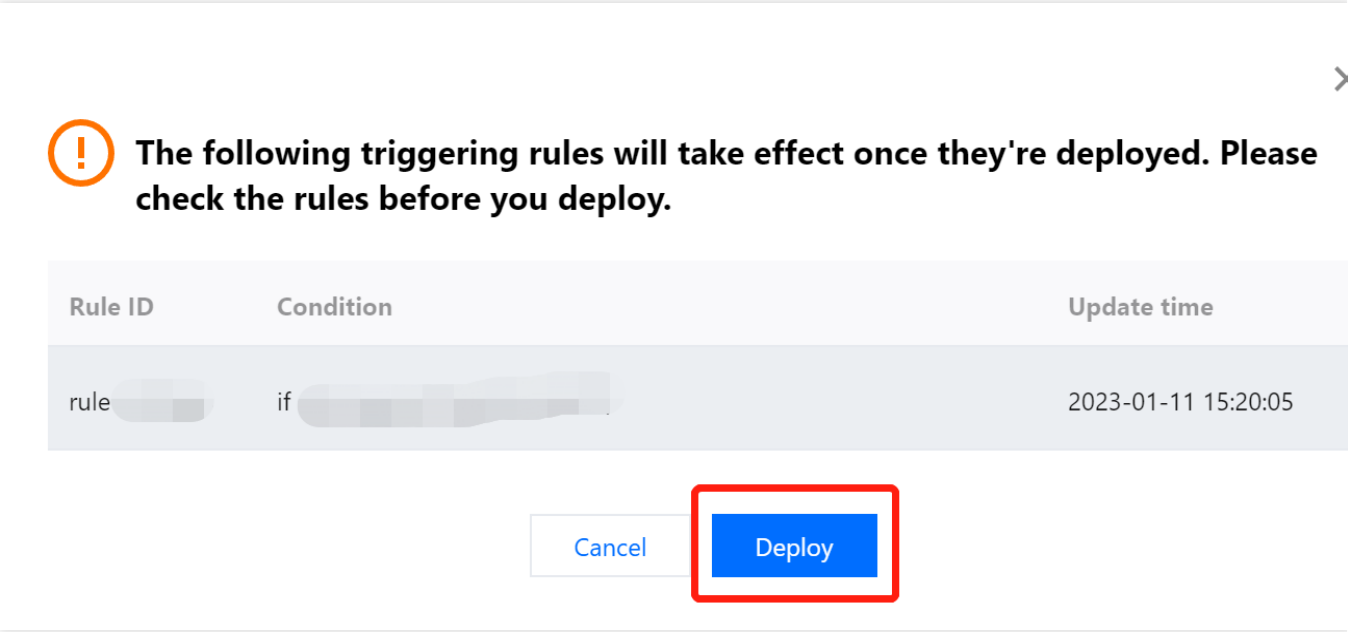
1. On the function management page, select the function you want to modify and click the **function name**. On the basic information page of the function details, click **Edit Code**.



2. On the function information page, click **Save and deploy or Ctrl + S** after modifying the function code.

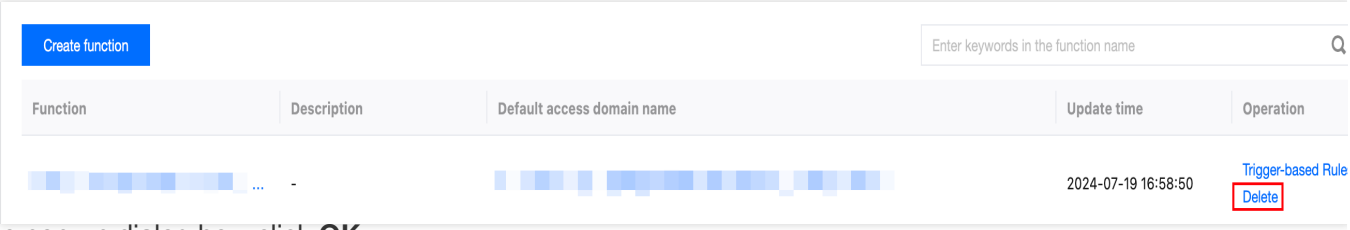


3. Modify the code and click **Save and deploy**. If you have configured a triggering rule for the function, a note will be displayed as follows:



Deleting the Function

1. If you need to delete a newly created function, you can go to the function management page, select the function you want to delete, and click the **Delete** button in the operation column.



2. In the pop-up dialog box, click **OK**.

Note

Once deleted, the function cannot be restored. The triggering rules of the function will also be deleted.

Web Debugging

Last updated：2024-08-26 10:34:34

The web debugging page is mainly used to debug the execution results of edge functions. You can enter this debugging page by clicking **Edit Code** within the edge function.

Directions

1. Log in to the EdgeOne console, click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the function management page, click the **specific function name** and then click **Edit Code** to enter the web debugging page. If no function has been created, refer to the guide on creating and deploying a function to create a function first.
4. After adjusting the code, click **Save and deploy** to quickly deploy the adjusted code to the current website. Then you can experience the features of the current website such as console debugging, HTTP request and response debugging, and request preview. For detailed operations, see Supported Web Debugging Features.

Note:
To effectively perform the console debugging, HTTP request and response debugging, and request preview operations, you must ensure that the default domain name of the function has taken effect, or the function has been configured with triggering rules. Please perform the above operations after confirming that the default domain name of the function has taken effect or the function triggering rules have been configured.

Supported Web Debugging Features

Web debugging supports the following debugging operations of the current website:
Console debugging: Supports printing the console messages, including LOG, INFO, NOTICE, DEBUG, WARN, and ERROR messages.
LOG: Outputs general messages about program status, variable values, and so on, to help developers get real-time information on program execution.
INFO: Outputs prompt messages about operation success, data load completion, and so on, to provide positive feedback for developers.
DEBUG: Outputs debugging messages, to help understanding the details of program execution.
WARN: Outputs warning messages, to describe program issues that are not errors but worth attention.
ERROR: Outputs error messages, to record the program execution errors that should be resolved by developers.

©2013-2025 Tencent Cloud International Pte. Ltd.

NOTICE: Outputs notice messages at a level between INFO and WARN, to remind developers of important but not urgent situations.

HTTP request and response debugging: Supports adding HTTP request headers and sending HTTP requests, and enables users to quickly view the response content, including the response status, response header, and response body.

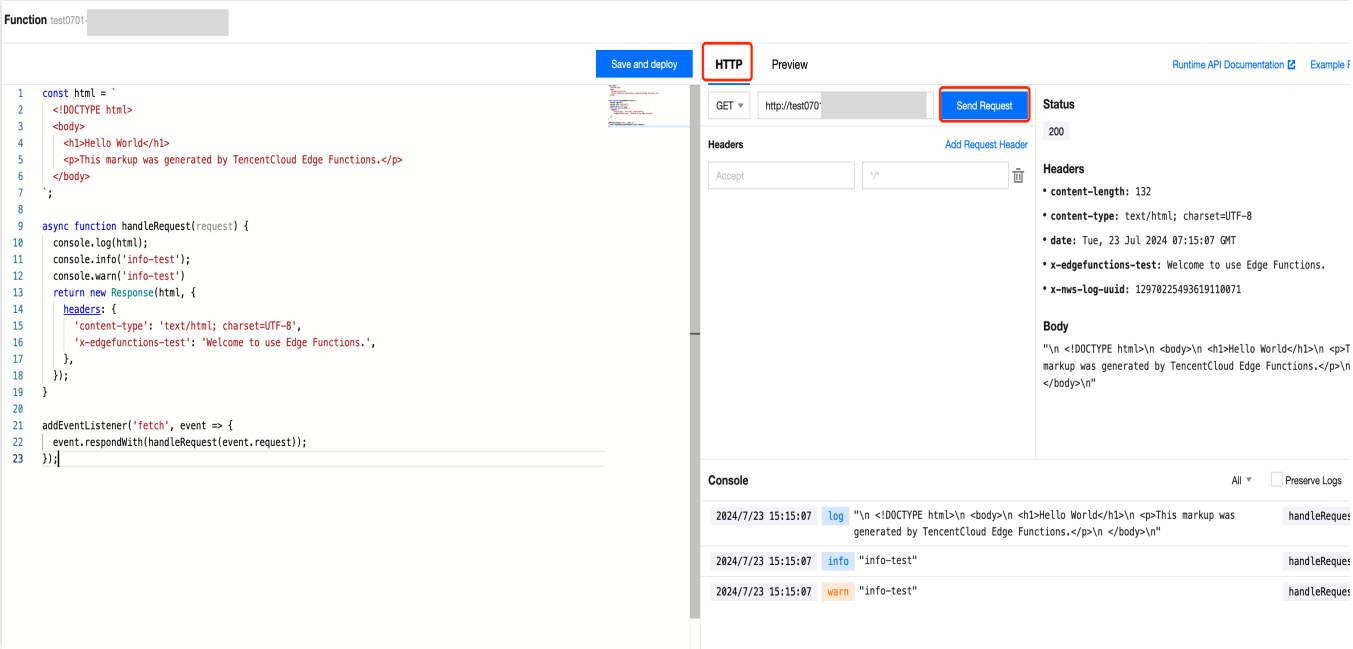
Custom HTTP headers: Allows users to add or modify HTTP request headers to simulate different request scenarios.

Instant response preview: Enables users to view the HTTP response, including the response status code, response header, and response body immediately after a request is sent.

Request preview: Supports viewing the response body content of the request results, and enables users to intuitively preview and analyze the corresponding data in the console.

Console Debugging

On the **HTTP** page, you can select a request method from GET, POST, PUT, PATCH, DELETE, HEAD, and OPTIONS, and click **Send Request** to print the console messages in the code. The types include LOG, INFO, NOTICE, DEBUG, WARN, and ERROR.

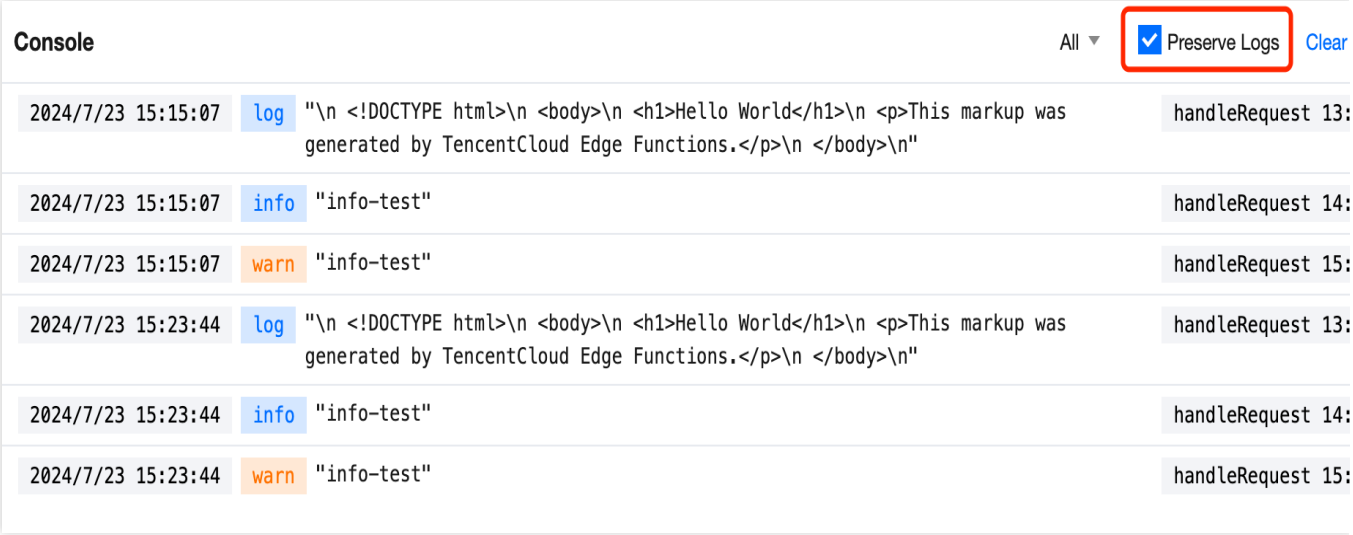


Filtering Console Log Types

On the **HTTP** page, select **All** in the dropdown list. It supports filtering a specific type of console messages, as shown in the figure below:

Preserving All Debugging Logs

On the **HTTP** page, check **Preserve Logs** to preserve the console messages after clicking Send Request each time, as shown in the figure below:



Clearing Console Logs

On the **HTTP** page, click **Clear Logs** to clear the console messages already printed currently.

Console message composition:

Time: time point when the console message is printed, e.g., 2024/3/25 20:01:42.

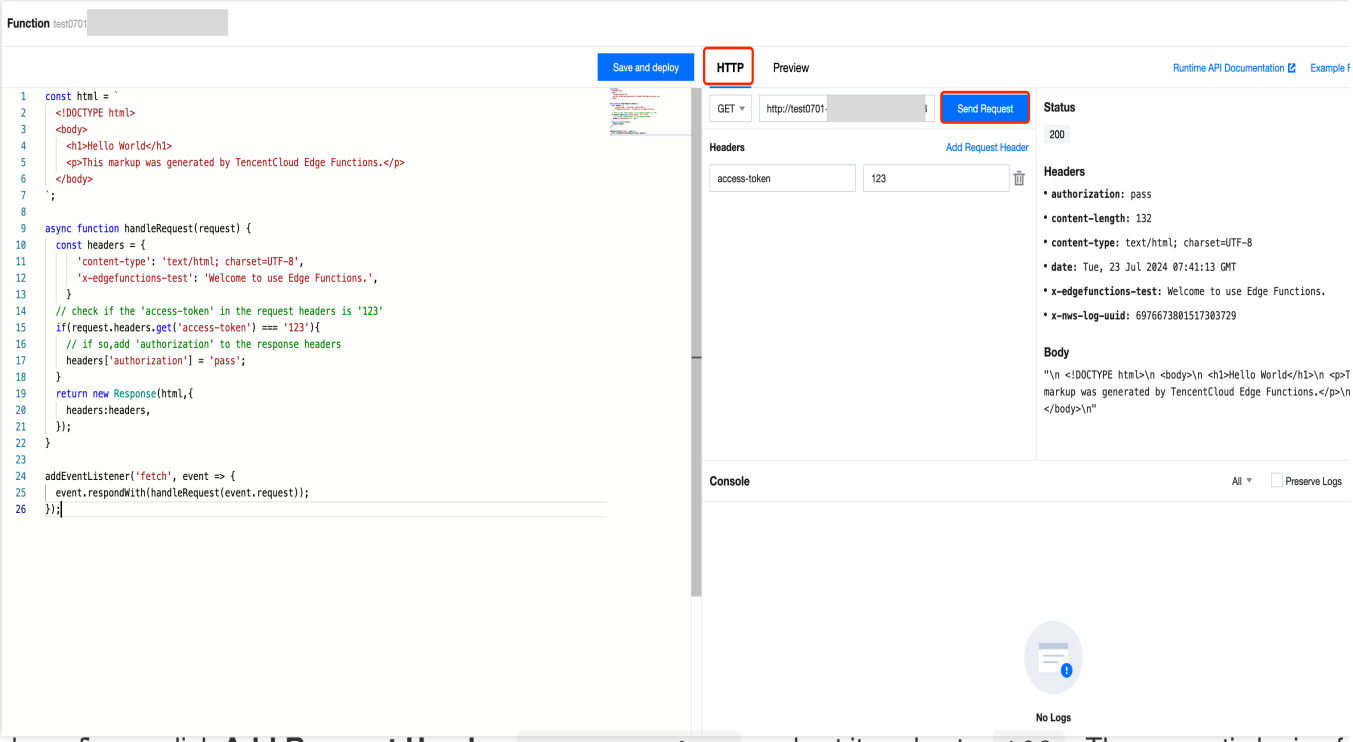
Type: type of the console message, including LOG, INFO, NOTICE, DEBUG, WARN, and ERROR.

Content: printed content for a specific type of console message, such as info-test.

Function name and position: Specific function to which the console message belongs, and its location (row and column) in the code. For example, `handleRequest 11:12` indicates that the console message belongs to the handleRequest function and is located at row 11 and column 12 in the code.

HTTP Request and Response Debugging

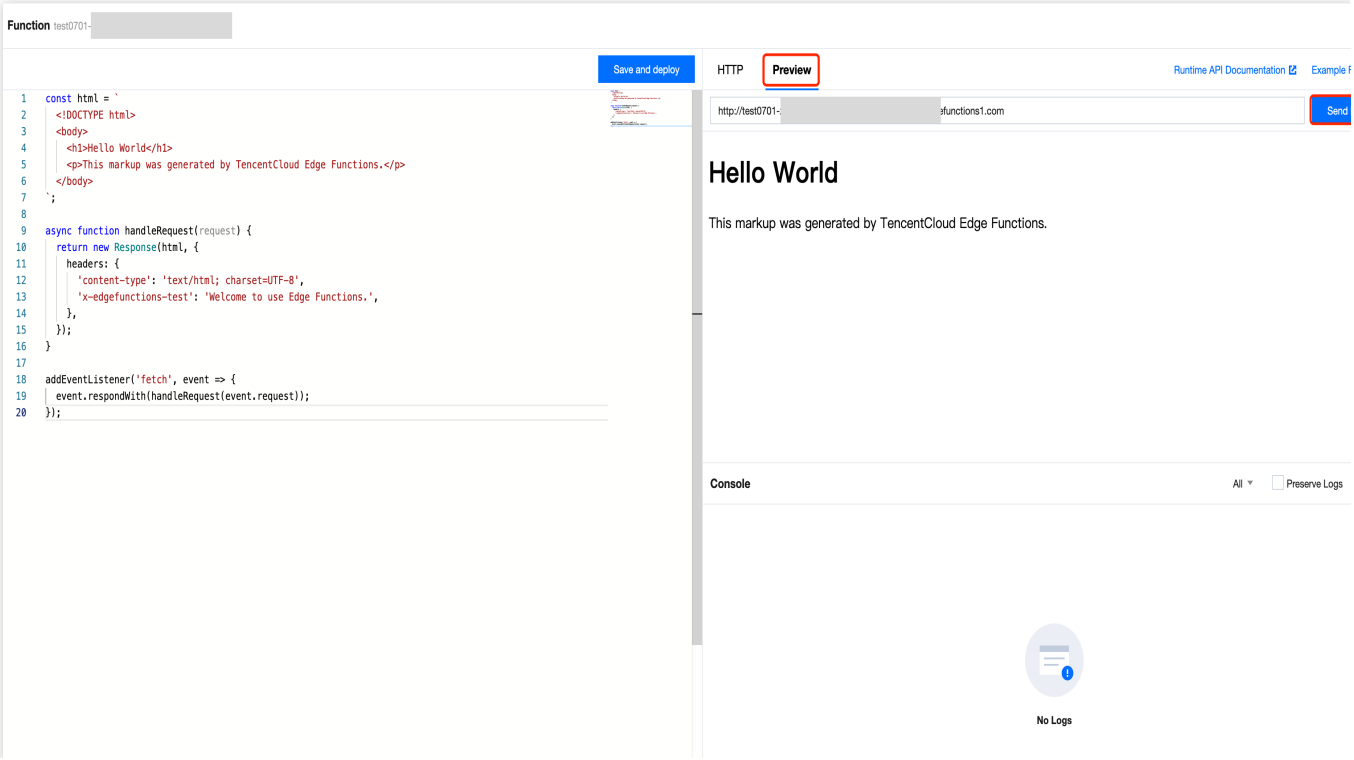
On the **HTTP** page, add the headers required for testing based on the scenario, and click **Send Request** to get the response status code, response header, and response body.



In the above figure, click **Add Request Header** `access-token` and set its value to `123`. The semantic logic of the code is to determine whether the request header contains `access-token` with the value `123`, and if yes, return the response header `authorization` with its value set to `pass`.

Request Preview

On the **Preview** page, click **Send Request** to directly render and preview the content of the response body.


Function test0701-
Save and deploy
HTTP
Preview
Runtime API Documentation
Example f
http://test0701-
functions1.com
Send
const html = `
<!DOCTYPE html>
<body>
<h1>Hello World</h1>
This markup was generated by TencentCloud Edge Functions.
</body>
`;
async function handleRequest(request) {
return new Response(html, {
headers: {
'content-type': 'text/html; charset=UTF-8',
'x-edgefunctions-test': 'Welcome to use Edge Functions.',
},
});
}
addEventListener('fetch', event => {
event.respondWith(handleRequest(event.request));
});
Hello World
This markup was generated by TencentCloud Edge Functions.
Console
All
Preserve Logs
No Logs

Function Trigger

Last updated：2023-12-14 17:47:37

Scenarios

This document guides you how to manage function triggering rules.

Add, delete, modify and search for a triggering rule.

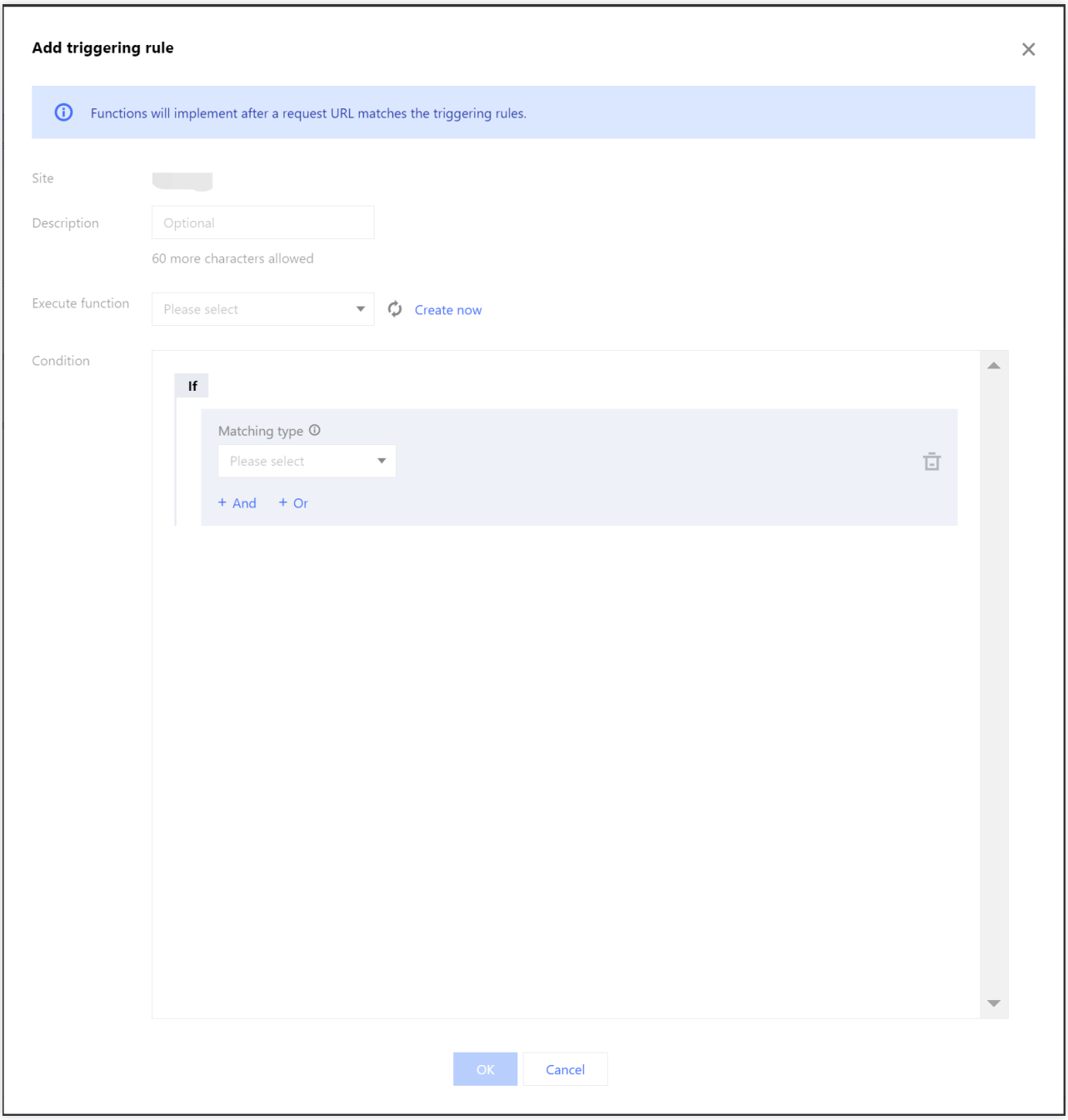
Adjust the priority of the triggering rules by reordering them in the list. If the request URL matches multiple triggering rules, only the first hit rule is executed.

Directions

Creating a triggering rule

1. Log in to the EdgeOne console and click **Site List** in the left sidebar. In the site list, click the target site.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the **Function Trigger** page, click the

icon to the right of the rule list.



Parameter description:

Site: The name of the current site is displayed by default.

Description: (Optional) It can contain up to 60 characters.

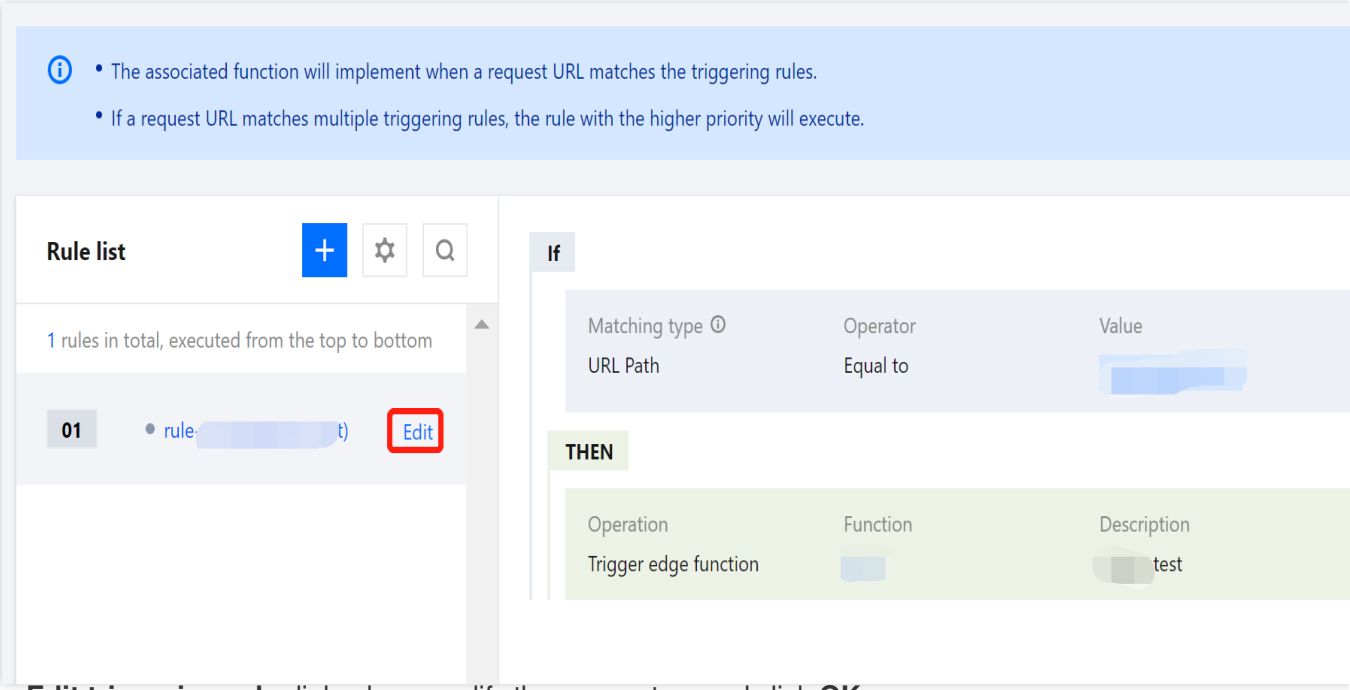
Condition: Specify the matching type, operator, and value as needed. For more information, see Rule Engine.

Execute function: Select a function from the drop-down list.

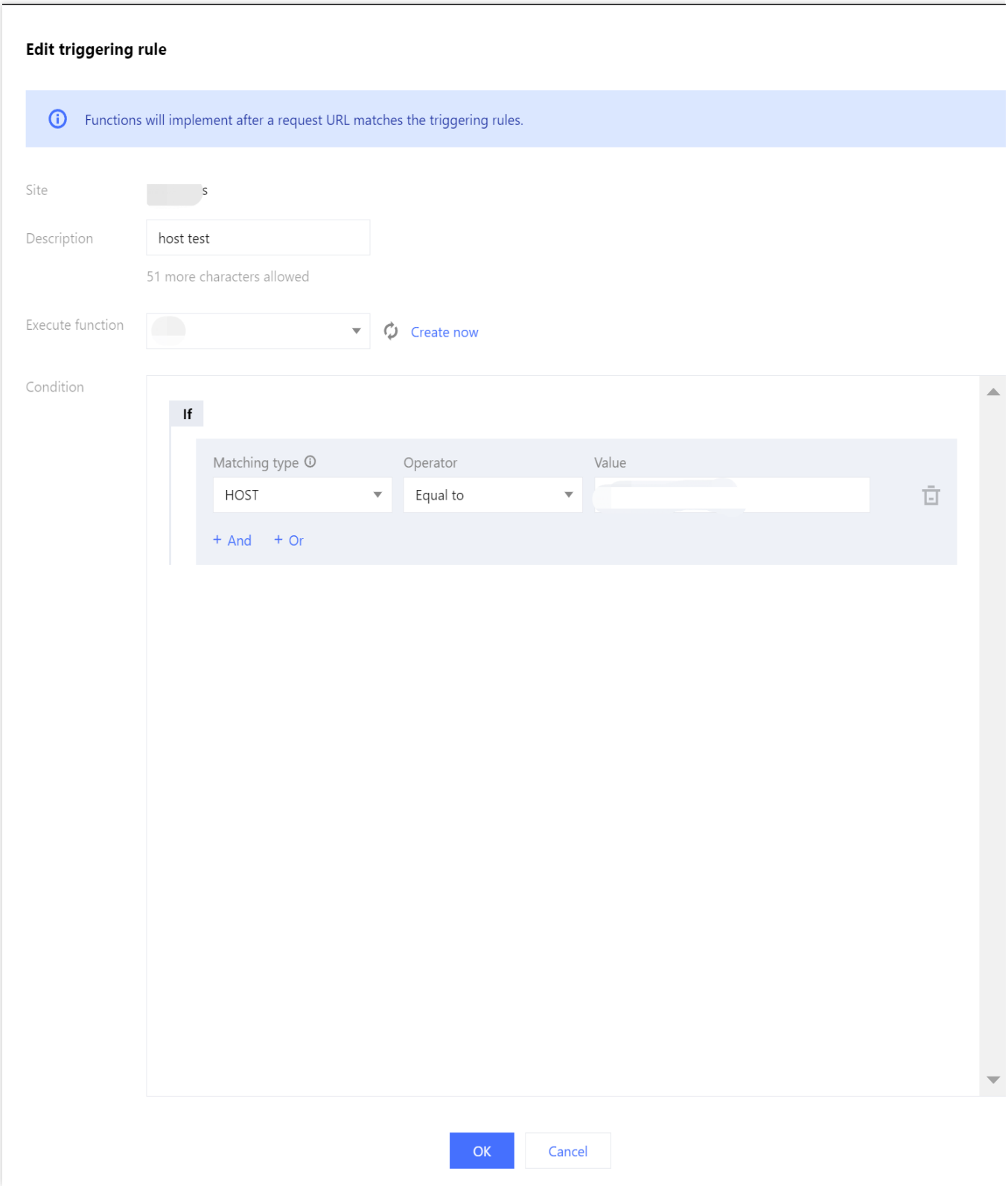
4. Click **OK** to complete the creation.

Editing the triggering rule

1. On the **Function Trigger** page, find the rule and click **Edit**.



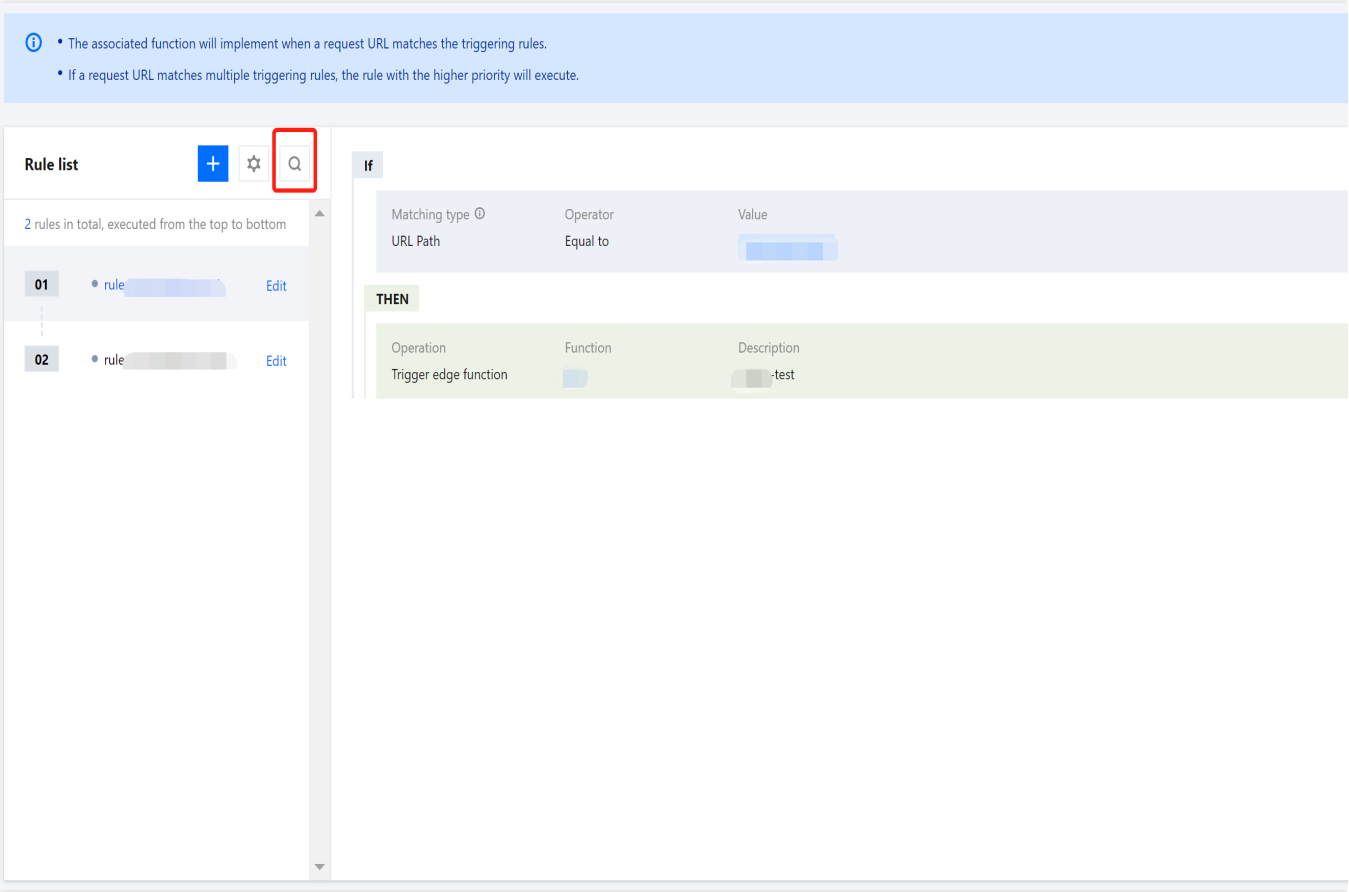
2. In the **Edit triggering rule** dialog box, modify the parameters and click **OK**.



Searching for the triggering rule

On the **Function Trigger** page, click the

icon to the right of the rule list and enter a keyword of the rule ID in the search box to search for the triggering rule.



Deleting the triggering rule

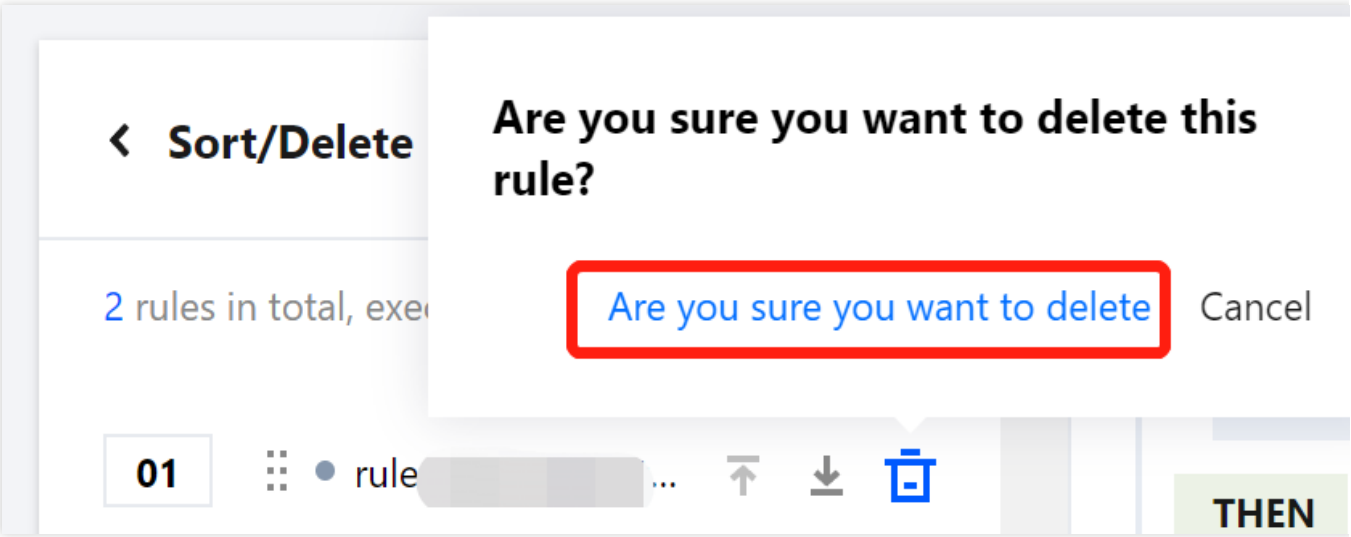
1. On the **Function Trigger** page, click the

icon to the right of the rule list.

2. Find the rule to delete and click the

icon.

3. In the pop-up dialog box, click **Confirm to delete**.



Adjusting the priority of the triggering rule

1. On the **Function Trigger** page, click the

icon to the right of the rule list.

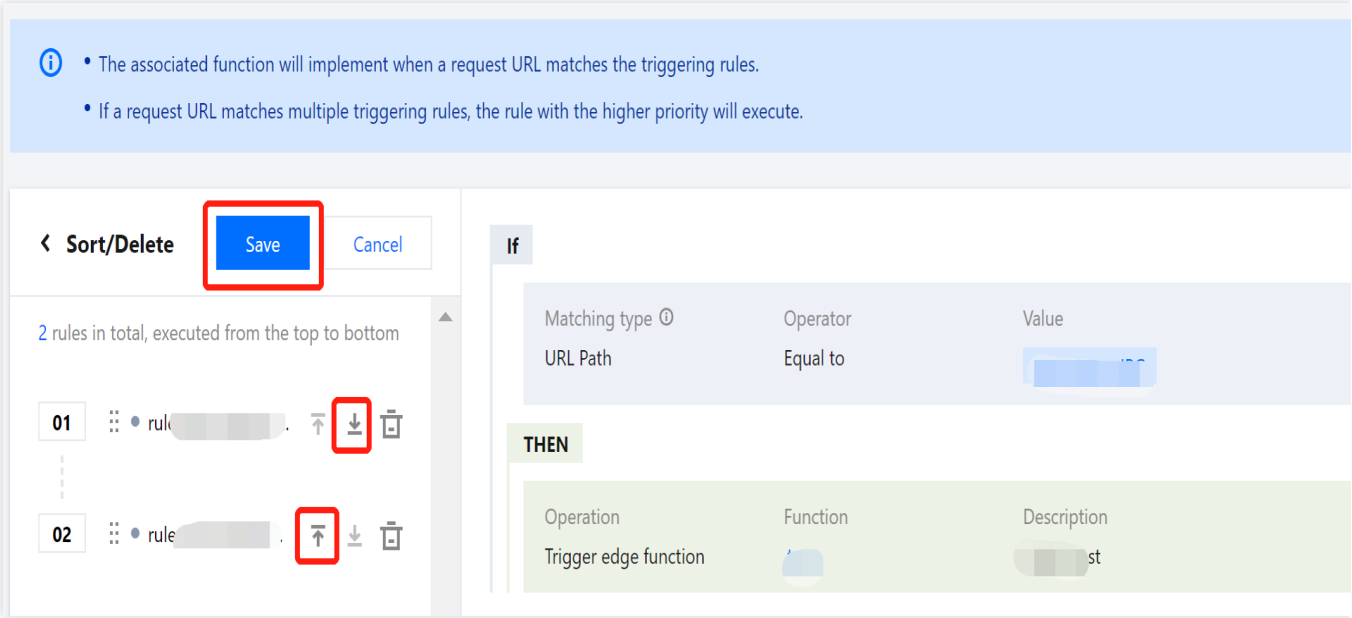
2. Find the rule. Click the

icon to move the rule up by one row or the

icon to move the rule down by one row. Then, click **Save**.

Reminder

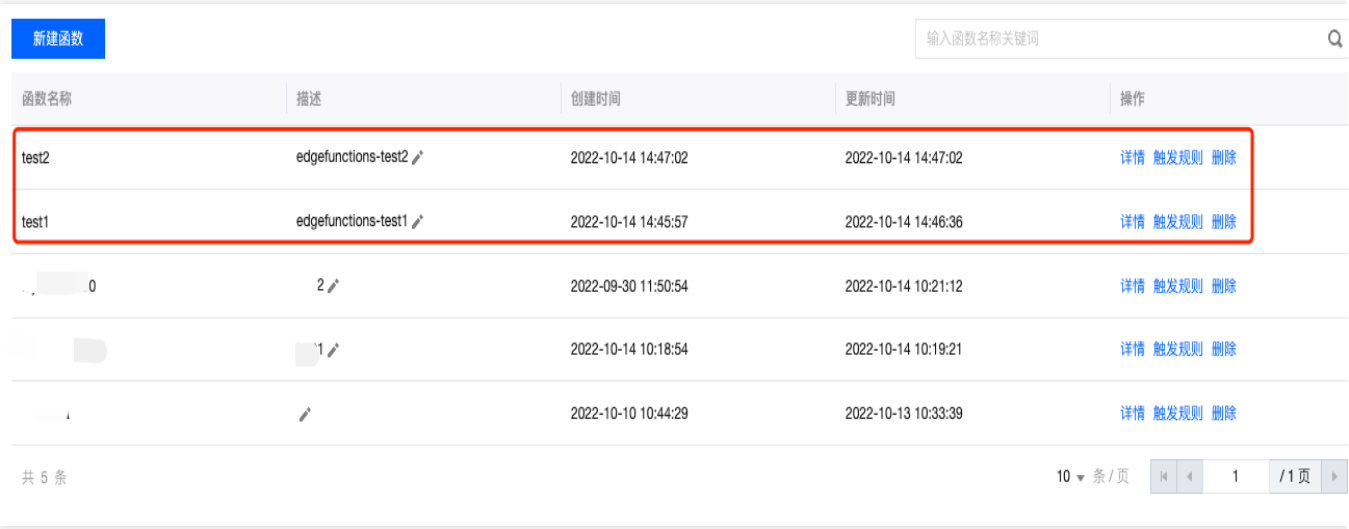
If the request URL matches multiple triggering rules, such as Rules 01 and 02 in the following figure, only the triggering rule on the top will be executed. In this example, only Rule 01 will be executed.



Use Cases

The following section describes how to adjust the execution order of multiple triggering rules that match the request URL:

1. On the Function Management page, two functions with the same triggering rule are already created.



Code of the `test1` function:

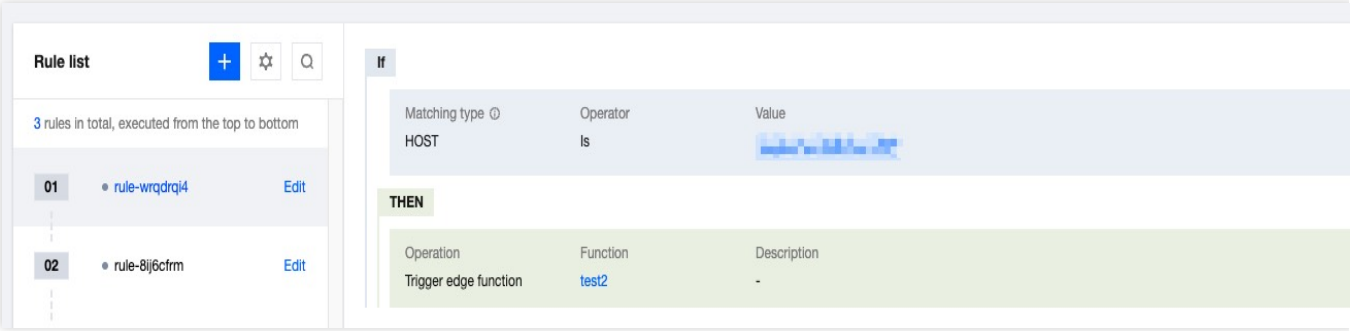
```
const html = `
  <!DOCTYPE html>
  <body>
 <h1>The test 1, Hello World</h1>
 <p>This markup was generated by a TencentCloud Edge Functions.</p>
 <a href="https://cloud.tencent.com/product/teo">Tencent Cloud EdgeOne</a>
```

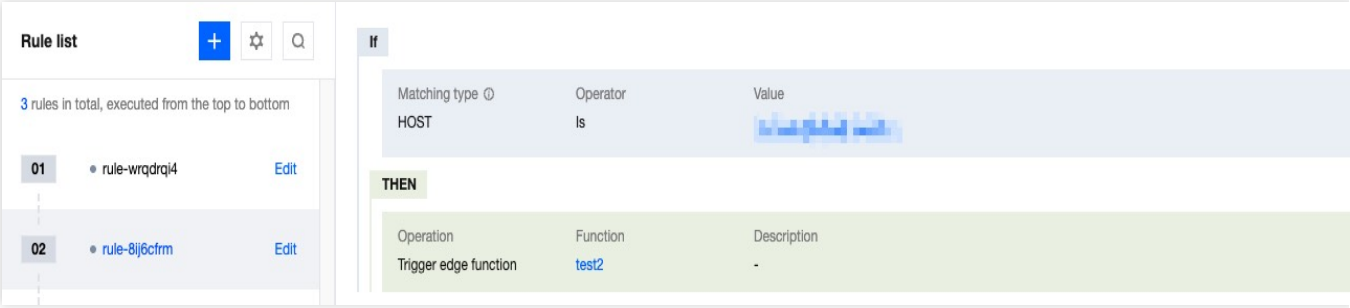
```
  </body>
`;

async function handleRequest(request) {
  return new Response(html, {
 headers: {
   'content-type': 'text/html; charset=UTF-8',
    'x-edgefunctions-test': 'Welcome to use Edge Functions.',
 },
  });
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Code of the `test2` function:

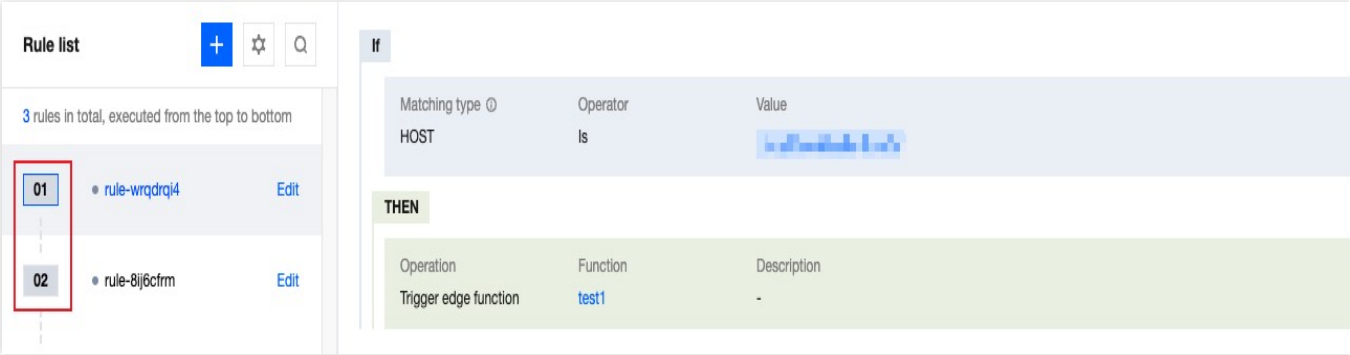
```
const html = `
  <!DOCTYPE html>
  <body>
 <h1>The test 2, Hello World</h1>
 <p>This markup was generated by a TencentCloud Edge Functions.</p>
 <a href="https://cloud.tencent.com/product/teo">Tencent Cloud EdgeOne</a>
  </body>
`;

async function handleRequest(request) {
  return new Response(html, {
 headers: {
   'content-type': 'text/html; charset=UTF-8',
    'x-edgefunctions-test': 'Welcome to use Edge Functions.',
 },
  });
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

2. The triggering rules are displayed on the **Function Trigger** page, as shown in the following figure.





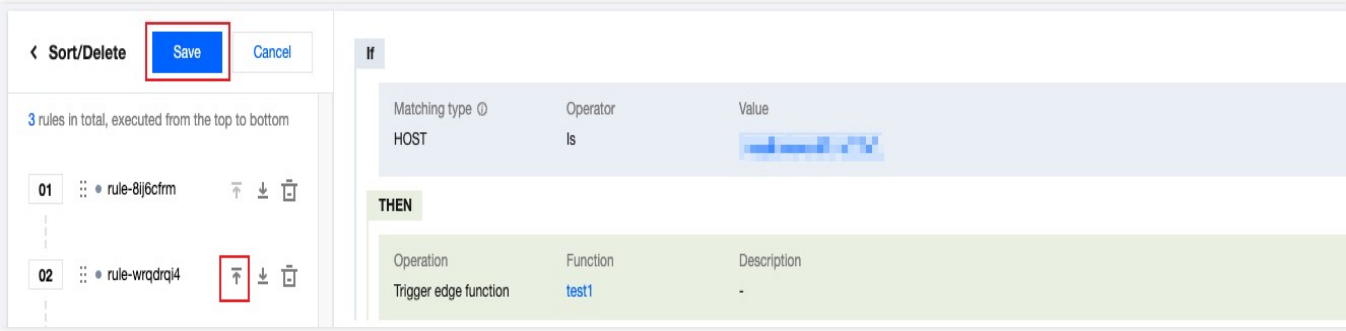
3. The triggering rule of the `test1` function is listed in position 01, and that of the `test2` function is in position 02, as shown in the following figure.



4. Enter the URL that contains the triggering rule in the address bar of a browser and press Enter. The following results are returned.

5. Click the

icon of the `test2` function and then click **Save**.



6. The triggering rule of the `test2` function is adjusted to position 01, and that of the `test1` function is adjusted to position 02.
7. Enter the URL that contains the triggering rule in the address bar of a browser and press Enter. The response is as below:

Environment Variable

Last updated：2024-08-23 15:48:33

Overview

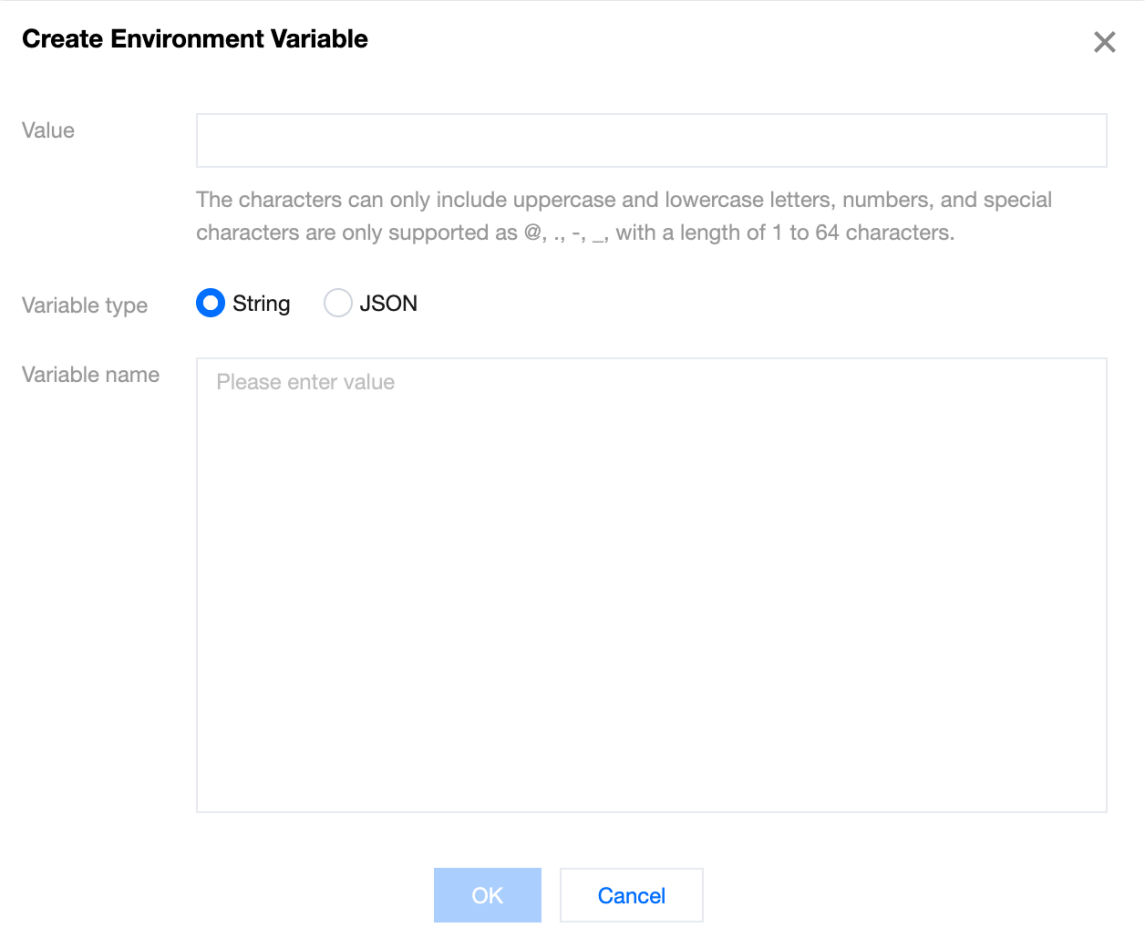
Environment variables are key-value pairs set in the runtime environment of edge functions, which can be accessed as global variables in edge function scripts without the need for explicit importing or initialization. By using environment variables, function code can be decoupled from configuration. Their common uses are as follows:
Configuration decoupling: Environment variables allow developers to configure edge functions without modifying the function's code, meaning developers can set different variable values for different environments (such as development, testing, and production) to control function behavior.
Security: By storing sensitive information (like API keys) in environment variables instead of hardcoding them into the code, application security is enhanced. This approach reduces the risk of leakage of sensitive information in the code repository and simplifies key management.
Flexibility: Environment variables provide great flexibility for edge functions, allowing developers to adjust the behavior of edge functions as needed. For example, developers can use environment variables to achieve grayscale release by controlling different user groups to access different versions of code or configuration.

Directions

1. Log in to the EdgeOne console, and click **Site List** in the left menu bar. Subsequently, in the Site List, click the **Site** you want to configure.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the Function Management page, click the **specific function name**. At the bottom of the page, find the **Environment Variables** module. If the function has not been created, follow the Creating and Deploying a Function guide to complete the function creation first.

Creating and Deploying Environment Variables

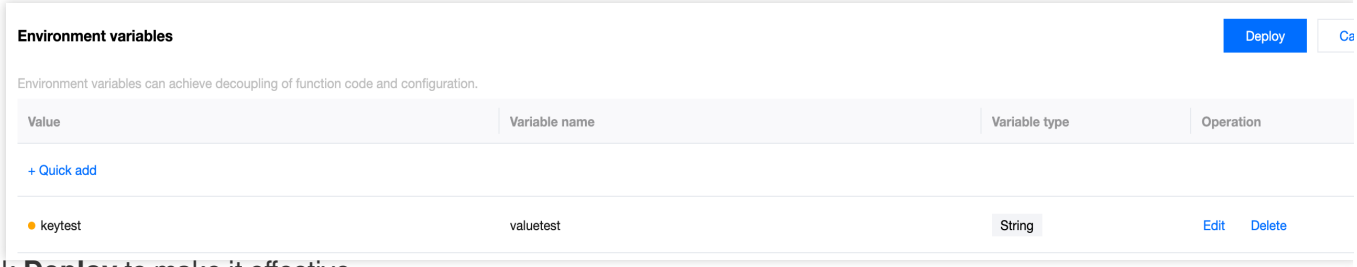
1. In the Environment Variables module, click **Quick add** and configure the relevant parameters.



Parameter Name	Description
Variable name	Required. It can only include uppercase and lowercase letters, digits, and special characters limited to `@, ., -, ,` . The length is restricted to 1 to 64 characters. The variable name cannot be duplicated and cannot be modified after creation. For example, keytest.
Variable type	Required. It supports String and JSON types. String: After this option is selected, the input variable value content will be saved as a string. For specific usage, refer to the Edge Functions Referencing Environment Variables section for the variable type String. JSON: After this option is selected, the input variable value content will be saved as a JSON array. The edge functions will automatically parse the variable value into a JavaScript object without the need to manually call JSON.parse(). For specific usage, refer to the Edge Functions Referencing Environment Variables section for the variable type JSON.
Value	Required. It supports a maximum of 5 KB. For example, if the type is String, enter valuetest as the variable value. If the type is JSON, the variable value will be validated to check if the input content follows the JSON data structure. If it does not, there will be an exception prompt.

©2013-2025 Tencent Cloud International Pte. Ltd.

2. Click **OK** to complete the creation of the environment variable. A single edge function supports creating up to 64 environment variables.



3. Click **Deploy** to make it effective.

Edge Functions Referencing Environment Variables

If the environment variables referenced by the function do not contain special characters `@, ., -, ,` , they can be referenced in the form of env.envname. For example, if the environment variable envname is `keytest` , the way to reference it in edge function code is `env.keytest` . For specific usage, refer to the section for the variable type String.

If the name of the environment variable referenced by the function contains special characters `@, ., -, ,` , it can be referenced in the form of env['envname']. For example, if the environment variable envname is `test-@.-a` , the way to reference it in edge function code is `env['test-@.-a']` .

Below are sample codes for edge functions referencing environment variables of types String and JSON. Developers can adjust them according to actual situations.

Variable Type: String

Through the above steps to create and deploy environment variables, create an environment variable named `keytest` with the value `valuetest` . The edge function reference is as follows:

```
// Entry function
addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});

// Function to handle requests
async function handleRequest(request) {
  // Obtain a value from an environment variable, and this environment variable nee
  const valueFromEnv = env.keytest;
  // Create response.
  const response = new Response(valueFromEnv, {
    headers: {
      'Content-Type': 'text/plain' // Set the response Content-Type.
    }
  });
```

```
  // Return the response.
  return response;
}
```

The semantic summary of the above code is to obtain the environment variable that the function has created and deployed, and respond it to the client as text.

Deploy the code, and you can view the result by accessing the trigger rules or default access of the function.

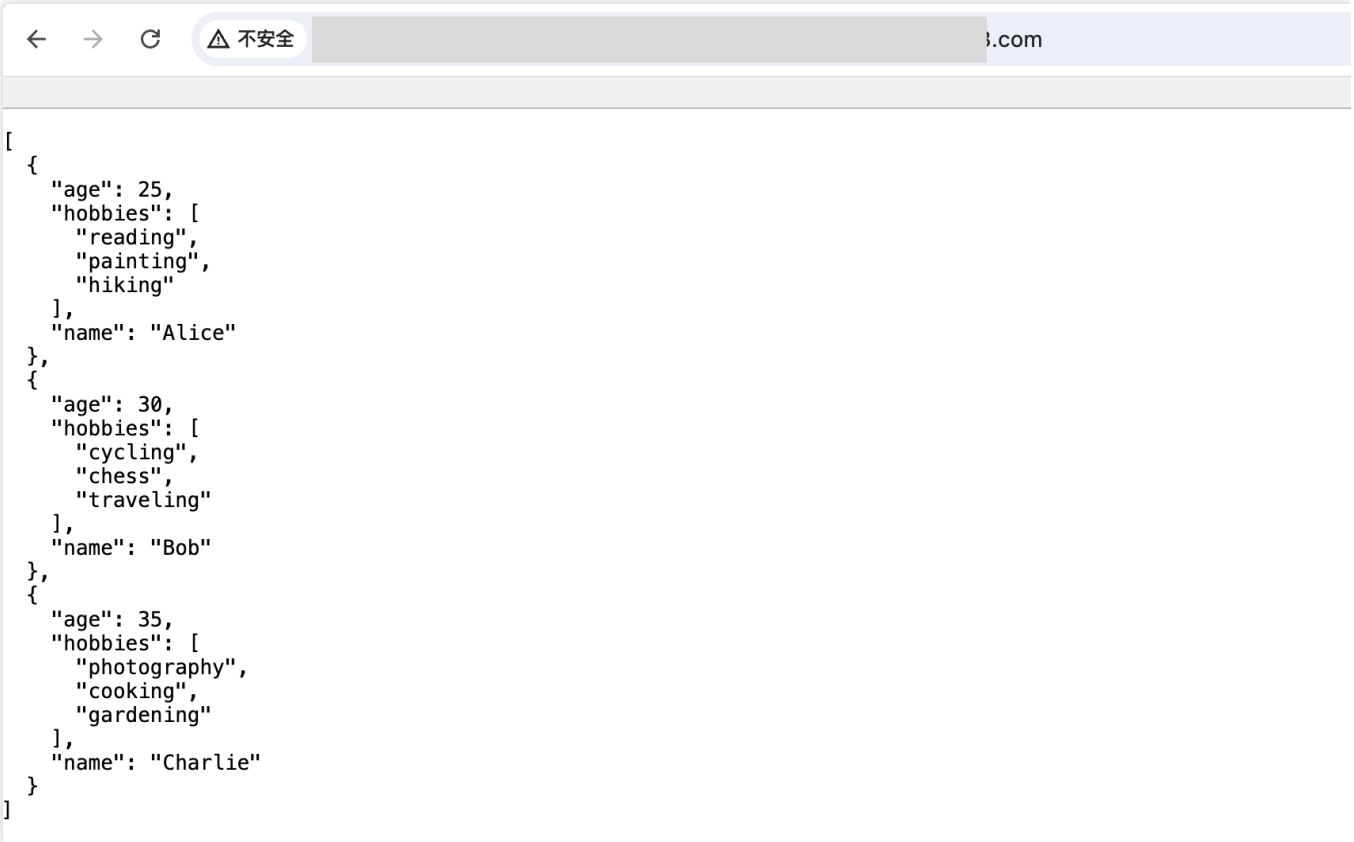
Variable Type: JSON

Repeat the steps to create and deploy environment variables, and create and deploy an environment variable of type JSON named `keytestjson` , as shown below:

```
 // Entry function
addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});

async function handleRequest(request) {
// Obtain a value from an environment variable, and this environment variable needs
const myJsonData = env.keytestjson;
  // Create response body.
  const response = new Response(JSON.stringify(myJsonData), {
    headers: {
      'Content-Type': 'application/json'
    }
  });
  // Return the response.
  return response;
}
```

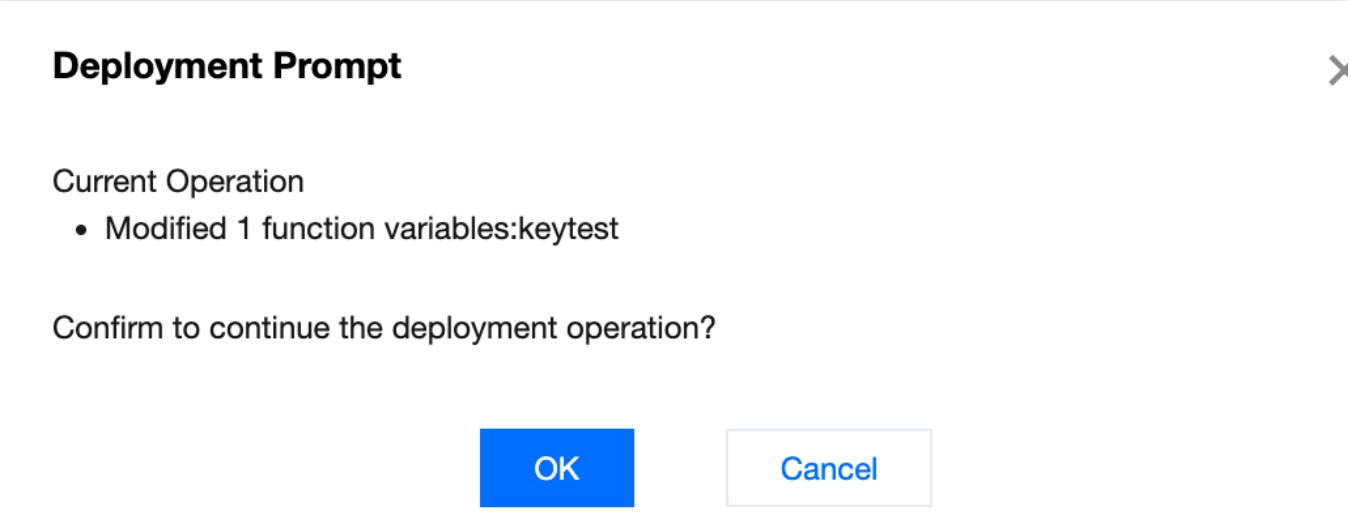
The semantic summary of the above code is to obtain the environment variable that the function has created and deployed, and respond it to the client as JSON data.

Deploy the code, and you can view the result by accessing the trigger rules or default access of the function.



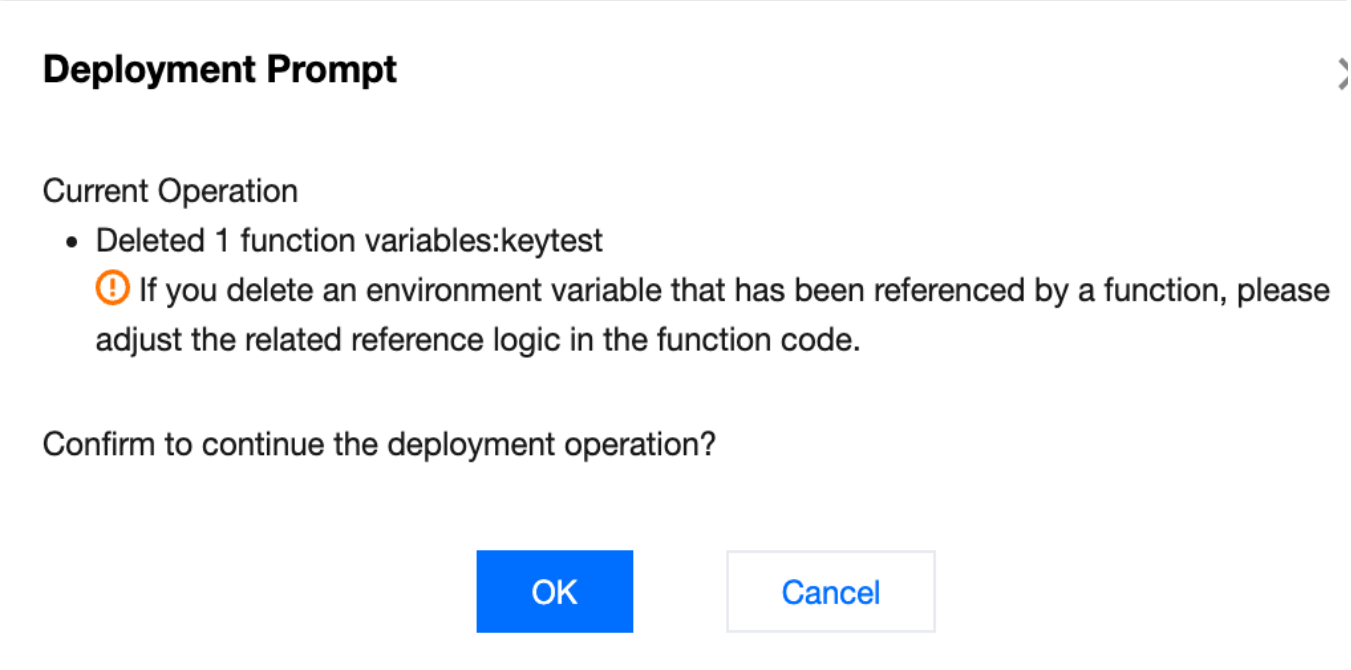
Editing Environment Variables

1. In the Environment Variables module, click Edit for a specific environment variable. In the pop-up window, you can change the variable type and the variable value. Click **OK** to save the edited content.
2. In the Environment Variables module**,** click **Deploy**. The pop-up window will display the specific updates for this operation. Click **OK** to complete the deployment.



Deleting Environment Variables

In the Environment Variables module, click **Delete** for a specific environment variable**,** and then click **Deploy**. The pop-up window will display the specific environment variable being deleted in this operation. Click **OK** to complete the deployment.



Note:

If the environment variables referenced by the function are deleted through the above operations, adjust the logic of related references in the function code.

Runtime APIs
addEventListener

Last updated：2023-03-08 11:26:27

This API is used to register an event listener for a target. The event listener triggers an edge function when the specified type of event is delivered to the target. Only one event listener takes effect for the same type of event. Only `fetch` request events are supported currently. If a `fetch` event occurs after a `fetch` event listener is registered, the event listener generates a FetchEvent object to process the HTTP request.

Overview

```
function addEventListener(type: string, listener: (event: FetchEvent) => void): voi
```

Parameters

Parameter	Type	Required	Description
type	string	Yes	Event type. Only `fetch` request events are supported. If you specify a request event type other than `fetch`, the Edge Functions engine throws an `Error` exception.
listener	(event: FetchEvent) => void	Yes	Event listener that is used to process event callbacks. You can register a `fetch` event listener to generate FetchEvent objects.

Sample Code

```
// Register a fetch event listener.
addEventListener('fetch', (event) => {
  // Respond to the client.
  event.respondWith(new Response('Hello World!'));
});
```

©2013-2025 Tencent Cloud International Pte. Ltd.

References

MDN documentation: addEventListener

Sample Functions: Returning an HTML Page

Sample Functions: Returning a JSON Object

©2013-2025 Tencent Cloud International Pte. Ltd.

Cache

Last updated：2024-01-31 10:10:01

The **Cache** API is designed based on the standard Web API Cache. During the runtime of edge functions, a global `caches` object that provides a group of methods for cache-related operations is injected.

Note:

The cached content takes effect only on the current data node and is not automatically copied to other data nodes.

Constructor API

Use `caches.default` to fetch the default cache instance.

```
// Fetch the default cache instance.
const cache = caches.default;

// This method provides the same effect as `caches.default`.
await caches.open('default');
```

Use `caches.open` to create a cache instance with the specified namespace.

```
// Create a cache instance with the specified namespace.
const cache = await caches.open(namespace);
```

Parameters

The following table describes the parameters of the `caches.open(namespace)` method.

Parameter	Type	Required	Description
namespace	string	Yes	The namespace of the cache. The value "default" specifies the default instance. You can also use caches.default to fetch a default instance.

Methods

match

```
cache.match(request: string | Request, options?: MatchOptions): Promise<Response |
```

The match() method fetches the Response cache associated with the request and returns a Promise object. If the cache exists, the object contains the Response object. If not, the object contains undefined.

Note:

The **cache.match()** method does not forward requests to the origin. If the cache expires, the method throws a 504 error.

Parameters

Parameter	Type	Required	Description
request	string | Request	Yes	The request object. The following method and headers are supported: GET Only the GET method is supported. If this parameter is set to a string, the string is used as a URL to construct a Request object. Range If the request contains a Range header and the cached response contains the Accept-Ranges header, the 206 response is returned. If-Modified-Since If the request contains a If-Modified-Since header and the cached response contains a Last-Modified header with the same value as that of the If-Modified-Since header, the 304 response is returned. If-None-Match If the request contains a If-None-Match header and the cached response contains an ETag header with the same value as that of the If-Modified-Since header, the 304 response is returned.
options	MatchOptions	No	The options.

MatchOptions

Parameter	Type	Example	Description
ignoreMethod	boolean	true	Specifies whether to ignore the request method. If you set this parameter to true, the request is considered to be a GET request regardless of its actual method.

put

```
cache.put(request: string | Request, response: Response): Promise<undefined>
```

The put() method tries to add a response to the cache by using the given request as the cache key. A `Promise<undefined>` object is returned regardless of whether caching is successful.

Note:

The put() method returns a 413 error if the Cache-Control header of the object specified in the **response** parameter instructs not to cache.

Parameters

Parameter	Type	Required	Description
request	string | Request	Yes	The cache key. GET The `request` parameter supports only the GET method. If other methods are specified, an error is thrown. string If the `request` parameter is set to a string, the string is used as a URL to construct a Request object.
response	Response	Yes	The cache content. Cache-Control Valid values: s-maxage, max-age, no-store, no-cache, and private. The values no-store, no-cache, and private indicate no caching. If you use these values, the `cache.put` method returns a 413 error. Pragma If Cache-Control is not specified and Pragma is set to no-cache, no caching is performed. ETag If the `request` parameter of the `cache.match` method contains a If-None-Match header, you can associate it with ETag. Last-Modified if the `request` parameter of the `cache.match` method contains a If-Modified-Since header, you can associate it with Last-Modified. 416 Range Not Satisfiable If the object specified in the `response` parameter is 416 Range Not Satisfiable, no caching is performed.

Parameter limits

The `cache.put` method throws an error in the following scenarios:

The `request` parameter specifies a method other than the GET method.

The status code of the `response` parameter is 206 Partial Content.

The `response` parameter contains a Vary: * header.

delete

```
cache.delete(request: string | Request, options?: DeleteOptions): Promise<boolean>
```

The delete() method deletes the response associated with the request from the cache. If no network exception occurs, a Promise containing `true` is returned. Otherwise, a Promise containing `false` is returned.

Parameters

Parameter	Type	Required	Description
request	string | Request	Yes	The cache key. GET The request parameter supports only the GET method. string If the `request` parameter is set to a string, the string is used as a URL to construct a Request object.
options	DeleteOptions	No	The options.

DeleteOptions

Parameter	Type	Example	Description
ignoreMethod	boolean	true	Specifies whether to ignore the request method. If you set this parameter to true, the request is considered to be a GET request regardless of its actual method.

References

MDN documentation: Cache
Sample Functions: Caching POST Requests
Sample Functions: Using the Cache API

©2013-2025 Tencent Cloud International Pte. Ltd.

Cookies

Last updated：2024-01-30 17:12:01

The **Cookies** API provides a group of methods for you to manage cookies.

Note:

The unique keys of Cookies objects are in the format of `name + domain + path`. You can manage Cookies objects based on the unique keys.

Constructor API

```
const cookies = new Cookies(cookieStr?: string, isSetCookie?: boolean);
```

Parameters

Parameter	Type	Required	Description
cookieStr	string	No	The Cookie string or Set-Cookie string.
isSetCookie	boolean	No	Specifies whether the value of the cookieStr parameter is a Set-Cookie string. Default value: false.

Methods

get

```
cookies.get(name?: string): null | Cookie | Array<Cookie>;
```

The get() method obtains the Cookie object of the specified name. If multiple objects are matched, a Cookie array is returned.

Parameters

Parameter	Type	Required	Description
name	string	No	The name of Cookie object. Valid options: Default name Obtains all Cookie objects. Specified name

			Obtains the Cookie object of the specified name. If multiple objects are matched, a Cookie array is returned.

Cookie

The following table describes the attributes of the `Cookie` object. For more information, see Set-Cookie.

Attribute	Type	Read-only	Description
name	string	Yes	The name of the Cookie object.
value	string	Yes	The value of the Cookie object.
domain	string	Yes	The host to which the Cookie object will be sent.
path	string	Yes	The path to which the Cookie object will be sent.
expires	string	Yes	The maximum effective time of the Cookie object. The value meets the HTTP Date header standards.
max_age	string	Yes	The number of seconds until the Cookie object expires.
samesite	string	Yes	Controls whether the Cookie object is sent with cross-site requests, providing some protection against cross-site request forgery (CSRF) attacks.
httponly	boolean	Yes	Forbids JavaScript from accessing the Cookie object. The attribute is carried only by HTTP requests.
secure	boolean	Yes	Specifies that the Cookie object can be carried only by HTTPS requests.

set

```
cookies.set(name: string, value: string, options?: Cookie): boolean;
```

The set() method adds cookies in overwrite mode. If `true` is returned, cookies are successfully added. If `false` is returned, cookies fail to be added because the number of cookies exceeds the upper limit. For more information, see Cookie limits.

Note:

Cookies are added in overwrite mode based on unique keys in the format of `name + domain + path`.

Parameters

Parameter	Type	Required	Description

name	string	Yes	The name of the Cookie object.
value	string	Yes	The value of the Cookie object.
Cookie	string	No	The configuration items of the Cookie object.

append

```
cookies.append(name: string, value: string, options?: Cookie): boolean;
```

The append() method appends cookies in scenarios where multiple values correspond to the same name. If `true` is returned, cookies are successfully appended. If `false` is returned, cookies fail to be appended because the value already exists or the number of cookies exceeds the upper limit. For more information, see Cookie limits.

Note:

Cookies are appended based on unique keys in the format of `name + domain + path`.

remove

```
cookies.remove(name: string, options?: Cookie): boolean;
```

The remove() method deletes cookies.

Note:

Cookies are deleted based on unique keys in the format of `name + domain + path`.

Parameters

Parameter	Type	Required	Description
name	string	Yes	The name of the Cookie object.
options	Cookie	Yes	The configuration items of the Cookie object. The configuration items `domain` and `path` support the wildcard character (*), indicating that all items are matched.

Use Limits

Automatic escape of special characters

The following characters are automatically escaped if they are contained in the value of the `name` attribute: `" ( ) , / : ; ? < = > ? @ [ ] \\ { }`. `0x00~0x1F` and `0x7F~0xFF`.

The following characters are automatically escaped if they are contained in the value of the `value` attribute: `, ; " \\`. `0x00~0x1F` and `0x7F~0xFF`.

Cookie limits

The size of the Cookie attribute `name` cannot exceed 64 bytes.

The accumulated size of the Cookie attributes `value, domain, path, expires, max_age, and samesite` cannot exceed 1 KB.

The total length of all fields after escape of cookies cannot exceed 4 KB.

The total number of Cookie objects contained in cookies cannot exceed 64.

Sample Code

```
function handleEvent(event) {
  const response = new Response('hello world');

  // Generate a Cookies object.
  const cookies = new Cookies('ssid=helloworld; expires=Sun, 10-Dec-2023 03:10:01 G

  // Set the response header Set-Cookie.
  response.setCookies(cookies);

  return response;
}

addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});
```

References

MDN documentation: Set-Cookie

Sample Functions: Performing an A/B Test

Sample Functions: Setting Cookies

©2013-2025 Tencent Cloud International Pte. Ltd.

Encoding

Last updated：2024-01-30 16:56:26

The encoder and decoder are designed based on standard Web APIs TextEncoder and TextDecoder.

TextEncoder

The TextEncoder API takes code point streams as inputs and generates `UTF-8` byte streams as outputs. For more information, see TextEncoder.

Constructor API

```
// The TextEncoder() constructor does not have any parameters.
const encoder = new TextEncoder();
```

Attributes

```
// encoder.encoding
readonly encoding: string;
```

The name of the encoding algorithm that is used by the encoder. The value is fixed to `UTF-8` .

Methods

encode

```
encoder.encode(input?: string | undefined): Uint8Array
```

The encoder.encode() method takes code point streams as inputs and generates `UTF-8` byte streams as outputs.

Note:

The maximum length of **input** is 300 MB. An exception will be thrown if the maximum length is exceeded.

`encoder.encode` parameters

Parameter	Type	Required	Description
input	string | undefined	No	The text to be encoded.

encodeInto

```
encoder.encodeInto(input: string, destination: Uint8Array): EncodeIntoResult;
```

The encoder.encodeInto() method takes code point streams as inputs, generates `UTF-8` byte streams as outputs, and writes the outputs to a `destination` byte array.

Parameters

Parameter	Type	Required	Description
input	string	Yes	The text to be encoded.
destination	Uint8Array	Yes	The object in which the encoded text is stored.

Return value: `EncodeIntoResult`

Parameter	Type	Description
read	number	The number of UTF-16 units that have been converted to UTF-8.
written	number	The number of bytes that are modified in the destination Uint8Array.

TextDecoder

The TextDecoder API takes byte streams as inputs and generates code point streams as outputs. For more information, see TextDecoder.

Construction API

```
const decoder = new TextDecoder(label?: string | undefined, options?: DecoderOption
```

Parameters

Note:

The label parameter does not support the following values:

iso-8859-16

hz-gb-2312

csiso2022kr, iso-2022-kr

Parameter	Type	Required	Description
label	string | undefined	No	The name of the decoding algorithm that is used by the decoder. Default value: UTF-8. For the valid values of label, see Encoding API Encodings.
options	DecoderOptions | undefined	No	The configuration items of the decoder.

DecoderOptions

The following table describes the configuration items of the decoder.

Parameter	Type	Default value	Description
fatal	boolean	false	Specifies whether to throw an exception when decoding fails.
ignoreBOM	boolean	false	Specifies whether to ignore byte-order marker.

Attributes

encoding

```
// decoder.encoding
readonly encoding: string;
```

The name of the decoding algorithm that is used by the decoder.

fatal

```
// decoder.fatal
readonly fatal: boolean;
```

Specifies whether to throw an exception when decoding fails.

ignoreBOM

```
// decoder.ignoreBOM
readonly ignoreBOM: boolean;
```

Specifies whether to ignore byte-order marker.

Methods

decode

```
const result = decoder.decode(buffer?: ArrayBuffer | ArrayBufferView | undefined, o
```

Note:

The maximum length of **buffer** is 100 MB. An exception will be thrown if the maximum length is exceeded.

Parameter	Type	Required	Description
buffer	ArrayBuffer | ArrayBufferView | undefined	No	The byte stream to be decoded. The maximum length of buffer is 100 MB. An exception will be thrown if the maximum length is exceeded.

options	DecodeOptions	No	The configuration items for decoding.

DecodeOptions

The following table describes the configuration items for decoding.

Parameter	Type	Default value	Description
stream	boolean	false	Specifies whether to perform decoding in streaming mode. Default value: false. Valid values: true Perform decoding in streaming mode. Use this value if data is processed in chunks and additional chunks are expected. false Do not perform decoding in streaming mode. Use this value if the current chunk is the last one or data is not chunked.

Sample Code

```
function handleEvent(event) {
  // Encoder
  const encoder = new TextEncoder();
  const encodeText = encoder.encode('hello world');

  // Decoder
  const decoder = new TextDecoder();
  const decodeText = decoder.decode(encodeText);

  // Response
  const response = new Response(JSON.stringify({
    encodeText: encodeText.toString(),
    decodeText,
  }));

  return response;
}

addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});
```

References

MDN documentation: TextEncoder

MDN documentation:TextDecoder

MDN documentation: Encoding API Encodings

Sample Functions: Protecting Data from Tampering

©2013-2025 Tencent Cloud International Pte. Ltd.

Fetch

Last updated：2025-04-25 10:53:13

The `Fetch` API is designed based on the standard Web API Fetch. You can add `fetch` to the edge function runtime to initiate an async request to fetch remote.

Description

```
function fetch(request: string | Request, requestInit?: RequestInit): Promise<Respo
```

Parameters

Parameter name	Type	Required	Description
request	string | Request	Yes	Requested resource.
requestInit	RequestInit	No	Initial configuration items of the request object. For more information, see RequestInit.

Advanced Features

With `fetch`, you can pass in specific parameters for finer configuration of EdgeOne node caching, fetching from the origin, image processing and redirection.

Accessing EdgeOne nodes or fetching from the origin

When a client request comes to a domain name domain name connected to EdgeOne (such as www.example.com), if the request triggers an edge function, `fetch(www.example.com)` is executed in the edge function. The request is directed to the cache on the EdgeOne node. If no cache is hit, the request is redirected to the origin.

Restrictions and requirements:

1. The domain name is connected to EdgeOne and the request triggers the edge function.
2. HOST of `request.url` specified in `fetch(request)` is the same as the HOST of client request URL.
3. HOST of `request.headers.host` specified in `fetch(request)` is the same as the HOST of client request URL.

©2013-2025 Tencent Cloud International Pte. Ltd.

Use `fetch(event.request)` to obtain the cache from EdgeOne nodes and pull resources from the origin if no cache is hit.

```
addEventListener('fetch', (event) => {
  //Use `fetch(event.request)` to obtain the cache from EdgeOne nodes and pull reso
  const response = fetch(event.request);

  event.respondWith(response);
});
```

Use `fetch(url)` to obtain the cache from EdgeOne nodes and pull resources from the origin if no cache is hit.

```
addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});

async function handleEvent(event) {
  const { request } = event;
  const urlInfo = new URL(request.url);
  // origin-pull URL rewrite
  const url = `${urlInfo.origin}/h5/${urlInfo.pathname}`;

  // fetch(url) collected EdgeOne CDN Cache and origin-pull.
  const response = await fetch(url);
  return response;
}
```

Image processing

You can use `requestInit.eo.image` to scale images or convert the image format. For details, see ImageProperties.

Note:

To use `fetch(request, requestInit)` to process images, the requirements for using `fetch` to obtain cache and pull from the origin must be met.

Timeout settings

`Fetch` supports configuring the request timeout using `requestInit.eo.timeoutSetting`, with a default timeout of 15 seconds and a maximum of 300 seconds.

```
fetch(url, {
    eo: {
        timeoutSetting: {
            connectTimeout: 60000,  // Connection timeout, unit: milliseconds
            readTimeout: 60000,     // Response read timeout, unit: milliseconds
```

```
                writeTimeout: 60000,    // Request send timeout, unit: milliseconds
            }
        }
    })
```

Redirection

`fetch` supports `3xx` redirect status codes. You can use the second parameter `requestInit.redirect` to specify the redirect status code. For more information, see RequestInit.

Redirect rules conform to the Fetch Standard. The follow rules vary based on the status code.

Status code	Redirect Rule
301 and 302	Replaces the POST method with the GET method.
303	Replaces all request methods except for HEAD and GET with the GET method.
307 and 308	Retains the original request method.

Important

The redirect address is obtained from the `Location` response header. If `Location` does not exist, no redirect action is taken.

The value of the `Location` can be an absolute URL or a relative URL. For more information, see RFC-3986: URI Reference.

Runtime limits

If you use `fetch` to initiate a request in an edge function, take note of the following limits:

Number of times: Each time an edge function runs, `fetch` can be initiated for a maximum of 64 times. If the limit is exceeded, the exceeding requests fail and exceptions are returned.

Number of concurrencies: Each time an edge function runs, `fetch` can be initiated at a maximum concurrency of 8. If the limit is exceeded, the exceeding requests are postponed until a running `fetch` is resolved.

Time limit: The default timeout for requests initiated with `fetch` in Edge functions is 15 seconds, with a maximum setting of 300 seconds.

Note

Each redirect is counted as a request and has a higher priority than newly-initiated `fetch` requests.

References

MDN documentation: Fetch

Sample functions: Obtaining Remote Resources and Responding to the Client

©2013-2025 Tencent Cloud International Pte. Ltd.

FetchEvent

Last updated：2024-01-30 16:48:21

A **FetchEvent** object represents any incoming HTTP request event. Edge Functions processes HTTP requests by registering `fetch` event listeners.

Overview

In Edge Functions, use addEventListener to register a `fetch` event listener to generate an HTTP request event FetchEvent, thereby processing HTTP requests.

Note:

The `FetchEvent` object cannot be constructed directly. You can use `addEventListener` to register a `fetch` event listener to obtain the `event` object.

```
// `event` is the `FetchEvent` object.
addEventListener('fetch', (event) => {
  event.respondWith(new Response('hello world!'));
});
```

Attributes

ClientId

```
// event.clientId
readonly clientId: string;
```

The ClientId attribute specifies the ID allocated by Edge Functions for each request.

request

```
// event.request
readonly request: Request;
```

The request attribute specifies the HTTP request object initiated by the client. For more information, see Request.

Methods

respondWith

```
event.respondWith(response: Response | Promise<Response>): void;
```

Edge Functions takes over requests from the client and uses this method to return custom responses.

Note:

In the `fetch` event callback of the `addEventListener` event listener, the `event.respondWith()` method is used to respond to the client. If this method is not invoked, Edge Functions forwards the current request back to the origin.

Parameters

Parameter	Type	Required	Description
response	Response Promise<Response>	Yes	The response to the HTTP request of the client.

waitUntil

```
event.waitUntil(task: Promise<any>): void;
```

The waitUntil() method is used to notify Edge Functions to wait until a `Promise` -based task is completed, extending the event processing lifecycle.

Parameters

Parameter	Type	Required	Description
task	Promise<Response>	Yes	The `Promise`-based task.

passThroughOnException

```
event.passThroughOnException(): void;
```

The passThroughOnException() method is used to prevent runtime error responses. If the function code throws an unhandled exception, Edge Functions forwards the request back to the origin, enhancing the service availability.

Sample Code

If the `event.respondWith` method is not invoked, Edge Functions forwards the current request back to the origin.

```
function handleRequest(request) {
```

```
  return new Response('Edge Functions, Hello World!');
}

addEventListener('fetch', event => {
  const request = event.request;
  // If the request URL contains the string /ignore/, Edge Functions forwards the c
  if (request.url.indexOf('/ignore/') !== -1) {
    // The event.respondWith method is not invoked.
    return;
  }

  // Customize content in the edge function to respond to the client.
  event.respondWith(handleRequest(request));
});
```

If the function code throws an unhandled exception, Edge Functions forwards the current request back to the origin.

```
addEventListener('fetch', event => {
  // If the function code throws an unhandled exception, Edge Functions forwards th
  event.passThroughOnException();
  throw new Error('Throw error');
});
```

References

MDN documentation: FetchEvent

Sample Functions: Returning an HTML Page

Sample Functions: Using the Cache API

Headers

Last updated：2025-03-04 15:37:54

The **Headers** API is designed based on the standard Web API Headers. You can use this API to perform operations on HTTP request and response headers.

Constructor API

```
const headers = new Headers(init?: object | Array<[string, string]> | Headers);
```

Parameters

Parameter	Type	Required	Description
init	object | Array<[string, string]> | Headers	No	The HTTP header that is used to pre-populate the Headers object. Valid parameter types: object The Constructor API enumerates all enumerable attributes that are included in the specified object and pre-populates the attributes to the new Headers object. Array<[string, string]> Each element in the array is a key-value pair. Example: [key, value]. The Constructor API traverses the array and pre-populates the key-value pairs to the new Headers object. Headers The Constructor API copies all fields from an existing Headers object to the new Headers object.

Methods

append

```
headers.append(name: string, value: string): void;
```

The append() method appends a new value to a header that is specified by the `Headers` object. If the header does not exist, the append() method directly adds the header.

Note:

The `headers` object has the following write-in limits:

The name cannot exceed 255 bytes, and can only consist of digits, uppercase and lowercase letters, -, and _. Otherwise, an exception will be thrown.

The value cannot exceed 4,095 bytes. Leading and trailing whitespace characters '\\n', '\\r', '\\t', and space will be automatically removed, and the string cannot contain '\\0', '\\n', or '\\r' in the middle. Otherwise, an exception will be thrown.

Parameters

Parameter	Type	Required	Description
name	string	Yes	The name of the HTTP header that you want to add to the Headers object.
value	string	Yes	The value of the HTTP header that you want to add.

delete

```
headers.delete(name: string): void;
```

The delete() method deletes the specified header from the `Headers` object.

Parameters

Parameter	Type	Required	Description
name	string	Yes	The name of the HTTP header that you want to delete from the Headers object.

entries

```
headers.entries(): iterator;
```

The entries() method obtains an array of all key-value pairs from the `Headers` object. The key-value pairs are in the format of [name, value]. For valid return values, see Iteration protocols.

forEach

```
headers.forEach(callback: (name: string, value: string) => void | number): void;
```

The forEach() method traverses all headers that are included in the `Headers` object. If `callback` returns a non-zero value, traversal is stopped.

Note:

The `forEach` method does not comply with Web API standards. Edge Functions extends the functionality of the method based on Web API standards to provide an efficient way to traverse headers.

get

```
headers.get(name: string): string;
```

The get() method obtains the value of the specified header from the `Headers` object.

getSetCookie

```
headers.getSetCookie(): Array<string>
```

This method returns an array, containing all values of the Set-Cookie headers.

has

```
headers.has(name: string): boolean;
```

The has() method checks whether the `Headers` object contains the specified header.

keys

```
headers.keys(): iterator;
```

The keys() method obtains all keys that are included in the `Headers` object. For valid return values, see Iteration protocols.

set

```
headers.set(name: string, value: string): void;
```

©2013-2025 Tencent Cloud International Pte. Ltd.

The set() method sets a new value for an existing header in the `Headers` object. If the header does not exist, the set() method directly adds the header.

Note:

The `headers` object has the following write-in limits:

The name cannot exceed 255 bytes, and can only consist of digits, uppercase and lowercase letters, -, and _. Otherwise, an exception will be thrown.

The value cannot exceed 4,095 bytes. Leading and trailing whitespace characters '\\n', '\\r', '\\t', and space will be automatically removed, and the string cannot contain '\\0', '\\n', or '\\r' in the middle. Otherwise, an exception will be thrown.

values

```
headers.values(): iterator;
```

The values() method obtains all values that are included in the `Headers` object. For valid return values, see Iteration protocols.

Sample Code

```
function handleEvent() {
  const headers = new Headers({
    'my-header-x': 'hello world',
  });

  const response =  new Response('hello world',{
    headers,
  });
  return response;
}

addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});
```

References

MDN documentation: Headers

Sample Functions: Protecting Data from Tampering

Sample Functions: Authenticating a Request Header

Sample Functions: Modifying a Response Header

Request

Last updated：2025-06-18 16:24:42

The **Request** API represents an HTTP request. It is designed based on the standard Web API Request.

Note

In Edge Functions, you can obtain a `Request` object by using any of the following methods:

Create a `Request` object for the Fetch API by using the `Request` constructor.

Use the `FetchEvent` object event.request to obtain a `Request` object.

Constructor API

```
const request = new Request(input: string | Request, init?: RequestInit)
```

Parameters

Parameter name	Type	Required	Description
input	string | Request	Yes	A URL string or a `Request` object.
options	RequestInit	No	Initial configuration items of the `Request` object.

RequestInit

The following table describes the initial configuration items of the Request object.

Name	Type	Required	Default value	Description
method	string	No	GET	Request method. Examples: `GET` and `POST` .
headers	Headers	No	-	Headers that you want to add to the request.
body	string |Blob | ArrayBuffer | ArrayBufferView |ReadableStream	No	-	Request body.
redirect	string	No	follow	Redirect mode. Valid values:

				`manual` , `error` , and `follow` .
maxFollow	number	No	12	The maximum number of redirects allowed.
version	string	No	HTTP/1.1	HTTP version. Valid values: `HTTP/1.0` , `HTTP/1.1` , and `HTTP/2.0` .
copyHeaders	boolean	No	-	Specifies whether to copy the headers of the `Request` object. **This parameter is not in the Web API specifications.**
eo	RequestInitEoProperties	No	-	Specifies the behavior that Edge Functions adopts in processing the request. **This parameter is not in the Web API specifications.**

RequestInitEoProperties

This parameter specifies the behavior that Edge Functions adopts in processing the request. It is not in the Web API specifications.

Parameter name	Type	Required	Description
resolveOverride	string	No	Overrides the original domain name resolution for the fetch request. You can specify a domain name or IP address that meets the following requirements: The IP address cannot contain a scheme or port number. For an IPv6 address, you do not need to enclose it in a pair of square brackets.
Image	ImageProperties	No	Image processing configurations

ImageProperties

fetch supports image processing. The configurations are the same as of site acceleration. For more information, see Resizing and Converting Images.

Parameter name	Type	Required	Description
format	string	No	Convert an image into a specified format. Valid values: `jpg` , `gif` , `png` , `bmp` , `webp` , `avif` , `jp2` , `jxr` , `heif` .

long	number	No	Specify the length of the long side, and automatically scale the short side (if not specified)
short	number	No	Specify the length of the short side, and automatically scale the long side (if not specified)
width	number	No	Specify the width, and automatically scale the height (if not specified)
height	number	No	Specify the height, and automatically scale the width (if not specified)

Attributes

body

```
// request.body
readonly body: ReadableStream;
```

Request body. For more information, see ReadableStream.

bodyUsed

```
// request.bodyUsed
readonly bodyUsed: boolean;
```

Indicates whether the request body is read.

headers

```
// request.headers
readonly headers: Headers;
```

Request headers. For more information, see Headers.

method

```
// request.method
readonly method: string;
```

The request method. Default value: `GET` .

redirect

```
// request.redirect
```

```
readonly redirect: string;
```

The request redirect mode. Valid values: `follow` , `error` , and `manual` . Default value: `follow` .

maxFollow

```
// request.maxFollow
readonly maxFollow: number;
```

The maximum number of redirects.

url

```
// request.url
readonly url: string;
```

The request URL.

version

```
// request.version
readonly version: string;
```

The HTTP version that is used by the request.

eo

```
// request.version
readonly eo: IncomingRequestEoProperties;
```

Other information provided by Edge Functions about the current request. For more information, see IncomingRequestEoProperties.

IncomingRequestEoProperties

The client request object event.request contains an `eo` attribute.

Name	Type	Description	Example
geo	GeoProperties	Location of the client who initiates the request.	-
clientIp	string	Client IP address	192.168.1.1
uuid	string	Unique identifier of the request, same as EO-LOG-UUID	-

GeoProperties

Location of the client who initiates the request.

Name	Type	Description	Example
asn	number	ASN	132203
countryName	string	Country name	Singapore
countryCodeAlpha2	string	ISO-3611 alpha2 code of the country	SG
countryCodeAlpha3	string	ISO-3611 alpha3 code of the country	SGP
countryCodeNumeric	string	ISO-3611 numeric code of the country.	702
regionName	string	Region name	-
regionCode	string	Region code	AA-AA
cityName	string	City name	singapore
latitude	number	Latitude	1.29027
longitude	number	Longitude	103.851959

Methods

Important

When using a method to obtain the request body, the size of the `HTTP body` is capped at 1 MB. If the threshold is exceeded, an OverSize exception is returned. In this case, we recommend that you use request.body to read the request body in streaming mode. For more information, see ReadableStream.

arrayBuffer

```
request.arrayBuffer(): Promise<ArrayBuffer>;
```

The arrayBuffer() method reads the request body and returns a promise that resolves with an ArrayBuffer.

blob

```
request.blob(): Promise<Blob>;
```

The blob() method reads the request body and returns a promise that resolves with a Blob.

clone

```
request.clone(copyHeaders?: boolean): Request;
```

The clone() method creates a clone of a request object.

Parameter

Parameter name	Type	Required	Description
copyHeaders	boolean	No	Specifies whether to copy the request headers of the original object. Default value: `false` . Valid values: true Copy the request headers of the original object. false Reference the request headers of the original object.

json

```
request.json(): Promise<object>;
```

The json() method reads the request body and returns a promise that resolves with the parsing result of the body text as `json` .

text

```
request.text(): Promise<string>;
```

The text() method reads the request body and returns a promise that resolves with a String.

formData

```
request.formData(): Promise<FormData>;
```

The formData() method takes a Response stream, reads it to completion, and returns a promise that resolves with a FormData.

Parameter

Parameter name	Type	Required	Description

cookies	Cookies	No	A new Cookies object.

Sample Code

```
async function handleRequest() {
  const request = new Request('https://www.tencentcloud.com/');
  const response = await fetch(request);
  return response;
}

addEventListener('fetch', (event) => {
  event.respondWith(handleRequest());
});
```

References

MDN documentation: Request

Sample Functions: Using the Cache API

Sample Functions: Performing Redirect Based on the Request Location

©2013-2025 Tencent Cloud International Pte. Ltd.

Response

Last updated：2024-05-07 16:15:40

The **Response** API represents the response to an HTTP request. It is designed based on the standard Web API Response.

Note

In Edge Functions, you can obtain a `Response` object by using the following methods:

Create a `Response` object by using the `Response` constructor API. This object can be used as the value of the `response` parameter in the event.respondWith method.

Use Fetch to obtain a `Response` object.

Constructor API

```
const response = new Response(body?: string | ArrayBuffer | Blob | ReadableStream |
```

Parameters

Parameter	Type	`Required`	Description
body	string | ArrayBuffer | Blob | ReadableStream | null | undefined	Yes	The body of the `Response` object.
init	ResponseInit	No	The initial configuration items of the `Response` object.

ResponseInit

Parameter	Type	Required	Description
status	number	No	The status code for the response.
statusText	string	No	The status message for the response. The maximum length is 4,095 bytes. If the length exceeds the upper limit, the extra content will be truncated.
headers	Headers	No	The headers associated with the response.

©2013-2025 Tencent Cloud International Pte. Ltd.

Attributes

body

```
// response.body
readonly body: ReadableStream;
```

The response body. For more information, see ReadableStream.

bodyUsed

```
// response.bodyUsed
readonly bodyUsed: boolean;
```

Indicates whether the response body is read.

headers

```
// response.headers
readonly headers: Headers;
```

The response headers. For more information, see Headers.

ok

```
// response.ok
readonly ok: boolean;
```

Indicates whether the response was successful. If the status code ranges from 200 to 299, the response was successful.

status

```
// response.status
readonly status: number;
```

The status code for the response.

statusText

```
// response.statusText
readonly statusText: string;
```

The status message for the response.

©2013-2025 Tencent Cloud International Pte. Ltd.

url

```
// response.url
readonly url: string;
```

The URL of the response.

redirected

```
// response.redirected
readonly redirected: boolean;
```

Indicates whether the response is the result of a redirect.

redirectUrls

```
// response.redirectUrls
readonly redirectUrls: Array<String>
```

All URLs for redirection.

Methods

Note

If the size of the `HTTP body` obtained by using a method exceeds 1 MB, the OverSize exception will be thrown. In this case, we recommend that you use response.body to read the response body in streaming mode. For more information, see ReadableStream.

arrayBuffer

```
response.arrayBuffer(): Promise<ArrayBuffer>;
```

The arrayBuffer() method takes a Response stream, reads it to completion, and returns a promise that resolves with an ArrayBuffer.

blob

```
response.blob(): Promise<Blob>;
```

The blob() method takes a Response stream, reads it to completion, and returns a promise that resolves with a Blob.

clone

```
response.clone(copyHeaders?: boolean): Request;
```

The clone() method creates a clone of a response object.

Parameters

Parameter	Type	Required	Description
copyHeaders	boolean	No	Specifies whether to copy the response headers of the original object. Default value: `false` . Valid values: true Copy the response headers of the original object. false Reference the response headers of the original object.

json

```
response.json(): Promise<object>;
```

The json() method takes a Response stream, reads it to completion, and returns a promise which resolves with the parsing result of the body text as `json` .

text

```
response.text(): Promise<string>;
```

The text() method takes a Response stream, reads it to completion, and returns a promise that resolves with a String.

formData

```
response.formData(): Promise<FormData>;
```

The formData() method takes a Response stream, reads it to completion, and returns a promise that resolves with a FormData.

Static Methods

error

```
Response.error(): Response;
```

The error() method returns a new Response object that contains network error information.

redirect

```
Response.redirect(url: string | URL, status?: number): Response;
```

The redirect() method returns a Response object resulting in a redirect to the specified URL.

Parameters

Parameter	Type	Required	Description
url	string	Yes	Redirect URL
status	number	No	Status code for the response. Valid values: 301, 302, 303, 307, and 308. Default value: 302.

Sample Code

```
addEventListener('fetch', (event) => {
  const response =  new Response('hello world');
  event.respondWith(response);
});
```

References

MDN documentation: Response
Sample Functions: Returning an HTML Page
Sample Functions: Modifying a Response Header
Sample Functions: Performing an A/B Test

©2013-2025 Tencent Cloud International Pte. Ltd.

Streams

ReadableStream

Last updated：2024-01-30 14:59:48

The **ReadableStream** API represents a readable stream or readable end. It is designed based on the standard Web API ReadableStream.

Note:

A `ReadableStream` object cannot be constructed directly. You can use TransformStream to construct a `ReadableStream` object.

Overview

```
// Use TransformStream to construct a ReadableStream object.
const { readable } = new TransformStream();
```

Attributes

```
// readable.locked
readonly locked: boolean;
```

The locked attribute indicates whether a stream is locked.

Note:

A stream is locked in the following scenarios:

The stream has no more than one activated `reader`. Before the `reader` calls the `releaseLock()` method, the stream is locked.

The stream is in being piped. The stream is locked until the piping ends.

Methods

Note:

Before you use any of the following methods, make sure that the stream is not locked. Otherwise, an exception is returned.

getReader

```
readable.getReader(options?: ReaderOptions): ReadableStreamDefaultReader | Readable
```

The getReader() method creates a `reader` and locks the current stream until the `reader` calls the `releaseLock()` method.

Parameters

Parameter	Type	Required	Description
options	ReaderOptions	Yes	The configuration items for generating the reader.

ReaderOptions

The following table describes the parameters of the `ReaderOptions` object.

Parameter	Type	Required	Description
mode	string	No	Reader The type of the `reader`. Default value: `undefined.` Valid values: undefined Create a reader of the ReadableStreamDefaultReader type. byob Create a reader of the ReadableStreamBYOBReader type.

pipeThrough

```
readable.pipeThrough(transformStream: TransformStream, options?: PipeToOptions): Re
```

The pipeThrough() method pipes the data of the current readable stream to the writable side of the `transformStream` and returns the readable side of the `transformStream` .

Note:

During the piping, the `writable` side of the current stream is locked.

Parameters

Parameter	Type	Required	Description
transformStream	TransformStream	Yes	The destination to which the current stream is piped.
options	PipeToOptions	Yes	The configuration items for piping the stream.

PipeToOptions

The following table describes the configuration items for piping the stream.

Parameter	Type	Required	Description
preventClose	boolean	No	If the value is true, the writable stream is not closed along with the readable stream.
preventAbort	boolean	No	If the value is true, the writable stream is not stopped when an exception occurs on the readable stream.
preventCancel	boolean	No	If the value is true, the writable stream is not closed when the readable stream is incorrect.
signal	AbortSignal	No	If `signal` is stopped, ongoing pipe operations are stopped.

pipeTo

```
readable.pipeTo(destination: WritableStream, options?: PipeToOptions): Promise<void
```

The pipeTo() method pipes the current readable stream to the `destination` writable stream.

Note:

During the piping, the `destination` of the current stream is locked.

Parameters

Parameter	Type	Required	Description
destination	WritableStream	Yes	The writable stream.
options	PipeToOptions	Yes	The configuration items for piping the stream.

tee

```
readable.tee(): [ReadableStream, ReadableStream];
```

The tee() method tees the current readable stream and returns two independent branches.

cancel

```
readable.cancel(reason?: string): Promise<string>;
```

The cancel() method ends the current stream.

References

MDN documentation: ReadableStream

Sample Functions: Merging Resources and Responding in Streaming Mode

Sample Functions: Rewriting a m3u8 File and Configuring Authentication

©2013-2025 Tencent Cloud International Pte. Ltd.

ReadableStreamBYOBReader

Last updated：2024-01-30 15:05:35

The **ReadableStreamBYOBReader** API defines a reader for a readable stream. It is designed based on the standard Web API ReadableStreamBYOBReader. `BYOB` is an abbreviation of bring your own buffer. A ReadableStreamBYOBReader object allows to read data from streams and write the read data to the buffer, thereby reducing replicas.

Note:

A `ReadableStreamBYOBReader` object cannot be constructed directly. You can use the ReadableStream.getReader method to construct a `ReadableStreamBYOBReader` object.

Overview

```
// Use TransformStream to construct a ReadableStream object.
const { readable } = new TransformStream();

// Use the ReadableStream object to obtain the reader.
const reader = readable.getReader({
  mode: 'byob',
});
```

Attributes

```
// readable.locked
readonly locked: boolean;
```

The locked attribute returns a Promise object. If the stream is closed, the status of the Promise object is `fulfilled`. If an exception occurs on the stream or the lock on the reader is released, the status of the Promise object is `rejected`.

Methods

read

```
reader.read(bufferView: ArrayBufferView): Promise<{value: ArrayBufferView, done: bo
```

The read() method reads data from the stream and writes the read data to the `bufferView` on the buffer.

Note:

You cannot initiate the next stream reading operation until the current stream reading operation ends.

Returned values

The `reader.read` method returns a Promise object that contains the read data and the reading status.

If a chunk is available, the Promise object is in the `fulfilled` status and contains an object in the `{ value: theChunk, done: false }` format.

If the stream is closed, the status of the Promise object is switched to `fulfilled`, and an object in the `{ value: theChunk, done: true }` format is contained.

If an exception occurs on the stream, the Promise object is in the `rejected` status, and the relevant error information is included.

cancel

```
reader.cancel(reason?: string): Promise<string>;
```

The cancel() method closes the stream and ends the reading operation.

releaseLock

```
reader.releaseLock(): void;
```

The releaseLock() method cancels the association with the stream and releases the lock on the stream.

References

MDN documentation: ReadableStreamBYOBReader

©2013-2025 Tencent Cloud International Pte. Ltd.

ReadableStreamDefaultReader

Last updated：2024-01-30 15:08:11

The **ReadableStreamDefaultReader** API defines a reader for a readable stream. It is designed based on the standard Web APIs ReadableStreamDefaultReader.

Note:

A `ReadableStreamDefaultReader` object cannot be constructed directly. You can use the ReadableStream.getReader method to construct a `ReadableStreamDefaultReader` object.

Overview

```
// Use TransformStream to construct a ReadableStream object.
const { readable } = new TransformStream();

// Use the ReadableStream object to obtain the reader.
const reader = readable.getReader();
```

Attributes

closed

```
// reader.closed
readonly closed: Promise<void>;
```

The closed attribute returns a Promise object. If the stream is closed, the status of the Promise object is `fulfilled`. If an exception occurs on the stream or the lock on the reader is released, the status of the Promise object is `rejected`.

Methods

read

```
reader.read(): Promise<{value: Chunk, done: boolean}>;
```

The read() method reads data from the stream.

Note:

You cannot initiate the next stream reading operation until the current stream reading operation ends.

Returned values

The `reader.read` method returns a Promise object that contains the read data Chunk and the reading status.
If a chunk is available, the Promise object is in the `fulfilled` status and contains an object in the `{ value: theChunk, done: false }` format.
If the stream is closed, the Promise object is in the `fulfilled` status and contains an object in the `{ value: undefined, done: true }` format.
If an exception occurs on the stream, the Promise object is in the `rejected` status, and the relevant error information is included.

Chunk

The `Chunk` parameter indicates the data to be read from the stream.

```
type Chunk = string | ArrayBuffer | ArrayBufferView;
```

cancel

```
reader.cancel(reason?: string): Promise<string>;
```

The cancel() method closes the stream and ends the reading operation.

releaseLock

```
reader.releaseLock(): void;
```

The releaseLock() method cancels the association with the stream and releases the lock on the stream.

References

MDN documentation: ReadableStreamDefaultReader

TransformStream

Last updated：2024-01-30 15:32:47

A **TransformStream** consists of a readable stream and a writable stream. It is designed based on the standard Web API TransformStream.

Constructor API

```
const { readable, writable } = new TransformStream(transformer?: any, writableStrat
```

Parameters

Parameter	Type	Required	Description
transformer	any	No	This parameter is not supported. The values do not take effect and are ignored automatically.
writableStrategy	WritableStrategy	No	The strategy for the writable side.

WritableStrategy

Parameter	Type	Required	Description
highWaterMark	number	Yes	The size of the writable buffer in bytes. Default value: 32K. Maximum value: 256K. If you enter a value greater than 256K, the value is changed to 256K automatically.

Attributes

readable

```
readonly readable: ReadableStream;
```

The readable stream. For more information, see ReadableStream.

writable

```
readonly writable: WritableStream;
```

©2013-2025 Tencent Cloud International Pte. Ltd.

The writable stream. For more information, see WritableStream.

Sample Code

```
async function handleEnterRoom() {
  // Generate readable streams and writeable streams.
  const { readable, writable } = new TransformStream();
  // Fetch a remote resource.
  const response = await fetch('https://www.tencentcloud.com/');
  // Respond to the client in streaming mode.
  response.body.pipeTo(writable);

  return new Response(readable, response);
}

addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});
```

References

MDN documentation: TransformStream

Sample Functions: Merging Resources and Responding in Streaming Mode

Sample Functions: Rewriting a m3u8 File and Configuring Authentication

©2013-2025 Tencent Cloud International Pte. Ltd.

WritableStream

Last updated：2024-01-30 16:00:52

The **WritableStream** API represents a writable stream or writable side. It is designed based on the standard Web API WritableStream.

Note:

A `WritableStream` object cannot be constructed directly. You can use TransformStream to construct a `WritableStream` object.

Overview

```
// Use TransformStream to construct a WritableStream object.
const { writable } = new TransformStream();
```

Attributes

locked

```
// writable.locked
readonly locked: boolean;
```

The locked attribute indicates whether the stream is locked.

Note:

A stream is locked in the following scenarios:

The stream has no more than one activated `writer`. Before the `writer` calls the `releaseLock()` method, the stream is locked.

The stream is in being piped. The stream is locked until the piping ends.

highWaterMark

```
// writable.highWaterMark
readonly highWaterMark: number;
```

The size of the writable buffer in bytes. Default value: 32K. Maximum value: 256K. If you enter a value greater than 256K, the value is changed to 256K automatically.

Methods

Note:

Before you use any of the following methods, make sure that the stream is not locked. Otherwise, an exception is returned.

getWriter

```
writable.getWriter(): WritableStreamDefaultWriter;
```

The getWriter() method creates a writer and locks the current stream until the writer calls the releaseLock() method. For more information about the returned values, see WritableStreamDefaultWriter.

close

```
writable.close(): Promise<void>;
```

The close() method closes the current stream.

abort

```
writable.abort(reason?: string): Promise<string>;
```

The abort() method stops the current stream.

References

MDN documentation: WritableStream

Sample Functions: Merging Resources and Responding in Streaming Mode

Sample Functions: Rewriting a m3u8 File and Configuring Authentication

©2013-2025 Tencent Cloud International Pte. Ltd.

WritableStreamDefaultWriter

Last updated：2024-01-30 16:03:02

The **WritableStreamDefaultWriter** API defines a writer for a writable stream. It is designed based on the standard Web API WritableStreamDefaultWriter.

Note:

A `WritableStreamDefaultWriter` object cannot be constructed directly. You can use the WritableStream.getWriter method to construct a `WritableStreamDefaultWriter` object.

Overview

```
// Use TransformStream to construct a WritableStream object.
const { writable } = new TransformStream();

// Use the WritableStream object to obtain the writer.
const writer = writable.getWriter();
```

Attributes

closed

```
// writer.closed
readonly closed: Promise<void>;
```

The closed attribute returns a Promise object. If the stream is closed, the status of the Promise object is `fulfilled`. If an exception occurs on the lock on the writer is released, the status of the Promise object is `rejected`.

ready

```
// writer.ready
readonly ready: Promise<void>;
```

The ready attribute returns a Promise object. When the size required by the internal queue of the stream changes from non-positive to positive, the Promise object is in the fulfilled status, indicating that it no longer applies backpressure.

desiredSize

```
// writer.desiredSize
readonly desiredSize: number;
```

The desiredSize attribute returns the size required to fill the internal queue of the stream.

Methods

write

```
writer.write(chunk: Chunk): Promise<void>;
```

The write() method writes the `Chunk` data to the stream.

Note:

You cannot call the `write` method to initiate the next stream writing operation until the current stream writing operation ends.

Parameters

Parameter	Type	Required	Description
chunk	Chunk	Yes	The chunk of data to be written to the stream.

Chunk

The `Chunk` parameter indicates the data to be written to the stream.

```
type Chunk = string | ArrayBuffer | ArrayBufferView;
```

close

```
writer.close(): Promise<void>;
```

The close() method closes the current stream.

abort

```
writer.abort(reason?: string): Promise<string>;
```

The abort() method stops the current stream.

releaseLock

```
writer.releaseLock(): void;
```

©2013-2025 Tencent Cloud International Pte. Ltd.

The releaseLock() method cancels the association with the stream and releases the lock on the stream.

References

MDN documentation: WritableStreamDefaultWriter

Web Crypto

Last updated：2025-06-23 14:59:01

The **Web Crypto API** is designed based on the standard Web API Web Crypto API. The Web Crypto API provides a set of functions for common cryptographic operations. Performing cryptographic operations by using the Web Crypto API is significantly faster than performing them purely in JavaScript.

Note:

The `Crypto` object cannot be constructed directly. It is globally injected during the runtime of edge functions. Directly use the global crypto instance.

Overview

```
// Perform encoding.
const encodeContent = new TextEncoder().encode('hello world');
// Use crypto to perform SHA-256 hashing. The resulting hash is a promise that fulf
const sha256Content = await crypto.subtle.digest(
  { name: 'SHA-256' },
  encodeContent
);
const result = new Uint8Array(sha256Content);
```

Attributes

```
// crypto.subtle
readonly subtle: SubtleCrypto;
```

The Web Crypto API supports common cryptographic operations, such as hashing, signature signing and verification, and encryption and decryption. For more information, see SubtleCrypto.

Methods

getRandomValues

```
crypto.getRandomValues(buffer: TypedArray): TypedArray;
```

The getRandomValues() method generates a random value, fills the buffer with the random value, and returns the buffer.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameters

Parameter	Type	Required	Description
buffer	Int8Array |Uint8Array |Uint8ClampedArray |Int16Array |Uint16Array |Int32Array |Uint32Array |BigInt64Array |BigUint64Array	Yes	The buffer of random values. The value cannot exceed 65,536 bytes in length. For more information, see TypedArray.

randomUUID

```
crypto.randomUUID(): string;
```

The randomUUID() method generates a new random (version 4) UUID.

SubtleCrypto

The SubtleCrypto API supports common cryptographic operations, such as hashing, signature signing and verification, and encryption and decryption. For more information, see SubtleCrypto.

Note:

SubtleCrypto methods are classified into the following types based on features:

Encryption methods: `encrypt/decrypt` , `sign/verify` , and `digest` . Such methods can be used to implement security-related features such as privacy and identity verification.

Key management methods: `generateKey` , `deriveKey` , and `importKey/exportKey` . Such methods can be used to manage keys.

digest

```
crypto.subtle.digest(algorithm: string | object, data: ArrayBuffer): Promise<ArrayB
```

The decrypt() method returns a Promise object that fulfills with the generated data digest (hash). For more information, see SubtleCrypto.digest.

encrypt

```
crypto.subtle.encrypt(algorithm: object, key: CryptoKey, data: ArrayBuffer): Promis
```

The encrypt() method returns a Promise object that fulfills with the encrypted data. For more information, see SubtleCrypto.encrypt.

For the `RSA-OAEP` algorithm, the data length must not exceed modulusLength/8 - (2 * hLen) - 2, where hLen is determined as follows:

SHA-1: hLen = 20 bytes

SHA-256: hLen = 32 bytes

SHA-384: hLen = 48 bytes

SHA-512: hLen = 64 bytes

For the `AES-CTR`, `AES-CBC`, and `AES-GCM` algorithms, the data length is limited to 1 MB.

decrypt

```
crypto.subtle.decrypt(algorithm: object, key: CryptoKey, data: ArrayBuffer): Promis
```

The decrypt() method returns a Promise object that fulfills with the decrypted data. For more information, see SubtleCrypto.decrypt.

For the `RSA-OAEP` algorithm, the data length is modulusLength/8.

For the `AES-CTR`, `AES-CBC`, and `AES-GCM` algorithms, the data length is limited to 1 MB.

sign

```
crypto.subtle.sign(algorithm: string | object, key: CryptoKey, data: ArrayBuffer):
```

The sign() method returns a Promise object that fulfills with the signature. For more information, see SubtleCrypto.sign.

verify

```
crypto.subtle.verify(algorithm: string | object, key: CryptoKey, signature: BufferS
```

The verify() method returns a Promise object that fulfills with the signature verification result. For more information, see SubtleCrypto.verify.

generateKey

```
crypto.subtle.generateKey(algorithm: object, extractable: boolean, keyUsages: Array
```

The generateKey() method returns a Promise object that fulfills with a CryptoKey or a CryptoKeyPair. For more information, see SubtleCrypto.generateKey.

deriveKey

```
crypto.subtle.deriveKey(algorithm: object, baseKey: CryptoKey, derivedKeyAlgorithm:
```

The deriveKey() method returns a Promise object that fulfills with a CryptoKey. For more information, see SubtleCrypto.deriveKey.

©2013-2025 Tencent Cloud International Pte. Ltd.

importKey

```
crypto.subtle.importKey(format: string, keyData: BufferSource, algorithm: string |
```

The importKey() method returns a Promise object that fulfills with a CryptoKey. For more information, see SubtleCrypto.importKey.

exportKey

```
crypto.subtle.exportKey(format: string, key: CryptoKey): Promise<ArrayBuffer>;
```

The exportKey() method returns a Promise object that fulfills with an ArrayBuffer containing the key. For more information, see SubtleCrypto.exportKey.

deriveBits

```
crypto.subtle.deriveBits(algorithm: object, baseKey: CryptoKey, length: integer): P
```

The deriveBits() method returns a Promise object that fulfills with an ArrayBuffer of pseudo-random bits. For more information, see SubtleCrypto.deriveBits.

wrapKey

```
crypto.subtle.wrapKey(format: string, key: CryptoKey, wrappingKey: CryptoKey, wrapA
```

The wrapKey() method returns a Promise object that fulfills with an ArrayBuffer containing the wrapped key. For more information, see SubtleCrypto.wrapKey.

unwrapKey

```
crypto.subtle.unwrapKey(format: string, wrappedKey: ArrayBuffer, unwrappingKey: Cry
```

The unwrapKey() method returns a Promise object that fulfills with the unwrapped CryptoKey. For more information, see SubtleCrypto.unwrapKey.

CryptoKey

A `CryptoKey` represents a key that is generated by using an encryption algorithm. For more information, see CryptoKey. The CryptoKey object cannot be constructed directly. You can use the following methods to generate CryptoKey objects:

crypto.subtle.generateKey

crypto.subtle.importKey
crypto.subtle.deriveKey
crypto.subtle.unwrapKey

The following table describes the `CryptoKey` attributes.

Attribute	Type	Read-only	Description
type	string	Yes	The type of the key.
extractable	boolean	Yes	Specifies whether the key can be exported.
algorithm	object	Yes	The algorithm for which this key can be used and the associated parameters.
usages	Array<string>	Yes	The usage of the key.

CryptoKeyPair

A `CryptoKeyPair` represents a key pair that is generated by using an encryption algorithm. For more information, see CryptoKeyPair. The CryptoKeyPair object cannot be constructed directly. You can use the following method to generate CryptoKeyPair objects:

crypto.subtle.generateKey

The following table describes the `CryptoKeyPair` attributes.

Attribute	Type	Read-only	Description
privateKey	CryptoKey	Yes	The private key. For encryption and decryption algorithms, this key is used for decryption. For signature signing and verification algorithms, this key is used for signature signing.
publicKey	CryptoKey	Yes	The public key. For encryption and decryption algorithms, this key is used for encryption. For signature signing and verification algorithms, this key is used for signature verification.

Supported Algorithms

Edge Functions supports all algorithms that are defined in the WebCrypto specification. The following table describes the algorithms.

Algorithm	encrypt()	sign()	wrapKey()	deriveKey()	generateKey()	importKey()	exportK

	decrypt()	verify()	unwrapKey()	deriveBits()			
RSASSA-PKCS1-v1_5	-	✓	-	-	✓	✓	✓
RSA-PSS	-	✓	-	-	✓	✓	✓
RSA-OAEP	✓	-	✓	-	✓	✓	✓
ECDSA	-	✓	-	-	✓	✓	✓
ECDH	-	-	-	✓	✓	✓	✓
HMAC	-	✓	-	-	✓	✓	✓
AES-CTR	✓	-	✓	-	✓	✓	✓
AES-CBC	✓	-	✓	-	✓	✓	✓
AES-GCM	✓	-	✓	-	✓	✓	✓
AES-KW	-	-	✓	-	✓	✓	✓
HKDF	-	-	-	✓	-	✓	-
PBKDF2	-	-	-	✓	-	✓	-
SHA-1	-	-	-	-	-	-	-
SHA-256	-	-	-	-	-	-	-
SHA-384	-	-	-	-	-	-	-
SHA-512	-	-	-	-	-	-	-
MD5	-	-	-	-	-	-	-

Sample Code

```
function uint8ArrayToHex(arr) {
    return Array.prototype.map.call(arr, (x) => ((`0${x.toString(16)}`).slice(-2)))
}
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
async function handleEvent(event) {
    const encodeArr = TextEncoder().encode('hello world');
    // Execute MD5.
    const md5Buffer = await crypto.subtle.digest({ name: 'MD5' }, encodeArr);
    // Generate a hexadecimal string.
    const md5Str = uint8ArrayToHex(new Uint8Array(md5Buffer));

    const response = new Response(md5Str);
    return response;
}

addEventListener('fetch', async (event) => {
  event.respondWith(handleEvent(event));
});
```

References

MDN documentation: Web Crypto API

MDN documentation: SubtleCrypto

MDN documentation: CryptoKey

MDN documentation: CryptoKeyPair

Sample Functions: Protecting Data from Tampering

Sample Functions: Rewriting a m3u8 File and Configuring Authentication

Sample Functions: Caching POST Requests

©2013-2025 Tencent Cloud International Pte. Ltd.

Web standards

Last updated：2024-11-18 14:57:21

Edge Functions designs a serverless code execution environment based on the **V8 JavaScript engine** and provides the following standardized Web APIs.

JavaScript Standard Built-in Objects

Edge Functions supports all JavaScript standard built-in objects. For more information, see Standard built-in objects.

URL

```
const urlInfo = new URL('https://www.tencentcloud.com/');
```

The URL API is used to parse, construct, standardize, and encode URLs. For more information, see URL.

Blob

```
const blob = new Blob(['hello', 'world'], { type: 'text/plain' });
```

The Blob API represents a file-like object of immutable and raw data. For more information, see Blob.

Base64

btoa

```
function btoa(data: string | ArrayBuffer | ArrayBufferView): string;
```

The btoa() method performs Base64 encoding. Unicode strings are not supported. For more information, see btoa().

atob

```
function atob(data: string): string;
```

The atob() method performs Base64 decoding. Unicode strings are not supported. For more information, see atob().

©2013-2025 Tencent Cloud International Pte. Ltd.

btoaUTF8

```
function btoaUTF8(data: string): string;
```

The btoaUTF8() method performs Base64 encoding. Unicode strings are supported.

atobUTF8

```
function atobUTF8(data: string): string;
```

The atobUTF8() method performs Base64 decoding. Unicode strings are supported.

Timer

setTimeout

```
setTimeout(func: function): number;
setTimeout(func: function, delay: number): number;
setTimeout(func: function, delay: number, ...args: any[]): number;
```

Functioning as an ordinary timer, it executes the designated function upon expiration. For more details, see setTimeout.

clearTimeout

```
clearTimeout(timeoutID: number): void;
```

It is an ordinary timer to clear the designated `timeoutID`. For more information, see clearTimeout.

setInterval

```
setInterval(func: function): number;
setInterval(func: function, delay: number): number;
setInterval(func: function, delay: number, ...args: any[]): number;
```

Functioning as a recurring timer, it implemets the designated function upon expiration. For more information, see setInterval.

clearInterval

```
clearInterval(intervalID: number): void;
```

It is used to clear a recurring timer. For more details, see clearInterval.

setImmediate

```
setImmediate(func: function): number;
setImmediate(func: function, ...args: any[]): number;
```

Functioning as an immediate timer, it executes the designated function after the EDGE-FUNCTION stack has been cleared. For more information, see setImmediate.

clearImmediate

```
clearImmediate(immediateID: number): void;
```

It is used to clear an immediate timer. For more information, see setImmediate.

EventTarget and Event

EventTarget

```
const eventTarget = new EventTarget();
```

The EventTarget API allows objects to publish and subscribe to events. For more information, see EventTarget.

Event

```
const event = new Event('type name');
```

The Event API represents a basic event. For more information, see Event.

AbortSignal and AbortController

AbortSignal

```
const signal = AbortSignal.abort();
```

The AbortSignal API represents a signal object that allows you to stop a request. For more information, see AbortSignal.

AbortController

```
const controller = new AbortController();
```

The AbortController API represents a controller object that allows you to stop a request. For more information, see AbortController.

CompressionStream and DecompressionStream

CompressionStream

```
const { readable, writable } = new CompressionStream('gzip');
```

CompressionStream supports compression methods including `gzip`, `deflate`, and `br`. For more details, see CompressionStream.

DecompressionStream

```
const { readable, writable } = new DecompressionStream('gzip');
```

DecompressionStream supports decompression methods including `gzip`, `deflate`, and `br`. For more details, see DecompressionStream.

©2013-2025 Tencent Cloud International Pte. Ltd.

Images
ImageProperties

Last updated：2024-11-15 16:47:37

OrientationEdge functions support custom image processing behaviors when you initialize a Fetch request or initializing a Request object, through the ImageProperties parameter in RequestInitEoProperties. ImageProperties is a non-standard option not included in the Web APIs. Here is how to use it.
When you initialize a Fetch request, set the ImageProperties parameter as follows:

```
addEventListener('fetch', (event) => {
  const response = fetch(event.request, { eo: { image: { format: "avif" } } });
  event.respondWith(response);
});
```

When you initialize a Request object, set the ImageProperties parameter as follows:

```
async function handleRequest() {
  const request = new Request("https://www.example.com/test.jpg", { eo: { image: {
  const response = await fetch(request);
  return response;
}

addEventListener('fetch', (event) => {
  event.respondWith(handleRequest());
});
```

Edge Functions support multiple image processing capabilities by setting different ImageProperties parameters. This includes Image Format Conversion, Image Quality Conversion, Image Scaling, Image Cropping, with specific configurations as follows.

Format Conversion

By specifying the `format` parameter, the original image is converted to the specified format.

Parameter Name	Type	Required	Description
format	string	No	Convert the original image to a specified format. Supports for `jpg`, `gif`, `png`, `bmp`, `webp`, `avif`, `jp2`, `jxr`, `heif`.
avoidSizeIncrease	boolean	No	When this parameter is set, if the processed image size

			exceeds that of the original, the original image will be returned without processing.

Example:

```
// Convert the original image to AVIF format.
eo: { image: { format: "avif" } }
```

Quality Conversion

By specifying the `quality` parameter, the quality of the image is adjusted.

Parameter Name	Type	Required	Description
quality	string | number	No	The absolute quality of the image, with a value range of 0 to 100. Use the minimum value between the original image quality and the specified quality. Append "!" after quality to enforce the specified value, for example, "90!".
rquality	string | number	No	The relative quality of the image, with a value range of 0 to 100, is based on the original image quality. For example, if the original image quality is 80 and rquality is set to 80, the resulting processed image will have a quality of 64 (calculated as 80x80%).
lquality	string | number	No	The minimum quality of the image, with a value range of 0 to 100. Set the minimum value of the quality parameter for the result image. For example, if the original image quality is 85, and lquality is set to 80, the quality of the processed image remains at 85. For example, if the original image quality is 60, and lquality is set to 80, the quality of the processed image will be enhanced to 80.
avoidSizeIncrease	boolean	No	When this parameter is set, if the processed image size exceeds that of the original, the original image will be returned without processing.

Note:

The quality conversion parameters have an order of precedence. By default, `lquality` > `quality` > `rquality`. If all are configured simultaneously, only the parameter with the highest priority takes effect.

Example:

```
// Set the absolute quality of the image to 80.
eo: { image: { quality: 80 } }

// Set the relative quality of the image to 80.
eo: { image: { rquality: 80 } }

// Set the minimum quality of the image to 80.
eo: { image: { lquality: 80 } }
```

Image Scaling

Image scaling operations are controlled by the `fit` parameter. By setting different `fit` parameters, various types of scaling operations can be achieved.

Parameter Name	Type	Required	Description
fit	string	No	Set different image scaling and cropping modes, with the following supported values for the fit parameter: contain: (Default) Scale the image while maintaining its aspect ratio. cover: Scale the image without maintaining its aspect ratio according to pixel values. percent: Scale the image without maintaining its aspect ratio according to a percentage value.

Note:
The behaviors corresponding to different `fit` parameters will be explained in detail below.

Proportional Scaling

When the `fit` parameter is not set or the `fit` parameter is set to `contain`, the image is scaled while maintaining its aspect ratio.

Parameter Name	Type	Required	Description
width	string | number	No	Set the target image width to be scaled to width pixels. When only the width parameter is set, the height is scaled proportionally.
height	string | number	No	Set the target image height to be scaled to height pixels. When only the height parameter is set, the width is scaled proportionally.
long	string |	No	Set the target image's long edge to be scaled to long pixels. Set the

	number		short edge to be scaled proportionally.
short	string | number	No	Set the target image's short edge to be scaled to short pixels. Set the long edge to be scaled proportionally.

Note:

The above parameters have an order of precedence, with `long` > `short` > (`width` | `height`). If multiple parameters are set simultaneously, only the parameter with the highest priority takes effect.

If both `width` and `height` parameters are set simultaneously, they define the maximum dimensions (`width` and `height`) for the thumbnail, with both width and height being scaled proportionally.

In addition to the above features, proportional scaling based on the number of pixels is also supported.

Parameter Name	Type	Required	Description
area	string | number	No	For proportional scaling, the total number of pixels in the scaled image does not exceed area.

Example:

```
// Set proportional scaling based on the long edge, with the length of the scaled l
eo: { image: { long: 100 } }

// Set proportional scaling based on the short edge, with the length of the scaled
eo: { image: { short: 100 } }

// Set proportional scaling based on the width, with the scaled width being 100 px.
eo: { image: { width: 100 } }

// Set proportional scaling based on the height, with the scaled height being 100 p
eo: { image: { height: 100 } }

// Set proportional scaling based on both height and width, with the maximum scaled
eo: { image: { width: 100, height: 100 } }

// Set proportional scaling based on the total number of pixels, with the total sca
eo: { image: { area: 10000 } }
```

Non-Proportional Scaling

When the `fit` parameter is set to `cover` , the image is scaled without maintaining its aspect ratio according to pixel values.

Parameter	Type	Required	Description

Name			
width	string | number	No	Set the target image width to be scaled to width pixels.
height	string | number	No	Set the target image height to be scaled to height pixels.

Note:

In this mode, both the `width` and `height` parameters must be set simultaneously. If only one is set, the image will be processed to either `width` x `width` or `height` x `height` .

Example:

```
// Set non-aspect ratio scaling based on pixel values, scaling the width to 100 px
eo: { image: { fit: 'cover', width: 100, height: 100 } }
```

When the `fit` parameter is set to `percent` , the image is scaled without maintaining its aspect ratio according to a percentage value.

Parameter Name	Type	Required	Description
width	string | number	No	Set the target image width to be scaled to width%, with a value range of 0 to 100. When only the width parameter is set, the height remains unchanged.
height	string | number	No	Set the target image height to be scaled to height%, with a value range of 0 to 100. When only the height parameter is set, the width remains unchanged.

Note:

In this mode, the `width` and `height` parameters operate independently. If both `width` and `height` are set simultaneously, the image will be processed to `width%` x `height%` .

Example:

```
// Set non-proportional scaling based on width percentage, with the scaled width be
eo: { image: { fit: 'percent', width: 50 } }

// Set non-proportional scaling based on height percentage, with the scaled height
eo: { image: { fit: 'percent', height: 50 } }

// Set non-proportional scaling based on both width and height percentages, with th
eo: { image: { fit: 'percent', width: 50, height: 50 } }
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Image Cropping

Image cropping is controlled by the `fit` parameter. By setting different `fit` parameters, various types of cropping operations can be achieved.

Parameter Name	Type	Required	Description
fit	string	No	Set different image scaling and cropping modes, with the following supported values for the fit parameter: cut: Perform cropping to the image. crop: Perform scaling and cropping to the image.

Note:

The behaviors corresponding to different `fit` parameters will be explained in detail below.

Original Image Cropping

When the `fit` parameter is set to `cut`, the original image is cropped.

Parameter Name	Type	Required	Description
width	string | number	No	Crop the target image to a width of width pixels. When only the width parameter is set, the height remains unchanged.
height	string | number	No	Crop the target image to a height of height pixels. When only the height parameter is set, the width remains unchanged.
gravity	string | { [key: string]: number | string }	No	The anchor point for image cropping. Can be set to 3x3 grid orientation values or coordinate object. For orientation values, see the 3x3 Grid Orientation Diagram. The coordinate object is in the form of { x: 100, y: 100 }. x represents a horizontal offset of x pixels from the top-left corner of the image rightwards. y represents a vertical offset of y pixels from the top-left corner of the image downwards.

Note:

If the `gravity` parameter is not set, the default cropping will occur at the top-left corner (northwest).

Example:

```
// Set to perform original cropping based on width, with the cropped width being 10
eo: { image: { fit: 'cut', width: 100 } }

// Set to perform original cropping based on height, with the cropped height being
eo: { image: { fit: 'cut', height: 100 } }

// Set to perform original cropping based on both width and height, with the croppe
eo: { image: { fit: 'cut', width: 100, height: 100 } }

// Set to perform original cropping based on both width and height, specifying the
eo: { image: { fit: 'cut', width: 100, height: 100, gravity:'center' } }

// Set to perform original cropping based on both width and height, specifying the
eo: { image: { fit: 'cut', width: 100, height: 100, gravity: { x: 100, y: 100 } }
```

Scaled Cropping

When the `fit` parameter is set to `crop`, the image is scaled and cropped.

Parameter Name	Type	Required	Description
width	string | number	No	Set the target image width to be cropped to width pixels. When only the width parameter is set, the height remains unchanged.
height	string | number	No	Set the target image height to be cropped to height pixels. When only the height parameter is set, the width remains unchanged.
gravity	string	No	The anchor point for image cropping. Can be set to 3x3 grid orientation values. For orientation values, see the 3x3 Grid Orientation Diagram.

Note:

In this mode, the `gravity` parameter must be set to 3x3 grid orientation values. If the `gravity` parameter is not set, it defaults to be cropped at the center point (center).

Example:

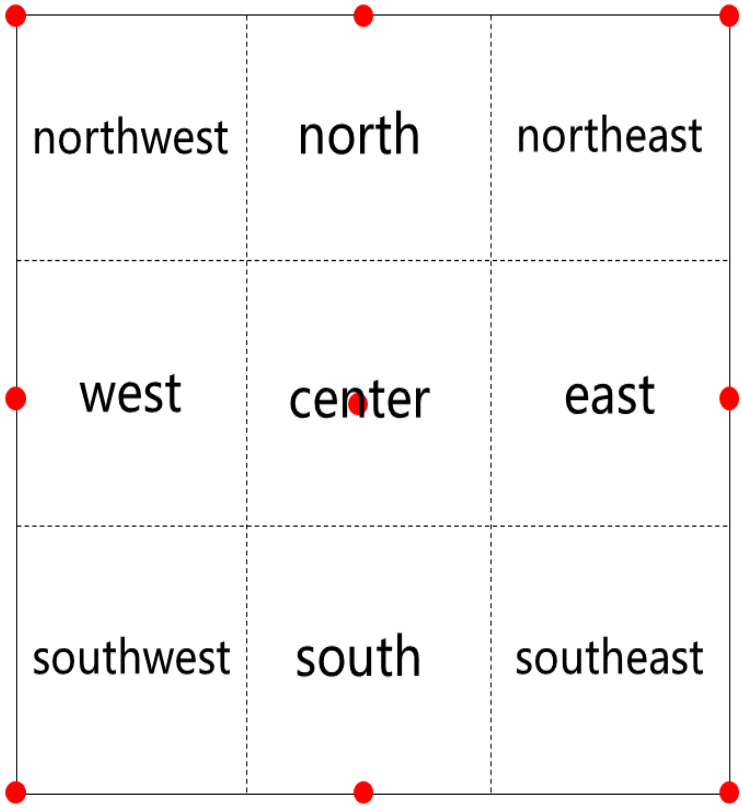
```
// Set to perform scaled cropping based on width, with the cropped width being 100
eo: { image: { fit: 'crop', width: 100 } }

// Set to perform scaled cropping based on height, with the cropped height being 10
eo: { image: { fit: 'crop', height: 100 } }

// Set to perform scaled cropping based on both width and height, with the cropped
eo: { image: { fit: 'crop', width: 100, height: 100 } }
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
// Set to perform scaled cropping based on both width and height, specifying an anc
eo: { image: { fit: 'crop', width: 100, height: 100, gravity:'northwest' } }
```

3x3 Grid Orientation Diagram

The 3x3 grid orientation diagram provides positional references for various image operations. The red dots serve as origin points for each region (after regions being selected via the gravity parameter, displacement operations reference the respective origins).



Sample Functions
Example Overview

Last updated：2024-08-27 10:09:40

Content Generation

Title	Example Introduction
Returning an HTML Page	Validates the request header and value. Responds if the validation is passed, and rejects if not.
Returning a JSON Object	Generates a JSON object and responds to the client with it.

Access Control

Title	Example Introduction
Authenticating a Request Header	Validates the request header and value. Responds if the validation is passed, and rejects if not.
Protecting Data from Tampering	Calculates the SHA-256 signature of the request body and compares it with the signature generated by the origin server. If they are consistent, it indicates that the content has not been tampered with; otherwise, a 416 status code is returned for response.
Rewriting a m3u8 File and Configuring Authentication	Controls the permissions for .m3u8 and .ts fragment resources.
Custom Referer Rules	Implements custom access control based on the Referer request header.
Remote Authentication	Forwards the request to a specified remote authentication server. Allows access if the validation is passed and denies if not.
HMAC Digital Signature	Uses the Web Crypto API to implement HMAC-SHA256 signature and stores the signature information in the request header, to complete data integrity check or identity authentication in cooperation with the origin server.

©2013-2025 Tencent Cloud International Pte. Ltd.

Redirect

Title	Example Introduction
Performing Redirect Based on the Request Location	Implements an automatic redirect to the target URL of the client region.
301 Redirect	Implements an automatic and permanent 301 redirect of client requests to the preset website address.
Batch Redirect	Through the predefined redirect mapping table, implements an automatic redirect of specific paths to corresponding URLs.

Obtaining Client Information

Title	Example Introduction
Obtaining Client IP Address	Obtains the client IP address through the rule engine and responds to the client.
Obtaining Client URL Information	Based on the incoming HTTP request, returns the detailed information of the request URL as an HTML page.

Geo Location

Title	Example Introduction
Customization Based on Client Geo Location	Provides customized content based on the country where the client is located.
Obtaining Client Geo Location Information	Responds with the client geo location information, including ASN, country name, country code, region name, region code, city name, longitude, and latitude.

Image Processing

Title	Example Introduction
Adaptive Image Resize	Implements adaptive scaling based on the User-Agent request header.

©2013-2025 Tencent Cloud International Pte. Ltd.

Image Adaptive WebP	Implements automatic format conversion to WebP based on the Accept request header.

Header Processing

Title	Example Introduction
Modifying a Response Header	Customizes the response headers by adding, modifying, and deleting.
Naming a Downloaded File	Modifies the Content-Disposition information in the response header to change the name of the downloaded file based on the fileName parameter in the request URL.

Cache

Title	Example Introduction
Using the Cache API	Obtains remote resources through the Fetch API and caches resources to EdgeOne edge nodes through the Cache API.
Caching POST Requests	Calculates an SHA-256 signature as the cache key for the POST request body and caches it to an edge node using the Cache API.

Responding in Streaming Mode

Title	Example Introduction
Responding in Streaming Mode	Obtains remote resources and responds to the client in streaming mode.
Merging Resources and Responding in Streaming Mode	Remotely obtains multiple video resources for reading and concatenation in streaming mode, and then responds to the client.

Fetch

Title	Example Introduction

©2013-2025 Tencent Cloud International Pte. Ltd.

Fetch Remote Resources	Obtains remote resources through the Fetch API and responds to the client.

Cookie

Title	Example Introduction
Setting Cookies	Obtains remote resources through the Fetch API and responds to the client.

Testing

Title	Example Introduction
Performing an A/B Test	Saves session information through Cookies and performs A/B testing on requests.

©2013-2025 Tencent Cloud International Pte. Ltd.

301 Redirect

Last updated：2024-08-26 17:09:00

This example demonstrates how to use the HTTP 301 status code to automatically and permanently redirect client requests to a preset website address. It is commonly used for permanent site migration or merging.

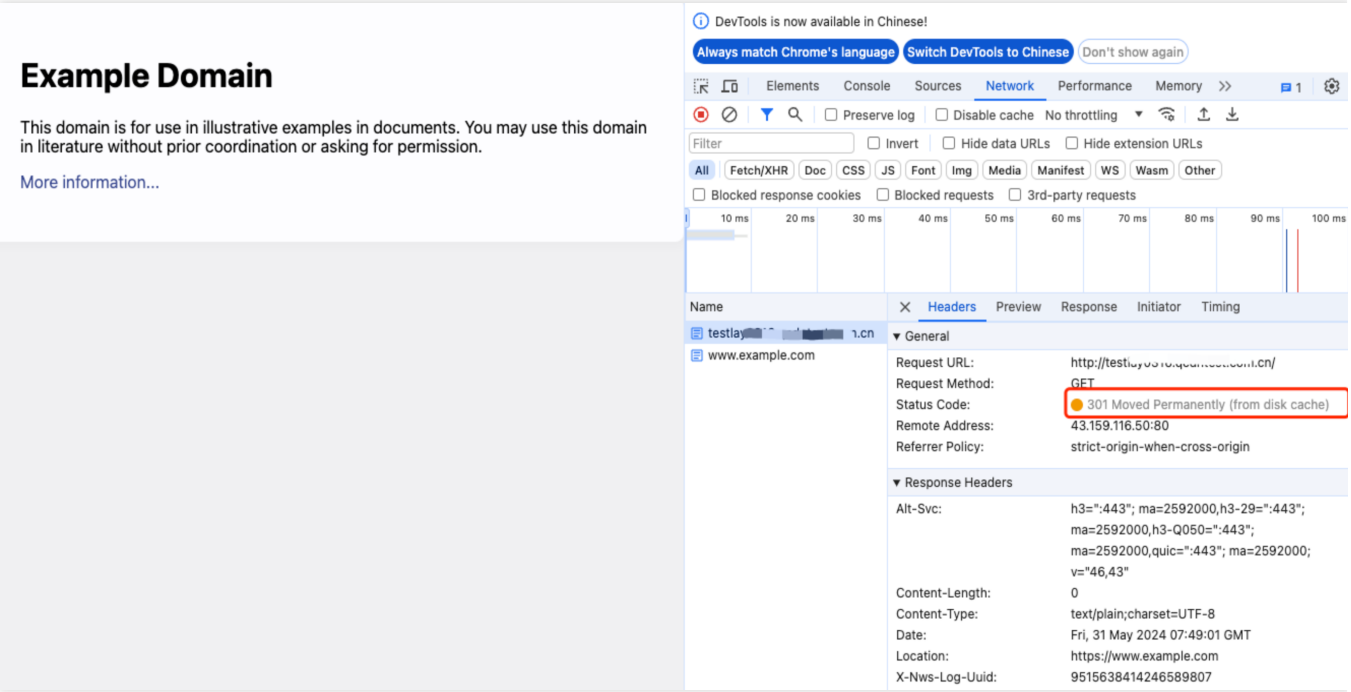
Sample Code

```
// Target redirect address
const destinationLocation = 'https://www.example.com';

// HTTP status code, used for permanent redirects
const statusCode = 301;

// Async function used to handle requests
async function handleRequest(request) {
    // Return a redirect response with the specified target location and status cod
    return Response.redirect(destinationLocation, statusCode);
}

// Add a fetch event listener to intercept requests and respond using the handleReq
addEventListener("fetch", event => {
    event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a triggering rule of the edge function to preview the effect of the sample code.

©2013-2025 Tencent Cloud International Pte. Ltd.



Related References

Runtime APIs: Request

Runtime APIs: Response

Obtaining Client URL Information

Last updated：2024-08-26 17:10:10

This example captures incoming HTTP requests and returns an HTML page displaying the detailed information about the request URL. It can be used for debugging and displaying the request parameters, paths, and sources.

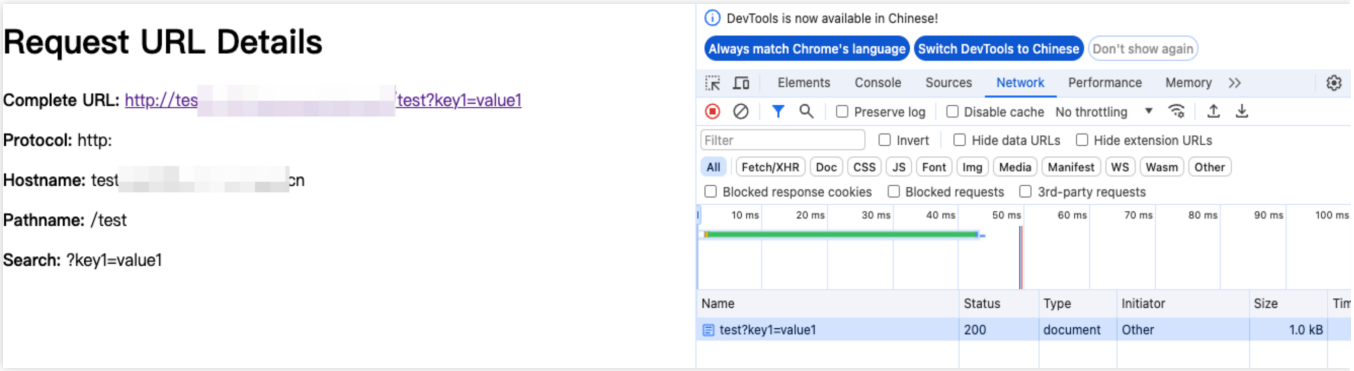
Sample Code

```
// Add a fetch event listener, which is triggered when a request is incoming. It us
addEventListener('fetch', event => {
    event.respondWith(handleRequest(event.request))
})

function handleRequest(request) {
    // Parse the request URL.
    const url = new URL(request.url)

    // Extract various components of the URL.
    const {
        href, // Full URL
        protocol, // Protocol (e.g., http:)
        hostname, // Host name (e.g., example.com)
        port, // Port (if specified)
        pathname, // Path (e.g., /path)
        search, // Query string (e.g., ?query=123)
        hash // Fragment after the hash sign (e.g., #section)
    } = url

    // Construct the HTML response content.
    const htmlContent = `
        <!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
            <title>Request URL Details</title>
        </head>
        <body>
            <h1>Request URL Details</h1>
            <p><strong>Complete URL:</strong> <a href="${href}">${href}</a></p>
            <p><strong>Protocol:</strong> ${protocol}</p>
            <p><strong>Hostname:</strong> ${hostname}</p>
```

```
            ${port ? `<p><strong>Port:</strong> ${port}</p>` : ''}
            <p><strong>Pathname:</strong> ${pathname}</p>
            <p><strong>Search:</strong> ${search}</p>
            ${hash ? `<p><strong>Hash:</strong> ${hash}</p>` : ''}
        </body>
        </html>
    `;

    // Return a response in HTML format.
    return new Response(htmlContent, {
        headers: {
            'Content-Type': 'text/html;charset=UTF-8'
        }
    })
}
```

Sample Preview

In the address bar of the browser, enter a URL that matches a triggering rule of the edge function to preview the effect of the sample code.



Related References

Runtime APIs: Request
Runtime APIs: Response

Customization Based on Client Geo Location

Last updated：2024-08-27 10:09:19

This example captures incoming HTTP requests and provides customized welcome messages and geo location (latitude and longitude) information based on the country where the client is located. It can be used for offering global personalized customization experience to users.

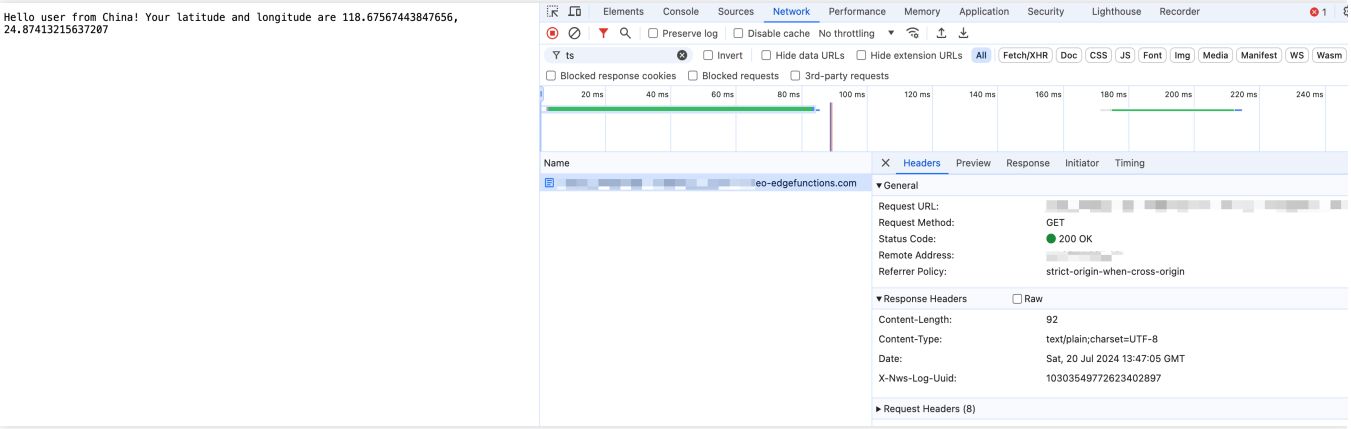
Sample Code

```
// Add a fetch event listener, which is triggered when a request is incoming. It us
addEventListener('fetch', event => {
    event.respondWith(handleRequest(event.request))
})
function handleRequest(request) {
    // Get the country code provided by EdgeOne.
    const countryCode = request.eo.geo.countryCodeAlpha2;
    // Select the corresponding language and welcome message based on the country c
    let responseText;
    switch (countryCode) {
        case 'CN': // China
            responseText =`Hello user from China! Your latitude and longitude are $
            break;
        case 'KR': // South Korea
            responseText = `한국의 사용자님 반갑습니다! 당신의 위도와 경도는 ${request.eo.
            break;
        case 'DE': // Germany
            responseText = `Willkommen in Deutschland! Ihre Breiten- und Längengrad
            break;
        case 'US': // USA
            responseText = `Hello from the USA! Your latitude and longitude are ${r
            break;
        default: // Respond in English by default in other cases.
            responseText = `Welcome to our service! Your latitude and longitude are
            break;
    }
    // Return a response.
    return new Response(responseText, {
        headers: {
            'Content-Type': 'text/plain;charset=UTF-8'
        }
    })
}
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Sample Preview

In the address bar of the browser, enter a URL that matches a triggering rule of the edge function to preview the effect of the sample code. If the client of the current request URL is located in China, the browser will respond with the welcome page for China, as well as the latitude and longitude of the client geo location.



Related References

Runtime APIs: addEventListener

Runtime APIs: Request

Runtime APIs: Response

Obtaining Client Geo Location Information

Last updated：2024-08-26 17:16:47

This example captures HTTP requests and responds with the user's geo location information, to generate an HTML page containing the autonomous system number (ASN), country, region, city, longitude and latitude information. It can be used for debugging or displaying the geo location information.

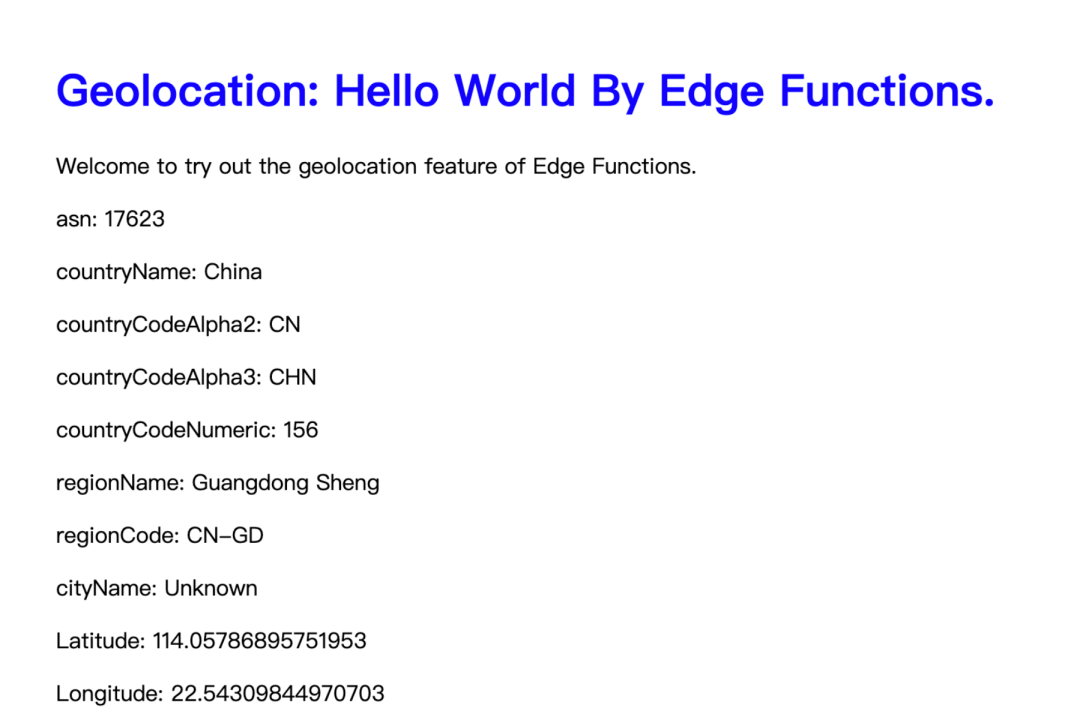
Sample Code

```
// Add a fetch event listener, which is triggered when a request is incoming. It us
addEventListener('fetch', event => {
    event.respondWith(handleRequest(event.request));
  });

  // Define an async function handleRequest, which is used to handle the incoming r
  async function handleRequest(request) {
    // Initialize an empty string to store the HTML content.
    let html_content = "";
    // Define the HTML style.
    let html_style = "body{padding:6em; font-family: sans-serif;} h1{color:#0000ff;

    // Get the user's geo location information.
    html_content += "<p> asn: " + request.eo.geo.asn + "</p>"; // ASN
    html_content += "<p> countryName: " + request.eo.geo.countryName + "</p>"; // C
    html_content += "<p> countryCodeAlpha2: " + request.eo.geo.countryCodeAlpha2 +
    html_content += "<p> countryCodeAlpha3: " + request.eo.geo.countryCodeAlpha3 +
    html_content += "<p> countryCodeNumeric: " + request.eo.geo.countryCodeNumeric
    html_content += "<p> regionName: " + request.eo.geo.regionName + "</p>"; // Reg
    html_content += "<p> regionCode: " + request.eo.geo.regionCode + "</p>"; // Reg
    html_content += "<p> cityName: " + request.eo.geo.cityName + "</p>"; // City na
    html_content += "<p> Latitude: " + request.eo.geo.latitude + "</p>"; // Latitud
    html_content += "<p> Longitude: " + request.eo.geo.longitude + "</p>"; // Longi

    // Construct the HTML response content.
    let html = `<!DOCTYPE html>
      <head>
        <title> Geolocation: Hello World By Edge Functions.</title>
        <style> ${html_style} </style>
      </head>
      <body>
        <h1>Geolocation: Hello World By Edge Functions.</h1>
        <p> Welcome to try out the geolocation feature of Edge Functions.</p>
```

```
        ${html_content}
      </body>`;

    // Return a new Response object, including the HTML content and corresponding h
    return new Response(html, {
      headers: {
        "content-type": "text/html;charset=UTF-8", // Set the Content-Type header o
      },
    });
  }
```

Sample Preview

In the address bar of the browser, enter a URL that matches a triggering rule of the edge function to preview the effect of the sample code.



Related References

Runtime APIs: addEventListener
Runtime APIs: Request
Runtime APIs: Response

Batch Redirect

Last updated：2024-08-26 17:19:39

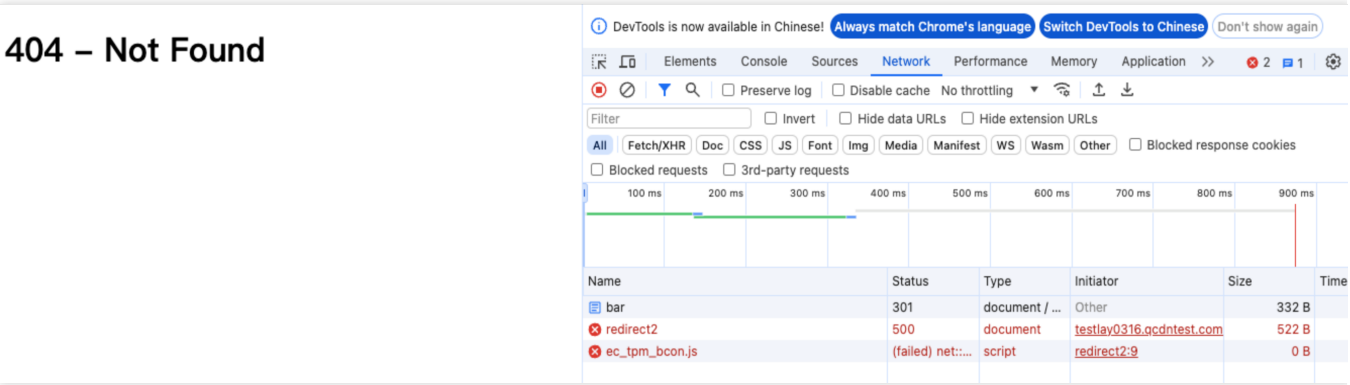
This example captures incoming HTTP requests and automatically redirects specified paths to corresponding URLs through a predefined redirect mapping table. It can be used for website migration or custom processing of error pages.

Sample Code

```
// Add a fetch event listener, which is triggered when a request is incoming. It us
async function handleRequest(request) {
    // Define the target external host name.
    const yourExternalHostname = "www.example.com";
    // Create mapping from paths to redirect URLs.
    const redirectMap = new Map([
        ["/foo", "https://" + yourExternalHostname + "/redirect1"],
        ["/bar", "https://" + yourExternalHostname + "/redirect2"],
        ["/baz", "https://" + yourExternalHostname + "/redirect3"],
    ]);
    // Parse the request URL.
    const url = new URL(request.url);
    // Get the path part of the URL.
    const path = url.pathname;
    // Check whether the path is in the redirect mapping, and if so, perform a redi
    if (redirectMap.has(path)) {
        return Response.redirect(redirectMap.get(path), 301);
    } else {
        // If the path is not in the mapping, return a 404 status code.
        return new Response('Not Found', { status: 404 });
    }
}

// When a request event occurs, use the handleRequest function to handle it.
addEventListener('fetch', event => {
    event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a triggering rule of the edge function. If the path contains /bar, a 301 redirect to `https://www.example.com/redirect2` will be performed automatically.
The preview is as follows. 404 is shown currently. Because no resources exist under the target host name www.example.com and the path /redirect2, you should replace the target host name and path with actual values.



Related References

Runtime APIs: Request
Runtime APIs: Response

URL rewrite based on regular expressions

Last updated：2025-01-15 10:43:38

This example captures incoming HTTP requests, matches them using regular expressions according to the URL path, and implements URL rewriting to access the index.html page when specific content is matched. It is applicable for any request under the specified directory to be uniformly rewritten to access the homepage (usually index.html), allowing different page requests to be handled through front-end routing.

Sample code

```
async function handleEvent(event) {
  const { request } = event;
  const urlInfo = new URL(request.url);
  // Regular expression matches /test/ followed by 1 or 2 path segments that are no
  const regexp = /^\\/test\\/([^\\/]+)(?:\\/([^\\/]+))?\\/?$/;
  // Check if the path matches the regular expression
  if (regexp.test(urlInfo.pathname)) {
    const matches = urlInfo.pathname.match(regexp);
    let newPathname = '/test/';
    // Construct a new pathname, which may be one or two according to the number of
    newPathname += matches[1]; // The first path segment
    if (matches[2]) {
      newPathname += '/' + matches[2]; // The second path segment, if any
    }
    // Ensure it ends with index.html
    newPathname += '/index.html';
    // Update the pathname of URLInfo
    urlInfo.pathname = newPathname;
  }

  // Make a request using the updated URLInfo
  const response = await fetch(urlInfo.toString(), {
    method: request.method,
    headers: request.headers,
    redirect: 'manual',
    body: request.body,
  });
  // return the response to the event
  return event.respondWith(response);
}

// call the handleEvent function for each fetch event
```

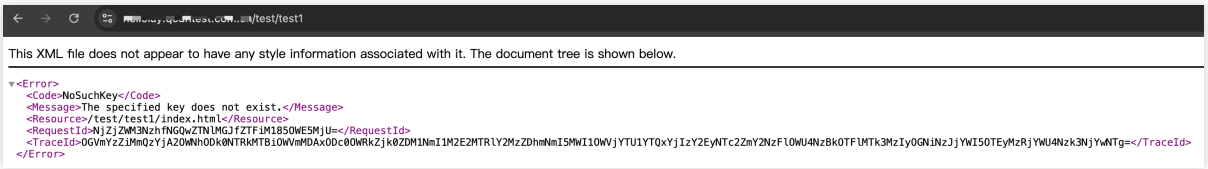
```
addEventListener('fetch', (event) => {
  handleEvent(event);
});
```

Sample Preview

1. Enter the URL matching the edge function trigger rules in the browser address bar, carry /test/segment1 in the path to achieve dynamic rewriting access `https://www.example.com/test/segment1/index.html`

2. Enter the URL matching the edge function trigger rules in the browser address bar, carry /test/segment1/segment2 in the path to achieve dynamic rewriting access `https://www.example.com/test/segment1/segment2/index.html`

3. Enter the URL matching the edge function trigger rules in the browser address bar, carry non-regular matching content in the path, such as: /test/test1



Related References

Runtime APIs: Request

Runtime APIs: Response

©2013-2025 Tencent Cloud International Pte. Ltd.

Returning an HTML Page

Last updated：2024-01-25 14:29:07

In this example, an edge function is used to generate an HTML page, and the HTML page is accessed and previewed from a browser.

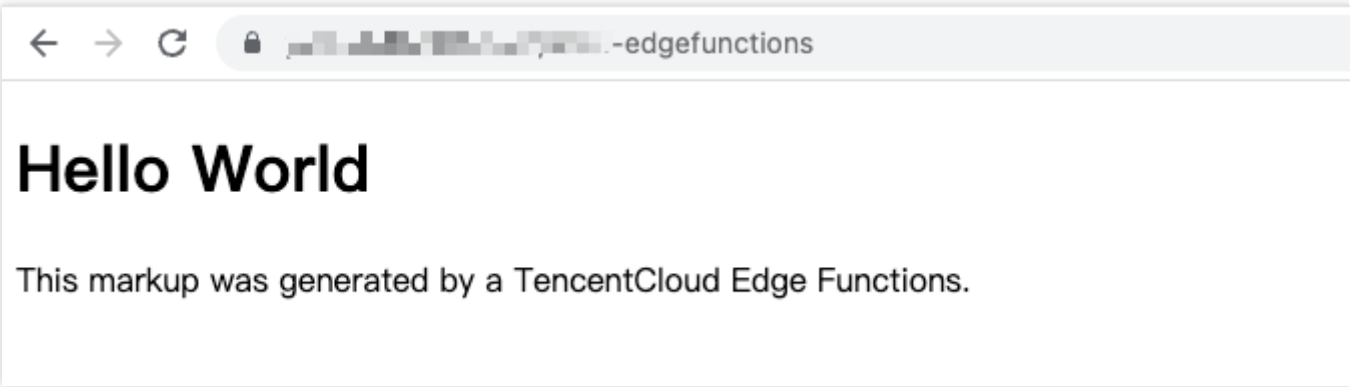
Sample Code

```
const html = `
  <!DOCTYPE html>
  <body>
    <h1>Hello World</h1>
    <p>This markup was generated by TencentCloud Edge Functions.</p>
  </body>
`;

async function handleRequest(request) {
  return new Response(html, {
    headers: {
      'content-type': 'text/html; charset=UTF-8',
      'x-edgefunctions-test': 'Welcome to use Edge Functions.',
    },
  });
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

©2013-2025 Tencent Cloud International Pte. Ltd.



References

Runtime APIs: addEventListener

Runtime APIs: Response

Runtime APIs: FetchEvent

©2013-2025 Tencent Cloud International Pte. Ltd.

Returning a JSON Object

Last updated：2024-01-25 14:26:56

In this example, an edge function is used to generate a JSON object, and the JSON object is accessed and previewed from a browser.

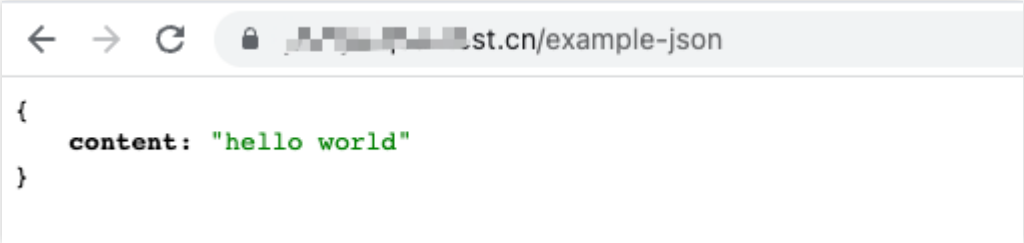
Sample Code

```
const data = {
  content: 'hello world',
};

async function handleRequest(request) {
  // Convert the JSON object to the String format.
  const result = JSON.stringify(data, null, 2);

  return new Response(result, {
    headers: {
      'content-type': 'application/json; charset=UTF-8',
    },
  });
}

addEventListener('fetch', event => {
  return event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

©2013-2025 Tencent Cloud International Pte. Ltd.



References

Runtime APIs: addEventListener

Runtime APIs: Response

Fetch Remote Resources

Last updated：2024-01-25 14:21:18

In this example, the Fetch API is called to fetch a remote jQuery.js resource and send the resource to a client in response to a request from the client.

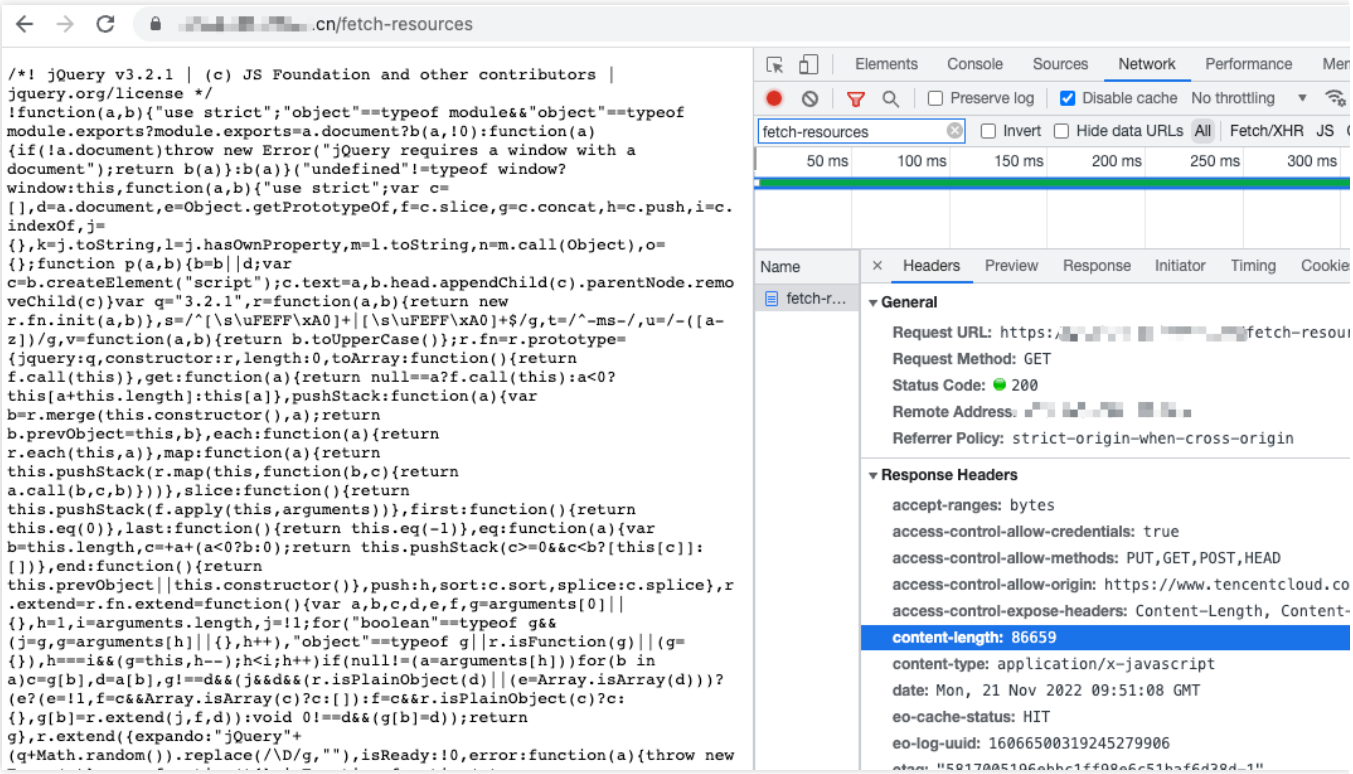
Sample Code

```
async function handleRequest(request) {
  // Fetch a remote resource.
  const response = await fetch('https://static.cloudcachetci.com/qcloud/main/script
  return response;
}

addEventListener('fetch', event => {
    return event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

©2013-2025 Tencent Cloud International Pte. Ltd.



References

Runtime APIs: Fetch

Authenticating a Request Header

Last updated：2024-01-25 11:43:50

This example demonstrates how to use an edge function to perform simple permission control by verifying the value of the `x-custom-token` request header. If the value is token-123456, access is allowed. Otherwise, access is denied.

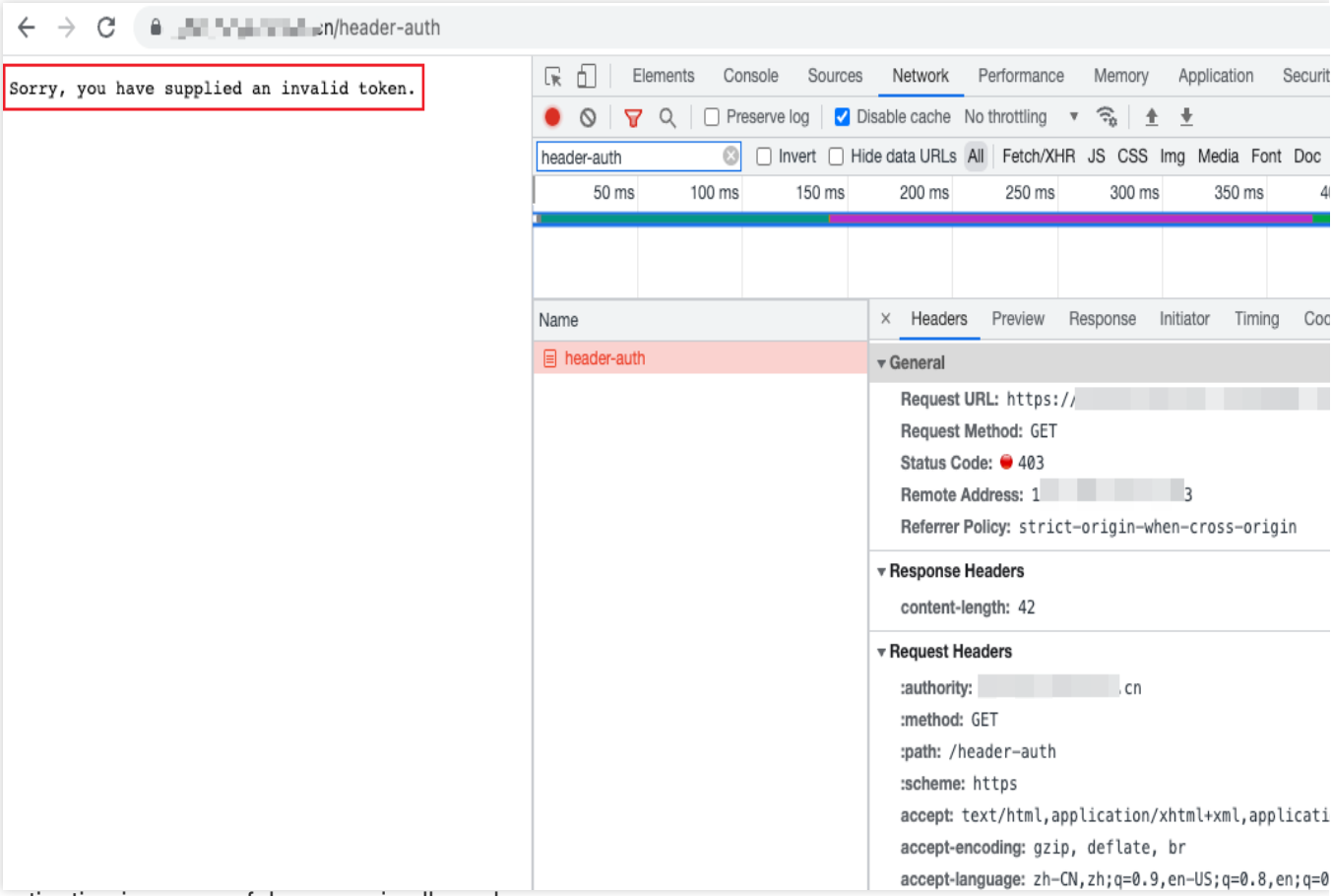
Sample Code

```
async function handleRequest(request) {
  const token = request.headers.get('x-custom-token');

  if (token === 'token-123456') {
    return new Response('hello world');
  }

  // Incorrect key supplied. Reject the request.
  return new Response('Sorry, you have supplied an invalid token.', {
    status: 403,
  });
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

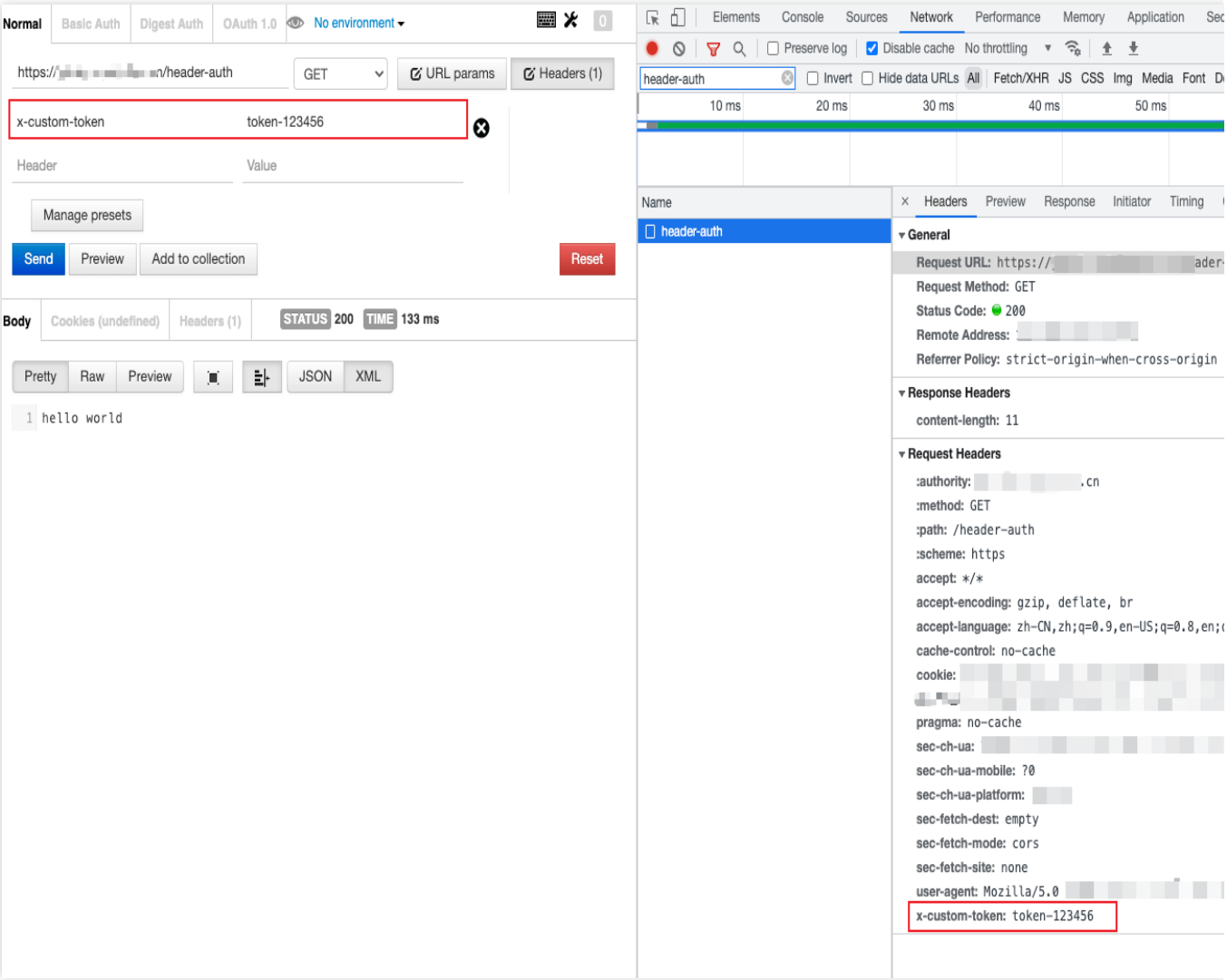
Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.
If authentication fails, access is denied.

©2013-2025 Tencent Cloud International Pte. Ltd.



If authentication is successful, access is allowed.



References

Runtime APIs: Headers

Runtime APIs: Response

Modifying a Response Header

Last updated：2023-11-24 15:10:13

This example uses the Fetch API to implement a reverse proxy for the domain name www.example.com of the site, and sets the HTTP response headers through Edge functions to achieve CORS (Cross-Origin Resource Sharing).

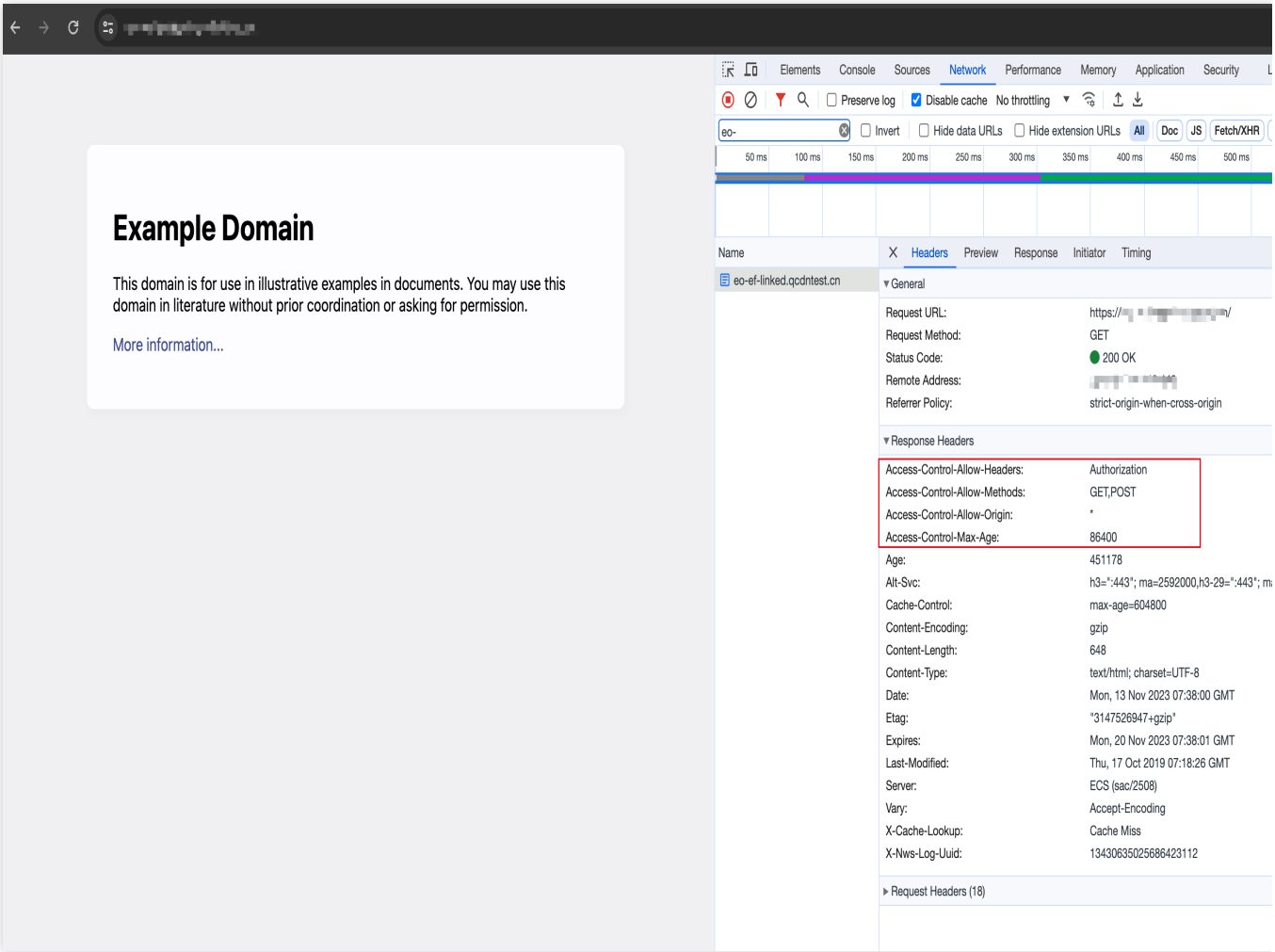
Sample Code

```
async function handleRequest(event) {
  const { request } = event;
  const urlInfo = new URL(request.url);

  const proxyRequest = new Request(`https://www.example.com${urlInfo.pathname}${url
    method: request.method,
    body: request.body,
    headers: request.headers,
    copyHeaders: true,
  });
  proxyRequest.headers.set('Host', 'www.example.com');

  // fetch reverse proxy
  const response = await fetch(proxyRequest);

  /** Add custom response headers **/
  // Specify which origins are allowed to access resources
  response.headers.append('Access-Control-Allow-Origin', '*');
  // Specify which HTTP methods (such as GET, POST, etc.) are allowed to access res
  response.headers.append('Access-Control-Allow-Methods', 'GET,POST');
  // Specify which HTTP headers can appear in the request header
  response.headers.append('Access-Control-Allow-Headers', 'Authorization');
  // How long the result of the pre-flight request can be cached
  response.headers.append('Access-Control-Max-Age', '86400');

  /** Delete response headers **/
  response.headers.delete('X-Cache');

  return response;
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event));
});
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.



References

Runtime APIs: Headers

Runtime APIs: Response

Performing an A/B Test

Last updated：2023-09-11 17:51:22

In this example, cookies are used to store session information and perform A/B testing on requests. This example demonstrates how to use an edge function to perform A/B testing.

Sample Code

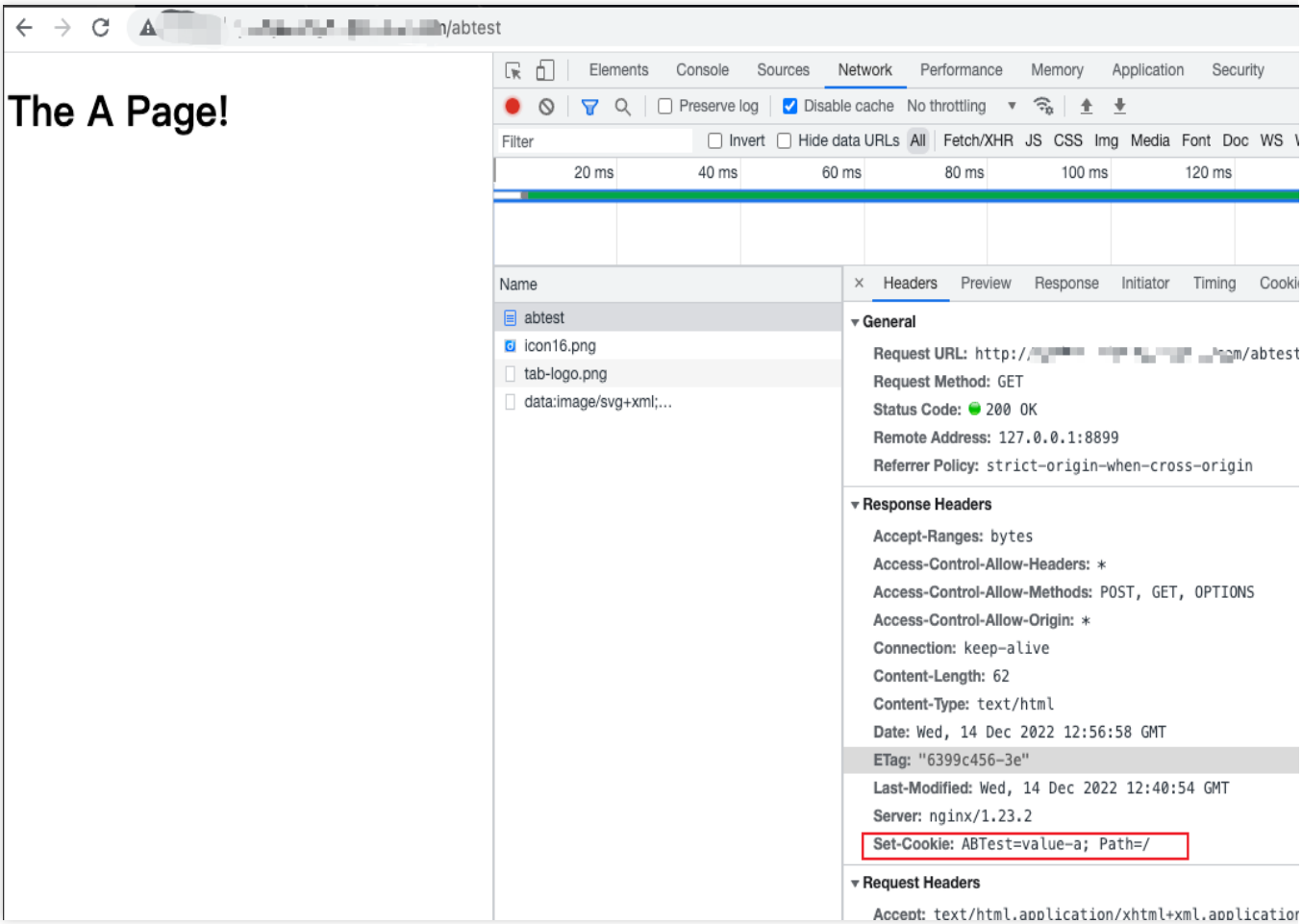
```
// cookie name
const COOKIE_NAME = 'ABTest';

// cookie value
const VALUE_A = 'index-a.html';
const VALUE_B = 'index-b.html';

// Root path, the origin must exist this path, and under this path, there are files
const BASE_PATH = '/abtest';

async function handleRequest(request) {
  const urlInfo = new URL(request.url);

  // Judge the URL path, if accessing non-abtest resources, directly responded.
  if (!urlInfo.pathname.startsWith(BASE_PATH)) {
    return fetch(request);
  }

  // Collected the current request's Cookie.
  const cookies = new Cookies(request.headers.get('cookie'));
  const abTestCookie = cookies.get(COOKIE_NAME);
  const cookieValue = abTestCookie?.value;

  // If the Cookie value is A test, Return index-a.html.
  if (cookieValue === VALUE_A) {
    urlInfo.pathname = `/${BASE_PATH}/${cookieValue}`;
    return fetch(urlInfo.toString());
  }

  // If the Cookie value is B test, Return index-b.html.
  if (cookieValue === VALUE_B) {
    urlInfo.pathname = `/${BASE_PATH}/${cookieValue}`;
    return fetch(urlInfo.toString());
  }
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
  // If the Cookie information does not exist, randomly grant the current request t
  const testValue = Math.random() < 0.5 ? VALUE_A : VALUE_B;

  urlInfo.pathname = `/${BASE_PATH}/${testValue}`;

  const response = await fetch(urlInfo.toString());

  cookies.set(COOKIE_NAME, testValue, { path: '/', max_age: 60 });
  response.headers.set('Set-Cookie', getSetCookie(cookies.get(COOKIE_NAME)));
  return response;
}

// Concatenate Set-Cookie.
function getSetCookie(cookie) {
  const cookieArr = [
    `${encodeURIComponent(cookie.name)}=${encodeURIComponent(cookie.value)}`,
  ];

  const key2name = {
    expires: 'Expires',
    max_age: 'Max-Age',
    domain: 'Domain',
    path: 'Path',
    secure: 'Secure',
    httponly: 'HttpOnly',
    samesite: 'SameSite',
  };

  Object.keys(key2name).forEach(key => {
    if (cookie[key]) {
      cookieArr.push(`${key2name[key]}=${cookie[key]}`);
    }
  });

  return cookieArr.join('; ');
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.



References

Runtime APIs: Cookies

Runtime APIs: Response

Setting Cookies

Last updated：2023-09-11 17:49:21

In this example, cookies are used to count the number of access requests. When a browser accesses the Edge Functions service, the number of access requests is increased by 1.

Sample Code

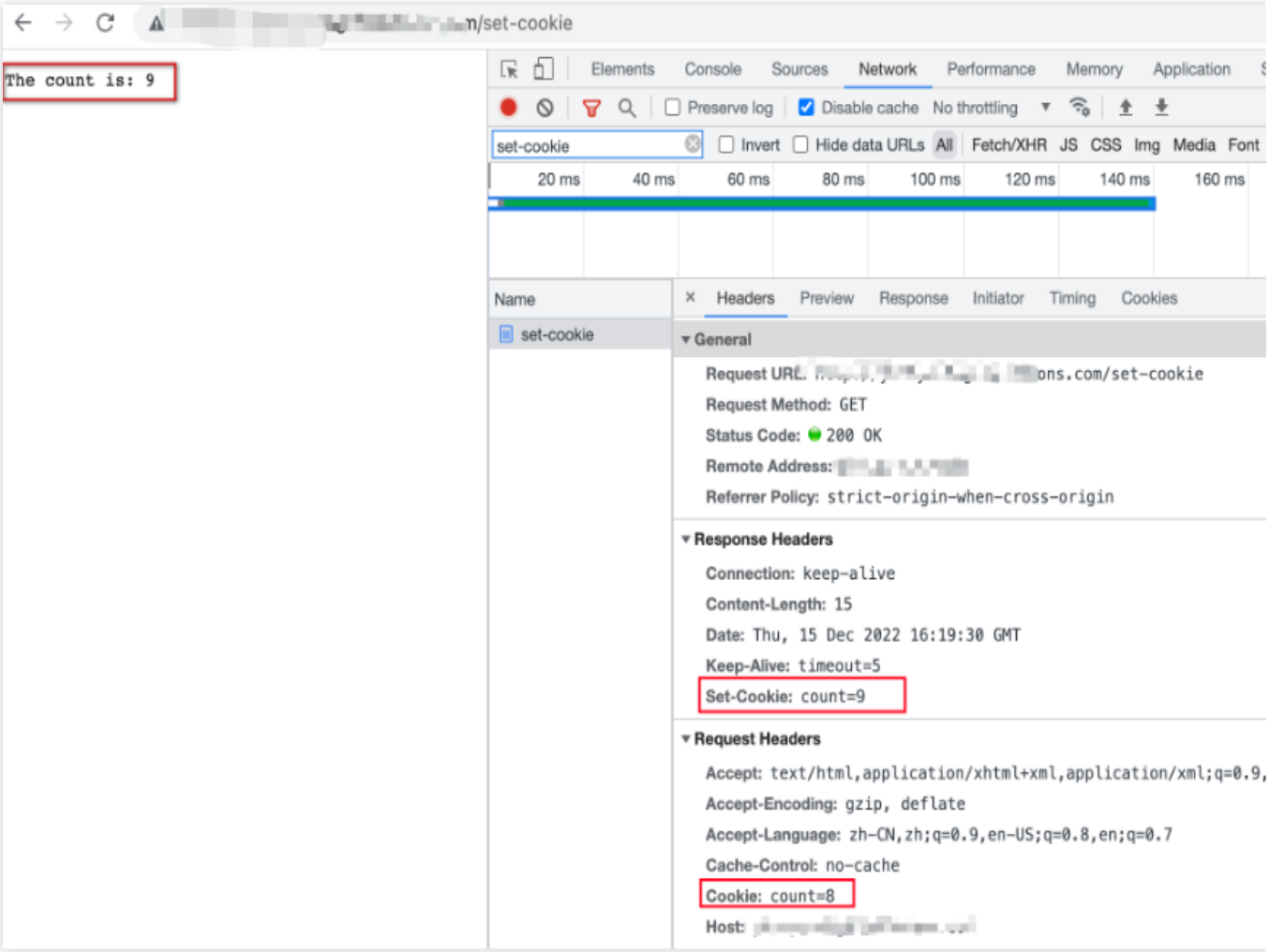
```
// cookie name
const COOKIE_NAME = 'count';

async function handleRequest(request) {
  // collected the current requests' Cookies and resolution into scope
  const cookies = new Cookies(request.headers.get('cookie'));
  const cookieCount = cookies.get(COOKIE_NAME);
  // count increment
  const count = Number(cookieCount && cookieCount.value || 0) + 1;
  // update Cookie's count
  cookies.set(COOKIE_NAME, String(count));

  const response = new Response(`The count is: ${count}`);
  // setting responded cookies
  response.headers.set('Set-Cookie', getSetCookie(cookies.get(COOKIE_NAME)));
  return response;
}

// concatenate Set-Cookie
function getSetCookie(cookie) {
  const cookieArr = [
    `${encodeURIComponent(cookie.name)}=${encodeURIComponent(cookie.value)}`,
  ];

  const key2name = {
    expires: 'Expires',
    max_age: 'Max-Age',
    domain: 'Domain',
    path: 'Path',
    secure: 'Secure',
    httponly: 'HttpOnly',
    samesite: 'SameSite',
  };
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
  Object.keys(key2name).forEach(key => {
    if (cookie[key]) {
      cookieArr.push(`${key2name[key]}=${cookie[key]}`);
    }
  });

  return cookieArr.join('; ');
}

addEventListener('fetch', (event) => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.



References

Runtime APIs: Cookies

Runtime APIs: Response

Runtime APIs: Request

Performing Redirect Based on the Request Location

Last updated：2023-11-24 15:08:21

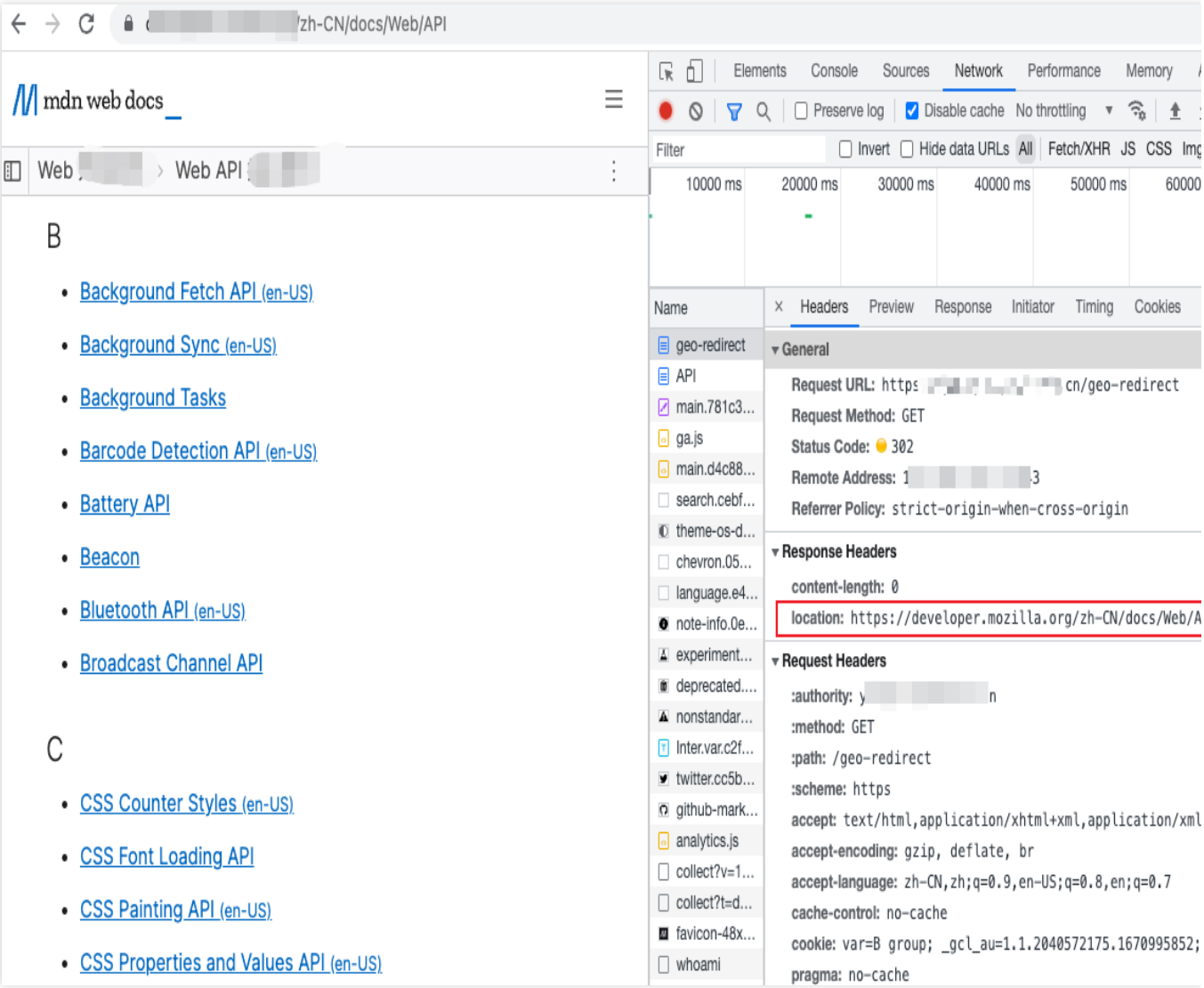
This example implements automatic redirection to the Target url of the region belonging to the Client by judging the region of the Client. It realizes the delivery of requests based on the region of the Client through Edge functions.

Sample Code

```
// The collection of URLs in all regions.
const urls = {
  CN: 'https://www.example.com/zh-CN',
  US: 'https://www.example.com/en-US',
};

// The default redirect URL.
const defaultUrl = 'https://www.example.com/en-US';

/**
 * Redirect to the target URL based on the region of the current request.
 * @param { Request } request
 */
function handleRequest(request) {
  // Obtain the region of the current request.
  const alpha2code = request.eo.geo.countryCodeAlpha2;
  // The target URL that you want to use for redirection.
  const url = urls[alpha2code] || defaultUrl;

  return Response.redirect(url, 302);
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.



References

Runtime APIs: Request

Runtime APIs: Response

Using the Cache API

Last updated：2024-01-25 11:42:05

In this sample edge function, the Fetch API is called to fetch a remote jQuery.js resource, and the Cache API is called to cache the resource to an EdgeOne edge node. The cache duration is set to 10s.

Sample Code

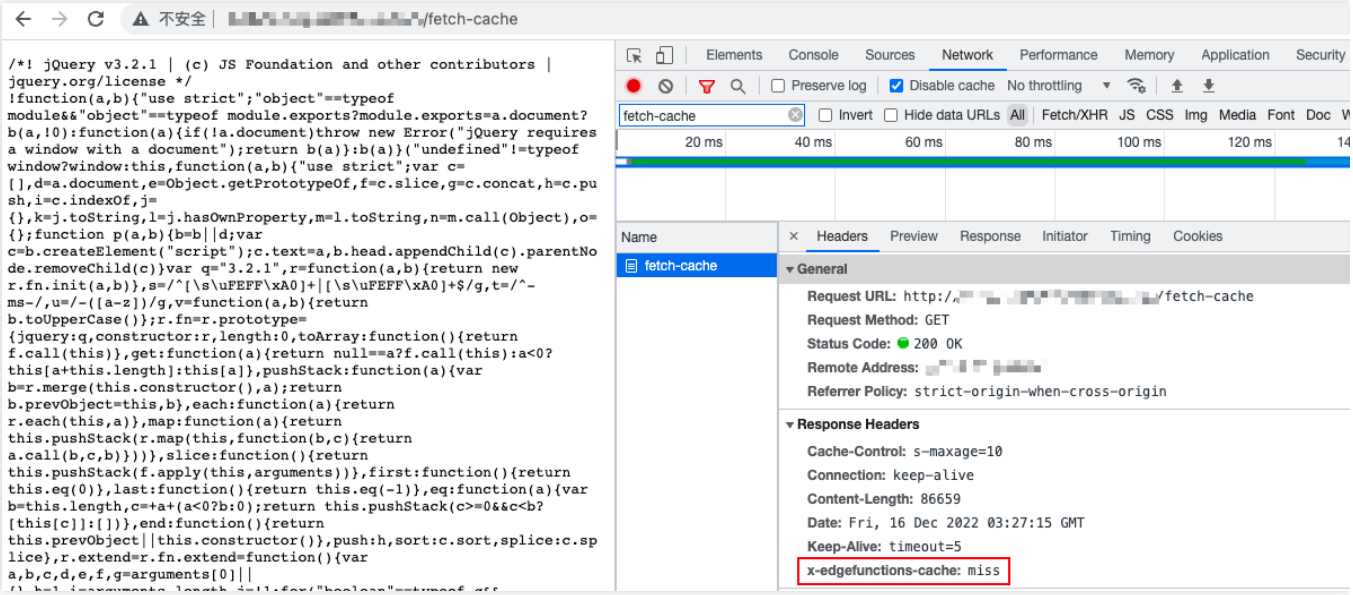
```
async function fetchJquery(event, request) {
  const cache = caches.default;
  // If the resource is not found in the cache, fetch the resource from the origin
  let response = await fetch(request);

  // Add the Cache-Control field to the response header and set the cache duration
  response.headers.append('Cache-Control', 's-maxage=10');
  event.waitUntil(cache.put(request, response.clone()));

  // Add an identifier indicating that the resource is not found in the cache to th
  response.headers.append('x-edgefunctions-cache', 'miss');
  return response;
}

async function handleEvent(event) {
  // The resource URL, which is also used as the cache key.
  const request = new Request('https://static.cloudcachetci.com/qcloud/main/scripts
  // Obtain the default cache instance.
  const cache = caches.default;

  try {
    // Fetch the associated resource from the cache. If the resource is already cac
    let response = await cache.match(request);

    // If the resource is not found in the cache, re-fetch the remote resource.
    if (!response) {
      return fetchJquery(event, request);
    }

    // Add an identifier indicating that the resource is found in the cache to the
    response.headers.append('x-edgefunctions-cache', 'hit');

    return response;
  } catch (e) {
```

```
    await cache.delete(request);
    // If the cache duration of the resource times out or another error occurs, re-
    return fetchJquery(event, request);
  }
}

addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});
```

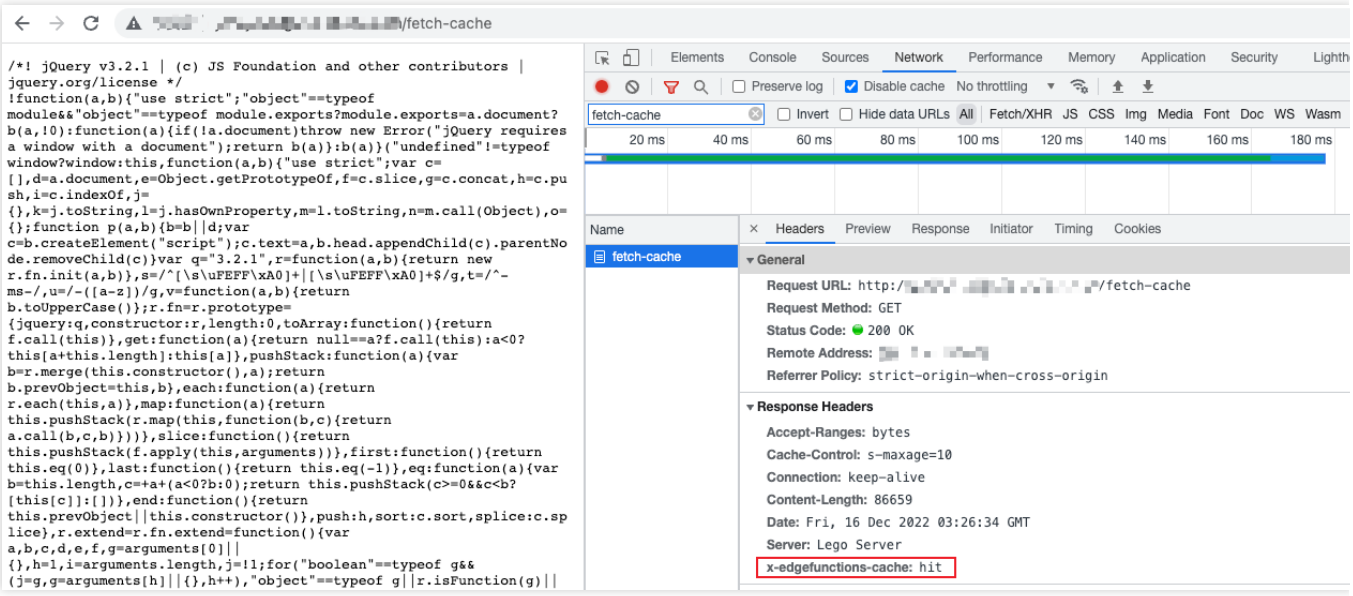
Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

The resource is not found in the cache.



The resource is found in the cache.



References

Runtime APIs: Cache

Runtime APIs: Fetch

Runtime APIs: FetchEvent

Runtime APIs: Response

Caching POST Requests

Last updated：2024-01-25 11:39:35

In this example, an SHA-256 signature is calculated for the request body of a POST request and used as a part of the cache key, and the Cache API is called to cache the response content. If the content is already stored in the cache, the cached content is sent to the client. Otherwise, the Fetch API is called to initiate a subrequest to fetch a remote resource. This example demonstrates how to use an edge function to cache POST requests.

Sample Code

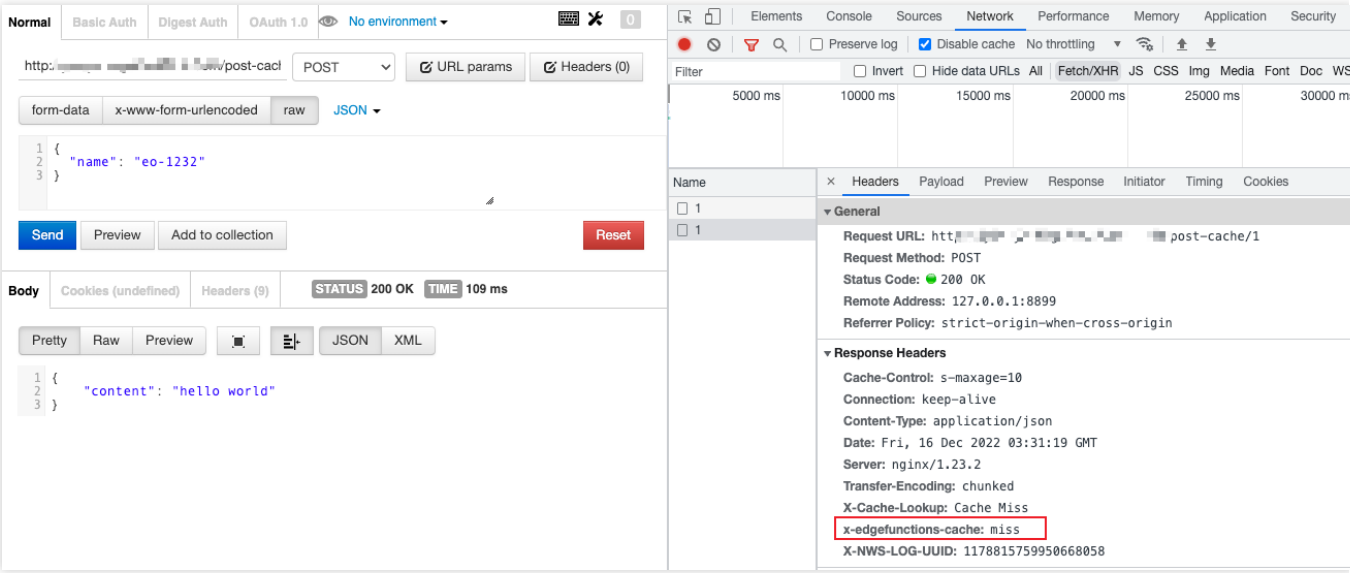
```
function uint8ArrayToHex(arr) {
  return Array.prototype.map.call(arr, (x) => (('0' + x.toString(16)).slice(-2))).j
}

// The SHA-256 signature digest.
async function sha256(message) {
  const msgBuffer = new TextEncoder().encode(message);
  const hashBuffer = await crypto.subtle.digest('SHA-256', msgBuffer);

  return uint8ArrayToHex(new Uint8Array(hashBuffer));
}

async function fetchContent(event, cacheKey) {
  const cache = caches.default;

  // If the resource is not found in the cache, fetch the resource from the origin
  const response = await fetch(event.request);

  // Add the Cache-Control field to the response header and specify the cache durat
  response.headers.set('Cache-Control', 's-maxage=10');
  event.waitUntil(cache.put(cacheKey, response.clone()));

  // Add an identifier indicating that the resource is not found in the cache to th
  response.headers.append('x-edgefunctions-cache', 'miss');

  return response;
}

async function handleRequest(event) {
  const request = event.request;
  const body = await request.clone().text();
```

```
  // // Calculate the hash value based on the request body.
  const hash = await sha256(body);

  // Use the hash value that is calculated based on the request body as a part of t
  const cacheKey = `${request.url}${hash}`;

  const cache = caches.default;

  try {
    // Fetch the associated response content from the cache.
    let response = await cache.match(cacheKey);

    if (!response) {
        return fetchContent(event, cacheKey);
    }

    // Add an identifier indicating that the resource is found in the cache to the
    response.headers.append('x-edgefunctions-cache', 'hit');

    return response;
  } catch (error) {
    await cache.delete(cacheKey);
    // If the cache duration of the resource times out or the resource is not found
    return fetchContent(event, cacheKey);
  }

  return response;
}

addEventListener('fetch', (event) => {
  try {
    const request = event.request;
    // Process a POST request.
    if (request.method.toUpperCase() === 'POST') {
      return event.respondWith(handleRequest(event));
    }
    // Non-POST request.
    return event.respondWith(fetch(request));
  } catch (e) {
    return event.respondWith(new Response('Error thrown ' + e.message));
  }
});
```

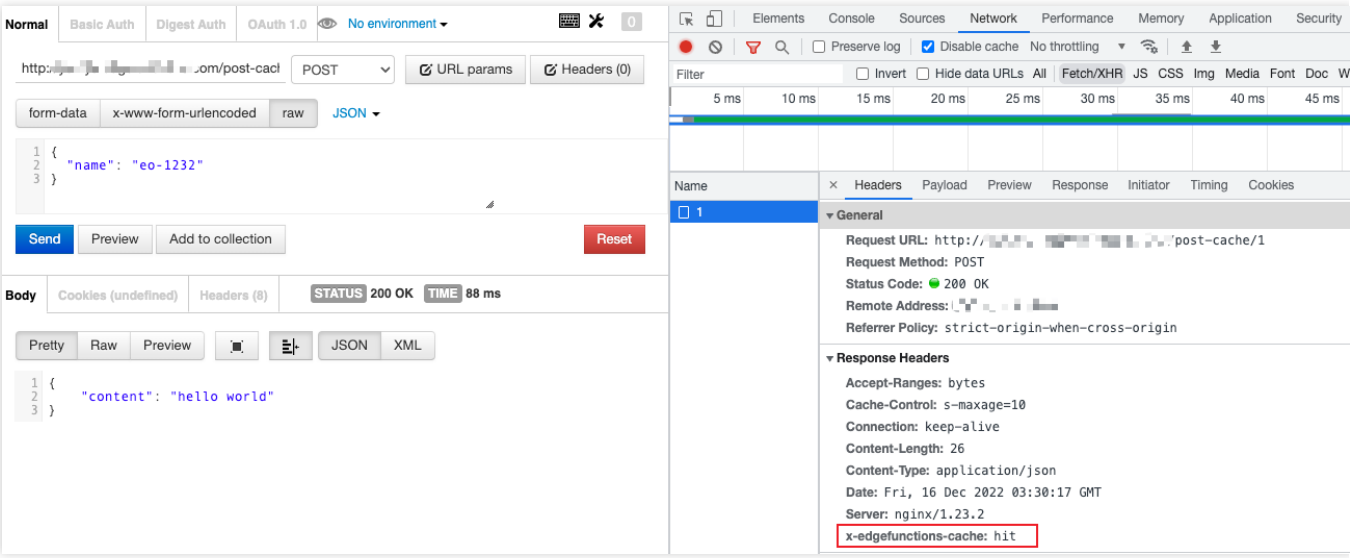
Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

The resource is not found in the cache.



The resource is found in the cache.



References

Runtime APIs: Fetch

Runtime APIs: Cache

Runtime APIs: Web Crypto

Responding in Streaming Mode

Last updated：2024-01-25 11:36:31

In this example, the Fetch API is called to fetch a remote jQuery.js resource, and the resource is sent to a client in streaming mode in response to a request from the client.

Sample Code

```
async function handleRequest(request) {
  const response = await fetch('https://static.cloudcachetci.com/qcloud/main/script

  if (response.status !== 200) {
    return response;
  }

  // Generate readable streams and writeable streams.
  const { readable, writable } = new TransformStream();
  // Respond to the client in streaming mode.
  response.body.pipeTo(writable);

  return new Response(readable, response);
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

©2013-2025 Tencent Cloud International Pte. Ltd.

```
> GET htt
> User-Agent:
> Hos
> Accept: */*
> Proxy-Connection: Keep-Alive
>
< HTTP/1.1 200 OK
< EO-Cache-Status: HIT
< EO-LOG-UUID: 5573999441187848225
< Access-Control-Expose-Headers: Content-Length, Content-Type, server_ip, x-nws-log-uuid
< Accept-Ranges: bytes
< Last-Modified: Tue, 18 Oct 2022 12:52:34 GMT
< x-cos-request-id: NjM3MjNkOWVfYTk3NWI3MDlfMjljN18xZTc2YzI3
< x-cos-hash-crc64ecma: 12262715282927589694
< Vary: Origin, Access-Control-Request-Headers, Access-Control-Request-Method
< Connection: keep-alive
< Date: Mon, 14 Nov 2022 13:07:42 GMT
< Access-Control-Allow-Methods: PUT,GET,POST,HEAD
< Access-Control-Allow-Credentials: true
< Server: tencent-cos
< Content-Type: application/x-javascript
< Access-Control-Allow-Origin: https://www.tencentcloud.com
< Etag: "5817005196ebbc1ff98e6c51baf6d38d-1"
< Transfer-Encoding: chunked
<
{ [data not shown]
```

References

Runtime APIs: TransformStream

Runtime APIs: Response

Merging Resources and Responding in Streaming Mode

Last updated：2025-06-18 16:27:43

In this example, three video clips are merged into one video, and the merged video is played on a client based on the order in which the video clips are merged. This example demonstrates how to use an edge function to fetch multiple remote resources, read and merge the resources in streaming mode, and respond to a client request by using the merged resource in streaming mode.

Sample Code

```
async function sequentialCombine(urls, destination) {
  try {
    // Process each URL in order
    for (let i = 0; i < urls.length; i++) {
      const url = urls[i];

      // Get the current clip
      const response = await fetch(url);

      if (!response.ok) {
        console.error(`Failed to obtain video clip: ${url}, Error code: ${response.
        continue;
      }

      // Get a readable stream
      const readable = response.body;

      // Execute pipeTo immediately to write the current clip to the target stream
      try {
        await readable.pipeTo(destination, {
          preventClose: true // Keep the stream open for subsequent writing
        });
      } catch (e) {
        console.error(`Stream processing errors (${url}): ${e.message}`);
      }
    }
  } catch (err) {
    console.error(`Merging video streams error: ${err.message}`);
  } finally {
    // Close the stream after all fragments are processed
```

```
    const writer = destination.getWriter();
    writer.close();
    writer.releaseLock();
  }
}

async function handleRequest(request) {
  // The URLs of the video clips.
  const urls = [
    'https://vod.example.com/stream-01.mov',
    'https://vod.example.comm/stream-02.mov',
    'https://vod.example.com/stream-03.mov',
  ];

  // Creating a Transformation Flow
  const { readable, writable } = new TransformStream();

  // Get and merge video clips in sequence
  sequentialCombine(urls, writable);

  // Returns the merged video stream response
  return new Response(readable, {
    headers: {
      'content-type': 'video/mp4',
    }
  });
}

// Listening for fetch events
addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the merged video. View the response header and verify that the video is transferred in chunked mode.

References

Runtime APIs: TransformStream

Runtime APIs: Response

©2013-2025 Tencent Cloud International Pte. Ltd.

Protecting Data from Tampering

Last updated：2024-01-25 11:32:07

In this example, an SHA-256 signature is calculated for a request body and compared with the signature that is generated by the origin server. If the signatures are the same, the response content is not tampered with. Otherwise, a 416 status code is returned, indicating that the response content is tampered with. This example demonstrates how to use an edge function to verify whether the response content sent from the origin server is tampered with.

Note:

Make sure that the origin server uses the same signature algorithm and tamper-proofing rules that are used in this example.

To use the tamper-proofing rules described in this example in the production environment, perform sorting on the calculated signature to prevent the signature from being cracked by attackers.

Sample Code

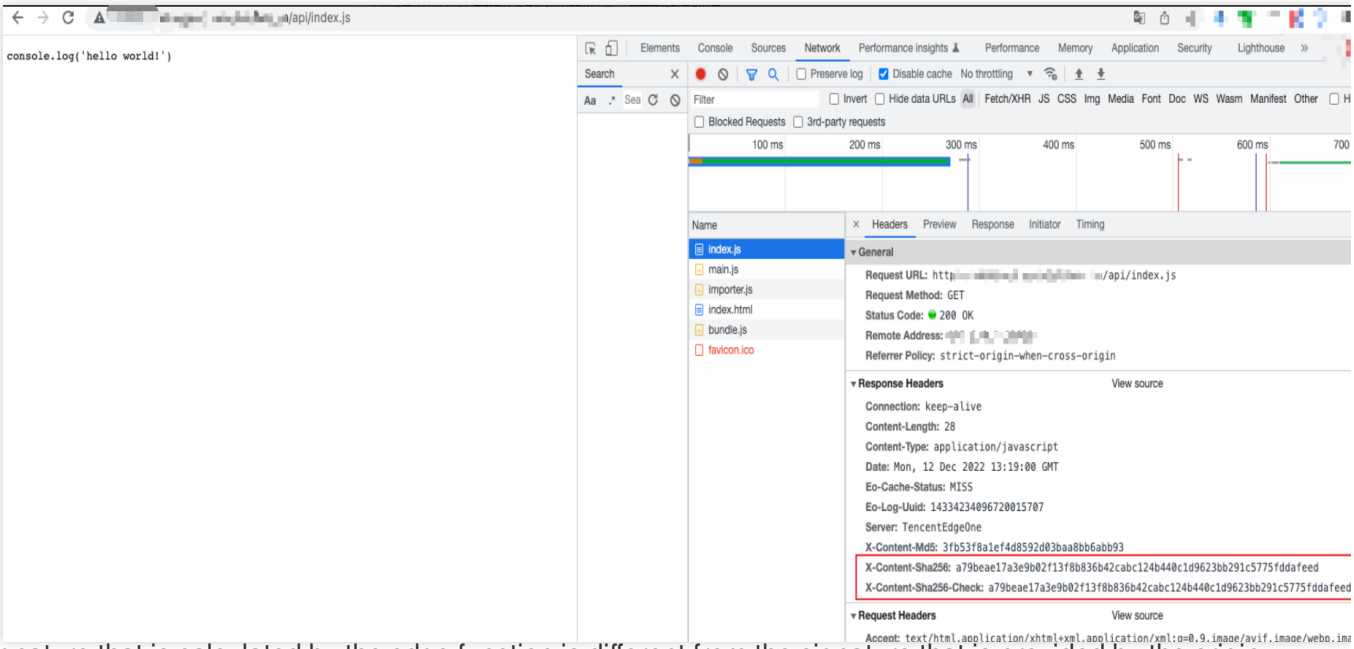
```
// The supported text file formats.
const textFileTypes = [
  'application/javascript',
  'text/html; charset=utf-8',
  'text/css; charset=utf-8',
  'text/xml; charset=utf-8'
];

// The supported image file format.
const imageFileTypes = [
  'image/jpeg'
];

function uint8ArrayToHex(arr) {
  return Array.prototype.map
    .call(arr, (x) => (('0' + x.toString(16)).slice(-2)))
    .join('');
}

/**
 * The following algorithms are supported: MD5, SHA-1, SHA-256, SHA-384, and SHA-51
 * Note: When you calculate a signature on the origin server, do not directly sign
 * Use the same method in the sample code to calculate the signature for comparison
 **/
async function checkAndResponse(response, hash, algorithm) {
  const headers = response.headers;
```

```
  let checkHash = 'sorry! not match';
  let data = null;
  const contentType = headers.get('Content-Type');
  if (textFileTypes.includes(contentType) || imageFileTypes.includes(contentType))
    data = await response.arrayBuffer();
  }
  let ret = await crypto.subtle.digest({name: algorithm}, data);
  checkHash = uint8ArrayToHex(new Uint8Array(ret));

  headers.append(`X-Content-${algorithm}-Check`, checkHash);
  // Compare the signature that is calculated in real time with the signature that
  if (checkHash !== hash) {
    return new Response(null, {
      headers,
      status: 416
    });
  }

  return new Response(data, {
    headers,
    status: 200
  });
}

async function handleEvent(event) {
  // Obtain the content that is returned by the origin server. If the content is ca
  const response = await fetch(event.request);
  if (response.status === 200) {
    const headers = response.headers;
    // The signature header of the content that is returned by the origin server.
    const hash = headers.get('X-Content-Sha256');
    if (hash) {
      // Check whether the calculated signature is the same as the signature that i
      return checkAndResponse(response, hash, 'Sha-256');
    }
  }

  return response;
}

addEventListener('fetch', event => {
  event.respondWith(handleEvent(event));
});
```

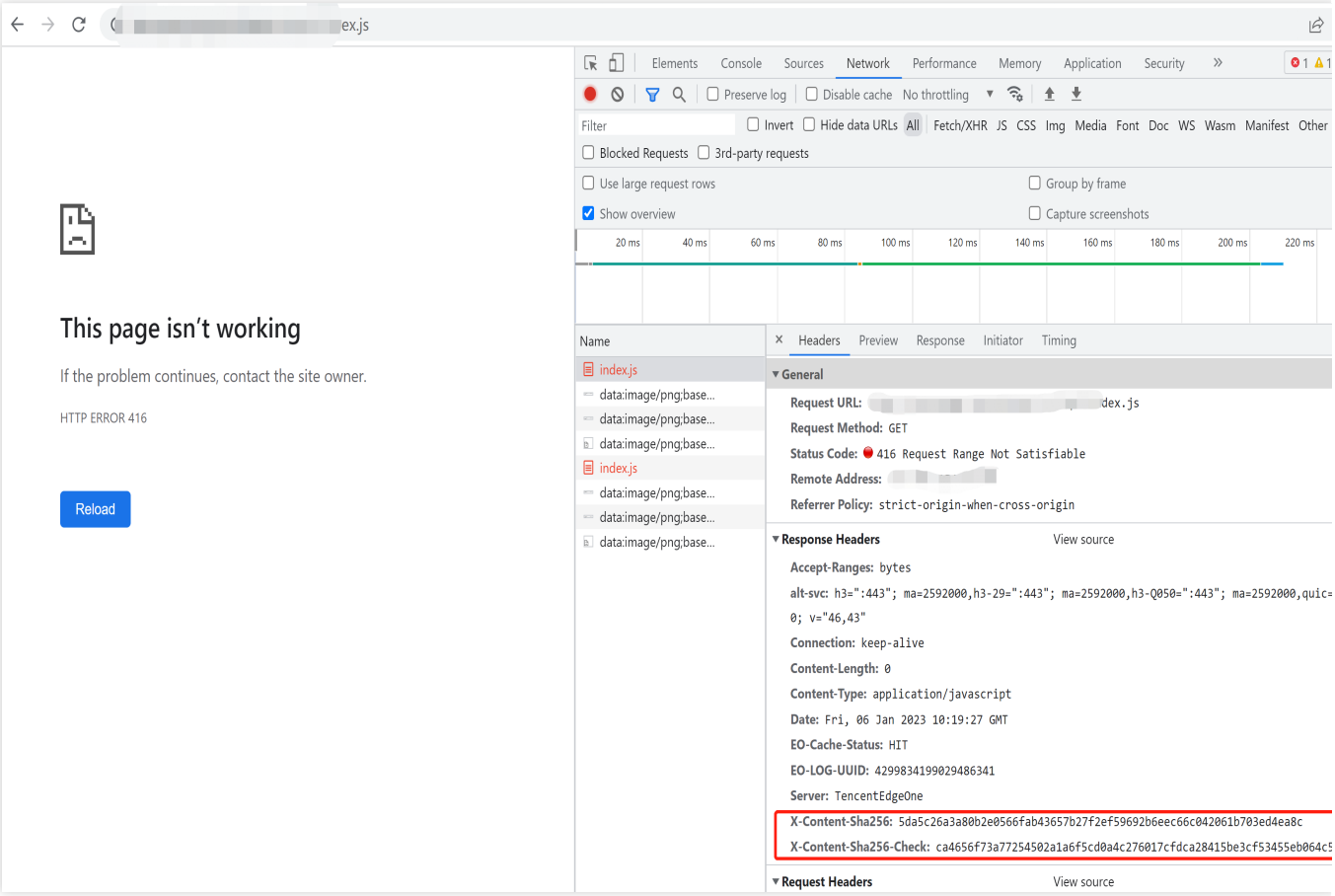
Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code.

The signature that is calculated by the edge function is the same as the signature that is provided by the origin server.



The signature that is calculated by the edge function is different from the signature that is provided by the origin server, and a 416 status code is returned.



References

Runtime APIs: Fetch

Runtime APIs: Web Crypto

Runtime APIs: Headers

Runtime APIs: Response

Rewriting a m3u8 File and Configuring Authentication

Last updated：2023-04-13 11:14:30

In this example, a .m3u8 file is rewritten to support TypeA authentication that controls access to .m3u8 files and .ts segments. You can modify the code as needed to support other authentication methods.

Sample Code

```
// The private key for TypeA authentication. Specify the key as needed and make sur
const PK = '0123456789';
// The validity period of the key for encryption and verification, in seconds.
const TTL = 60;
const KEY_NAME = 'key';
const UID = 0;
const SUFFIX_LIST = ['.m3u8', '.ts'];

addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event));
});

async function handleEvent(event) {
  try {
    const { request } = event;
    const urlInfo = new URL(request.url);
    const suffix = getSuffix(urlInfo.pathname);

    // Check whether the file extension is .m3u8 or .ts.
    if (!SUFFIX_LIST.includes(suffix)) {
      return fetch(request);
    }

    // TypeA authentication.
    const checkResult = await checkTypeA(urlInfo);
    if (!checkResult.flag) {
      return new Response(checkResult.message, {
        status: 403,
        headers: {
          'X-Auth-Err': checkResult.message
        },
      });
```

```
    }

    // Rewrite the .m3u8 file and respond.
    if (suffix === '.m3u8') {
      return fetchM3u8({
        request,
        querySign: {
          basePath: urlInfo.pathname.substring(0, urlInfo.pathname.lastIndexOf('/')
          ...checkResult.querySign,
        }
      });
    }

    // Respond with .ts resources.
    if (suffix === '.ts') {
      return fetchTs(request);
    }
  } catch (error) {
    return new Response(error.stack, { status: 544 });
  }

  return fetch(request);
}

async function checkTypeA(urlInfo) {
  const sign = urlInfo.searchParams.get(KEY_NAME) || '';
  const elements = sign.split('-');

  if (elements.length !== 4) {
    return {
      flag: false,
      message: 'Invalid Sign Format',
    };
  }

  const [ts, rand, uid, md5hash] = elements;
  if (ts === undefined || rand === undefined || uid === undefined || md5hash === un
    return {
      flag: false,
      message: 'Invalid Sign Format',
    };
  }

  if (!isNumber(ts)) {
    return {
      flag: false,
      message: 'Sign Expired',
```

```
    };
  }

  if (Date.now() > (Number(ts) + TTL) * 1000) {
    return {
      flag: false,
      message: 'Sign Expired',
    };
  }

  const hash = await md5([urlInfo.pathname, ts, rand, uid, PK].join('-'));
  if (hash !== md5hash) {
    return {
      flag: false,
      message: 'Verify Sign Failed',
    };
  }
  return {
    flag: true,
    message: 'success',
    querySign: {
      rand,
      uid,
      md5hash,
      ts,
    },
  };
}

async function fetchM3u8({ request, querySign }) {
  request.headers.delete('Accept-Encoding');
  let response = null;
  try {
    response = await fetch(request);
    if (response.status !== 200) {
      return response;
    }
  } catch (error) {
    return new Response('', {
      status: 504,
      headers: { 'X-Fetch-Err': 'Invalid Origin' }
    });
  }

  const content = await response.text();
  const lines = content.split('\\n');
```

```
  const contentArr = await Promise.all(
    lines.map(line => rewriteLine({ line, querySign }))
  );

  return new Response(contentArr.join('\\n'), response);
}

async function fetchTs(request) {
  let response = null;
  try {
    response = await fetch(request);
    if (response.status !== 200) {
      return response;
    }
  } catch (error) {
    return new Response('', {
      status: 504,
      headers: { 'X-Fetch-Err': 'Invalid Origin' }
    });
  }
  return response;
}

async function rewriteLine({ line, querySign }) {
  // Skip empty lines.
  if (/^\\s*$/.test(line)) {
    return line;
  }

  if (line.charAt(0) === '#') {
    // Process #EXT-X-MAP.
    if (line.startsWith('#EXT-X-MAP')) {
      const key = await createSign(querySign, line);
      line = line.replace(/URI="([^"]+)"/, (matched, p1) => {
        return p1 ? matched.replace(p1, `${p1}?key=${key}`) : matched;
      });
    }
    return line;
  }

  const key = await createSign(querySign, line);

  return `${line}?${KEY_NAME}=${key}`;
}

async function createSign(querySign, line) {
  const { ts, rand, uid = 0 } = querySign;
```

```
  const pathname = `${querySign.basePath}/${line}`;

  const md5hash = await md5([pathname, ts, rand, uid, PK].join('-'));
  const key = [ts, rand, uid, md5hash].join('-');

  return key;
}

function getSuffix(pathname) {
  const suffix = pathname.match(/\\.m3u8|\\.ts$/);
  return suffix ? suffix[0] : null;
}

function isNumber(num) {
  return Number.isInteger(Number(num));
}

function bufferToHex(arr) {
  return Array.prototype.map
    .call(arr, (x) => (x >= 16 ? x.toString(16) : '0' + x.toString(16)))
    .join('');
}

async function md5(text) {
  const buffer = await crypto.subtle.digest('MD5', TextEncoder().encode(text));
  return bufferToHex(new Uint8Array(buffer));
}
```

Sample Preview

In the address bar of the browser, enter a URL that matches a trigger rule of the edge function to preview the effect of the sample code. Sample URL: 如：http://www.example.com/index.m3u8?key=1678873033-123456-0-32f4xxxxcabcxxxx1602xxxx6756d8f4.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:1.416667,
0-282375.ts
#EXTINF:2.875000,
282376-675671.ts
#EXTINF:1.708333,
675672-905595.ts
#EXTINF:3.208333,
3.ts
#EXTINF:2.000000,
4.ts
#EXTINF:2.000000,
5.ts
#EXTINF:2.000000,
6.ts
#EXTINF:2.000000,
7.ts
#EXTINF:3.708333,
8.ts
#EXTINF:5.791667,
9.ts
#EXTINF:3.375000,
10.ts
#EXT-X-ENDLIST
```

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:1.416667,
0-282375.ts?key=1671181186-6767974945631037-0-58d3457cef4a1e3ce68a471b6966ef74
#EXTINF:2.875000,
282376-675671.ts?key=1671181186-6767974945631037-0-6c0a83fb9313c836734df13cad56d18a
#EXTINF:1.708333,
675672-905595.ts?key=1671181186-6767974945631037-0-948a4eaf14828bf8ff57120983b3b474
#EXTINF:3.208333,
3.ts?key=1671181186-6767974945631037-0-e10778ac5e94134a6e762de322cdc25e
#EXTINF:2.000000,
4.ts?key=1671181186-6767974945631037-0-3fb747e907e623822fbfbcb9cb066768
#EXTINF:2.000000,
5.ts?key=1671181186-6767974945631037-0-193aa1d267b6892a8f89754cf9fcf68a
#EXTINF:2.000000,
6.ts?key=1671181186-6767974945631037-0-e5e95cf873c8911ab7d0668716ea518a
#EXTINF:2.000000,
7.ts?key=1671181186-6767974945631037-0-bd882ed14c331d31c368682b436cc5f1
#EXTINF:3.708333,
8.ts?key=1671181186-6767974945631037-0-702fdc4c39f7499971ce3e0174cdd73a
#EXTINF:5.791667,
9.ts?key=1671181186-6767974945631037-0-eb91c87cadbf372adf95900f442aeace
#EXTINF:3.375000,
10.ts?key=1671181186-6767974945631037-0-2cfe4f6b1ea682820e020918e121ac71
#EXT-X-ENDLIST
```

References

Runtime APIs: Fetch

Runtime APIs: Web Crypto

Runtime APIs: Response

TypeA

Adaptive Image Resize

Last updated：2023-07-06 17:20:26

In this document, we demonstrate how to identify the client type via `User-Agent` in the request header, use fetch API to get the image from the origin, and convert it to the format required by the browser.

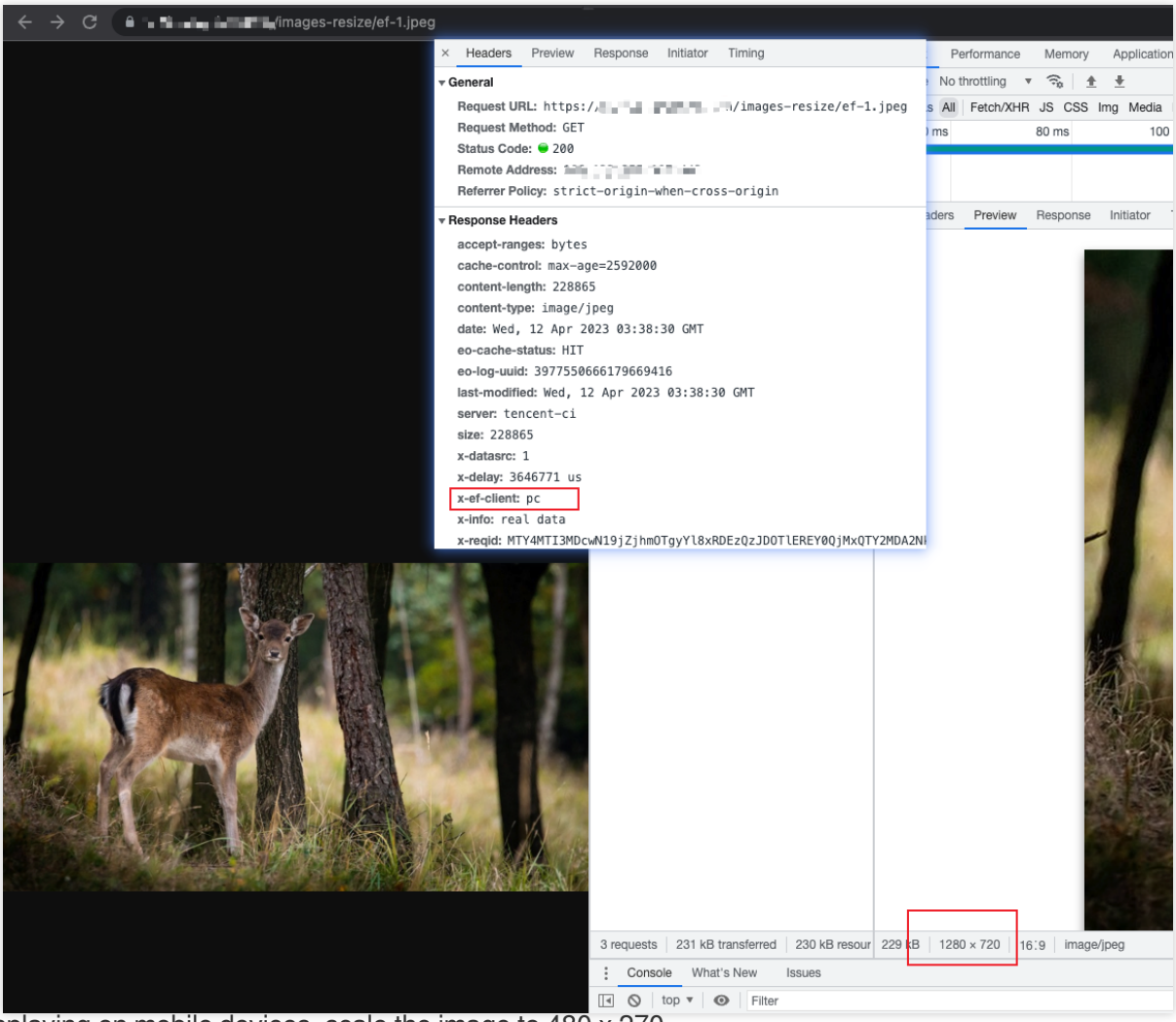
```
addEventListener('fetch', event => {
  // If the function code throws an unhandled exception, Edge Functions forwards th
  event.passThroughOnException();
  event.respondWith(handleEvent(event));
});

async function handleEvent(event) {
  const { request } = event;
  const urlInfo = new URL(request.url);
  const userAgent = request.headers.get('user-agent');

  // Request a non-image resource
  if (!/\\.(jpe?g|png)$/.test(urlInfo.pathname)) {
    return fetch(request);
  }

  // Image width on the mobile device
  let width = 480;
  const isPcClient = isPc(userAgent);

  // Image width on the PC
  if (isPcClient) {
    width = 1280;
  }

  // Scale the image
  const response = await fetch(request, {
    eo: {
      image: {
        width,
      }
    }
  });

  // Set the response header
  response.headers.set('x-ef-client', isPcClient ? 'pc' : 'mobile');
  return response;
}
```

```
// Identify the request client type
function isPc(userAgent) {
  const regex = /(phone|pad|pod|iPhone|iPod|ios|iPad|Android|Mobile|BlackBerry|IEMo

  if(regex.test(userAgent)) {
    return false;
  }

  return true;
}
```

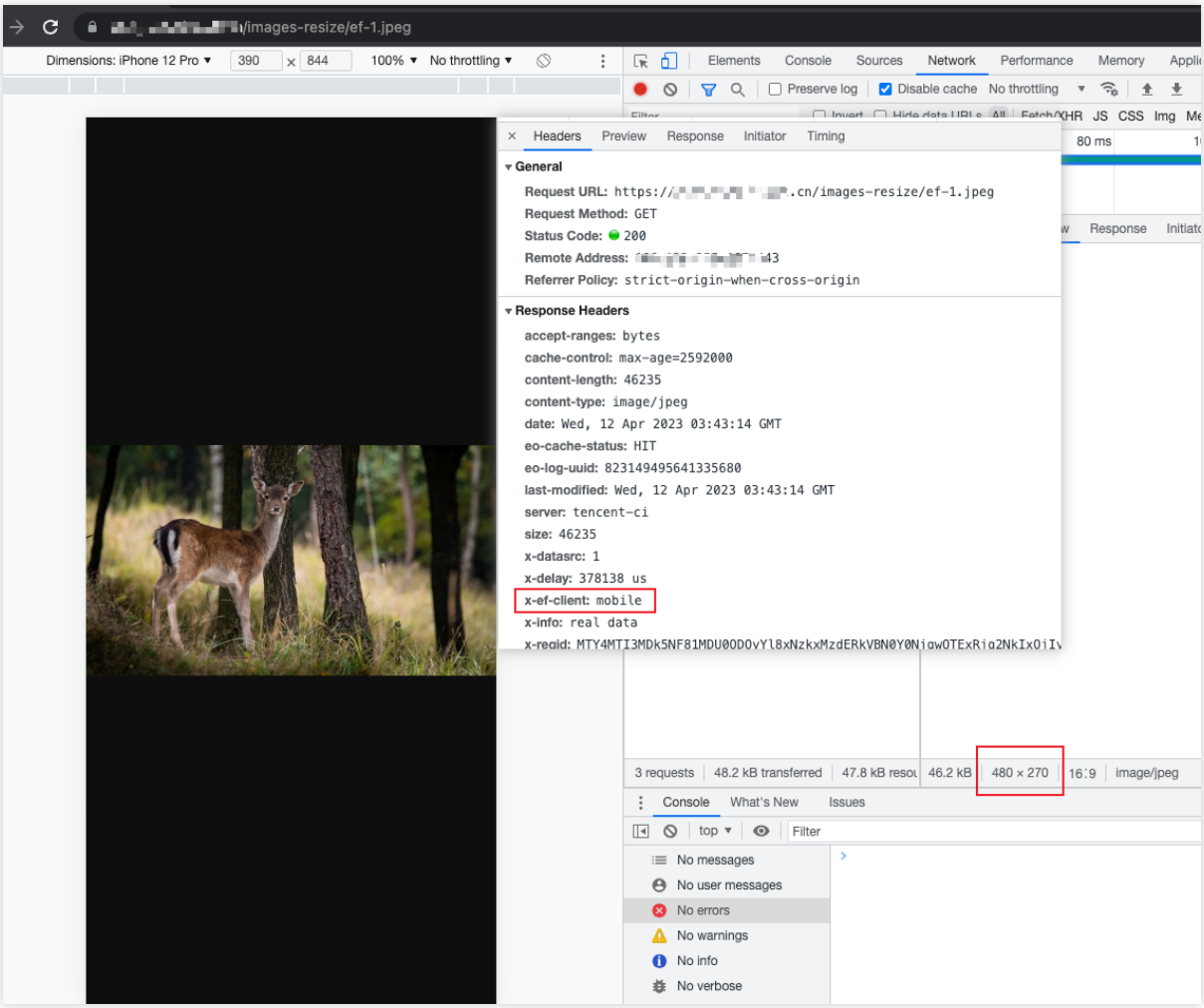
Sample Preview

In the address bar of the browser, enter a URL (such as `https://example.com/images-resize/ef-1.jpeg`) that matches a trigger rule of the edge function to preview the effect of the sample code.

For displaying on PC, scale the image to 1280 x 720.



For displaying on mobile devices, scale the image to 480 x 270.



References

Runtime APIs: Fetch

Runtime APIs: Headers

Runtime APIs: Response

Runtime APIs: Request

Image Adaptive WebP

Last updated：2024-08-26 16:48:20

This example determines whether the request header `Accept` contains `image/webp`. If so, the Edge function will automatically convert the image format to `WebP` and cache it on the EdgeOne edge node. If your Web application displays a large amount of `PNG` and `JPEG` format images and you want to automatically optimize the images at the edge to reduce traffic bandwidth costs, you can use Edge functions to implement a smooth upgrade, automatically converting `PNG` and `JPEG` format images to `WebP` format with 0 changes to the business code. For more image conversion formats, please refer to ImageProperties.

Note：

This example can execute the conversion feature normally only when the MIME type specified in the response header `Content-Type` of the source file is an image (`image/*`).

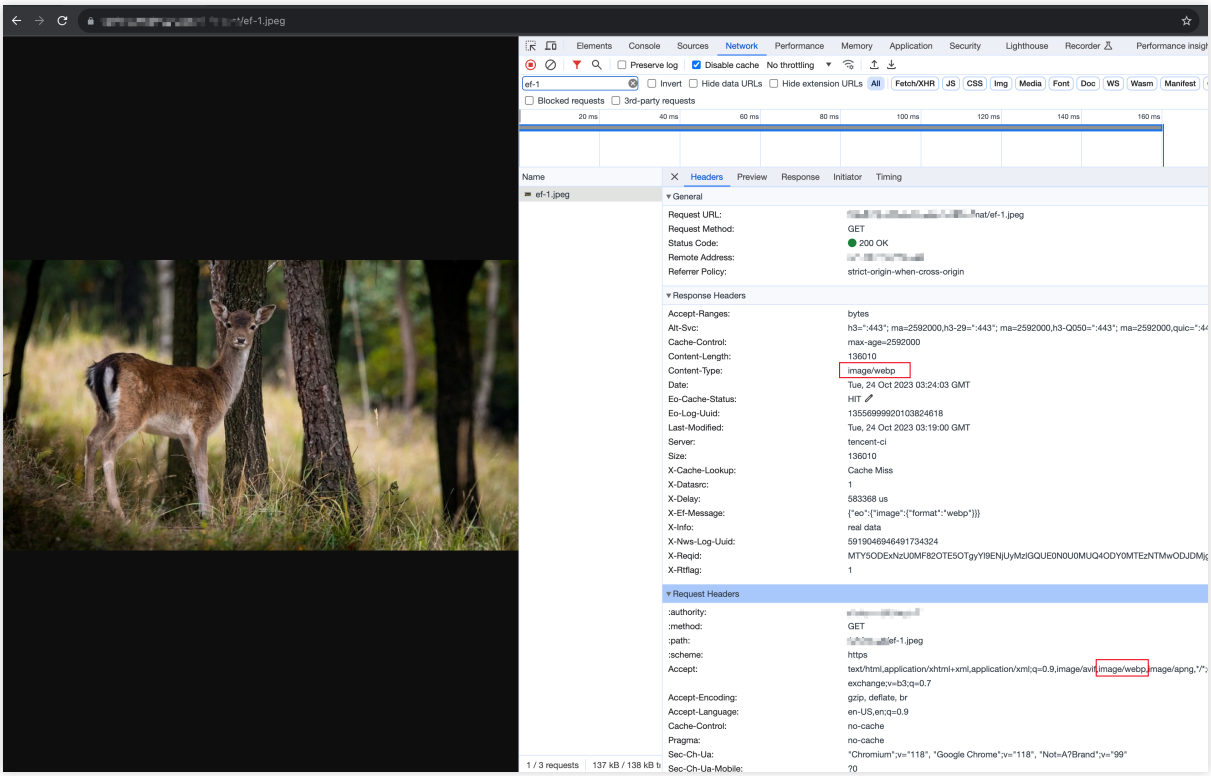
This example does not currently support converting `SVG` format images.

It is recommended to add image file extensions such as `.png` , `.jpeg` , and `.jpg` to the edge function triggering rules.

```
async function handleEvent(event) {
  const { request } = event;

  // Get the image type supported by the client
  const accept = request.headers.get('Accept');
  const option = { eo: { image: {} } };

   // Check whether the client supports WebP format images, if not, respond with th
  if (accept && accept.includes('image/webp')) {
    option.eo.image.format = 'webp';
  }

  const response = await fetch(request, option);
  return response;
}

addEventListener('fetch', event => {
   // When the function code throws an unhandled exception, the Edge function trans
  event.passThroughOnException();
  event.respondWith(handleEvent(event));
});
```

Example Preview

Enter the URL that matches the triggering rules of the Edge function in the address bar of the browser on the PC side and mobile side (e.g., `https://example.com/images-format/ef-1.jpeg`), and the image will be automatically converted to Webp format.



Related References

Runtime APIs: Fetch

Runtime APIs: Headers

Runtime APIs: Request

Runtime APIs: FetchEvent

Customize Referer restriction rules

Last updated：2024-08-26 16:48:20

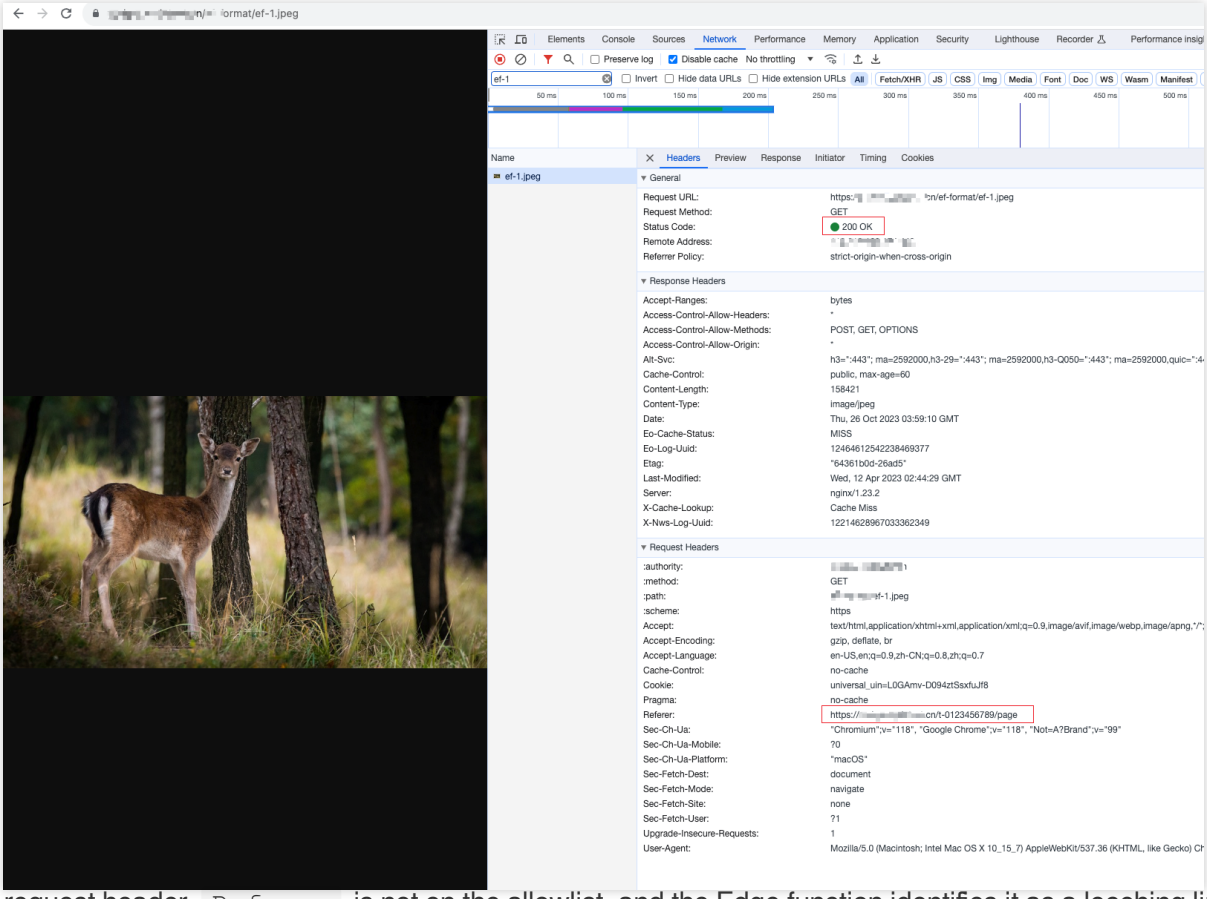
This example determines the source of a request by checking the Referer field in the HTTP request header. You can flexibly set the matching rules for the Referer field based on your needs. If the Referer field in a request is missing or its value does not match the preset domain name list, the edge function will block such requests and return a 403 status code to indicate that the access is denied. This is commonly used to restrict access to resources on websites.

```
async function handleRequest(request) {
  // Collect the Referer
  const referer = request.headers.get('Referer');

  // If the Referer is empty, access is denied
  if (!referer) {
    return new Response(null, { status: 403 });
  }

  // Set Referer allowlist
  const urlInfo = new URL(request.url);
  const refererRegExp = new RegExp(`^https?:\\/\\/${urlInfo.hostname}\\/t-[0-9a-z]{

  // If the Referer is not on the allowlist, access is denied
  if (!refererRegExp.test(referer)) {
    return new Response(null, { status: 403 });
  }

  // Normal request, access EdgeOne node cache or origin-pull
  return fetch(request);
}

addEventListener('fetch', event => {
  // When the function code throws an unhandled exception, the Edge function transm
  event.passThroughOnException();
  event.respondWith(handleRequest(event.request));
});
```

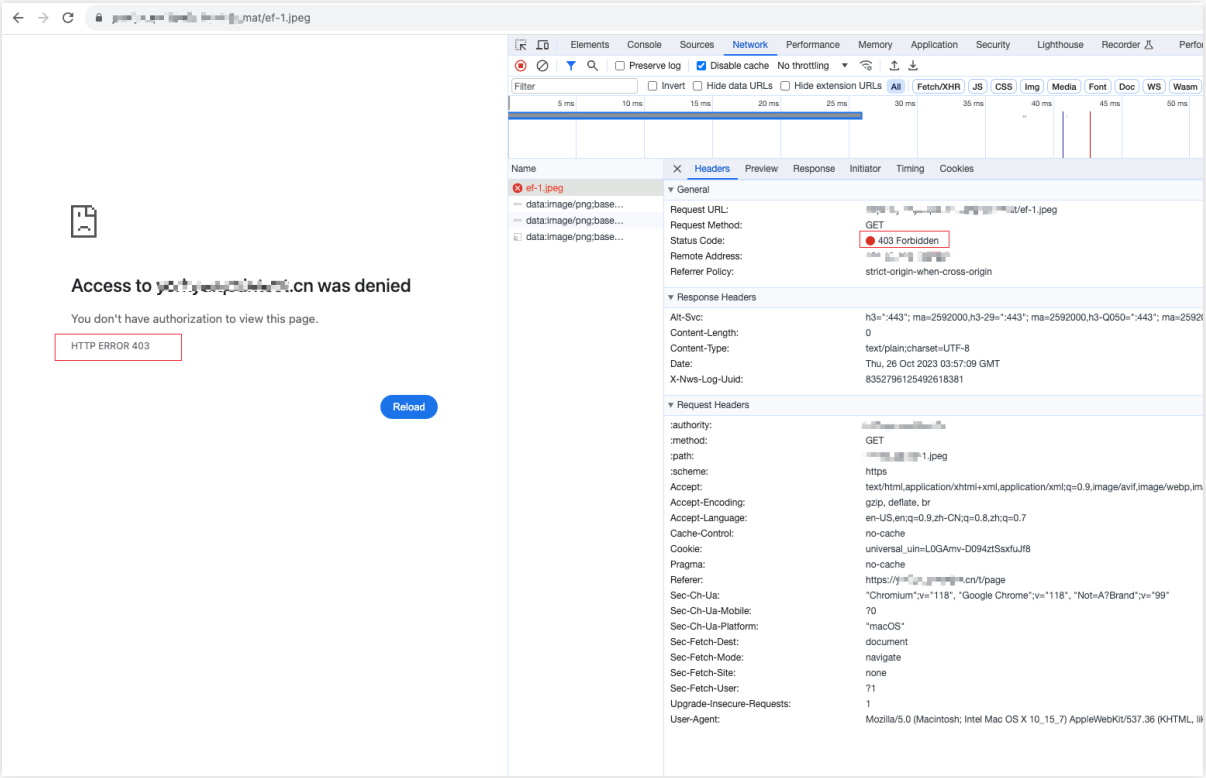
Example Preview

Enter the URL that matches the Edge function triggering rules in the address bar of the browser on the PC end and mobile end (e.g., `https://example.com/images/ef-1.jpeg`) to preview the example effect.

HTTP request header `Referer` is https://example.com/t-0123456789/page, and the Edge function responds normally to the image.



HTTP request header `Referer` is not on the allowlist, and the Edge function identifies it as a leeching link and responds with a 403 status code.



Related References

Runtime APIs: Fetch

Runtime APIs: Headers

Runtime APIs: Response

Remote Authentication

Last updated：2023-10-30 17:46:16

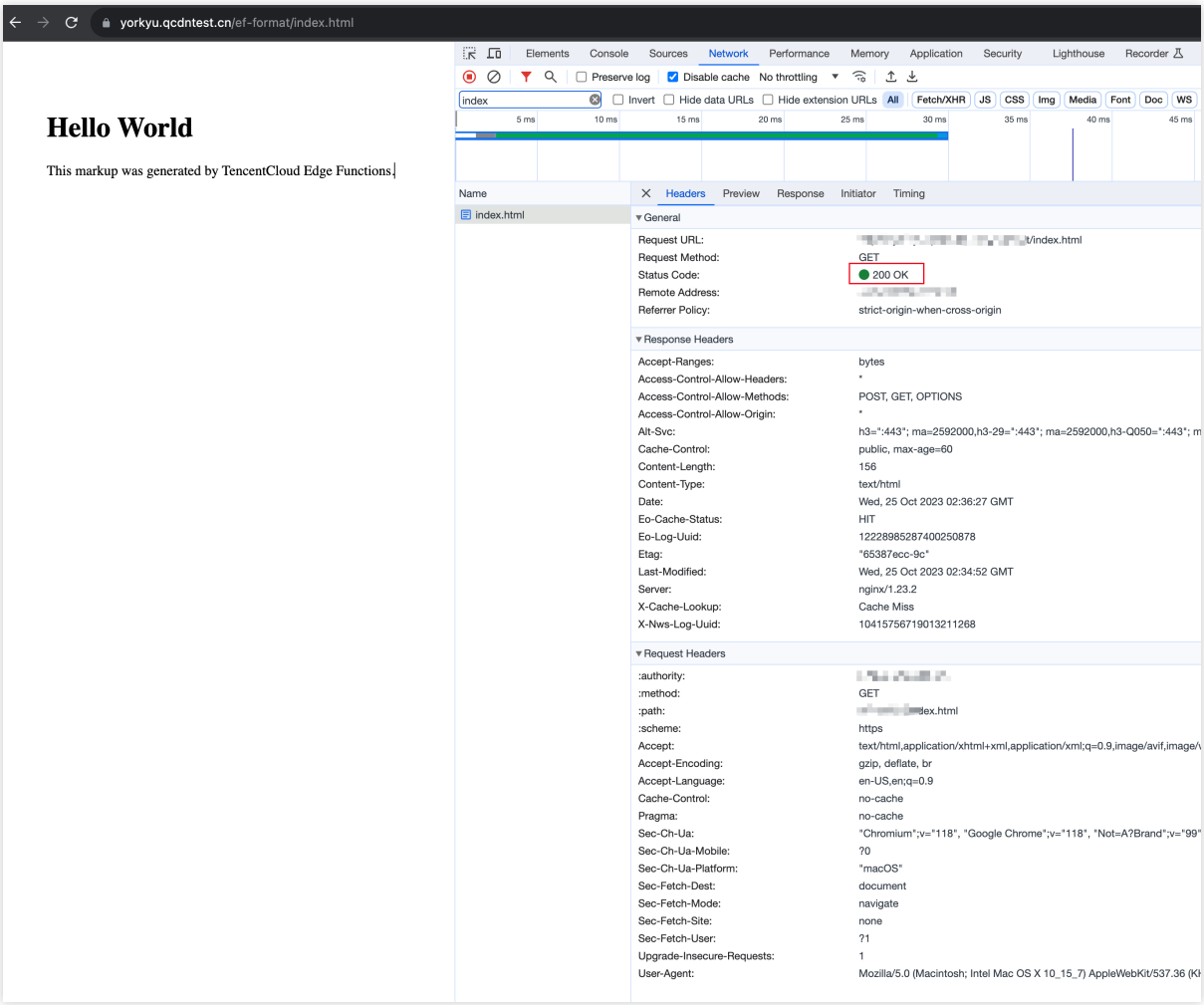
In order to avoid customers' resources being accessed by illegal users, this example transmits the request to the customer-specified remote authentication server. The authentication server verifies the user's request, and the Edge functions decide whether to allow access to the target resources based on the check result returned by the remote authentication server. If the authentication fails, the client will be responded with a 403 status code.

```
async function handleRequest(request) {
  // Remote authentication API address
  const checkAuthUrl = 'https://www.example.com/';
  // Initiate remote authentication
  const checkAuthRes = await fetch(checkAuthUrl);

  // Authentication passed, normal access to resources
  if (checkAuthRes.status === 200) {
    return fetch(request, {
      headers: request.headers,
    });
  }

  // Authentication failed, prohibit access to resources
  return new Response(null, {
    status: 403
  });
}

addEventListener('fetch', e => {
  e.respondWith(handleRequest(e.request));
});
```

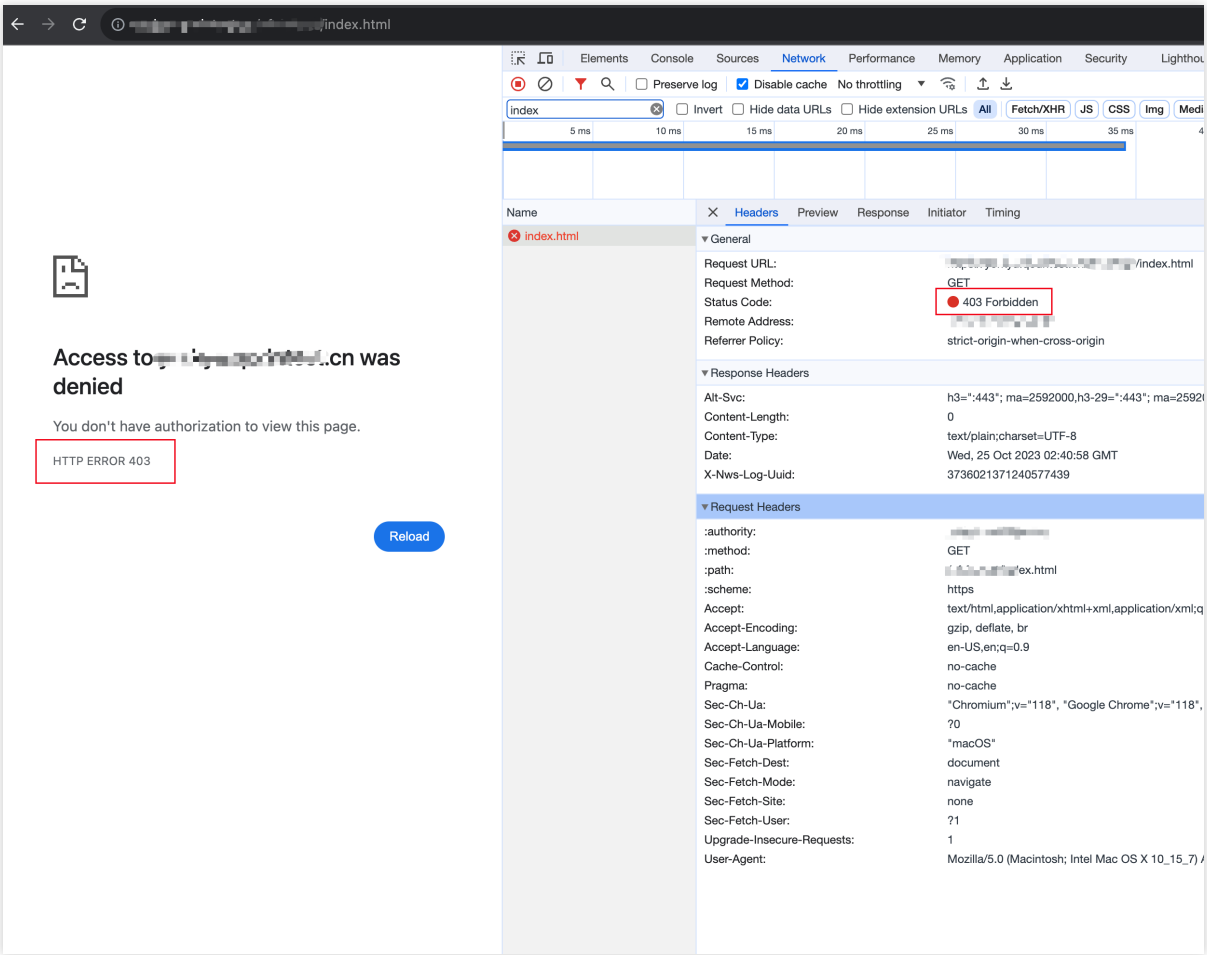
Example preview

Enter the URL that matches the triggering rules of the Edge functions in the address bar of the browser on both PC and mobile (e.g., `https://example.com/app/index.html`) to preview the example effect.

Authentication passed, normal access to resources.



Authentication failed, prohibit access to resources.



Related references

Runtime APIs: Fetch

Runtime APIs: Response

HMAC Digital Signature

Last updated：2023-12-13 10:54:00

Hash-based Message Authentication Code (HMAC) is a kind of message authentication code based on hash function, mainly used to verify data integrity and identity authentication. `Web Crypto API` is used in this example to achieve HMAC-SHA256 signature and store the signature information in a request header. It achieves data integrity or identity authentication collaborating with origin server. Developers can modify the code based on their specific needs.

Note

Cooperation with the origin server is required in this example, which means that the origin server must possess the corresponding signature verification algorithm.

For live network use of the code provided in this example, modification must be made in accordance with the comments.

Sample code

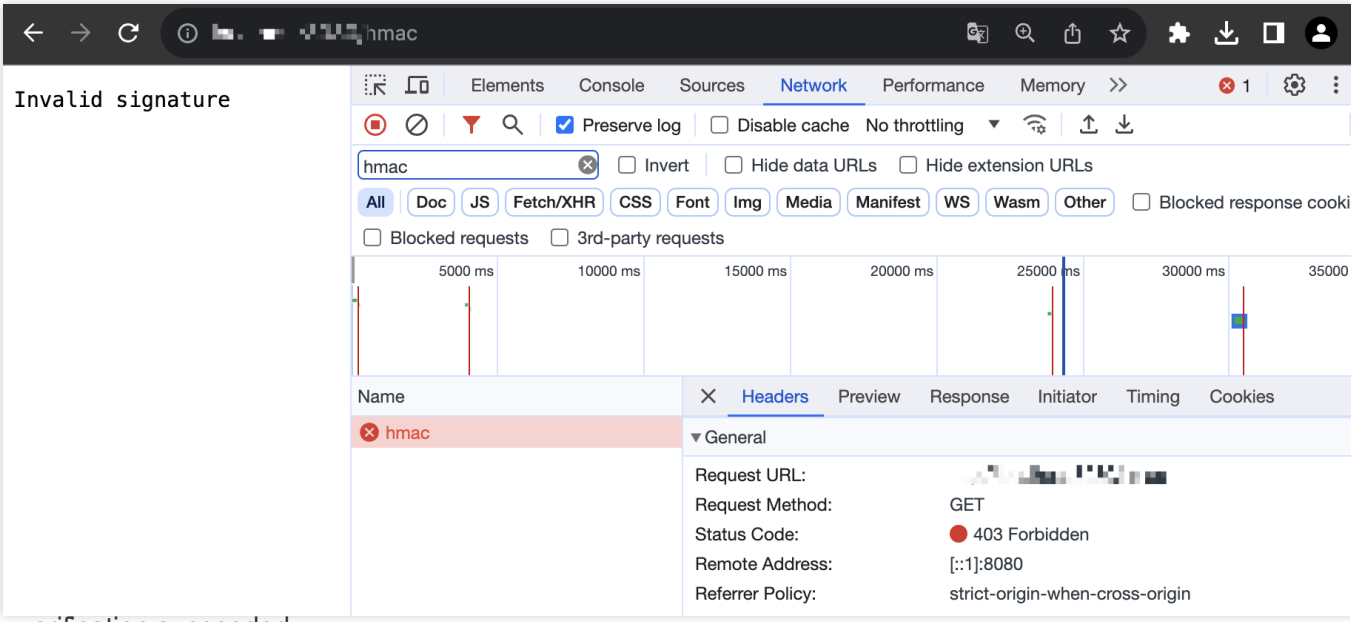
```
function uint8ArrayToHex(uint8Array) {
  return Array.prototype.map.call(uint8Array, x => ('0' + x.toString(16)).slice(-2)
}

async function generateHmac({ secretKey, message, hash }) {
  const encoder = new TextEncoder();
  const secretKeyBytes = encoder.encode(secretKey);
  const messageBytes = encoder.encode(message);

  const key = await crypto.subtle.importKey('raw', secretKeyBytes, { name: 'HMAC',

  const signature = await crypto.subtle.sign('HMAC', key, messageBytes);
  const signatureArray = new Uint8Array(signature);
  return uint8ArrayToHex(signatureArray);
}

async function handleRequest(request) {
  const secretKey = 'YOUR_SECRET_KEY';
  // Please modify the message to the information that needs to be signed, generall
  const message = 'YOUR_MESSAGE';

  // Choose one from SHA-1, SHA-256, SHA-384, and SHA-512 for hash.
  const hmac = await generateHmac({ secretKey, message, hash: 'SHA-256' });
  request.headers.set('Authorization', `HMAC-SHA256 ${hmac}`);
```

```
  return fetch(request);
}

addEventListener('fetch', event => {
  event.passThroughOnException();
  event.respondWith(handleRequest(event.request));
});
```

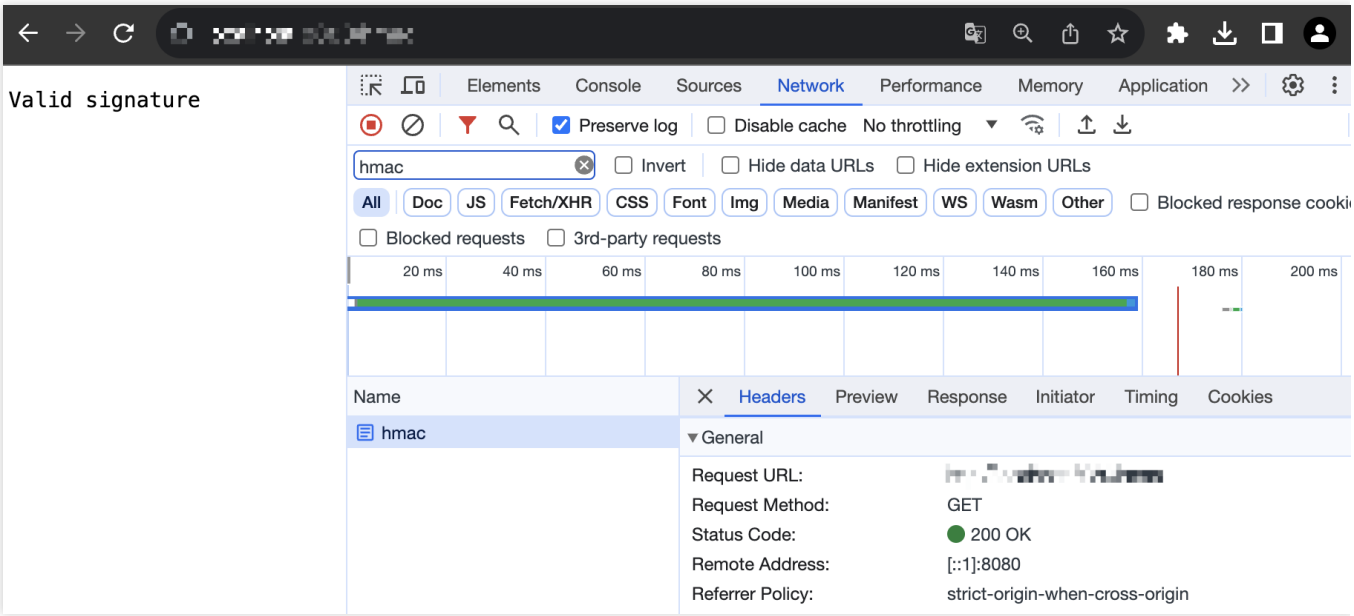
Sample Preview

Enter a URL (such as `https://example.com/hmac`) that matches the trigger rule of edge function in the address bar of a browser on both the PC and mobile terminal to preview the example effect.

Identity verification failed.



Identity verification succeeded.



Related References

Runtime APIs: Fetch

Runtime APIs: Web Crypto

Runtime APIs: Headers

Runtime APIs: Response

Naming a Downloaded File

Last updated：2023-12-13 10:55:46

In this example, the modification of the Content-Disposition in the response headers achieves the modification of the downloaded file name according to the `fileName` parameter in the request URL.

```
addEventListener('fetch', event => {
  event.passThroughOnException();
  event.respondWith(handleRequest(event.request));
});

async function handleRequest(request) {
  const url = new URL(request.url);
  const fileName = url.searchParams.get('fileName');

  const response = await fetch(request);

  // Evaluate the response status code and the search parameter
  if (response.status !== 200 || !fileName) {
    return response;
  }

  // Modify the Content-Disposition response header
  response.headers.append('Content-Disposition', `attachment; filename="${fileName}
  return response;
}
```

Sample Preview

Enter a URL (such as `https://example.com/origin.exe?fileName=modified.exe`) that matches the trigger rule of edge function in the address bar of the browser on both the PC and mobile terminal to preview the example effect.

Related References

Runtime APIs: Fetch

Runtime APIs: Response

©2013-2025 Tencent Cloud International Pte. Ltd.

Obtaining Client IP Address

Last updated：2025-06-18 16:29:56

Since the front end cannot directly obtain the client IP address, it's often necessary to obtain the client IP address through the server side or third-party services in multiple business scenarios. This example gets the client IP through the Request object's eo.clientIp property, and assembled into data in the form of JSON to respond to the clients, which succeeds in obtaining client IP address by the use of edge function.

```
function handleRequest(request) {
  // Obtain the client IP through the request.eo.clientIp
  const ip = request.eo.clientIp || '';

  // Respond with JSON data
  return new Response(JSON.stringify({ ip }), {
    headers: { 'content-type': 'application/json' },
  });
}

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request));
});
```

Sample Preview

Enter the URL that matches the Edge function trigger rule in your browser address bar to preview the example effect:

Related References

Runtime APIs: Fetch
Runtime APIs: Request
Runtime APIs: Response

©2013-2025 Tencent Cloud International Pte. Ltd.

Complex origin-pull URL rewriting

Last updated：2025-06-18 16:54:04

In some business scenarios, you need to modify client request URLs before sending them to the origin server. This example implements two complex URL rewriting scenarios using Edge functions:

1. **Path regex replacement**: Using regex capture groups for path replacement, replacing paths starting with `/a` with `/path-a` while preserving the subsequent path structure. Path regex replacement enables seamless mapping from old paths to new paths, for example when resource directory structures change during content management system updates. Regex capture groups preserve key parts of the original path to ensure content remains accessible while adapting to the new directory structure.
2. **Path case conversion**: Converting the entire path to lowercase, conforming to Web standards best practices. For websites using object storage services (such as Tencent Cloud Object Storage) as their origin, since object storage typically distinguishes between uppercase and lowercase, uniform conversion can simplify resource management and prevent resource access failures due to case sensitivity issues.

Sample Code

```
async function handleEvent(event) {
  try {
    const request = event.request;
    const url = new URL(request.url);
    let pathname = url.pathname;

    // Use regular expression for path replacement, replace /a or /a/xxx with /path
    if (pathname.startsWith('/a')) {
      const aPathRegex = /^\\/a(\\/.*)?$/;
      pathname = pathname.replace(aPathRegex, '/path-a$1');
    }

    // Path case conversion, directly convert the entire path to lowercase
    if (pathname.startsWith('/b')) {
      pathname = pathname.toLowerCase();
    }

    url.pathname = pathname;

    // Create a new request
    const newRequest = new Request(url.toString(), {
      method: request.method,
      headers: request.headers,
```

```
      body: request.body,
      redirect: 'manual'
    });

    const response = await fetch(newRequest);
    return event.respondWith(response);
  } catch (err) {
    console.log(err);
  }
}

addEventListener('fetch', event => {
  handleEvent(event);
});


javascript
```

Example Preview

Enter a URL that matches the Edge function trigger rules in the browser address bar to preview the example effect.

Path regex replacement: Replace paths starting with `/a` with `/path-a`, while preserving the subsequent path structure.

Path case conversion: Convert the entire path to lowercase.

Related References

Runtime APIs: Fetch
Runtime APIs: Request
Runtime APIs: Response

Best Practices

Overview

Last updated：2025-04-30 17:52:03

Title	Practice Overview
Adaptive Image Format Conversion via Edge Functions	This article introduces how to automatically convert image formats using edge functions without modifying the original client request URL. It describes how to automatically determine the image file format to return based on the User-Agent header in the client request and trigger the image format conversion.
EdgeOne achieves session persistence based on Client IP address	This article explains how to utilize EdgeOne 's edge functions and rule engine to ensure that requests from the same Client IP address are always forwarded to the same backend origin server , thus maintaining the continuity of the session and the consistency of data.
EdgeOne achieves Specify Origin Retrieval based on Client Geographic Features	This article explains how to utilize EdgeOne 's edge functions and rule engine to achieve origin retrieval to a specified origin server based on the user geographic location.

©2013-2025 Tencent Cloud International Pte. Ltd.

Adaptive Image Format Conversion via Edge Functions

Last updated：2023-07-10 09:48:54

This document describes how to the convert image format by using edge functions without changing the original client request URL. Edge functions can automatically convert the image according to the `User-Agent` header in the client request.

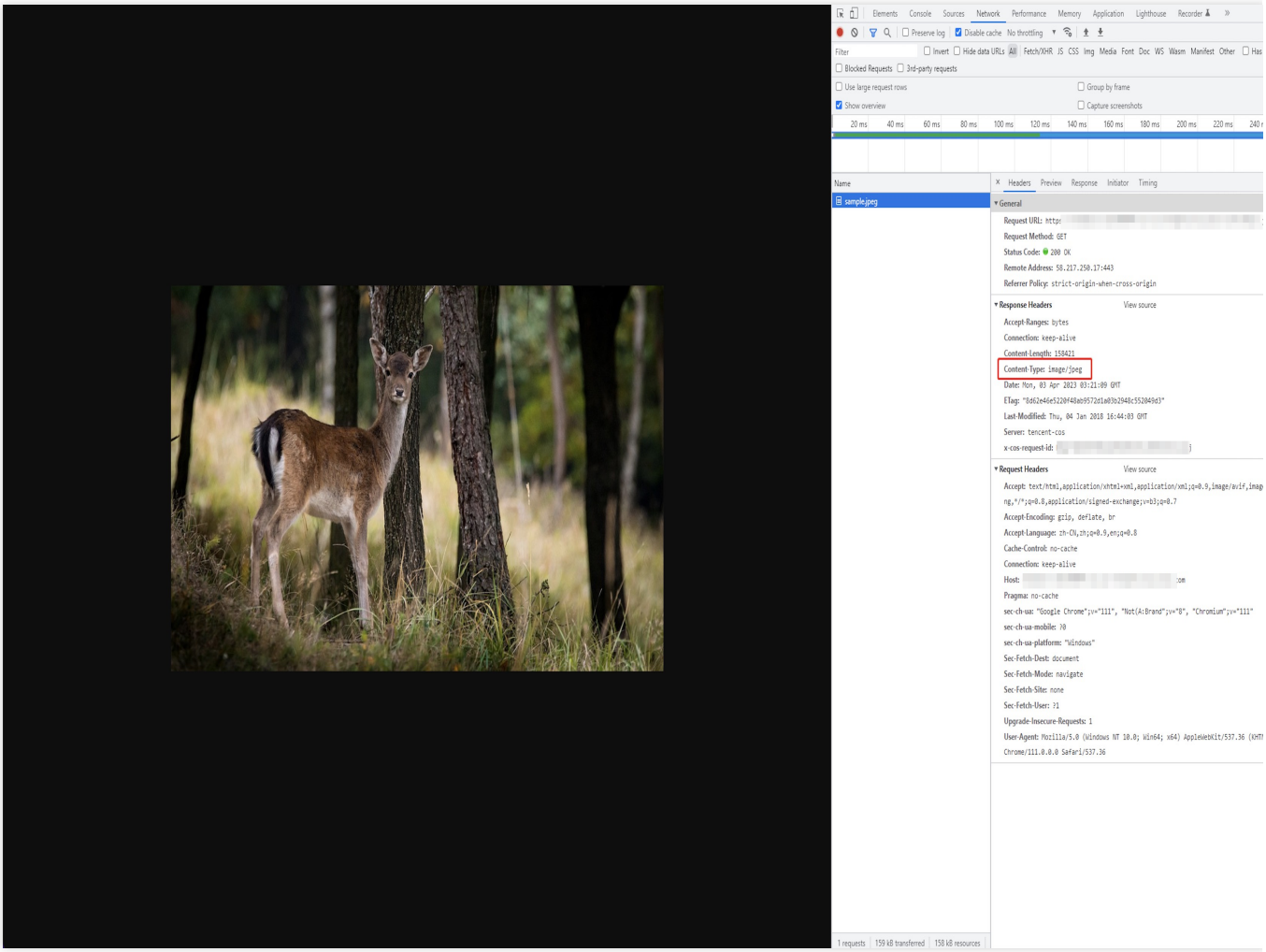
Background

For sites providing lots of images, adapting the image for different browser is a must. At the same time, they need to the compress the images without losing quality, so as to reduce the cost on data transfer. For example, they need to:
Return webp images for Chrome, Opera, Firefox and Edge browsers.
Return jp2 images for the Safari browser.
Return jxr images for the IE browser.
Return webp images for all the other browsers.

Edge functions can automatically convert the image according to the `User-Agent` header in the client request. If you want to proactively convert the image format in the request URL, you can also use the image processing feature.

Use Cases

`example.com` is connected to EdgeOne, with all the images stored under `http://image.example.com/image/` . You need to automatically convert all image files under this directory according to the browser type of the client. `https://image.example.com/image/test.jpg` is taken as the test image.



Directions

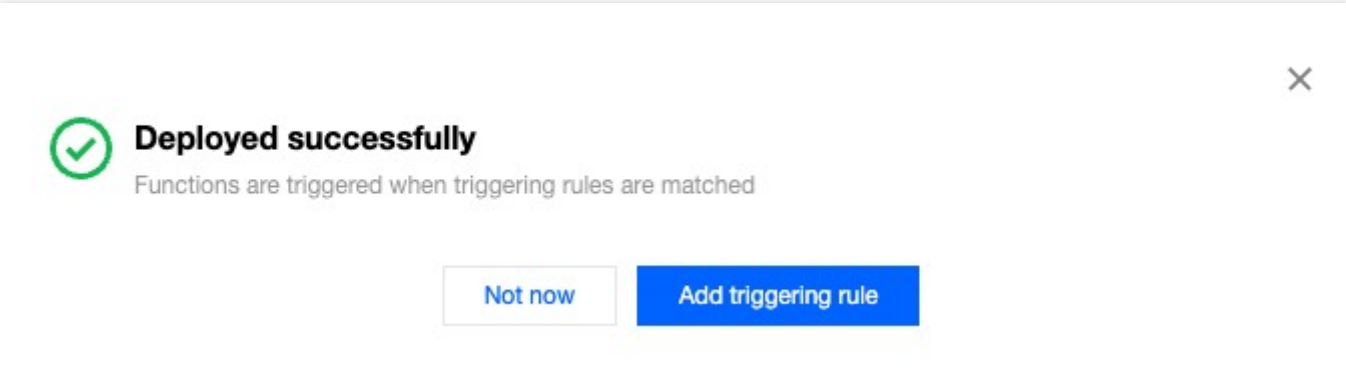
1. Log in to the EdgeOne console and click the target site in the site list to display second-level menus for site management.
2. In the left sidebar, click **Edge Functions** > **Function Management**.
3. On the **Function management** page, click **Create function**.
4. On the function creation page, enter the function name, description and codes. See below for the sample codes:

```
// Browser image format
const broswerFormat = {
  Chrome: 'webp',
  Opera: 'webp',
  Firefox: 'webp',
  Safari: 'jp2',
  Edge: 'webp',
  IE: 'jxr'
};
```

```
addEventListener('fetch', event => {
  // If the function code throws an unhandled exception, Edge Functions forwards th
  event.passThroughOnException();
  event.respondWith(handleEvent(event));
});

async function handleEvent(event) {
  const { request } = event;
  const userAgent = request.headers.get('user-agent');
  const bs = getBroswer(userAgent);
  const format = broswerFormat[bs];

  // Do not convert the image format
  if (!format) {
    return fetch(request);
  }

  // Convert image format
  const response = await fetch(request, {
    eo: {
      image: {
        format
      }
    }
  });

  // Set the response header
  response.headers.set('x-ef-format', format);

  return response;
}

function getBroswer(userAgent) {
  if (/Edg/i.test(userAgent)) {
    return 'Edge'
  }
  if (/Trident/i.test(userAgent)) {
    return 'IE'
  }
  if (/Firefox/i.test(userAgent)) {
    return 'Firefox';
  }
  if (/Chrome/i.test(userAgent)) {
    return 'Chrome';
  }
  if (/Opera|OPR/i.test(userAgent)) {
```

```
      return 'Opera';
    }
    if (/Safari/i.test(userAgent)) {
      return 'Safari'
    }
  }
```

5. Click **Create and deploy function**. Wait till the deployment completed. Click **Add trigger rule**.

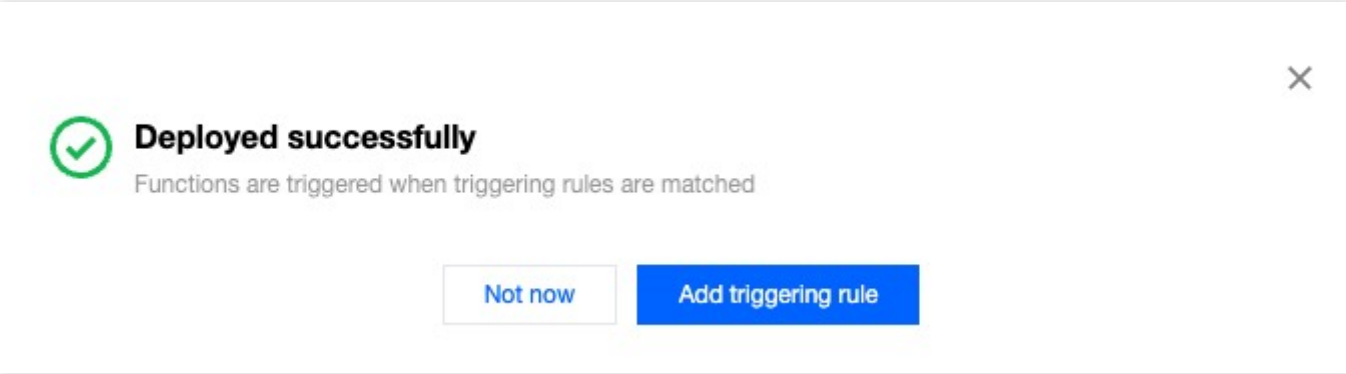


6. Configure the condition to trigger this function. In this case, you can configure two conditions, which are combined with AND.

When the request HOST equals to `Image.example.com` .

When the request URL Path equals to `/image/*` .

When both the conditions are met, the edge function is triggered to process the image automatically.



7. Click **Save**.

8. You can verify the functions in two ways:

Using curl to test

Browser Access Test

You can run a curl command with the specified User-Agent to test.

Chrome

Safari

IE

To test on Chrome browser on Mac/Linux OS, run the following command on the device: `curl --user-agent "chrome" https://image.example.com/image/test.jpg -i`

Check if `Content-Type` in the response is `image/webp`.

```
[redacted] ~ % curl --user-agent "chrome" h[redacted].jpg
HTTP/1.1 200 OK
x-ef-format: webp
EO-LOG-UUID: 3816446099087859674
Cache-Control: max-age=2592000
Last-Modified: Fri, 14 Apr 2023 08:14:28 GMT
Accept-Ranges: bytes
Connection: keep-alive
Date: Fri, 14 Apr 2023 08:14:28 GMT
X-RtFlag: 1
X-ReqId: MTY4MTQ2MDA2Nl83NzE5OTgyYl85QkZERTQ5OEFDQjE0N0Q4ODBBM0ZDMTQ5QjU4NzI4MA==
Size: 123220
X-DataSrc: 1
X-Info: real data
EO-Cache-Status: HIT
X-Delay: 2316668 us
Content-Type: image/webp
Server: tencent-ci
Content-Length: 123220
```

On Mac/Linux OS, run the following command on the device: `curl --user-agent "safari" https://image.example.com/image/test.jpg -i`

Check if `Content-Type` in the response is `image/jp2`.

```
[redacted] ~ % curl --user-agent "safari" [redacted].jpg
HTTP/1.1 200 OK
x-ef-format: jp2
EO-LOG-UUID: 299090522723185511
Cache-Control: max-age=2592000
Last-Modified: Fri, 14 Apr 2023 08:42:27 GMT
Accept-Ranges: bytes
Connection: keep-alive
Date: Fri, 14 Apr 2023 08:42:27 GMT
X-RtFlag: 1
X-ReqId: MTY4MTQ2MTc0NV83NzE5OTgyYl8zRjRCODlEODM2NDg0RTEyQTJGRTYyNTZDODI1MkEyMg==
Size: 121014
X-DataSrc: 1
X-Info: real data
EO-Cache-Status: MISS
X-Delay: 2747522 us
Content-Type: image/jp2
Server: tencent-ci
Content-Length: 121014
```

On Mac/Linux OS, run the following command on the device: `curl --user-agent "Trident" https://image.example.com/image/test.jpg -i`

Check if `Content-Type` in the response is `image/vnd.ms-photo`.

```
[redacted] ~ % curl --user-agent "Trident" [redacted].jpg -i
HTTP/1.1 200 OK
x-ef-format: jxr
EO-LOG-UUID: 16823953457232177833
Cache-Control: max-age=2592000
Last-Modified: Fri, 14 Apr 2023 08:49:50 GMT
Accept-Ranges: bytes
Connection: keep-alive
Date: Fri, 14 Apr 2023 08:49:50 GMT
X-RtFlag: 1
X-ReqId: MTY4MTQ2MjE5MF83NzE5OTgyYl80MDU1MjdGMjJDMTQ0RjEwODlBNjRFREFBMUUzMTkwNQ==
Size: 138140
X-DataSrc: 1
X-Info: real data
EO-Cache-Status: MISS
X-Delay: 97009 us
Content-Type: image/vnd.ms-photo
Server: tencent-ci
Content-Length: 138140
```

Access the test image address `https://image.example.com/image/test.jpg` with different browsers.

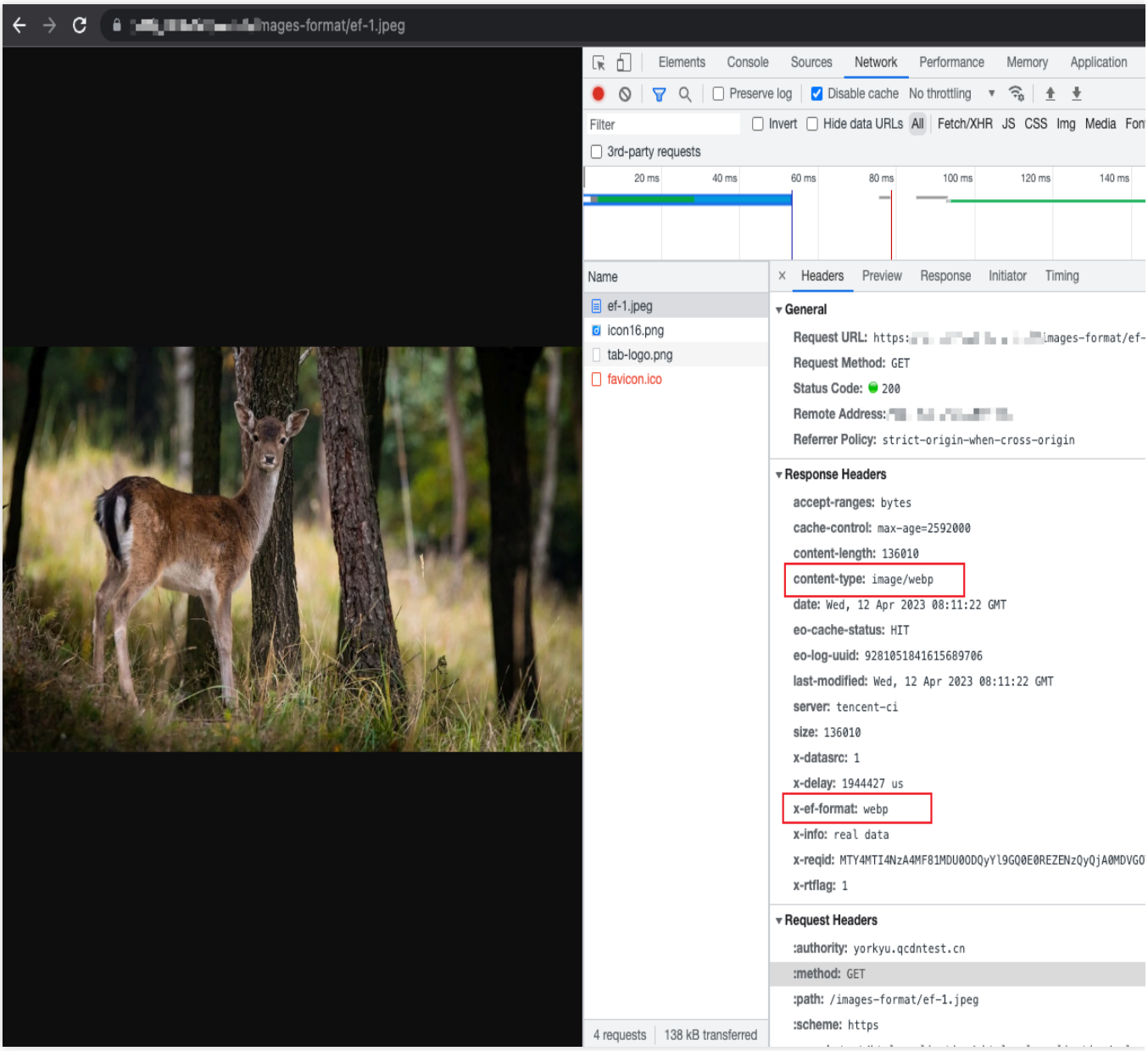
Check the format for image returned to see whether the edge function works properly.

Chrome

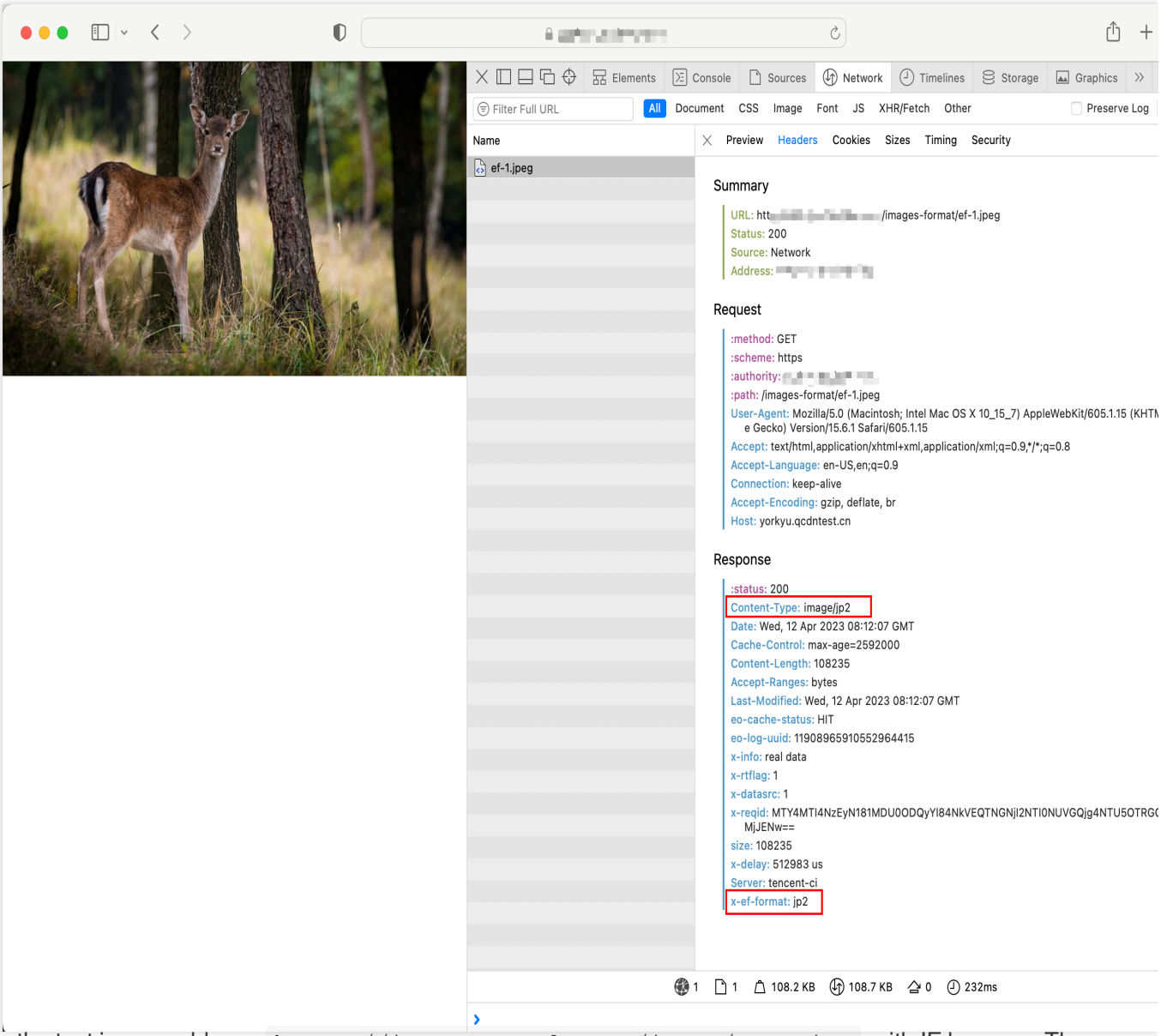
Safari

IE

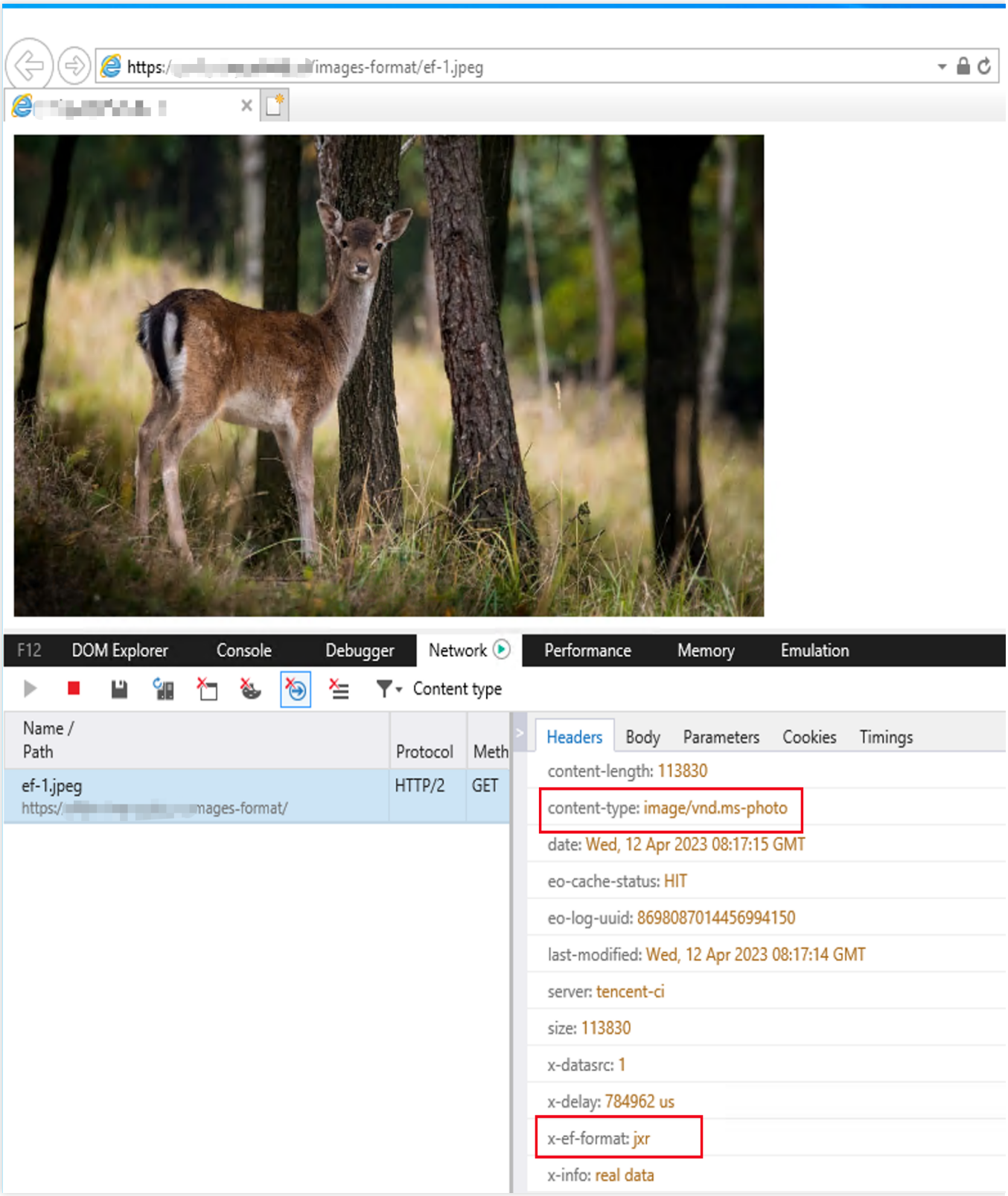
Access the test image address `https://image.example.com/image/test.jpg` wit Chrome browser. The responded image should be in webp format.



Access the test image address `https://image.example.com/image/test.jpg` with Safari browser. The responded image should be in jp2 format.



Access the test image address `https://image.example.com/image/test.jpg` with IE browser. The responded image should be in jxr format.



Relevant Documentation

Sample function: Image auto-adaptation

Resizing and Converting Images

©2013-2025 Tencent Cloud International Pte. Ltd.