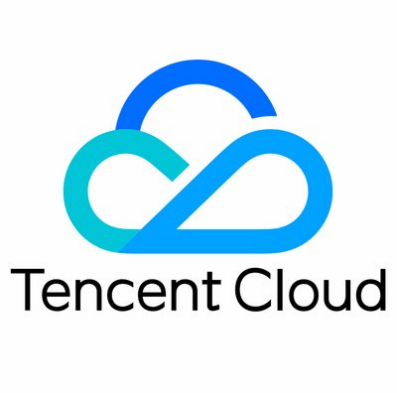
Tencent Cloud EdgeOne

Edge Functions

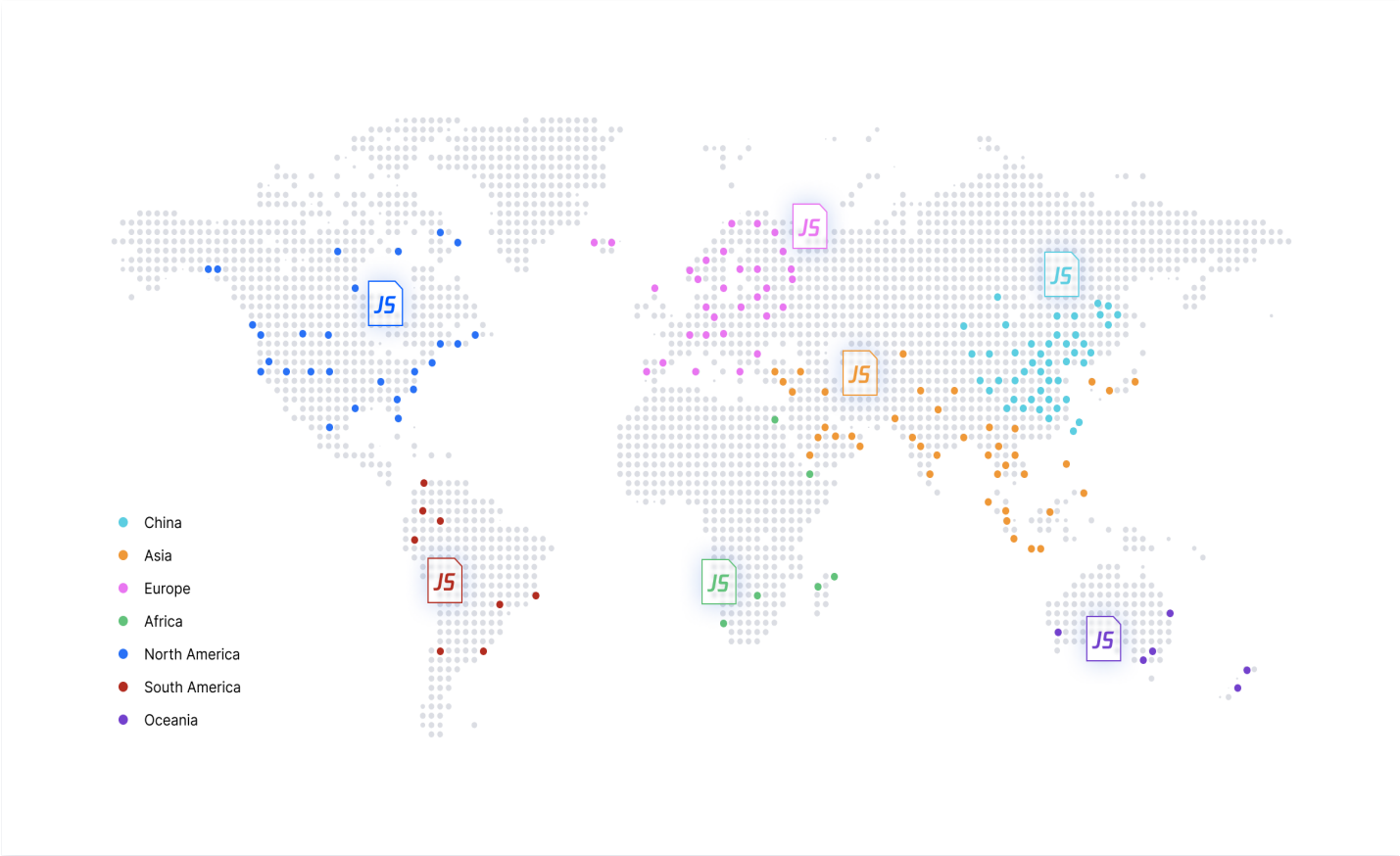


Contents

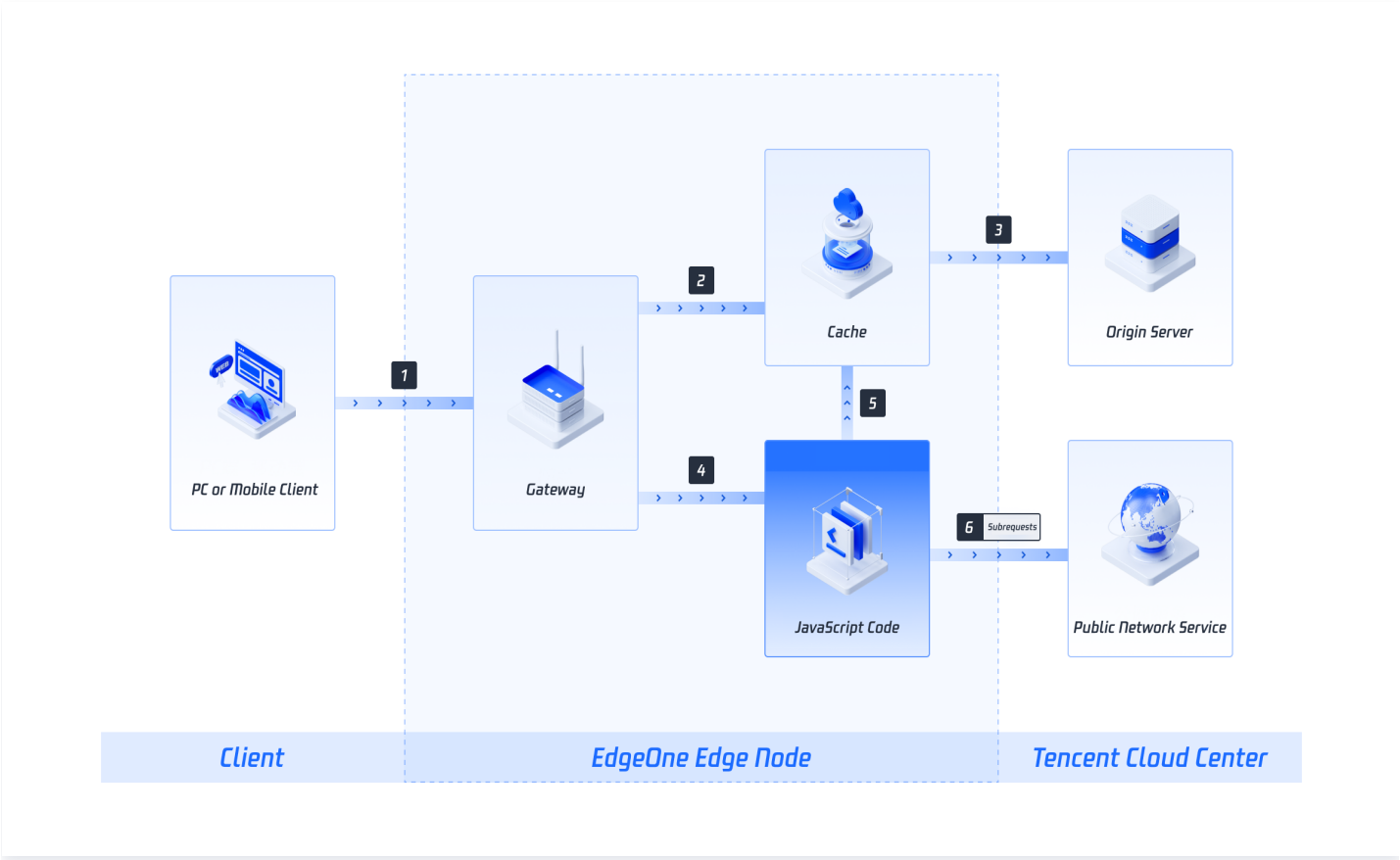
Edge Functions Overview

Last updated: 2023-09-07 15:14:51

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.



Principle Overview



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) Client request > Arrives at the EdgeOne edge node gateway > (2) If the node has a cache, the cache responds > (3) If the cache does not hit, the origin server responds.
2. If a client request hits the configured function triggering rule, the request is handled in one of the following processes:
   - (1) Client request > Arrives at the EdgeOne edge node gateway > (4) Edge function takes over and executes your JS code > (5) Sub-request accesses cache > (3) If the cache does not hit, the origin server responds.
   - (1) Client request > Arrives at the EdgeOne edge node gateway > (4) Edge function takes over and executes your JS code > (6) Sub-request accesses public network services.

Benefits

Distributed deployment

Edge Functions service is deployed distributedly on over 2,800 EdgeOne nodes.

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.

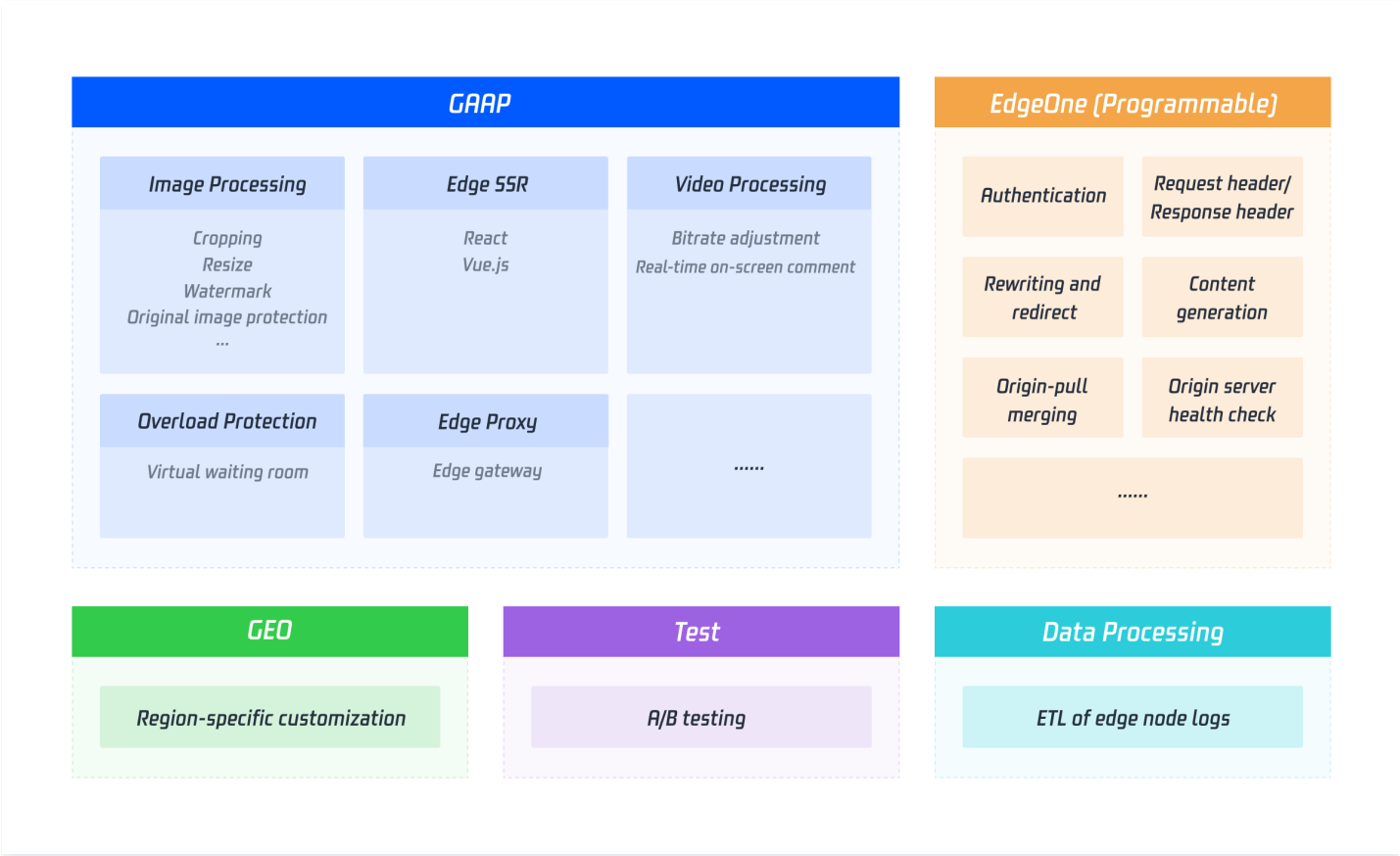
Scalability

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 your need to manage the servers (memory, CPU, network and more). You can focus on the development of business code.

Scenarios



Getting Started

Last updated: 2023-09-07 15:14:59

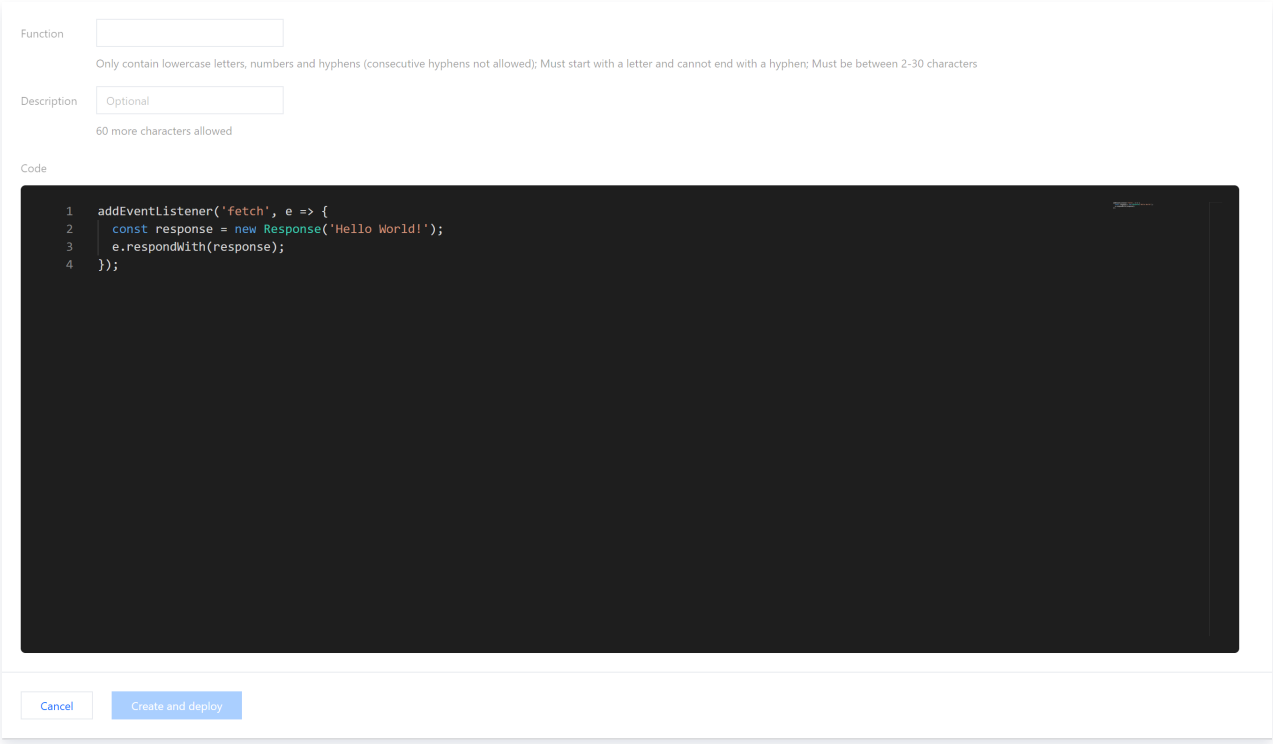
This document describes how to create a simple edge function and use the function to redirect a request to another URL and return the custom response header.

Prepare.

- Edge functions are configured on the site level. To use an edge function, you must first create a site. For more information, see Adding a Site.
- (Optional) After successfully establishing the site, you have the option to create a subdomain under this site for function testing. For more information, see Creating a Subdomain for Function Testing.

Step 1: Creating and Deploying a Function

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the Edge Functions page, click **Create Function**.
4. On the 'Create New Edge Function' page, configure the relevant parameters and click on **Create and Deploy**.



Parameter Description:

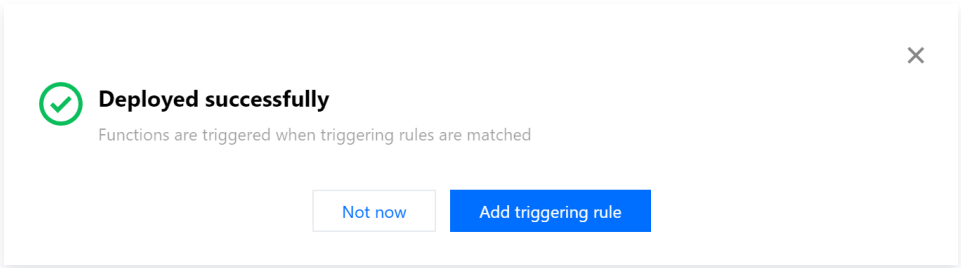
- Function: (Required) Example: test-edgefunctions.
- Description: (Optional) Example: Custom HTML page and response header.
- Code: Copy the following function code and paste it into the code editor in the console to replace the default code:

```
const html = `
  <!DOCTYPE html>
  <body>
```

```
 <h1>Hello World</h1>
 <p>This markup was generated by 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));
});
```

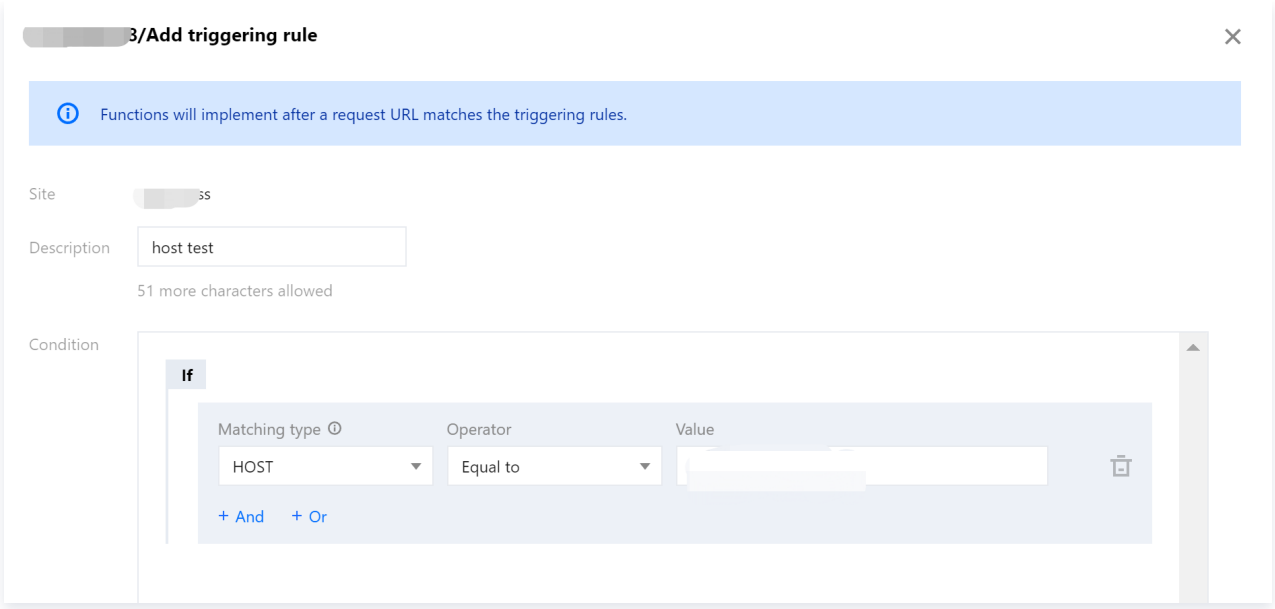
Note
The preceding code creates a custom HTML and adds the following custom response header: x-edgefunctions-test: Welcome to use Edge Functions.

5. If you see the following pop-up dialog box, the deployment is successful.



Step 2. Configuring a Triggering Rule

1. After successfully creating and deploying the function, follow the prompts to click **Add Trigger Rule**.
2. On the Add Trigger Rule page, select the match type as "HOST", the operator as "equals", and the value as the subdomain you have added. Click **Confirm** to create the trigger rule.



Step 3: Verifying the Edge Function

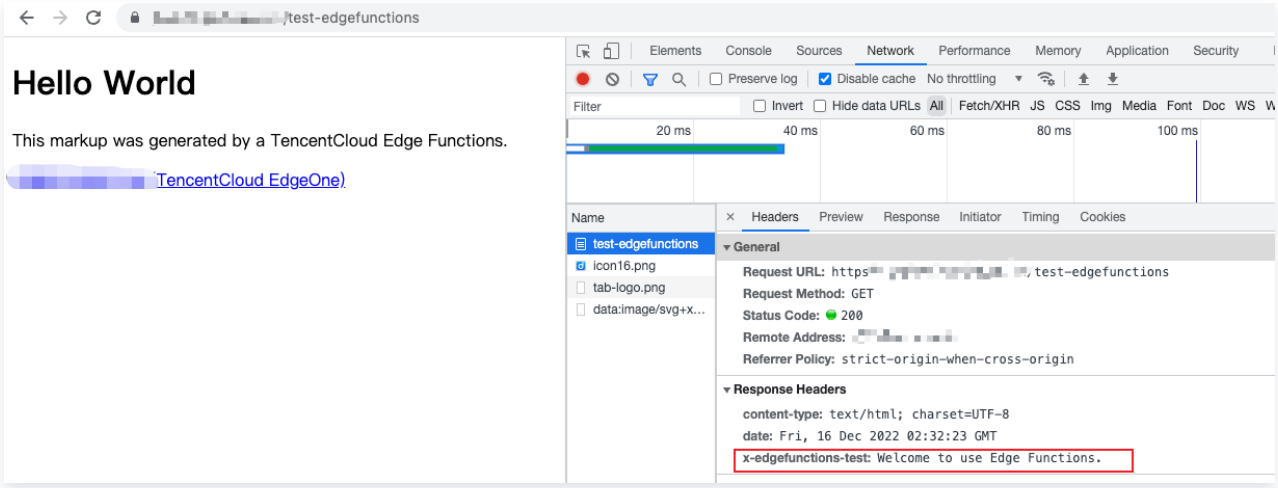
Prerequisites

1. You have created a site in Activated status and a subdomain for testing the edge function at this site.
2. You have created the edge function.
3. You have created the triggering rule.

Verification method

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

- **Browser Verification**: Enter a URL in the browser that matches the pre-set trigger condition to execute the function.

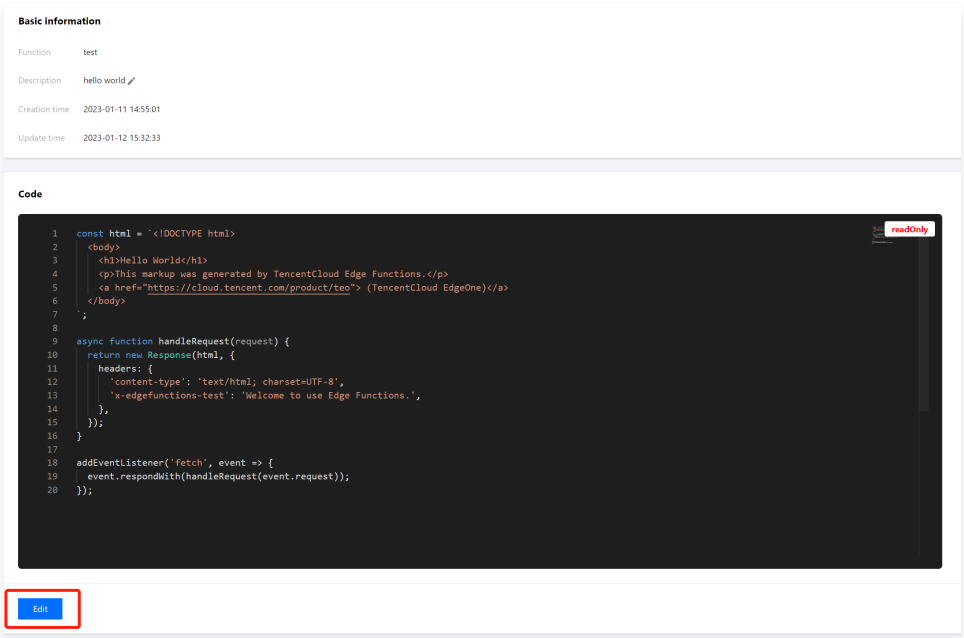


- **Curl Verification.**

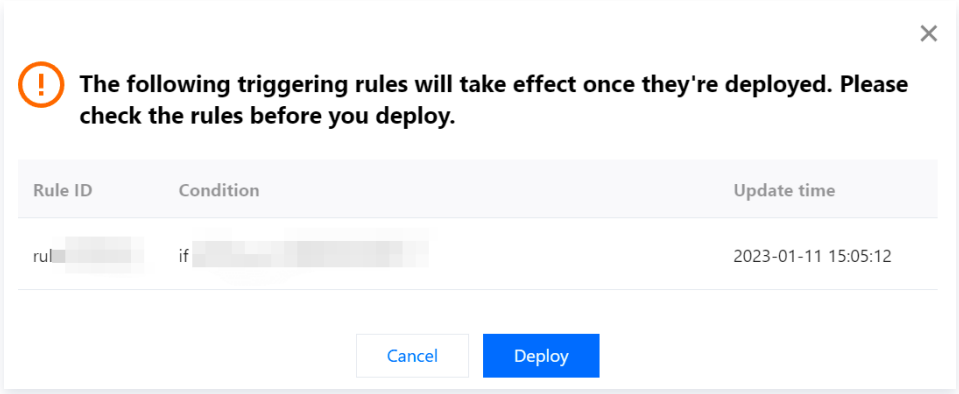
```
➜  ~ curl -i https://[redacted]ions
HTTP/2 200
x-edgefunctions-test: Welcome to use Edge Functions.
content-type: text/html; charset=UTF-8
content-length: 238

<!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>
➜  ~
```

(Optional) Step 4: Editing the Edge Function

1. If the deployed function does not meet your expectations, navigate to the Edge Function page, select the function you wish to modify, and click on **Details**.
2. On the function information page, click **Edit**, modify the code, and then click **Save and Deploy**.



3. Click **Deploy** as prompted to complete the function editing and redeployment process.



Operation Guide
Function management

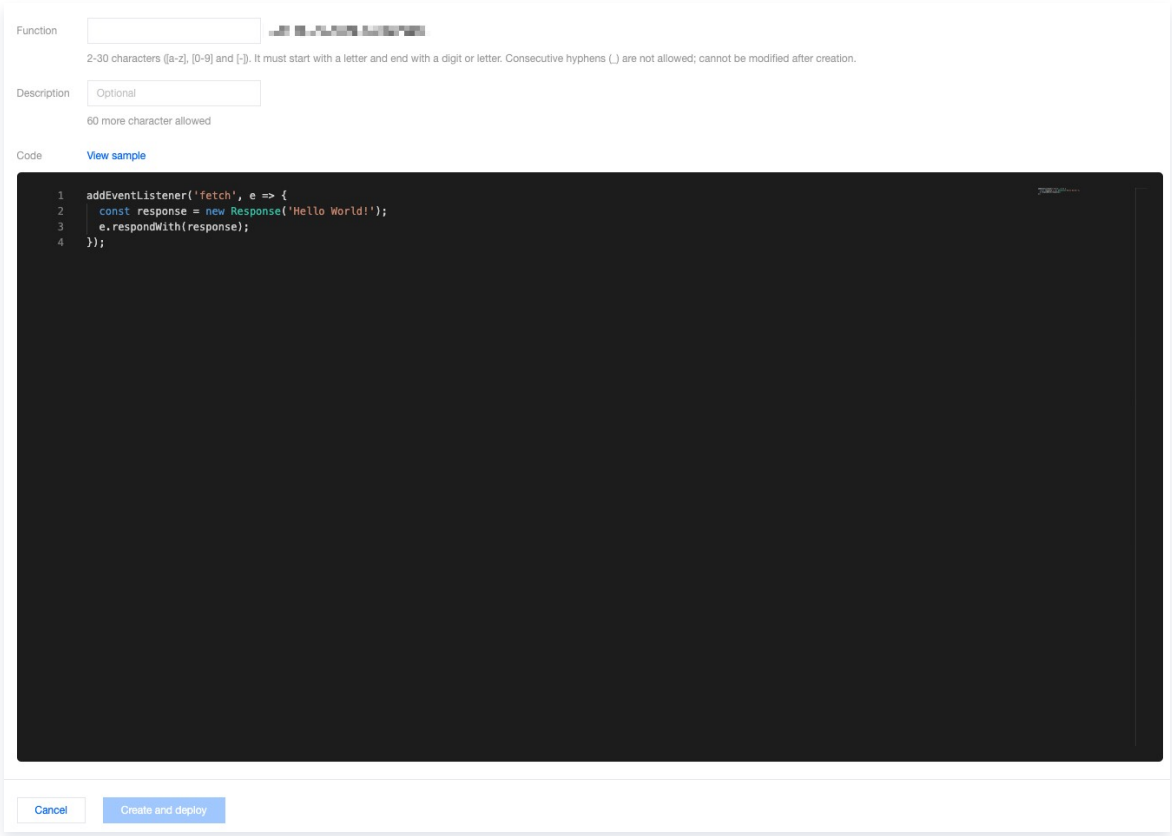
Last updated: 2023-09-07 15:15:06

Scenario

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

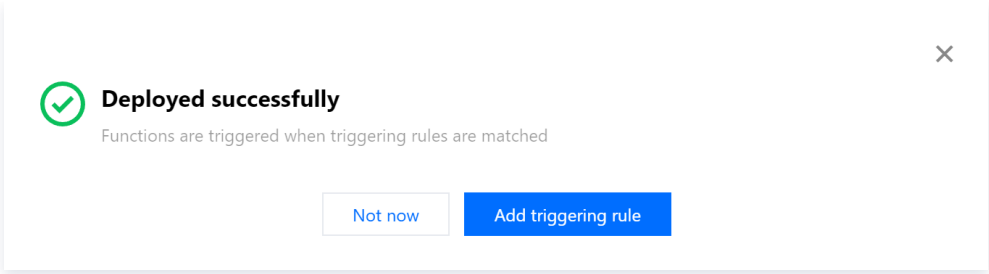
1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the Edge Functions page, click **Create New Function**.
4. On the new edge function page, configure the relevant parameters and click **Create and Deploy**.



Parameters:

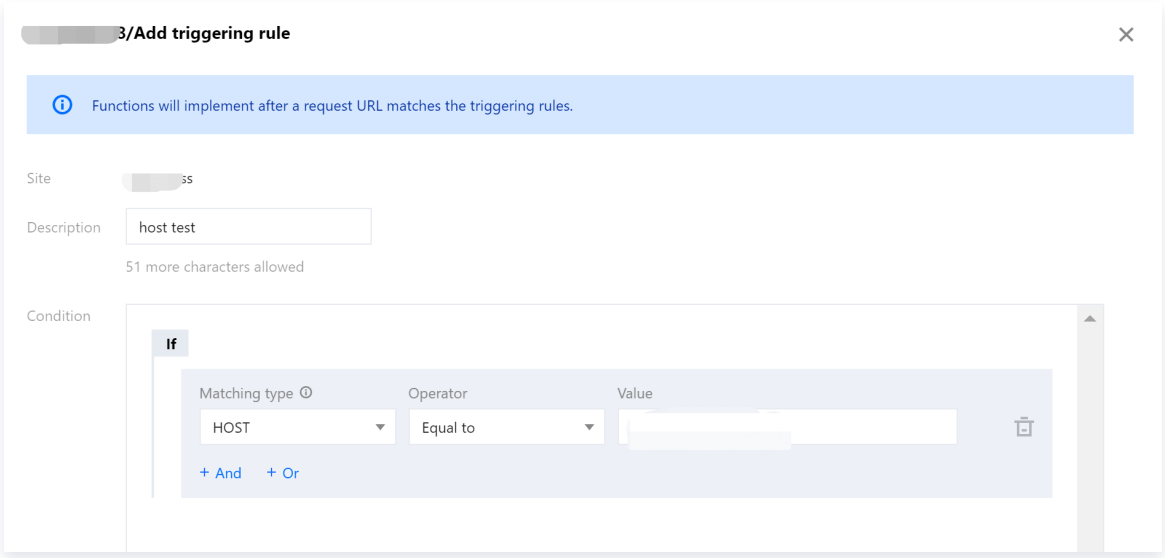
- Function: (Required) It can contain only 2-30 letters, digits, and hyphens and must start with a letter and end with a digit or letter.
- Description: (Optional) It can contain up to 60 characters.

5. If you see the following pop-up dialog box, the deployment is successful.

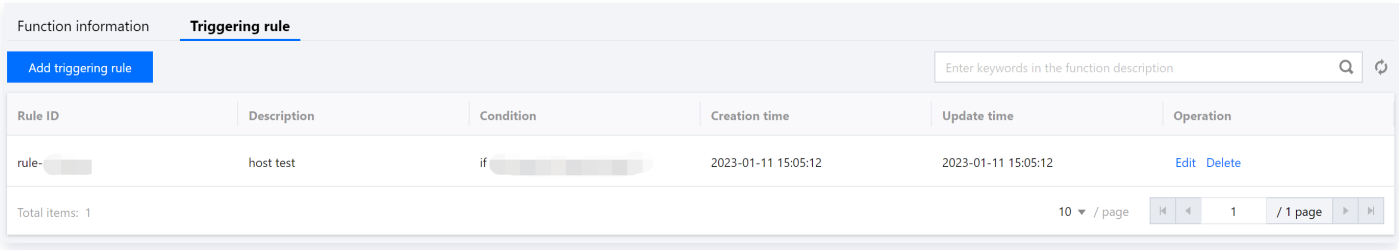


Configuring a Triggering Rule

1. After successfully creating and deploying the function, follow the prompts to click **Add Trigger Rule**.
2. On the **Add triggering rule** page, specify the matching type, operator, and value as needed.

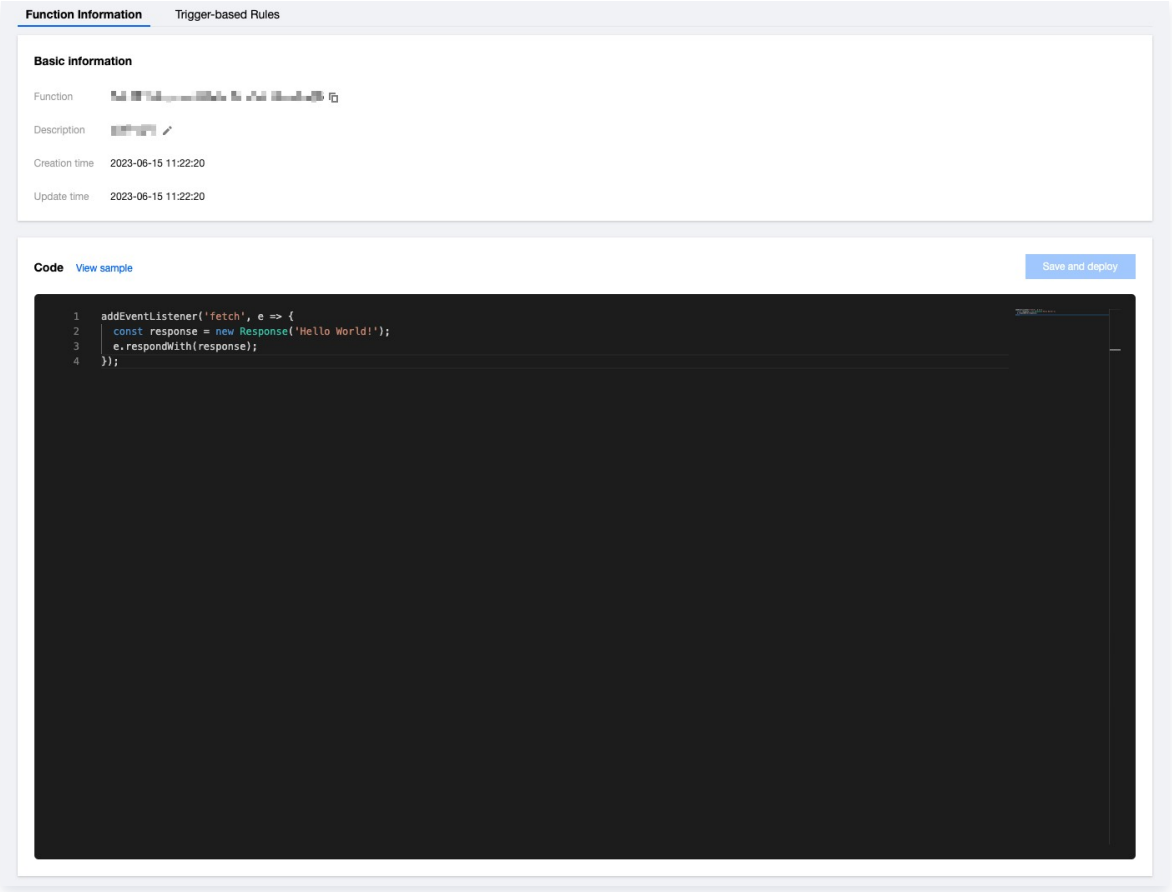


3. Click **OK** to finalize the creation of the trigger rule.

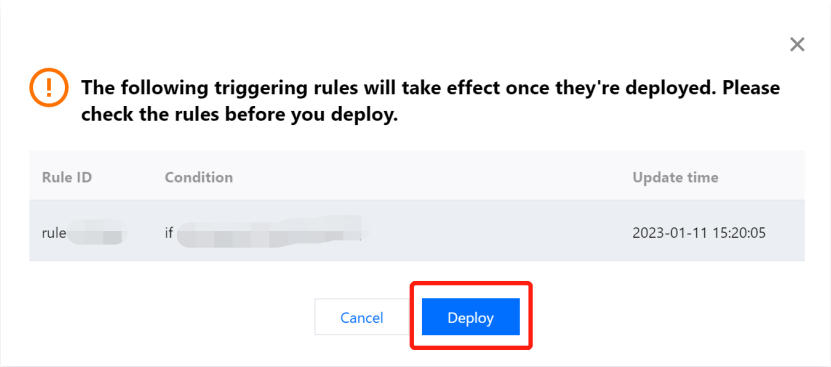


Editing the Function

1. If the deployed function does not meet your expectations, you can go to the Edge Functions page, select the function you wish to modify, click on its **function name**, and enter the function information page.

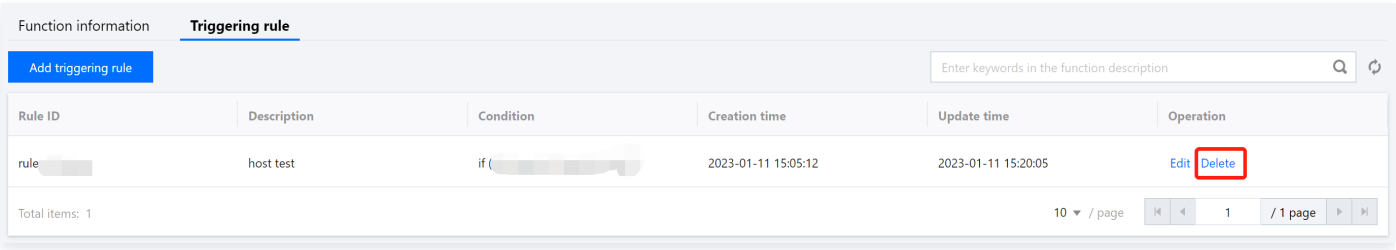


2. After modifying the function code, click **Save and Deploy**. If a trigger rule for this function already exists, a message will be displayed as follows.



Deleting the Function

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



2. In the pop-up dialog box, click **OK** to complete the deletion process.

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

Function Trigger

Last updated: 2023-09-07 17:30:29

Scenario

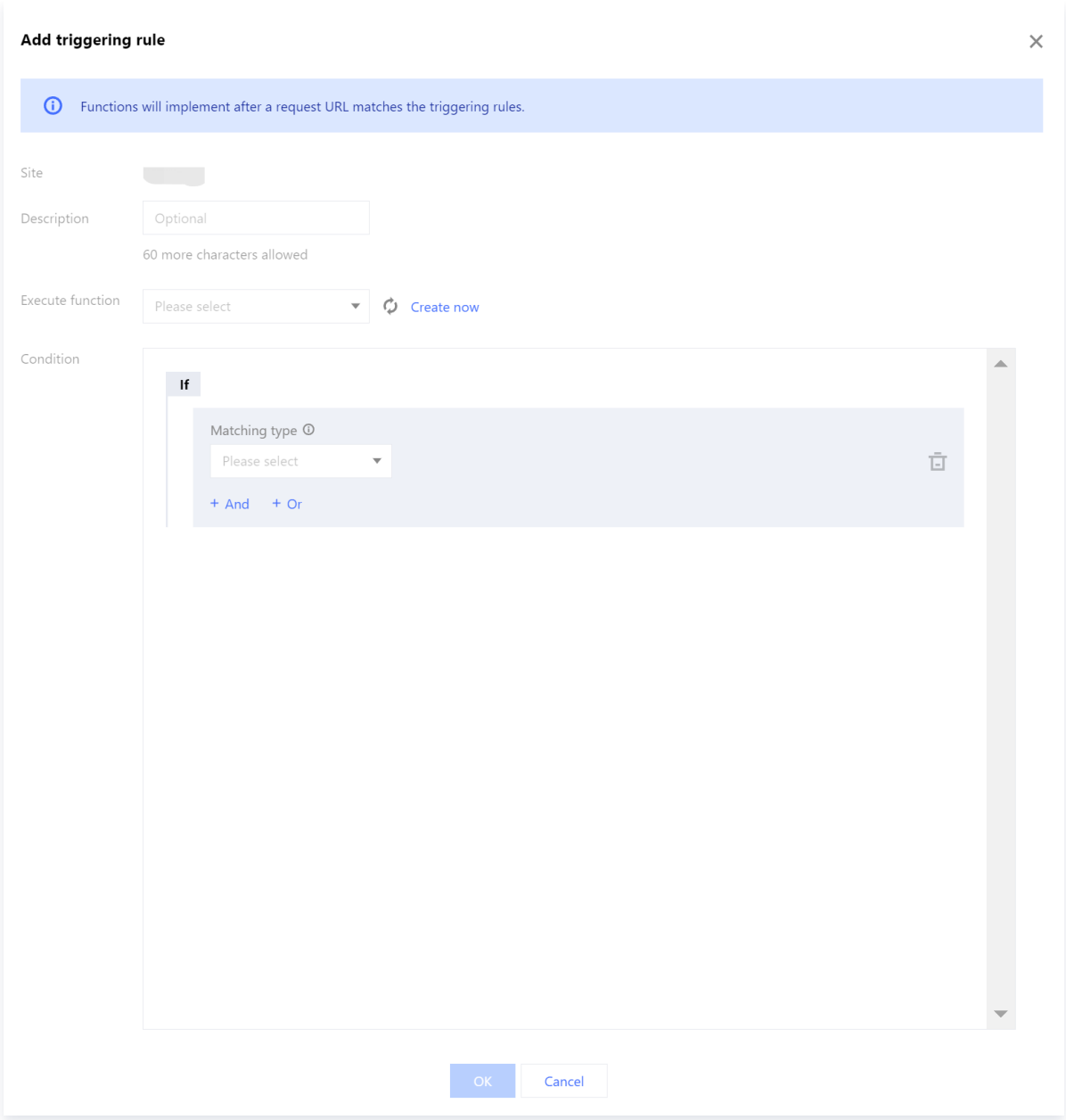
This document introduces how to work with function triggers.

- Add, delete, modify, and search for a trigger
- Adjust the priority of the triggering rule. If the request URL matches multiple triggering rules, you can reorder those triggering rules. Only the triggering rule on the top will be executed.

Directions

Creating a triggering rule

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Edge Functions** > **Function Management**.
3. On the trigger configuration page, click on the + on the right side of the rule list to configure the relevant parameters.



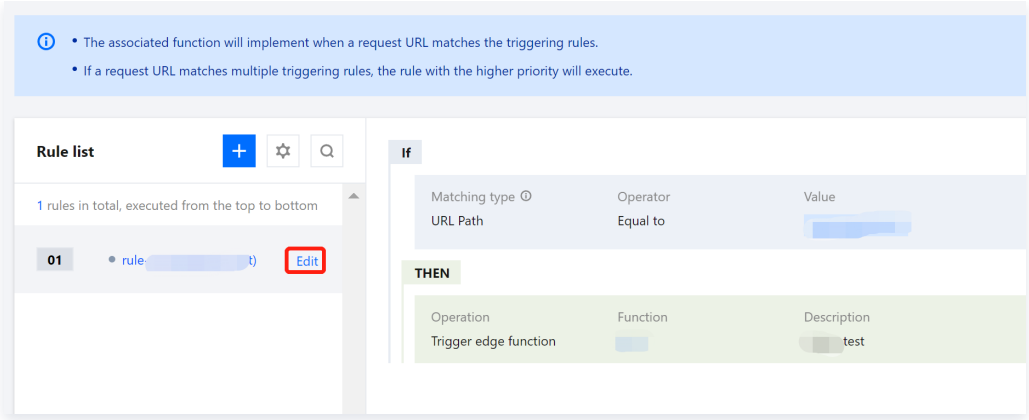
Parameters:

- Site: The name of the current site is displayed by default.
- Description: (Optional) It can contain up to 60 characters.
- Trigger Condition: Choose the match type, operator, and value as needed. For more details, see Rule Engine.
- Execute function: Select a created function from the drop-down list.

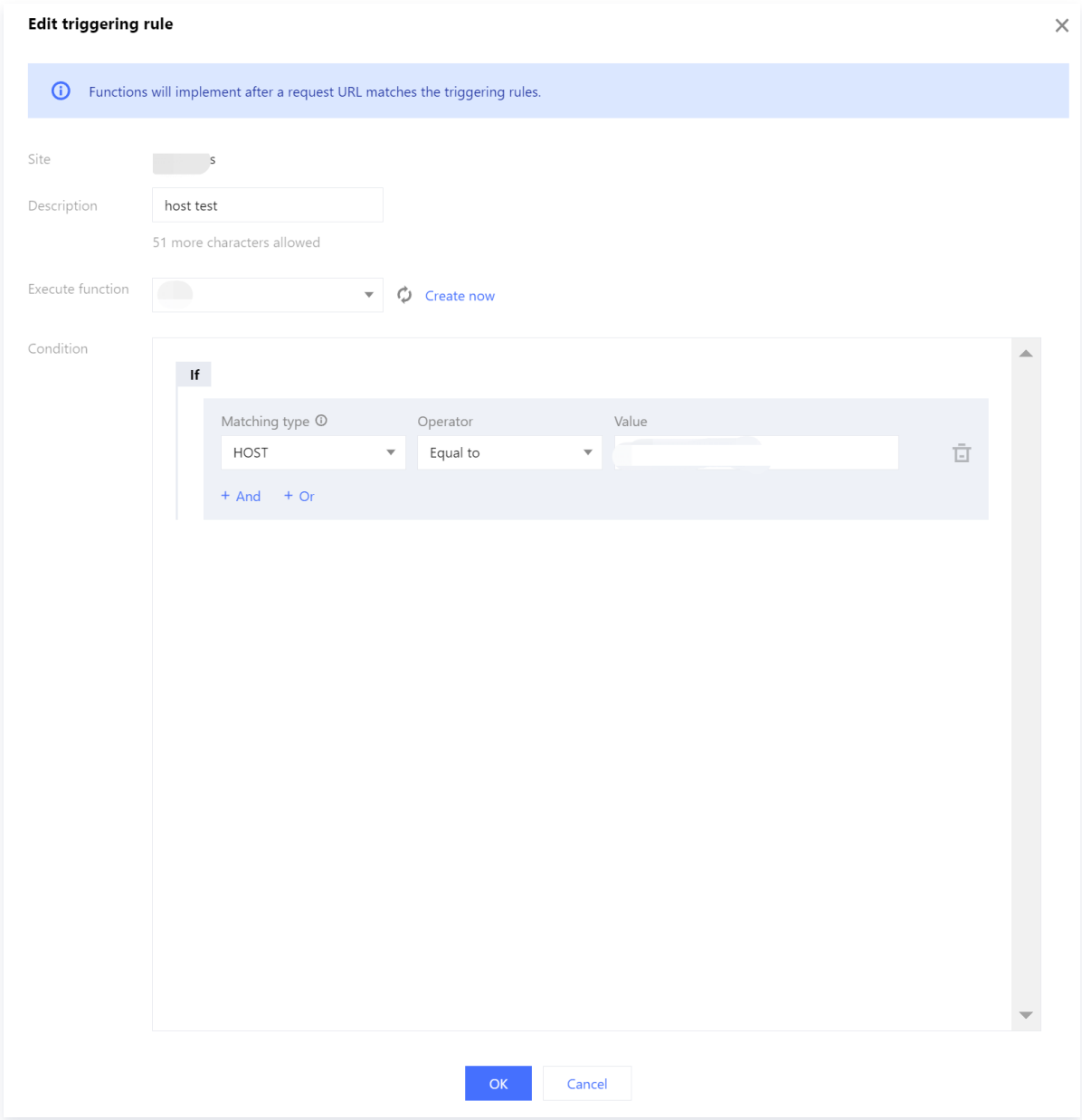
4. Click **OK** to finalize the creation of the trigger rule.

Editing the triggering rule

1. On the trigger configuration page, select the rule you wish to modify and click **Edit**.

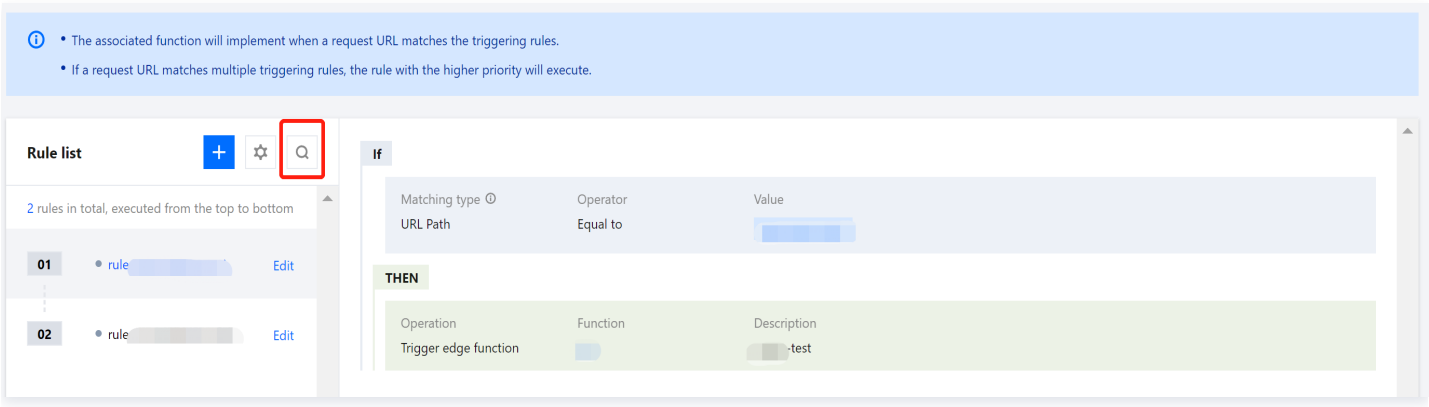


2. Within the Edit Trigger Rule dialog box, modify the relevant parameters and click **OK** to complete the editing of the trigger rule.



Searching for the triggering rule

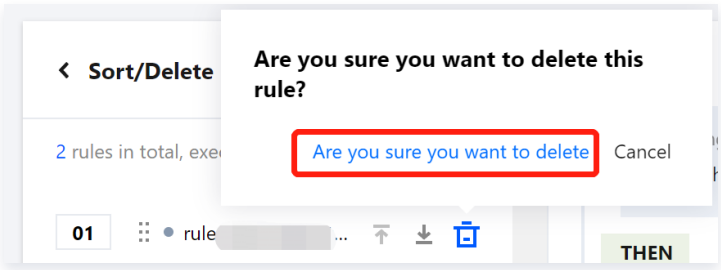
On the trigger configuration page, click on the 🔍 on the right side of the rule list. Fill in the keyword of the rule ID in the search input box to complete the query.



Deleting the triggering rule

1. On the trigger configuration page, click on the ⚙ icon on the right side of the rule list.
2. Select the rule you wish to remove and click on the 🗑 icon.

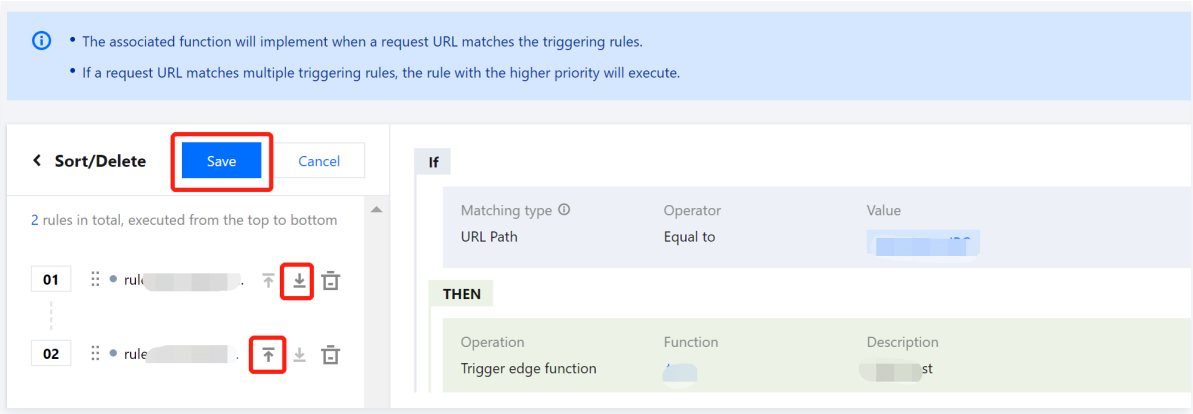
3. In the pop-up dialog box, click **Confirm to delete** to remove the trigger rule.



Adjusting the priority of the triggering rule

1. On the trigger configuration page, click on the ⚙ icon on the right side of the rule list.
2. Select the rule you wish to adjust. Click the ⤒ icon to move the rule up or the ⤓ icon to move the rule down. Click **Save** to complete the priority adjustment.

> **Note**
> 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:

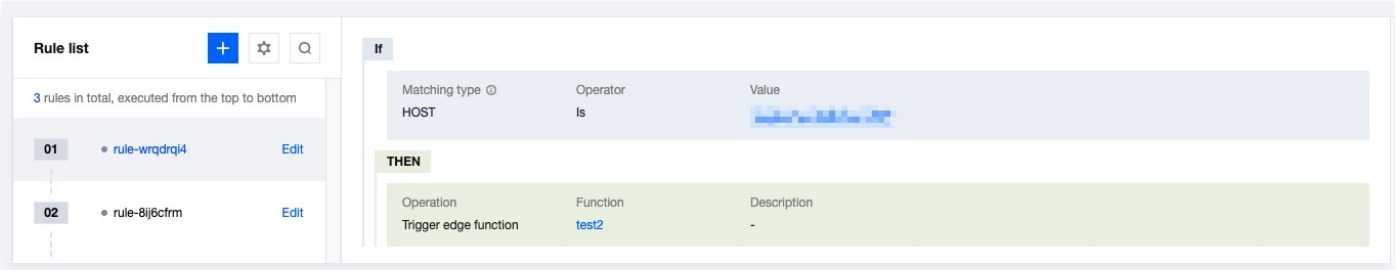
```
const html = `
  <!DOCTYPE html>
  <body>
 <h1>The test 1, Hello World</h1>
 <p>This markup was generated by a TencentCloud Edge Functions.</p>
 <a href="https://cloud.tencent.com/product/teo">Tencent Cloud EdgeOne</a>
  </body>
`;

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));
});
```

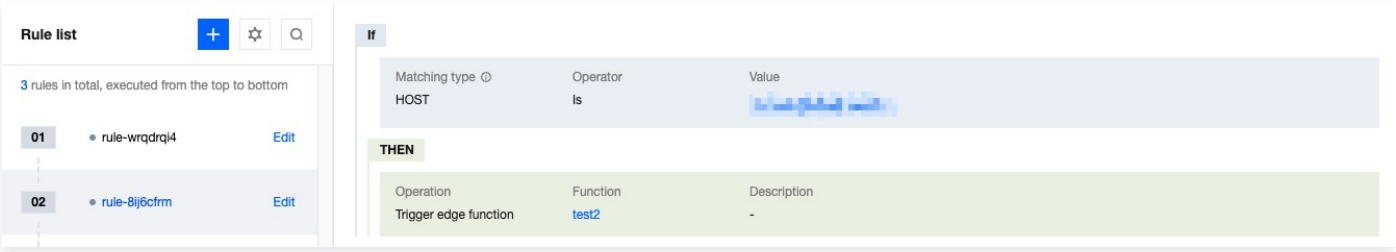
   Code of the `test2` function:

```
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: {
```

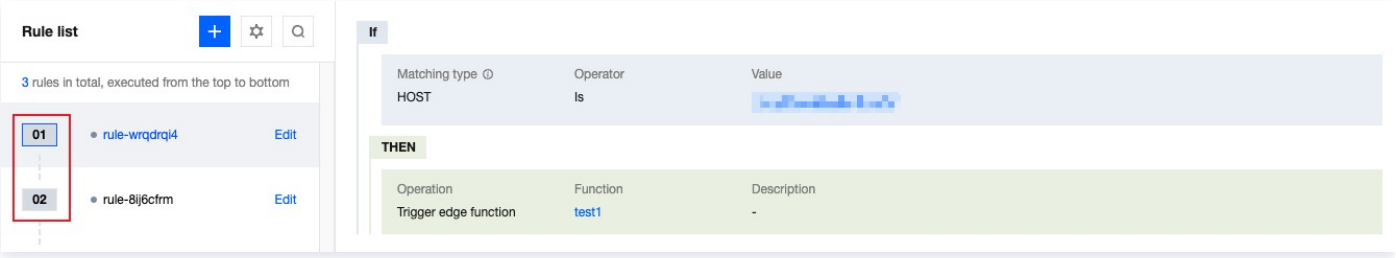
```
    'content-type': 'text/html; charset=UTF-8',
    'x-edgefunctions-test': 'Welcome to use Edge Functions.',
  },
  });
}

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.

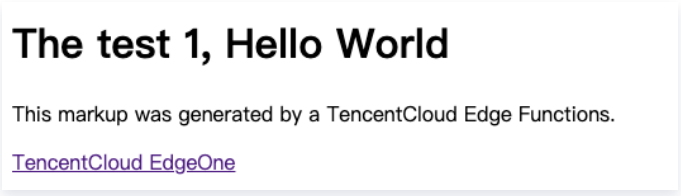




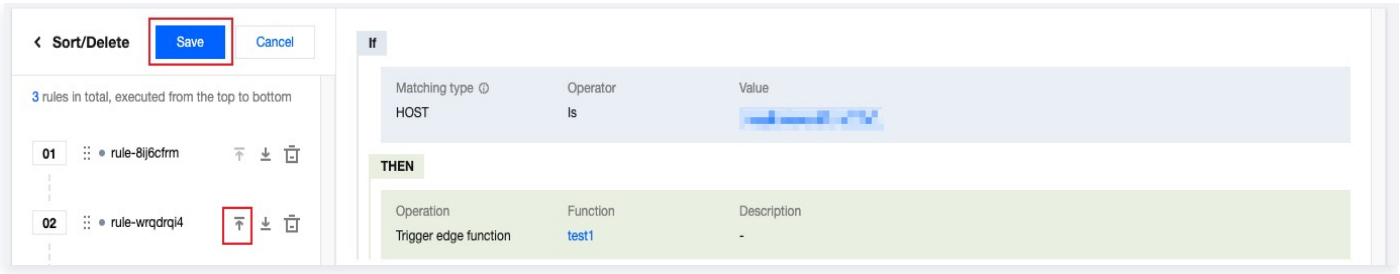
3. The triggering rule of the `test1` function is listed in position 01, and that of the `test2` function is in position 02, as shown in the following figure.



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

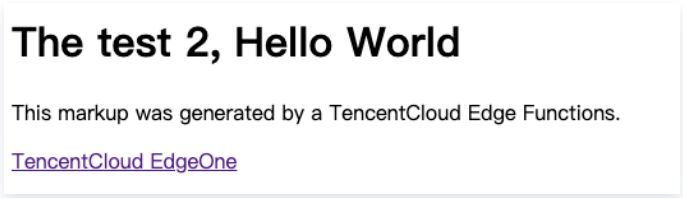


5. Click ⤒ for the `test2` function, then click **Save**.



The order of the triggering rules has been adjusted: `test2` is now in position 01 and `test1` is in position 02.

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



The above steps outline how to adjust the execution order of triggering rules when the request URL matches multiple rules.

Runtime APIs addEventListener

Last updated: 2023-09-07 15:15:18

The registration of an event listener serves as the entry point for Edge Functions. The `addEventListener` only supports the registration of a single event listener. Currently, only `fetch` request events are supported. By registering a `fetch` event listener, an HTTP request event FetchEvent is generated, thereby facilitating the processing of HTTP requests.

Description

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

Category

Parameter name	Local Disk Types	Required	Note
type	string	Supported	The event type to listen for. • Only `fetch` request events are supported at present. • If you specify a request event type other than `fetch`, the Edge Functions engine throws an `Error` exception.
listener	(event: FetchEvent) => void	Supported	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!'));
});
```

References

- MDN Documentation: addEventListener
- Sample Function: Returning an HTML Page
- Sample Functions: Returning a JSON Object

Cache

Last updated: 2023-09-07 15:15:25

Cache is designed based on the standard Web APIs Cache API. During the runtime of edge functions, a `caches` object is globally injected, providing a set of cache operation interfaces.

> **Note**
> The content of the cache is only valid in the current data node and will not be automatically replicated to other data nodes.

Constructor Function

- The default cache instance can be obtained using `caches.default`.

```
// 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 for a specific namespace.

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

Category

The parameters for the `caches.open(namespace)` method are explained below.

Parameter name	Local Disk Types	Required	Note
namespace	string	Supported	The namespace of the cache. • If the value is "default", it represents the default instance, and the default instance can also be directly obtained using `caches.default`.

Instance Methods

match

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

Retrieve the cache Response associated with the request. This returns a Promise object. If the cache exists, it contains the Response object, otherwise, it contains undefined.

> **Note**
> **cache.match** does not actively go back to the source internally. If the cache expires, a 504 error will be thrown.

Category

Parameter name	Local Disk Types	Required	Note
request	string | Request	Supported	The request object. The following method and headers are supported: • GET The request only supports the `GET` method. When the type is a string, it will be used as a URL to construct the Request object. • Range When the request contains a `Range` header, if the cached Response can support Range processing, a 206 response is returned. • If-Modified-Since When the request contains the If-Modified-Since header, if the cached Response has a Last-Modified header and the Last-Modified equals If-Modified-Since, a 304 response is returned. • If-None-Match When the request contains the If-None-Match header, if the cached Response has an ETag header and the ETag equals If-None-Match, a 304 response is returned.
options	MatchOptions	Not required	Options.

MatchOptions

Attribute	Local Disk Types	Sample value	Note
ignoreMethod	boolean	true	Determines whether to ignore the method of the Request. When set to true, the original method of the Request will be disregarded and treated as a GET request.

put

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

Attempt to add the response to the cache using the given request as the cache key. Regardless of whether the cache is successful, a `Promise<undefined>` object is returned.

> **Note**
> A 413 error is thrown when the Cache-Control header of the **response** object indicates no caching.

Category

Parameter name	Local Disk Types	Required	Note
request	string | Request	Supported	The cache key. • GET

			The request parameter supports only the GET method. If other methods are specified, an error is thrown. • string When the request parameter is of type string, it will be used as a URL to construct a Request object.
rsponse	Response	Supported	The cache content. • Cache-Control Supports s-maxage, max-age, no-store, no-cache, private; where no-store, no-cache, private all indicate no caching, `cache.put` will return a 413 error. • Pragma When Cache-Control is not set, and Pragma is set to no-cache, it indicates that caching is not permitted. • ETag When the `cache.match` parameter `request` includes the If-None-Match header, it can be used in conjunction with ETag. • Last-Modified When the `cache.match` parameter `request` includes the If-Modified-Since header, it can be used in conjunction with Last-Modified. • 416 Range Not Satisfiable When the `response` object is 416 Range Not Satisfiable, caching is temporarily suspended.

Parameter Limit

`cache.put` will throw a parameter error when using the following parameter values:

- The parameter `request` represents methods other than the GET method.
- The parameter `response` has a status code of 206 Partial Content.
- The parameter `response` includes the Vary: * header.

delete

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

Deletes the cache response associated with the request. Always returns a Promise containing true when no network error occurs, otherwise it contains false.

Category

Parameter name	Local Disk Types	Required	Note
request	string | Request	Supported	The cache key. • GET The parameter `request` only supports the GET method. • string When the `request` parameter is of type string, it is used as a URL to construct a Request object.
options	DeleteOptions	Not required	The options.

DeleteOptions

Attribute	Local Disk Types	Sample value	Note
ignoreMethod	boolean	true	Determines whether to ignore the method name of `request`. When set to true, the original method of the Request will be ignored and treated as a GET request.

References

- MDN Official Documentation: Cache
- Sample Function: Caching POST Requests
- Sample Functions: Using the Cache API

Cookies

Last updated: 2023-09-07 15:15:32

Cookies provides a set of cookie operation interfaces.

> **Note**
> The Cookies object manages the Cookie object collection using `name + domain + path` as a unique key.

Constructor Function

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

Category

Parameter name	Local Disk Types	Required	Note
cookieStr	string	Not required	A Cookie string or a Set-Cookie string.
isSetCookie	boolean	Not required	The parameter cookieStr determines whether it is a Set-Cookie string. The default value is false.

Method

get

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

Retrieves the Cookie object with the specified name. When multiple `name` matches exist, it returns an array of Cookie objects.

Category

Parameter name	Local Disk Types	Required	Note
name	string	Not required	`Cookie` name, the value description is as follows. • Default name Obtains all Cookie objects. • Specified name This represents the retrieval of a Cookie object with a specified name. When multiple matches exist, it returns an array of Cookie.

Cookie

The `Cookie` object has the following properties. For more details, refer to the MDN official documentation on Set-Cookie.

Attribute	Local Disk Types	Read-Only	Note

name	string	Supported	`Cookie` Name.
value	string	Supported	`Cookie` Value.
domain	string	Supported	The scope domain of `Cookie`.
path	string	Supported	The scope of the `Cookie`.
expires	string	Supported	`Cookie` maximum validity period, the value complies with the HTTP Date header standard.
max_age	string	Supported	`Cookie` expires after max_age seconds, unit in seconds (s).
samesite	string	Supported	Controls the protection against Cross-Site Request Forgery (CSRF) attacks for `Cookie`.
httponly	boolean	Supported	JavaScript access to `Cookie` is prohibited, and it is only carried by HTTP requests.
secure	boolean	Supported	`Cookie` is only carried with HTTPS request protocol.

set

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

Overwrite and add Cookie. Returns true, indicating successful addition, returns false, indicating failure to add (exceeded the number limit of cookies, see cookie size limit for details).

> **Note**
> Add a Cookie, using `name + domain + path` as a unique key to overwrite the existing one.

Category

Parameter name	Local Disk Types	Required	Note
name	string	Supported	`Cookie` Name.
value	string	Supported	`Cookie` Value.
Cookie	string	Not required	Cookie attribute configuration.

append

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

Appends a Cookie, applicable for scenarios with the same name but multiple values. Returns true if the addition is successful, and false if the addition fails (due to value duplication or exceeding the cookie quantity limit. For more details, please refer to cookie size limit).

> **Note**
> Append a Cookie using `name + domain + path` as a unique key.

remove

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

The remove() method deletes cookies.

> **Note**
> Deletes the Cookie using `name + domain + path` as a unique key.

Category

Parameter name	Local Disk Types	Required	Note
name	string	Supported	`Cookie` Name.
options	Cookie	Supported	Cookie attribute configuration items, where the attributes domain and path can support *, indicating a match for all.

Usage Limits

Automatic escape of special characters

- The name value containing characters `" ( ) , / : ; ? < = > ? @ [ ] \ { }`, `0x00~0x1F`, `0x7F~0xFF` will be automatically escaped.
- Values containing characters `, ; " \`, `0x00~0x1F`, `0x7F~0xFF` will be automatically escaped.

Cookies size limit

- The size of the Cookie attribute `name` cannot exceed 64 bytes.
- The cumulative size of the Cookie attributes `value, domain, path, expires, max_age, samesite` should not exceed 1KB.
- 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 cookies object
  const cookies = new Cookies('ssid=helloworld; expires=Sun, 10-Dec-2023 03:10:01 GMT; path=/; domain=.tencer

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

  return response;
}

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

References

- MDN Official Documentation: Set-Cookie
- Sample Function: Performing an A/B Test
- Example Function: Set Cookie

Encoding

Last updated: 2023-09-07 15:15:38

Designed based on the Web APIs standards, TextEncoder and TextDecoder have been implemented as the encoder and decoder respectively.

TextEncoder

The encoder accepts a stream of code points as input and outputs a `UTF-8` byte stream. Please refer to the MDN official documentation TextEncoder.

Constructor Function

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

Properties

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

The encoding type of the encoder, the current value is only `UTF-8`.

Method

encode

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

Accepts a stream of code points as input and outputs a `UTF-8` byte stream.

> **Note**
> **The maximum input length is 300M**; exceeding this limit will result in an exception.

- `encoder.encode` Parameters

Parameter name	Local Disk Types	Required	Note
input	string | undefined	Not required	The text to be encoded.

encodeInto

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

Accepts a stream of code points as input, outputs a `UTF-8` byte stream, and writes it into the `destination` byte array.

- Category

Parameter name	Local Disk Types	Required	Note
input	string	Supported	The text to be encoded.
destination	Uint8Array	Supported	The object in which the encoded text is stored.

- Return Value: `EncodeIntoResult`

Attribute	Local Disk Types	Note
read	number	The number of UTF-16 units that have been converted to UTF-8.
written	number	The number of bytes modified in the target Uint8Array.

TextDecoder

The decoder takes a byte stream as input and provides a stream of code points as output. Please refer to the MDN official documentation TextDecoder.

Construction method

```
const decoder = new TextDecoder(label?: string | undefined, options?: DecoderOptions | undefined): TextEncoder;
```

Category

> **Note**
> The parameter label does not currently support the following values:
> - iso-8859-16。
> - hz-gb-2312。
> - csiso2022kr，iso-2022-kr。

Parameter name	Local Disk Types	Required	Note
label	string | undefined	Not required	The type of decoder, with a default value of `UTF-8`. For optional `label` values, please refer to the MDN official documentation.
options	DecoderOptions | undefined	Not required	Decoder Configuration Items.

DecoderOptions

The following table describes the configuration items of the decoder.

Attribute	Local Disk Types	Default value	Note
fatal	boolean	false	Indicates whether an exception is thrown when decoding fails.
ignoreBOM	boolean	false	Indicates whether to ignore the byte-order marker.

Properties

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;
```

Indicates whether to ignore the byte-order marker.

Method

decode

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

> **Note**
> The maximum length of the `buffer` parameter is 100 MB. An exception will be thrown if this length is exceeded.

Parameter name	Local Disk Types	Required	Note
buffer	ArrayBuffer | ArrayBufferView | undefined	Not required	The byte stream to be decoded. • The maximum length of the buffer is 100M; exceeding this limit will throw an exception.
options	DecodeOptions	Not required	The configuration items for decoding.

DecodeOptions

The following table describes the configuration items for decoding.

Attribute	Local Disk Types	Default value	Note
stream	boolean	false	Specifies whether to perform decoding in streaming mode. Default value: false. Valid values: • true Indicates data is processed in `chunk` form, that is, stream decoding. • false Indicates whether the `chunk` has ended or the data has not been processed using chunk, i.e., non-stream decoding.

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);

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

  return response;
}

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

References

- MDN Official Documentation: TextEncoder
- MDN Documentation: TextDecoder
- MDN Official Documentation: Encoding Label
- Sample Function: Tamper-proof Verification

Fetch

Last updated: 2023-09-07 15:15:44

Designed based on the Fetch Standard, edge functions can use `fetch` to initiate asynchronous requests and access remote resources.

Description

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

Category

Parameter name	Local Disk Types	Required	Note
request	string | Request	Supported	The requested resource.
requestInit	RequestInit	Not required	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 accesses an accelerated domain of a site that has been connected to EdgeOne (for example, `www.example.com`), and this request triggers the execution of an edge function, the `fetch(www.example.com)` request implemented in this edge function will access the EdgeOne node cache. If the cache does not exist, it will fetch from the origin.

> **Note:** To use `fetch` to access EdgeOne node cache and fetch from the origin, the following conditions must be met.
> 1. The client accesses the accelerated domain of the EdgeOne access site, which in turn triggers the execution of the edge function.
> 2. The HOST in the `fetch(request)` specified `request.url` is identical to the HOST in the client request URL.
> 3. The `request.headers.host` specified by `fetch(request)` is identical to the HOST value in the client request header.

- `fetch(event.request)` retrieves cache and pulls from the origin in EdgeOne.

```
addEventListener('fetch', (event) => {
  // fetch(event.request) retrieves cache and pulls from the origin of EdgeOne CDN.
  const response = fetch(event.request);

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

- `fetch(url)` is used to retrieve EdgeOne cache and pull from the origin.

```
addEventListener('fetch', (event) => {
```

```
  const { request } = event;
  const urlInfo = new URL(request.url);
  const url = ${ urlInfo.origin }/h5/${ urlInfo.pathname };

  // fetch(url) to obtain EdgeOne CDN cache and fetch from the origin.
  const response = await fetch(url);

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

Processing image

`fetch` supports the input parameter `requestInit.eo.image` for image scaling or format conversion. For more details, refer to the parameter configuration items of ImageProperties for image processing.

> **Note:**
> To utilize `fetch(request, requestInit)` for image processing, it is imperative to meet the conditions for `fetch` to acquire EdgeOne node cache and retrieve from the origin simultaneously.

Redirection

`Fetch` supports `3xx` redirect status codes. The `requestInit.redirect` attribute can be used for configuration. For more redirect settings, please refer to RequestInit.

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

Status code	Redirection rules
301、302	Replaces the POST method with the GET method.
303	Replaces all methods except for HEAD and GET with the GET method.
307、308	Retains the original method.

> **Note**
> The redirect address is derived from the `Location` response header. If this header is absent, no redirection will occur.

- The value of the `Location` response header can be either an absolute URL or a relative URL. For further details, refer to RFC-3986: URI Reference.

Runtime Limits

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

- Frequency Limit: The total number of `fetch` requests that can be initiated in a single run of an edge function is 64. Any `fetch` requests exceeding this limit will fail and throw an exception.
- Concurrency Limit: During a single execution of an edge function, a maximum of 8 concurrent `fetch` requests are allowed. Any `fetch` requests exceeding this limit will be delayed until one of the running `fetch` requests is resolved.

> **Note**
> Each redirection is counted as a request and takes precedence over newly initiated `fetch` requests.

References

- MDN Documentation: Fetch
- Example Function: Returning Remote Resources
- Sample Function: Image Auto-Adaptation Format
- Sample function: Responsive Image Scaling

FetchEvent

Last updated: 2023-09-07 15:15:50

The **FetchEvent** represents any incoming HTTP request event. The Edge Functions handle HTTP requests by registering a `fetch` event listener.

Description

Within Edge Functions, the addEventListener is used to register a `fetch` event listener, generating a HTTP request event FetchEvent, thereby facilitating the handling of HTTP requests.

> **Note**
> Direct construction of `FetchEvent` objects is not supported. Use `addEventListener` to register a `fetch` event and obtain an `event` object.

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

Properties

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 HTTP request object initiated by the client. For more details, refer to Request.

Method

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 event listener `addEventListener`, the interface `event.respondWith()` must be invoked to respond to the client. If this interface is not invoked, the Edge Functions will forward the current request back to the origin.

Category

Parameter name	Local Disk Types	Required	Note
response	Response | Promise<Response>	Supported	The response to the client's HTTP request.

waitUntil

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

This is used to notify the Edge Functions to wait for the `Promise` to complete, thereby extending the lifecycle of the event handling.

Category

Parameter name	Local Disk Types	Required	Note
task	Promise<Response>	Supported	Awaiting the completion of Promise tasks.

passThroughOnException

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

This is used to prevent runtime response anomalies. If the function code throws an unhandled exception, Edge Functions will forward the request back to the origin, thereby enhancing the availability of the service.

Sample code

- If the `event.respondWith` interface is not invoked, the Edge Functions will forward the current request back to the origin server.

```
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 will forward the current request back to the ori
  if (request.url.indexOf('/ignore/') !== -1) {
    // The interface event.respondWith has not been invoked.
    return;
  }

  // In Edge Functions, customize content 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 the current request back to the
  event.passThroughOnException();
  throw new Error('Throw error');
});
```

References

- MDN Official Documentation: FetchEvent
- Sample Function: Returning an HTML Page
- Sample Functions: Using the Cache API

Headers

Last updated: 2023-09-07 15:15:57

Headers are designed based on the Web APIs standard Headers. They can be used for HTTP request and response header operations.

Constructor Function

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

Category

Parameter name	Local Disk Types	Required	Note
init	object | Array<[string, string]> | Headers	Not required	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 of the array is a `key/value` pair (such as: [key, value]). The constructor function traverses the array and initializes it into a new Headers object. • Headers The Constructor API copies all fields from an existing Headers object to the new Headers object.

Method

append

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

Append a new value on the header specified in the `headers` object. If the header does not exist, it will be added directly.

Category

Attribute	Local Disk Types	Required	Note
name	string	Supported	The name of the HTTP header that you want to delete from the Headers object.
value	string	Supported	The value of the HTTP header that you want to add.

delete

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

Removes the specified header from the `headers` object.

Category

Attribute	Local Disk Types	Required	Note
name	string	Supported	The name of the HTTP header that you want to delete from the Headers object.

entries

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

The getEntries() method retrieves all key-value pairs ([name, value]) from the `headers` object as an array. For return values, refer to the MDN official documentation: iterator.

forEach

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

Iterates through all headers of the `headers` object. If the `callback` returns a non-zero value, the iteration is terminated.

> **Note**
> `forEach` is a non-standard Web APIs method. To provide an efficient way to traverse headers, Edge Functions have extended this based on the Web APIs standard.

get

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

Retrieve the value of a specified header from the `headers` object.

has

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

Determine whether the `headers` object contains the specified header.

keys

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

The getKeys() method retrieves all keys contained in the `headers` object. For return values, refer to the MDN official documentation: iterator.

set

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

The set() method assigns a specified value to a `headers` object. If the header does not exist, a new `key/value` pair is added.

values

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

The getValues() method retrieves all values contained in the `headers` object. For return values, refer to the MDN official documentation: iterator.

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 Official Documentation: Headers
- Sample Function: Tamper-proof Verification
- Example Function: Request Header Authentication
- Sample Function: Modifying Response Headers

Request

Last updated: 2023-09-07 15:16:04

Request represents the HTTP request object, designed based on the Web APIs standard 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.
> - Obtain the current request's Request object by using the FetchEvent object event.request.

Constructor Function

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

Category

Parameter name	Local Disk Types	Required	Note
input	string | Request	Supported	A URL string or a Request object.
options	RequestInit	Not required	The initial configuration items of the Request object.

RequestInit

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

Attribute	Local Disk Types	Required	Default value	Note
method	string	Not required	GET	Request method. Examples: `GET` and `POST`.
headers	Headers	Not required	-	The headers that you want to add to the request.
body	string | Blob | ArrayBuffer | ArrayBufferView | ReadableStream	Not required	-	Request body.
redirect	string	Not required	follow	Redirect mode. Valid values: `manual`, `error`, and `follow`.
maxFollow	number	Not required	12	The maximum number of redirects allowed.
version	string	Not required	HTTP/1.1	HTTP version. Valid values: `HTTP/1.0`, `HTTP/1.1`, and `HTTP/2.0`.
copyHeaders	boolean	Not required	-	**Non-Web APIs Standard Option** indicates whether to

				copy the headers of the incoming Request object.
eo	RequestInitEoProperties	Not required	–	**Non-standard Web API options** are used to dictate the behavior of Edge Functions in processing the request.

RequestInitEoProperties

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

Parameter name	Local Disk Types	Required	Note
resolveOverride	string	Not required	Used to override the original domain name resolution under fetch request, supporting specified domain names or IP addresses. Further details are as follows: • 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	Not required	Image processing parameter configurations.

ImageProperties

Fetch supports image processing capabilities, with the parameter configurations explained below. These image processing configurations are identical to the image processing parameters used in site acceleration. You can refer to Image Processing for understanding the image processing capabilities within site acceleration.

Parameter name	Local Disk Types	Required	Note
format	string	Not required	Convert an image into a specified format. Valid values: `jpg` , `gif` , `png` , `bmp` , `webp` , `avif` , `jp2` , `jxr` , `heif` .
long	number	Not required	Specify the length of the long side, and automatically scale the short side (if not specified).
short	number	Not required	Specify the length of the short side, and automatically scale the long side (if not specified).
width	number	Not required	Specify the width, and automatically scale the height (if not specified).
height	number	Not required	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: `manual`.

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, as detailed below:

Attribute	Local Disk Types	Note	Sample value
geo	GeoProperties	This parameter describes the location information of the client request.	-

GeoProperties

This parameter describes the location information of the client request.

Attribute	Local Disk Types	Note	Sample value
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

Instance Methods

> **Note**
> 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 the parsing result of the body text as ArrayBuffer.

blob

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

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

clone

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

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

Category

Parameter name	Local Disk Types	Required	Note
copyHeaders	boolean	Not required	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.

getCookies

```
request.getCookies(): Cookies;
```

The getCookies() method obtains cookies from `request` headers. The cookies will be automatically parsed into Cookies objects.

setCookies

```
request.setCookies(cookies: Cookies): boolean;
```

The setCookies() method sets cookies for `request` headers.

Category

Parameter name	Local Disk Types	Required	Note
cookies	Cookies	Not required	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 Official Documentation: Request
- Sample Functions: Using the Cache API
- Example Function: Redirection Based on Request Region

Response

Last updated: 2023-09-07 15:16:11

The **Response** represents an HTTP response, designed based on the Web APIs standard Response.

> **Note**
> In Edge Functions, you can obtain a `Response` object by using any of the following methods:
> - Employ the `Response` constructor to create a Response object, which can be used for event.respondWith responses.
> - Utilize fetch to acquire the request response Response object.

Constructor Function

```
const response = new Response(body?: string | ArrayBuffer | Blob | ReadableStream | null | undefined, init?: Respon
```

Category

Parameter name	Local Disk Types	Required	Note
body	string | ArrayBuffer | Blob | ReadableStream | null | undefined	Supported	The body content of the `Response` object.
init	ResponseInit	Not required	The initialization configuration options for the `Response` object.

ResponseInit

Parameter name	Local Disk Types	Required	Note
status	number	Not required	The status code for the response.
statusText	string	Not required	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	Not required	The headers associated with the response.

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

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

The status code for the response.

statusText

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

The status message for the response.

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.

Instance 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 reads the response body to completion and returns a promise that resolves with the parsing result as an `ArrayBuffer`.

blob

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

The blob() method reads a Response stream to completion and returns a promise that resolves with the parsing result as a `Blob`.

clone

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

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

Category

Attribute	Local Disk Types	Required	Note
copyHeaders	boolean	Not required	Determines whether to duplicate the response headers. Default value: `false`. The valid values are as follows: • 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.

getCookies

```
response.getCookies(): Cookies;
```

The getCookies() method obtains cookies from `response` headers. The cookies will be automatically parsed into Cookies objects.

setCookies

```
response.setCookies(cookies: Cookies): boolean;
```

The setCookies() method sets cookies for `response` headers.

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 that facilitates a redirect to the specified URL.

Category

Attribute	Local Disk Types	Required	Note
url	string	Supported	Redirect URL
status	number	Not required	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 Official Documentation: Response
- Sample Function: Returning an HTML Page
- Sample Function: Modifying Response Headers
- Sample Function: Performing an A/B Test

Streams
ReadableStream

Last updated: 2023-09-07 15:16:18

ReadableStream, also known as the readable end, is designed based on the Web APIs standard ReadableStream.

> **Note**
> Direct construction of `ReadableStream` objects is not supported. They can be obtained through the use of TransformStream.

Description

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

Properties

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

The locked attribute indicates whether a stream is locked.

> **Note**
> The stream can be locked under the following circumstances:
> - A stream can have at most one active `reader`, and the stream remains locked until the `reader` invokes the `releaseLock()` method.
> - The stream is in being piped. The stream is locked until the piping ends.

Method

> **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 | ReadableStreamBYOBReader;
```

Create a `Reader`, and lock the current stream until the `Reader` calls releaseLock() to release the lock.

Category

Parameter name	Local Disk Types	Required	Note
options	ReaderOptions	Supported	The configuration items for generating the reader.

ReaderOptions

The properties of the `ReaderOptions` object are as follows.

Attribute	Local Disk Types	Required	Note
mode	string	Not required	`Reader` type, with a default value of `undefined`. The value explanation is as follows. • undefined Create a Reader of the ReadableStreamDefaultReader type. • byob Create a Reader of the ReadableStreamBYOBReader type.

pipeThrough

```
readable.pipeThrough(transfromStream: TransfromStream, options?: PipeToOptions): ReadableStream;
```

Pipeline processing of the stream. This transfers the current readable stream data to the writable end of the parameter `transfromStream`, and returns the readable end of `transfromStream`.

> **Note**
> During the pipeline transmission process, the current stream's `writable` end will be locked.

Category

Parameter name	Local Disk Types	Required	Note
transfromStream	TransfromStream	Supported	The destination to which the current stream is piped.
options	PipeToOptions	Supported	The configuration items for piping the stream.

PipeToOptions

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

Attribute	Local Disk Types	Required	Note
preventClose	boolean	Not required	When the value is `true`, it signifies that the closure of the readable stream will not result in the closure of the writable stream.
preventAbort	boolean	Not required	When the value is `true`, it indicates that an error in the readable stream will not cause the writable stream to terminate.
preventCancel	boolean	Not required	When the value is `true`, it indicates that an error in the writable stream will not result in the termination of the readable stream.
signal	AbortSignal	Not required	When `signal` is aborted, the ongoing transfer will be terminated.

pipeTo

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

Pipeline processing of the stream, transferring the current readable stream to the `destination` writable stream.

> **Note**
> During the pipeline transmission process, the current stream `destination` will be locked.

Category

Parameter name	Local Disk Types	Required	Note
destination	WritableStream	Supported	Writable stream.
options	PipeToOptions	Supported	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
- Example Function: Merging Resource Stream Responses
- Sample Function: m3u8 Rewrite and Authentication

ReadableStreamBYOBReader

Last updated: 2023-09-08 11:09:43

ReadableStreamBYOBReader is used for readable stream operations, designed based on the Web APIs standard ReadableStreamBYOBReader. `BYOB` (bring your own buffer) signifies the allowance of reading data from the stream into the buffer, thereby minimizing replicas to the greatest extent.

> **Note**
> Direct construction of `ReadableStreamBYOBReader` objects is not supported. They can be obtained using the ReadableStream.getReader method.

Description

```
// 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',
});
```

Properties

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

Returns a Promise object. If the stream is closed, the Promise status is `fulfilled`. If an error occurs in the stream or the read lock is released, the Promise status is `rejected`.

Method

read

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

Read data from the stream into the buffer `bufferView`.

> **Note**
> Invoking the `read` method to initiate the next stream read operation before the previous stream read operation has ended is not permitted.

Returned value

`reader.read` returns a Promise containing the read data and the reading status, as described below:

- If a chunk is available, the Promise is in a `fulfilled` state, containing an object in the format of `{ value: theChunk, done: false }`.
- If the stream is closed, the Promise transitions to a `fulfilled` state, containing an object in the format of `{ value: theChunk, done: true }`.
- If the stream encounters an error, the Promise is in a `rejected` state and contains relevant error information.

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 official documentation: ReadableStreamBYOBReader

ReadableStreamDefaultReader

Last updated: 2023-09-07 15:16:30

ReadableStreamDefaultReader is utilized for readable stream operations, designed based on the Web APIs standard ReadableStreamDefaultReader.

> **Note**
> Direct construction of `ReadableStreamDefaultReader` objects is not supported. They can be obtained using the ReadableStream.getReader method.

Description

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

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

Properties

closed

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

Returns a Promise object. If the stream is closed, the Promise status is `fulfilled`. If an error occurs in the stream or the read lock is released, the Promise status is `rejected`.

Method

read

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

The read() method reads data from the stream.

> **Note**
> Invoking the `read` method to initiate the next stream read operation before the previous stream read operation has ended is not permitted.

Returned value

`reader.read` returns a Promise containing the read data (Chunk) and the reading status, as detailed below:

- If a chunk is available, the Promise is in a `fulfilled` state, containing an object in the format of `{ value: theChunk, done: false }`.
- If the stream is closed, the Promise is in a `fulfilled` state, containing an object in the format of `{ value: undefined, done: true }`.
- If the stream encounters an error, the Promise is in a `rejected` state and contains relevant error information.

Chunk

The data `Chunk` read from the stream is described as follows:

```
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 official documentation: ReadableStreamDefaultReader

TransformStream

Last updated: 2023-09-07 15:16:35

TransformStream consists of a pair of streams, a readable stream referred to as the readable end, and a writable stream known as the writable end. It is designed based on the Web APIs standard TransformStream.

Constructor Function

```
const { readable, writable } = new TransformStream(transformer?: any, writableStrategy?: WritableStrategy);
```

Category

Parameter name	Local Disk Types	Required	Note
transformer	any	Not required	**Currently unsupported, passing values will not take effect and this parameter will be automatically ignored.**
writableStrategy	WritableStrategy	Not required	The strategy for the writable side.

WritableStrategy

Attribute	Local Disk Types	Required	Note
highWaterMark	number	Supported	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.

Properties

readable

```
readonly readable: ReadableStream;
```

The readable end. For more information, see ReadableStream.

writable

```
readonly writable: WritableStream;
```

The writable end. For more information, see WritableStream.

Sample code

```
async function handleEvent(event) {
  // Generate readable and writable ends
  const { readable, writable } = new TransformStream();
  // Retrieve remote resources
  const response = await fetch('https://www.tencentcloud.com/');
  // Stream response to the client
  response.body.pipeTo(writable);
```

```
  return new Response(readable, response);
}

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

References

- MDN Official Documentation: TransformStream
- Example Function: Merging Resource Stream Responses
- Sample Function: m3u8 Rewrite and Authentication

WritableStream

Last updated: 2023-09-07 15:16:42

WritableStream, also known as the writable end, is designed based on the Web APIs standard WritableStream.

> **Note**
> Direct construction of `WritableStream` objects is not supported. They can be obtained through the use of TransformStream.

Description

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

Properties

locked

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

The locked attribute indicates whether the stream is locked.

> **Note**
> The stream can be locked under the following circumstances:
> - A stream can have at most one active `writer`, and the stream remains locked until the `writer` invokes the `releaseLock()` method.
> - 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.

Method

> **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;
```

Create a writer and lock the current stream until the writer calls the releaseLock() method to release the lock. For the return value, refer to 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
- Example Function: Merging Resource Stream Responses
- Sample Function: m3u8 Rewrite and Authentication

WritableStreamDefaultWriter

Last updated: 2023-09-07 15:17:48

WritableStreamDefaultWriter is used for operations on writable streams. It is designed based on the Web APIs standard WritableStreamDefaultWriter.

> **Note**
> Direct construction of `WritableStreamDefaultWriter` objects is not supported. They can be obtained using the WritableStream.getWriter method.

Description

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

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

Properties

closed

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

Returns a Promise object. If the stream is closed, the Promise status is `fulfilled`. If an error occurs in the stream or the write lock is released, the Promise status 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.

Method

write

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

Write the `chunk` data into the stream.

> **Note**
> Invoking the `write` method to initiate the next write stream operation before the previous one ends is not permitted.

Category

Parameter name	Local Disk Types	Required	Note
chunk	Chunk	Supported	The chunk of data to be written to the stream.

Chunk

The data `Chunk` read from the stream is described as follows:

```
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;
```

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

References

- MDN Official Documentation: WritableStreamDefaultWriter

Web Crypto

Last updated: 2023-09-07 15:17:55

Web Crypto API is designed based on the Web APIs standard Web Crypto API. It offers a set of common encryption operation interfaces, and compared to encryption interfaces implemented purely in JavaScript, the `Web Crypto API` performs at a higher level.

> **Note**
> Direct construction of `Crypto` objects is not supported. During the edge function runtime, it is globally injected, and the global crypto instance can be used directly.

Description

```
// Perform encoding.
const encodeContent = new TextEncoder().encode('hello world');
// Utilize crypto to generate a SHA-256 hash value Promise<ArrayBuffer>.
const sha256Content = await crypto.subtle.digest(
  { name: 'SHA-256' },
  encodeContent
);
const result = new Uint8Array(sha256Content);
```

Properties

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

It provides common encryption operations, such as hashing, signing/verifying, encryption/decryption, etc. For more details, see SubtleCrypto.

Method

getRandomValues

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

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

Category

Attribute	Local Disk Types	Required	Note
buffer	Int8Array | Uint8Array | Uint8ClampedArray | Int16Array | Uint16Array | Int32Array | Uint32Array	Supported	Random number buffer, not exceeding 65536 bytes. For more details, refer to TypedArray.

	| BigInt64Array | BigUint64Array		

randomUUID

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

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

SubtleCrypto

It provides common encryption operations, such as: hashing, signing/verifying, encryption/decryption, etc. For more details, see MDN Official Documentation: SubtleCrypto.

> **Note**
> The **SubtleCrypto** encryption interface is divided into two categories based on functionality:
> - The encryption feature, including `encrypt/decrypt`, `sign/verify`, and `digest`, can be utilized to implement security functions such as privacy protection and identity verification.
> - Key management functions, including `generateKey`, `deriveKey`, and `importKey/exportKey`, can be utilized for key management.

digest

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

Returns a Promise object, which includes the generated data digest (hash). For more details, please refer to the MDN official documentation: SubtleCrypto.digest.

encrypt

```
crypto.subtle.encrypt(algorithm: object, key: CryptoKey, data: ArrayBuffer): Promise<ArrayBuffer>;
```

Returns a Promise object, which includes encrypted data. For more details, refer to MDN Official Documentation: SubtleCrypto.encrypt.

decrypt

```
crypto.subtle.decrypt(algorithm: object, key: CryptoKey, data: ArrayBuffer): Promise<ArrayBuffer>;
```

Returns a Promise object containing decrypted data. For more details, refer to MDN Official Documentation: SubtleCrypto.decrypt.

sign

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

Returns a Promise object, which includes the data signature. For more details, refer to MDN Official Documentation: SubtleCrypto.sign.

verify

```
crypto.subtle.verify(algorithm: string | object, key: CryptoKey, signature: BufferSource, data: ArrayBuffer): Promise<
```

Returns a Promise object, which includes the signature verification results. For more details, refer to MDN Official Documentation: SubtleCrypto.verify.

generateKey

```
crypto.subtle.generateKey(algorithm: object, extractable: boolean, keyUsages: Array<string>): Promise<CryptoKey
```

Returns a Promise object, containing the CryptoKey or CryptoKeyPair. For more details, refer to MDN Official Documentation: SubtleCrypto.generateKey.

deriveKey

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

Returns a Promise object, which includes the CryptoKey. For more details, refer to MDN Official Documentation: SubtleCrypto.deriveKey.

importKey

```
crypto.subtle.importKey(format: string, keyData: BufferSource, algorithm: string | object, extractable: boolean, keyU
```

Returns a Promise object, which includes the CryptoKey. For more details, refer to MDN Official Documentation: SubtleCrypto.importKey.

exportKey

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

Returns a Promise object, which includes the exported key ArrayBuffer. For more details, refer to MDN Official Documentation: SubtleCrypto.exportKey.

deriveBits

```
crypto.subtle.deriveBits(algorithm: object, baseKey: CryptoKey, length: integer): Promise<ArrayBuffer>;
```

Returns a Promise object, which includes a pseudo-random byte ArrayBuffer. For more details, refer to MDN Official Documentation: SubtleCrypto.deriveBits.

wrapKey

```
crypto.subtle.wrapKey(format: string, key: CryptoKey, wrappingKey: CryptoKey, wrapAlgo: string | object): Promise<
```

Returns a Promise object, which includes the encapsulated key ArrayBuffer. For more details, refer to MDN Official Documentation: SubtleCrypto.wrapKey.

unwrapKey

```
crypto.subtle.unwrapKey(format: string, wrappedKey: ArrayBuffer, unwrappingKey: CryptoKey, unwrapAlgo: string |
```

Returns a Promise object, which includes the unsealed CryptoKey. For more details, refer to MDN Official Documentation: SubtleCrypto.unwrapKey.

CryptoKey

`CryptoKey` represents a key generated by an encryption algorithm, for more details, refer to MDN Official Documentation CryptoKey. Direct construction of CryptoKey objects is not supported, use the following interface to generate keys:

- crypto.subtle.generateKey
- crypto.subtle.importKey
- crypto.subtle.deriveKey
- crypto.subtle.unwrapKey

The attributes of `CryptoKey` are described as follows.

Attribute	Local Disk Types	Read-Only	Note
type	string	Supported	The type of the key.
extractable	boolean	Supported	Specifies whether the key can be exported.
algorithm	object	Supported	The algorithm for which this key can be used and the associated parameters.
usages	Array<string>	Supported	The usage of the key.

CryptoKeyPair

`CryptoKeyPair` represents a pair of keys generated using an encryption algorithm. For more details, refer to MDN Official Documentation: CryptoKeyPair. Direct construction of CryptoKeyPair objects is not supported. Use the following interface to generate a key pair:

- crypto.subtle.generateKey

The attributes of `CryptoKeyPair` are described as follows.

Attribute	Local Disk Types	Read-Only	Note
privateKey	CryptoKey	Supported	For encryption and decryption algorithms, the private key is used for decryption. For signature algorithms, the private key is used for signing.
publicKey	CryptoKey	Supported	For encryption and decryption algorithms, the public key is used for encryption. For signature algorithms, the public key is utilized for signature verification.

Supported Algorithms

Edge Functions support all algorithms defined by the Web APIs standard WebCrypto, as detailed in the following table.

lgorithm	encrypt() decrypt()	sign() verify(	wrapKey()	deriveKey()	generateKey()	importKey()	exportKey()	digest()

		)	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))).join('');
}

async function handleEvent(event) {
  const encodeArr = TextEncoder().encode('hello world');
  // Execute MD5.
  const md5Buffer = await crypto.subtle.digest({ name: 'MD5' }, encodeArr);
  // Output 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 Official Documentation: Web Crypto API
- MDN Official Documentation: SubtleCrypto
- MDN Official Documentation: CryptoKey
- MDN Official Documentation: CryptoKeyPair
- Sample Function: Tamper-proof Verification
- Sample Function: m3u8 Rewrite and Authentication

Web standards

Last updated: 2023-09-07 15:18:01

The Edge Function, designed and implemented based on the **V8 JavaScript Engine**, provides a Serverless code execution environment, offering the following standardized Web APIs.

JavaScript Standard Built-in Objects

Edge Functions support all JavaScript Standard Built-in Objects. For more details, refer to the MDN Official Documentation: JavaScript Standard Built-in Objects.

URL

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

The URL API is used for parsing, constructing, normalizing, and encoding URLs. For more details, refer to the MDN Official Documentation: URL.

Blob

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

The Blob API represents immutable, raw data as pseudo-file objects. For more details, refer to the MDN Official Documentation: Blob.

Base64

btoa

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

The execution of Base64 encoding does not support Unicode strings. For more details, refer to the MDN Official Documentation: btoa.

atob

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

The execution of Base64 decoding does not support Unicode strings. For more details, refer to the MDN Official Documentation: atob.

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.

EventTarget and Event

EventTarget

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

Event publishing and subscription. For more details, refer to the MDN Official Documentation: EventTarget.

Event

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

Basic Events. For more details, refer to the MDN Official Documentation: Event.

AbortSignal and AbortController

AbortSignal

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

Abort Signal. For more information, see MDN Official Documentation: AbortSignal.

AbortController

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

Abort Controller. For more information, refer to the MDN Official Documentation: AbortController.

Sample Functions Returning an HTML Page

Last updated: 2023-09-07 15:18:14

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.

References

- Runtime APIs: addEventListener
- Runtime APIs: Response
- Runtime APIs: FetchEvent

Returning a JSON Object

Last updated: 2023-09-07 15:18:20

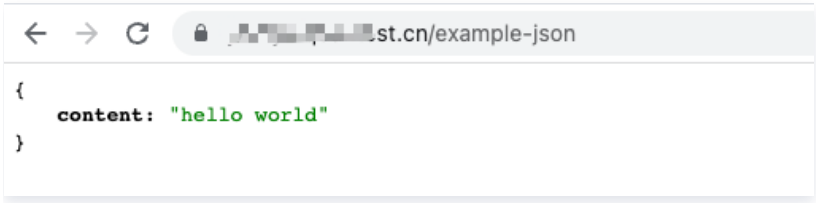
In this example, an edge function is used to generate a JSON object, and the JSON object is accessed and previewed from a browser.

Sample code

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

async function handleRequest(request) {
  // Convert JSON to string
  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.



References

- Runtime APIs: addEventListener
- Runtime APIs: Response

Fetch Remote Resources

Last updated：2023-09-07 15:18:26

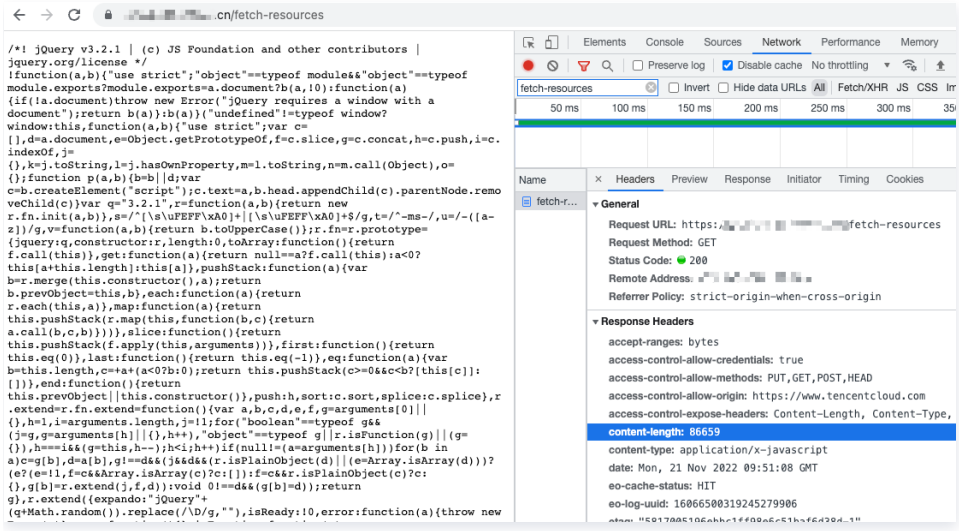
This example utilizes the Fetch API to retrieve the remote resource jQuery.js and respond to the client.

Sample code

```
async function handleRequest(request) {
  // Retrieve remote resources
  const response = await fetch('https://static.cloudcachetci.com/qcloud/main/scripts/release/common/vendors/jquer
  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.



References

- Runtime APIs: Fetch

Authenticating a Request Header

Last updated: 2023-09-07 15:18:31

This example validates the value of the request header `x-custom-token`. If the value equals 'token-123456', access is granted; otherwise, access is denied. This simple access control is implemented using edge functions.

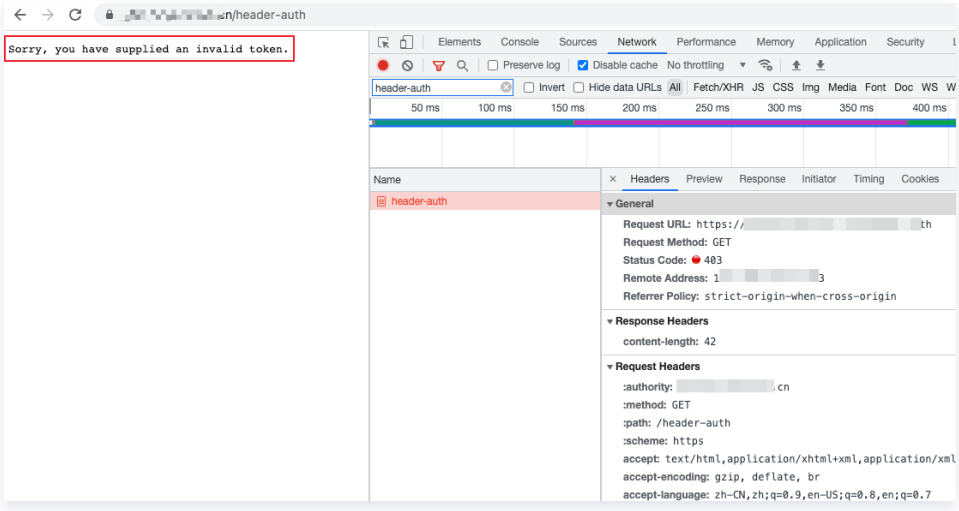
Sample code

```
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));
});
```

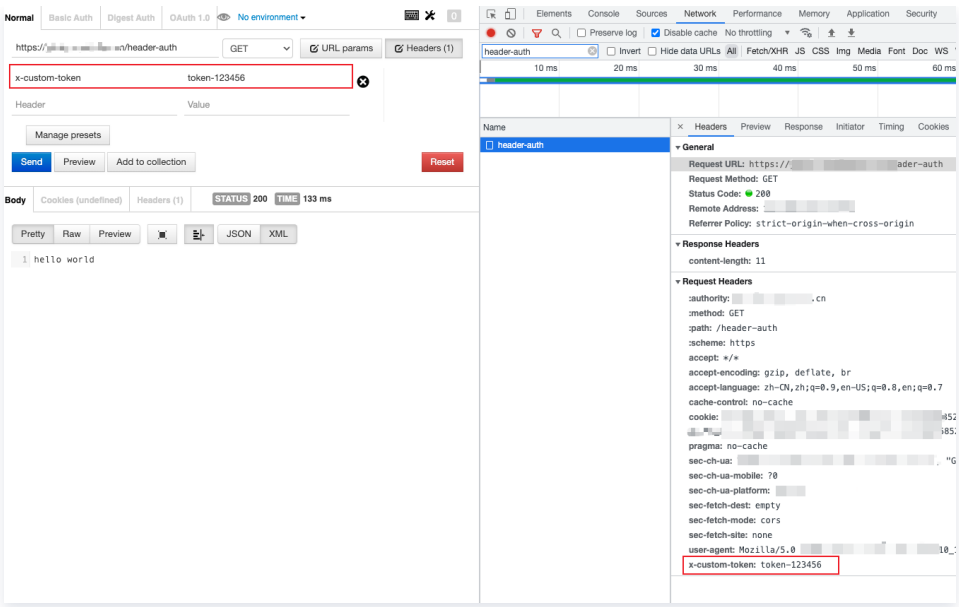
Sample Preview

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.



- If authentication is successful, access is allowed.



References

- Runtime APIs: Headers
- Runtime APIs: Response

Modifying a Response Header

Last updated: 2023-09-07 15:18:37

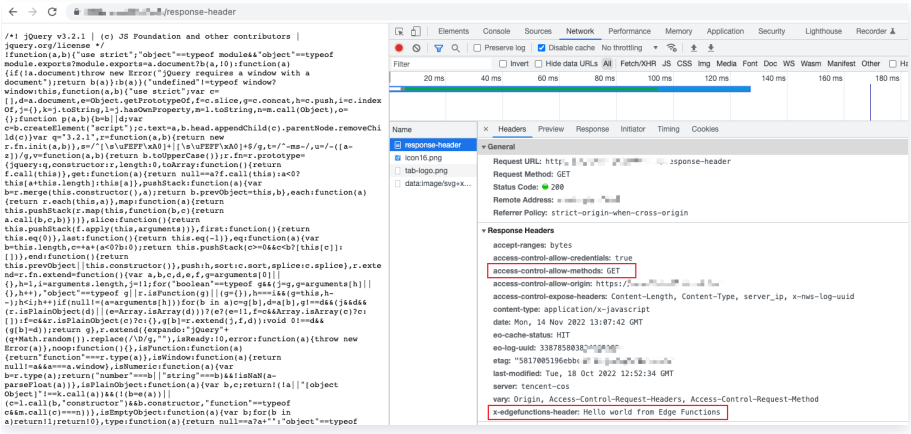
This example utilizes the Fetch API to retrieve the remote resource jQuery.js, modifies the response header, and returns it to the client.

Sample code

```
async function handleRequest() {
  const response = await fetch('https://static.cloudcachetci.com/qcloud/main/scripts/release/common/vendors/jquer

  // Add custom response header
  response.headers.append('x-edgefunctions-header', 'Hello world from edgefunction');

  // Remove response header
  response.headers.delete('x-cos-request-id');
  response.headers.delete('x-cos-hash-crc64ecma');

  // Modify request header
  response.headers.set('access-control-allow-methods', 'GET');

  return response;
}

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

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: 2023-09-07 15:18:43

This example employs cookies to preserve session information, thereby controlling A/B testing of requests. The scenario of A/B testing is implemented using edge functions.

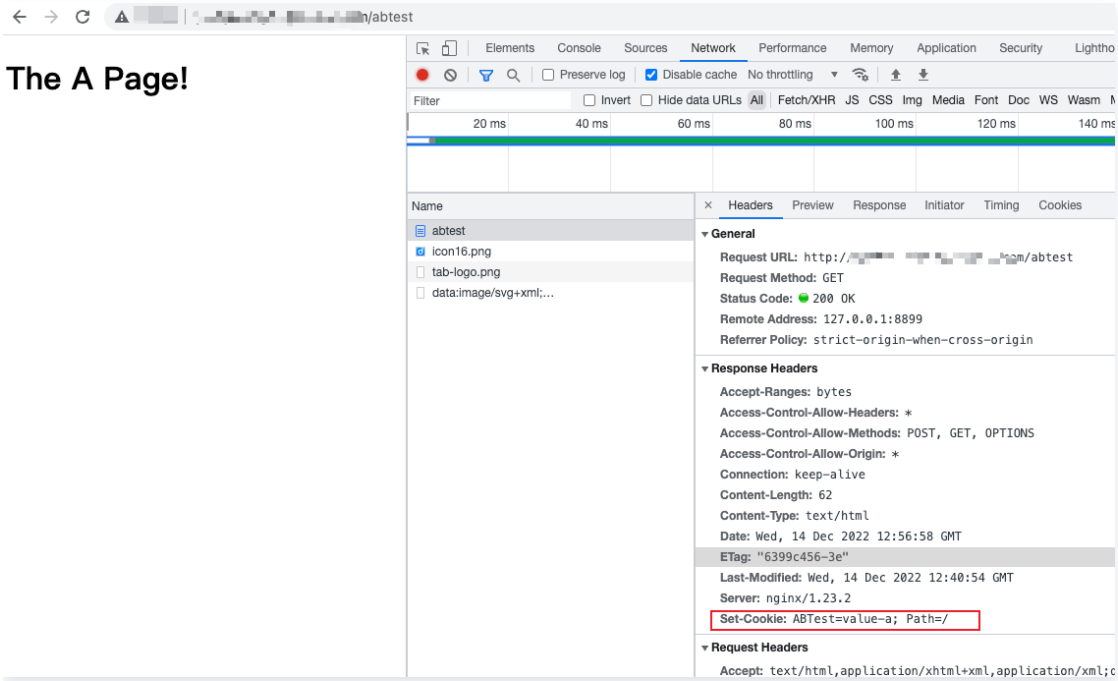
Sample code

```
// The name of the cookie.
const cookieName = 'ABTest';
// The values of the cookie.
const valueA = 'value-a';
const valueB = 'value-b';
// The paths of the webpage.
const pathA = '/path-a';
const pathB = '/path-b';

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

  // If the request has already entered the A/B test, then the corresponding content is directly returned.
  if (urlInfo.pathname.startsWith(pathA) || urlInfo.pathname.startsWith(pathB)) {
    return fetch(request);
  }

  // Obtaining the cookie of the current request.
  const cookies = request.getCookies();
  const cookie = cookies.get(cookieName);
  const cookieValue = cookie && cookie.value;

  // If the cookie value is for A testing, return the corresponding content.
  if (cookieValue === valueA) {
    urlInfo.pathname = pathA + urlInfo.pathname;
    return fetch(urlInfo.toString());
  }

  // If the cookie value is for B testing, return the corresponding content.
  if (cookieValue === valueB) {
    urlInfo.pathname = pathB + urlInfo.pathname;
    return fetch(urlInfo.toString());
  }

  // Randomly assign the current request to either A or B test.
  const testValue = Math.random() < 0.5 ? valueA : valueB;
  const path = testValue === valueA ? pathA : pathB;
  urlInfo.pathname = path + urlInfo.pathname;

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

  cookies.set(cookieName, testValue, { path: '/' });

  response.setCookies(cookies);
  return 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.



References

- Runtime APIs: Cookies
- Runtime APIs: Response

Setting Cookies

Last updated: 2023-09-07 15:18:50

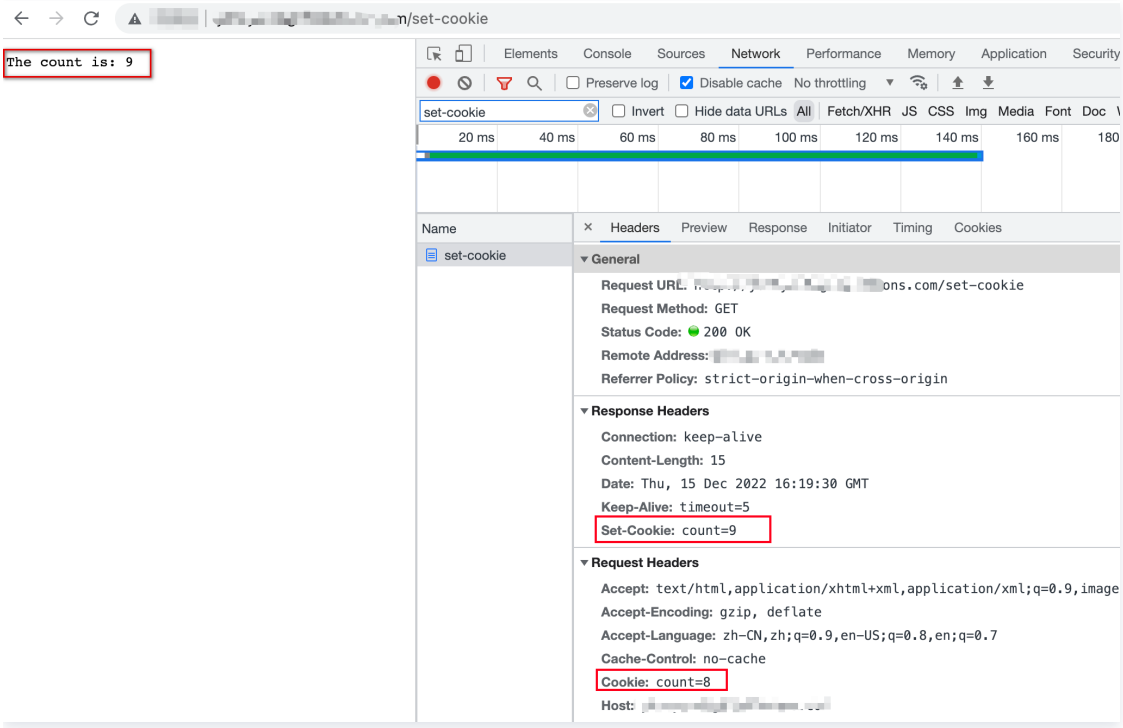
In this example, cookies are used to count the number of access requests. When a browser accesses the Edge Functions service, the number of access requests is increased by 1.

Sample code

```
async function handleRequest(request) {
  // Retrieve the cookies from the current request
  const cookies = request.getCookies();
  const cookieCount = cookies.get('count');
  // Increment the count
  const count = Number(cookieCount && cookieCount.value || 0) + 1;
  // Update the count in the cookie
  cookies.set('count', String(count));

  const response = new Response(The count  is :  $ { count } );
  // Set the response cookies
  response.setCookies(cookies);
  return 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.



References

- Runtime APIs: Cookies
- Runtime APIs: Response
- Runtime APIs: Request

Performing Redirect Based on the Request Location

Last updated: 2023-09-07 15:18:56

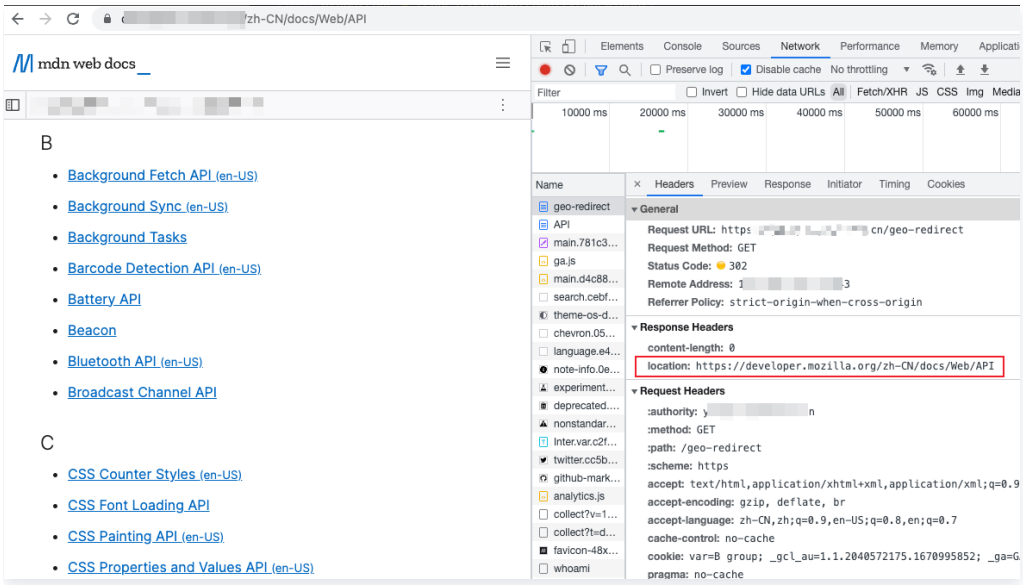
This example automatically redirects to the target URL of the client's region by determining the client's region. It demonstrates the use of edge functions to distribute requests based on the client's region.

Sample code

```
// The collection of URLs in all regions.
const urls = {
  CN: 'https://developer.mozilla.org/zh-CN/docs/Web/API',
  US: 'https://developer.mozilla.org/en-US/docs/Web/API',
};

// The default redirect URL.
const defaultUrl = 'https://developer.mozilla.org/en-US/docs/Web/API';

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

  return new Response(null, {
    headers: {
      location: url
    },
    status: 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: 2023-09-07 15:19:02

In edge functions, use the Fetch API to obtain the remote resource jQuery.js, and with the help of the Cache API, cache the resource at the EdgeOne edge node for a duration of 10 seconds.

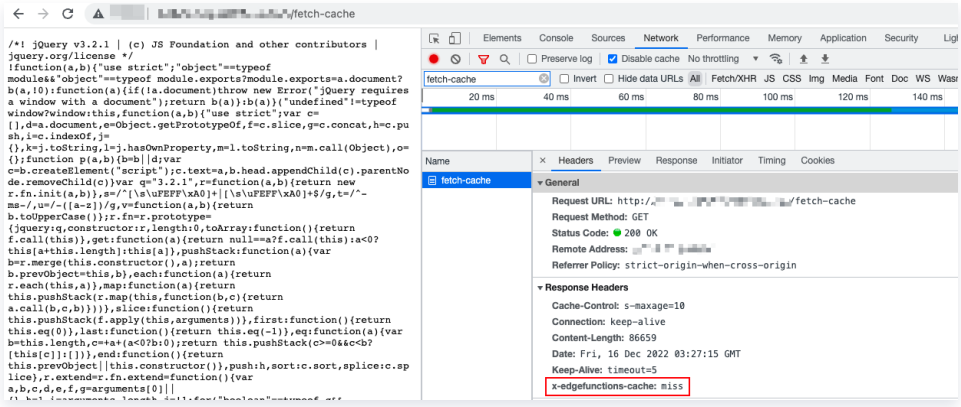
Sample code

```
async function fetchJquery(event, request) {
  const cache = caches.default;
  // Cache miss, origin-pull and cache
  let response = await fetch(request);

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

  // Cache miss, set response header identifier
  response.headers.append('x-edgefunctions-cache', 'miss');
  return response;
}

async function handleEvent(event) {
  // Resource address, also used as the cache key
  const request = new Request('https://static.cloudcachetci.com/qcloud/main/scripts/release/common/vendors/jquer
  // Cache the default instance
  const cache = caches.default;

  try {
    // Retrieve associated cache content. If the cache has expired, the underlying API will not actively pull from the or
    let response = await cache.match(request);

    // Cache does not exist, re-fetch remote resource
    if (!response) {
      return fetchJquery(event, request);
    }

    // Cache hit, set response header identifier
    response.headers.append('x-edgefunctions-cache', 'hit');

    return response;
  } catch (e) {
    await cache.delete(request);
    // Cache expiration or other exceptions, re-acquire remote resources
    return fetchJquery(event, request);
  }
}

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

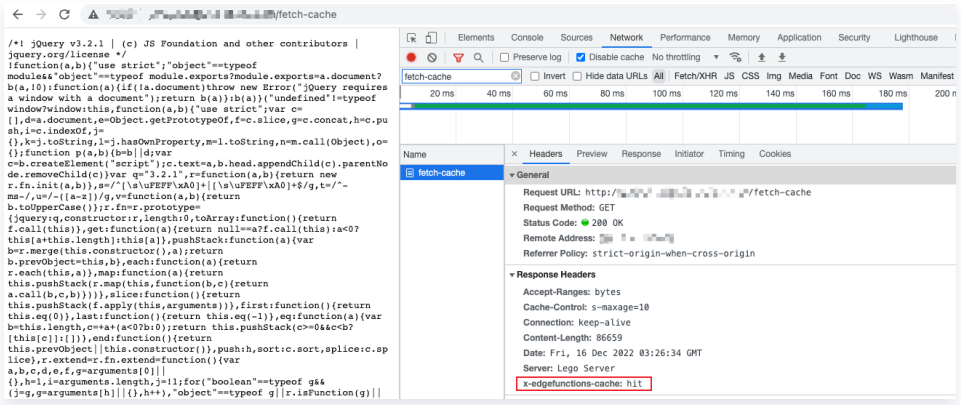
Sample Preview

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: 2023-09-07 15:19:08

This example calculates the SHA-256 signature of the POST request body as part of the cache key, and uses the Cache API to cache the response content. If a cache exists, it responds to the client with the cached content; otherwise, it uses the Fetch API to initiate a sub-request to obtain remote resources. This effectively implements caching of POST requests using edge functions.

Sample code

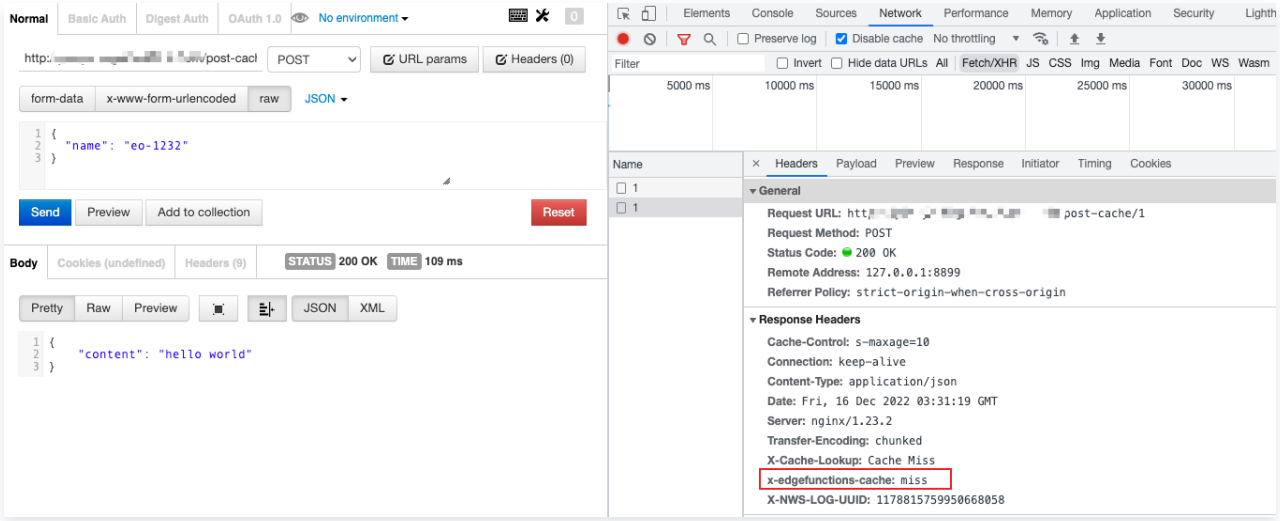
```
function uint8ArrayToHex(arr) {
  return Array.prototype.map.call(arr, (x) => (('0' + x.toString(16)).slice(-2))).join('');
}

// 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;

  // Cache miss, origin-pull and use cache.
  const response = await fetch(event.request);

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

  // Cache miss, set response header identifier
  response.headers.append('x-edgefunctions-cache', 'miss');

  return response;
}

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

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

  // The hash value calculated from the request body serves as part of the cacheKey.
  const cacheKey = ${ request.url }${ hash };

  const cache = caches.default;

  try {
    // Retrieve the associated cache Response.
    let response = await cache.match(cacheKey);

    if (!response) {
```

```
      return fetchContent(event, cacheKey);
    }

    // Cache hit, set response header identifier
    response.headers.append('x-edgefunctions-cache', 'hit');

    return response;
  } catch (error) {
    await cache.delete(cacheKey);
    // If the cache is expired or does not exist, re-fetch the remote resources.
    return fetchContent(event, cacheKey);
  }

  return response;
}

addEventListener('fetch', (event) => {
  try {
    const request = event.request;
    // Handling POST requests.
    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));
  }
});
```

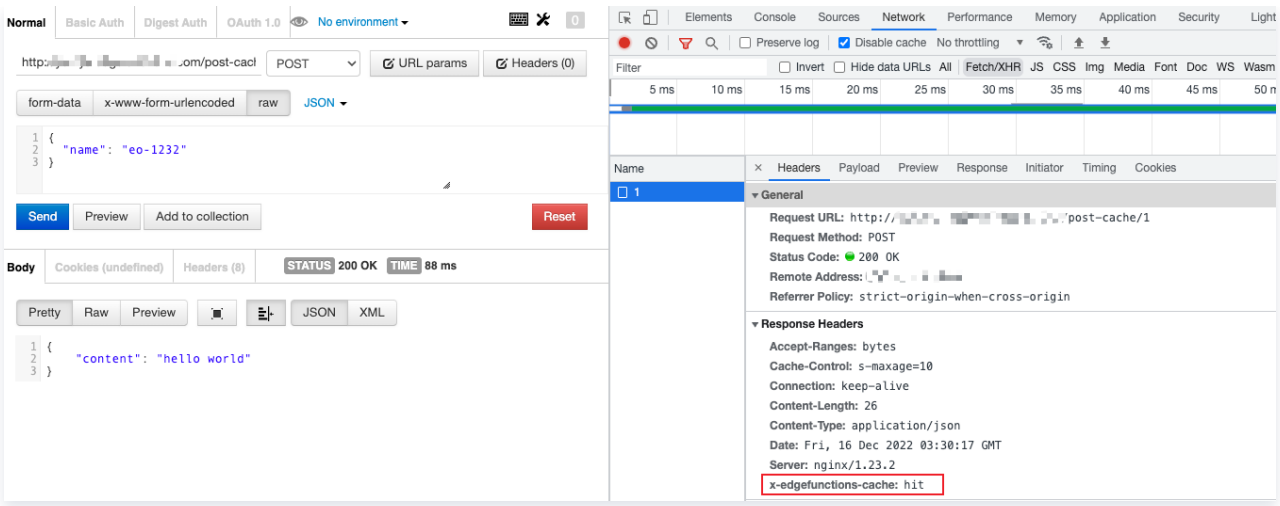
Sample Preview

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: 2023-09-07 15:19:13

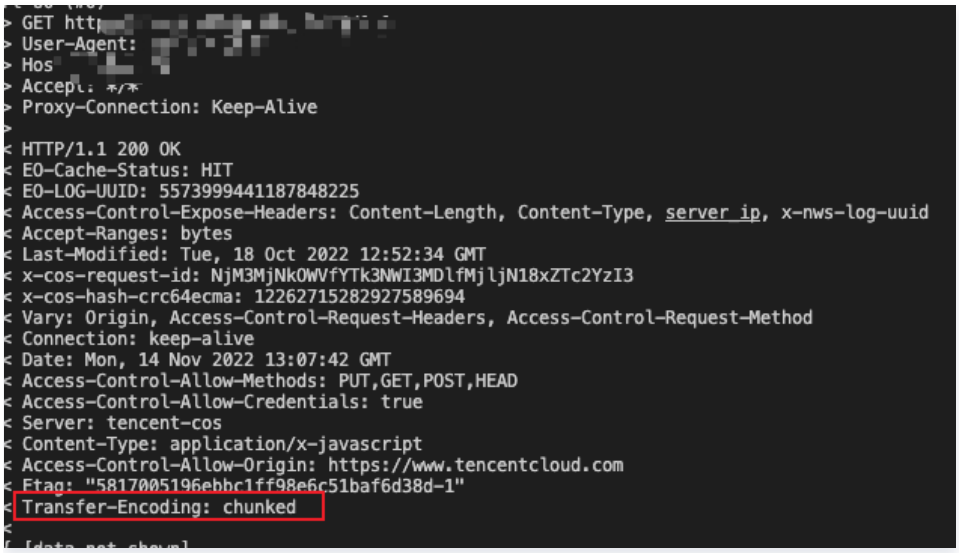
This example demonstrates the use of Fetch API to retrieve the remote resource jQuery.js and stream the response to the client.

Sample code

```
async function handleRequest(request) {
  const response = await fetch('https://static.cloudcachetci.com/qcloud/main/scripts/release/common/vendors/jquer

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

  // Generate readable and writable ends
  const { readable, writable } = new TransformStream();
  // Stream response to the client
  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.



References

- Runtime APIs: TransformStream
- Runtime APIs: Response

Merging Resources and Responding in Streaming Mode

Last updated: 2023-09-07 15:20:55

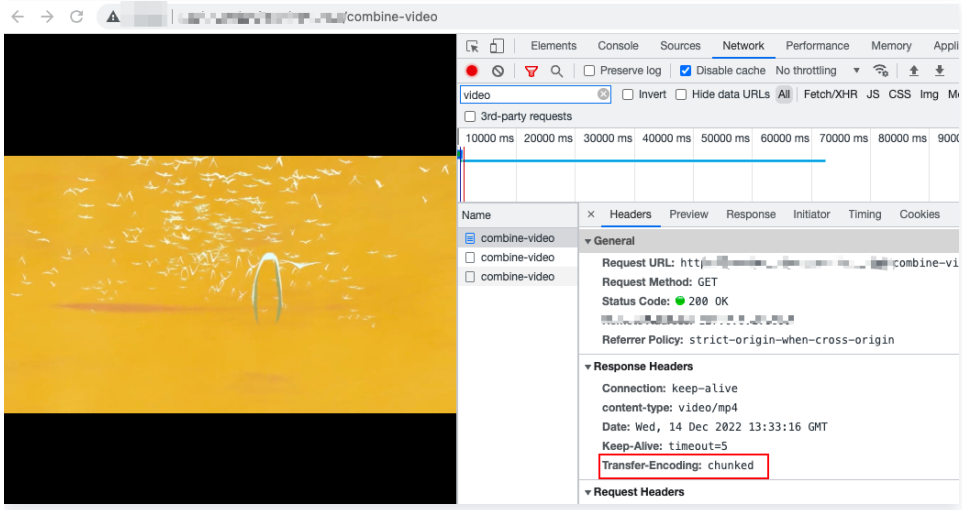
This example merges three videos into one, playing them in sequence on the client side. It demonstrates how to use edge functions to fetch multiple remote resources, stream-read and concatenate them, and finally stream-respond to the client.

Sample code

```
async function combine(readableVideos, destination) {
  for (const readable of readableVideos) {
    // Stream response
    await readable.pipeTo(destination, {
      preventClose: true
    });
  }

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

async function handleRequest(request) {
  // Video segment URL
  const urls = [
    'https://laputa-1257579200.cos.ap-guangzhou.myqcloud.com/stream-01.mov',
    'https://laputa-1257579200.cos.ap-guangzhou.myqcloud.com/stream-02.mov',
    'https://laputa-1257579200.cos.ap-guangzhou.myqcloud.com/stream-03.mov',
  ];

  const requests = urls.map(url => fetch(url));
  const responses = await Promise.all(requests);
  // Obtain the readable stream of the video segment
  const readableVideos = responses.map(res => res.body);

  const { readable, writable } = new TransformStream();
  // Merge videos
  combine(readableVideos, writable);

  return new Response(readable, {
    headers: {
      'content-type': 'video/mp4',
    }
  });
}

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

Sample Preview

Enter a URL that matches the trigger rule of the edge function in the browser's address bar to preview the merged video. By examining the response header, you can see that the video is transmitted in a chunked manner.



References

- Runtime APIs: TransformStream
- Runtime APIs: Response

Protecting Data from Tampering

Last updated: 2023-09-07 17:36:19

This example compares the SHA-256 checksum of the body with the checksum generated by the origin. If they match, the content has not been tampered with; otherwise, a 416 status code is returned, indicating that the content has been altered. This effectively verifies whether the content of the origin's response has been tampered with using edge functions.

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

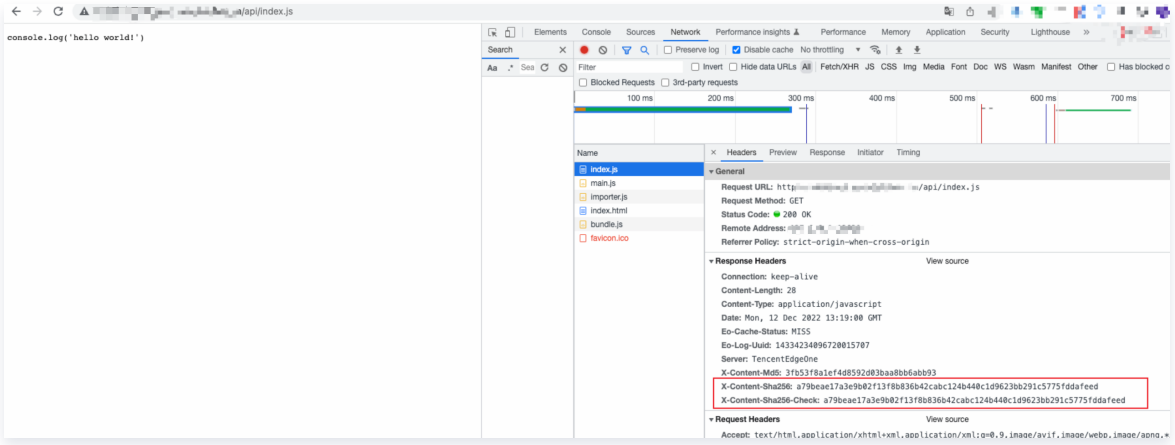
```
// The supported text file formats.
const textFileTypes = [
  'application/javascript',
  'text/html; charset=utf-8',
  'text/css; charset=utf-8',
  'text/xml; charset=utf-8'
];

// 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 algorithm here supports one of MD5, SHA-1, SHA-256, SHA-384, SHA-512, and is case-insensitive.
 Please note: When the origin server calculates the checksum, it should not directly sign the source file data. Instea
 Subsequently, the same method is employed here for calculation and comparison, thereby achieving the objective
 **/
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 real-time calculated signature with the origin signature. If the verification fails, return 416, indicati
  if (checkHash !== hash) {
    return new Response(null, {
      headers,
      status: 416
    });
  }

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

async function handleEvent(event) {
  // Retrieve the content returned by the origin server. If there is a cache on the EdgeOne node, it will not fetch from
  const response = await fetch(event.request);
  if (response.status === 200) {
    const headers = response.headers;
    // Origin content signature header
    const hash = headers.get('X-Content-Sha256');
    if (hash) {
      // Calculate whether the signatures match. The algorithm here supports MD5, SHA-1, SHA-256, SHA-384, SHA-5
      return checkAndResponse(response, hash, 'Sha-256');
    }
  }

  return response;
}

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

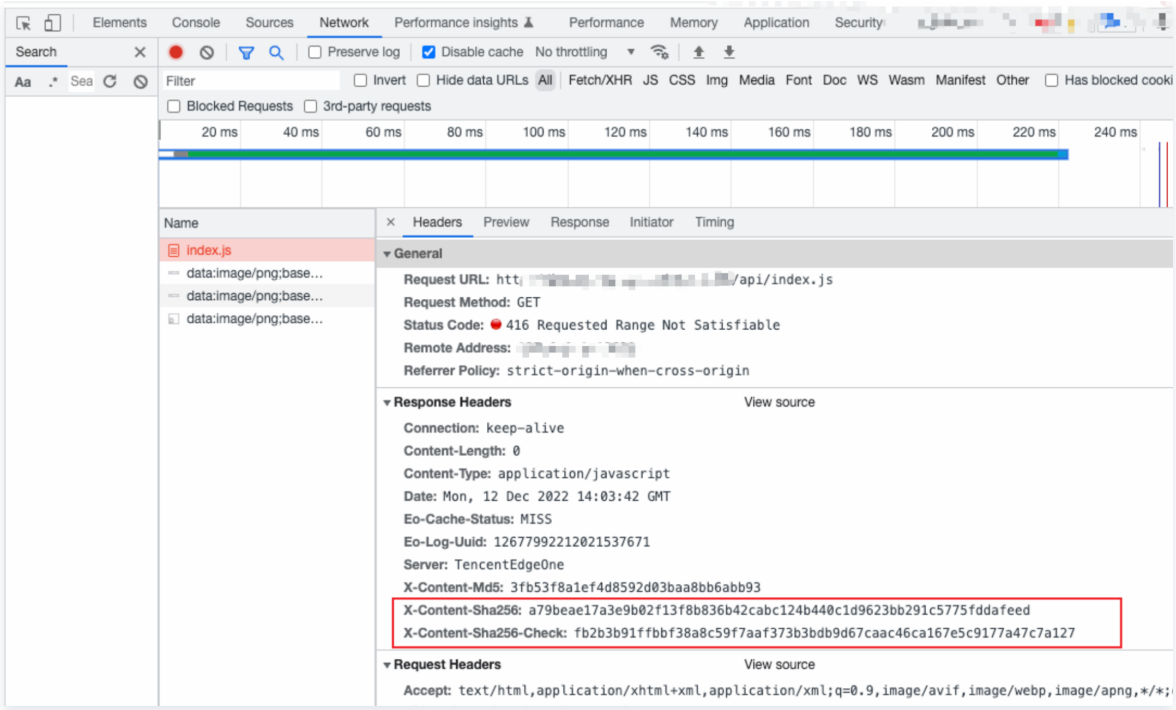
Sample Preview

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: 2023-09-07 15:21:09

In this instance, the .m3u8 file is restructured to incorporate Type A authentication, thereby facilitating access control for .m3u8 and .ts segment resources. Developers have the liberty to alter the code in accordance with their requirements to accommodate other authentication strategies.

Sample code

```
// The private key for TypeA authentication. Specify the key as needed and make sure that the key remains confider
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);

  // Verify if the file extension is either .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,
    }
   });
  }
```

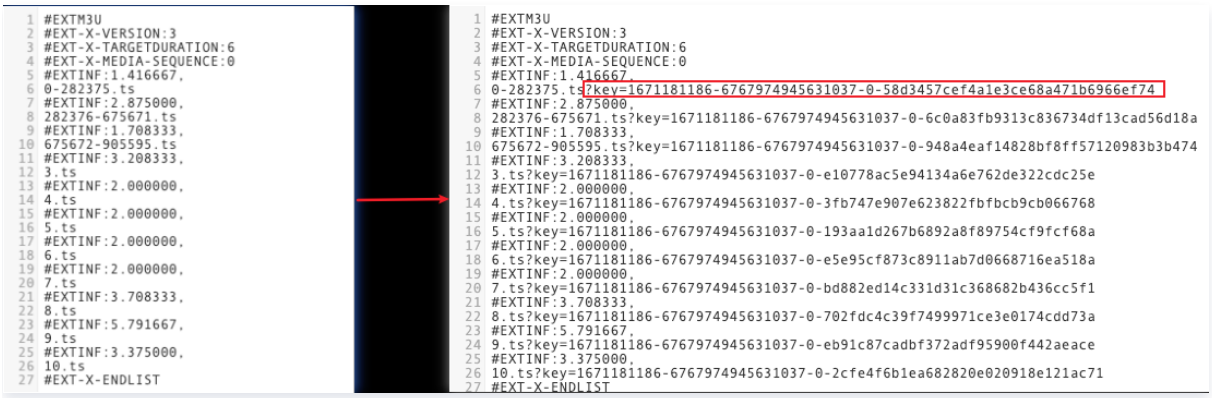
```
    // Respond with .ts resources.
    if (suffix === '.ts') {
      return fetchTs(request);
    }
  } catch (error) {
    return new Response(error.stack, { status: 544 });
  }

  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 === undefined) {
    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 (such as http://www.example.com/index.m3u8?key=1678873033-123456-0-32f4xxxxcabcxxxx1602xxxx6756d8f4) that corresponds to the trigger rule of the edge function to preview the effect of the sample code.



References

- Runtime APIs: Fetch
- Runtime APIs: Web Crypto
- Runtime APIs: Response
- Principles of Type A Authentication

Adaptive Image Format Conversion

Last updated: 2023-09-07 15:21:15

This example illustrates the process of identifying the browser type by extracting the `User-Agent` information from the request header, and utilizing the fetch API to retrieve the original image. Based on the identified browser type, the image is then converted to a compatible format, thereby achieving an adaptive image formatting effect. This method enhances page loading speed, optimizes image acceleration performance, and ultimately elevates the user experience on the website.

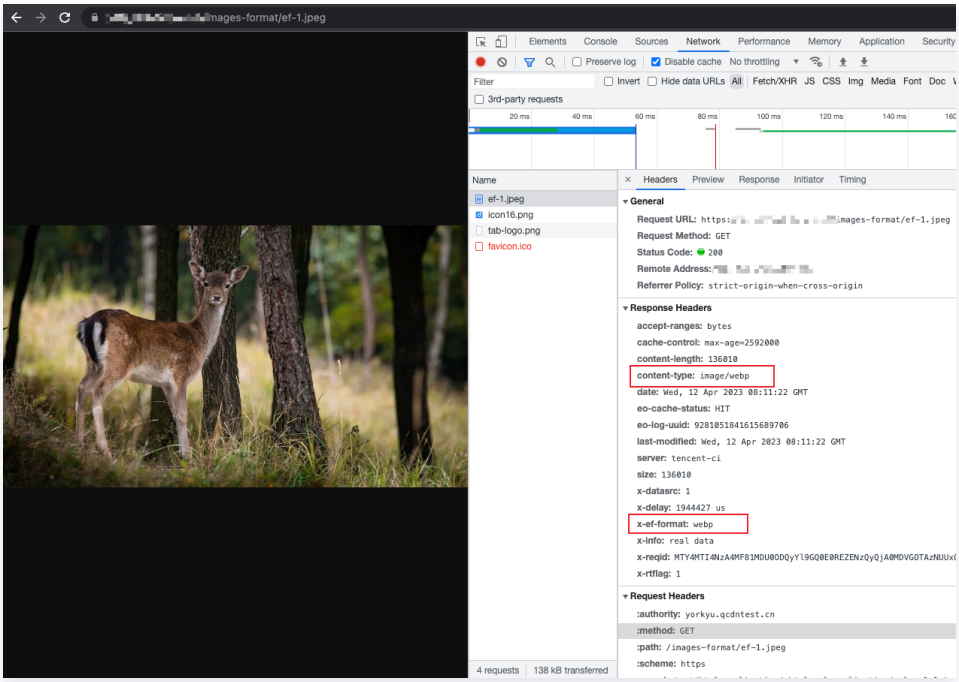
```
// Browser utilizes image format
const broswerFormat = {
  Chrome: 'webp',
  Opera: 'webp',
  Firefox: 'webp',
  Safari: 'jp2',
  Edge: 'webp',
  IE: 'jxr'
};

addEventListener('fetch', event => {
  // If the function code throws an unhandled exception, Edge Functions forwards the current request back to the ori
  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];

  // No need to 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'
  }
}
```

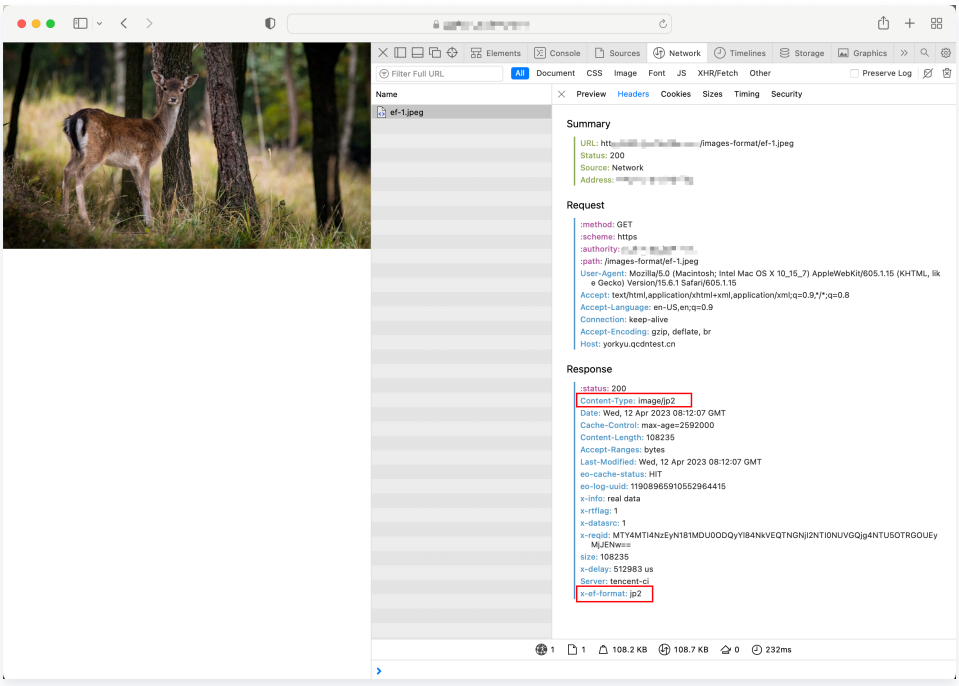
Sample Preview

In the address bar of the browser, enter a URL (such as `https://example.com/images-format/ef-1.jpeg`) that matches a trigger rule of the edge function to preview the effect of the sample code.

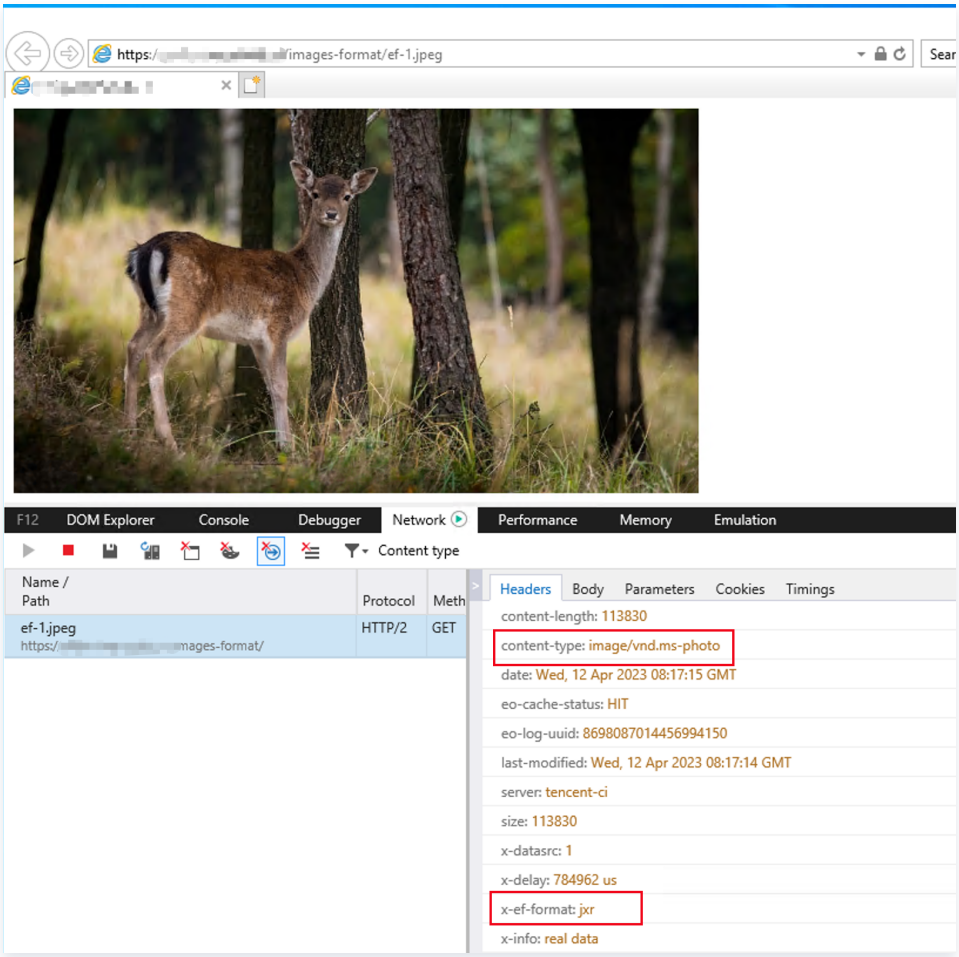
- When accessed via Google Chrome, the image is returned in webp format.



- Apple Safari: jp2



- When accessed via Microsoft's Internet Explorer, the image format responded is jxr.



References

- Runtime APIs: Fetch
- Runtime APIs: Headers

- Runtime APIs: Response
- Runtime APIs: Request

Adaptive Image Resize

Last updated: 2023-09-07 15:21:21

This example illustrates the process of identifying the client type by extracting the `User-Agent` information from the request header, and utilizing the fetch API to retrieve the original image. The image is then scaled according to the client type to achieve an adaptive scaling effect. This implementation method enhances the user experience on the website, ensuring that images are displayed in the optimal size across various devices.

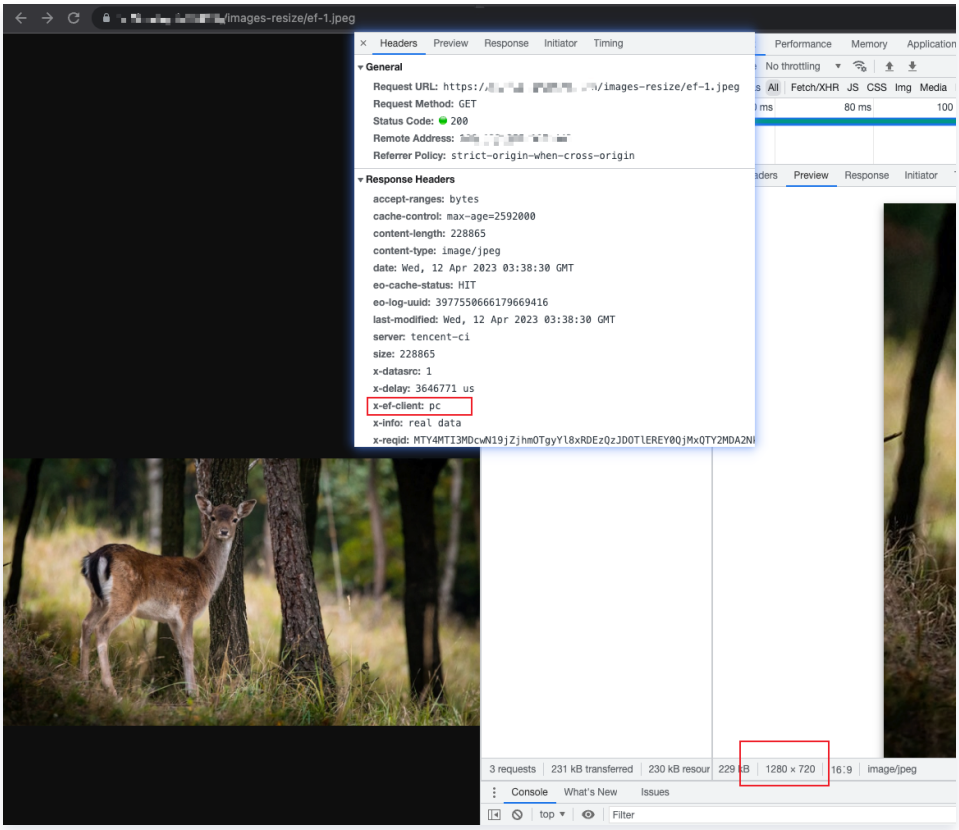
```
addEventListener('fetch', event => {
  // If the function code throws an unhandled exception, Edge Functions forwards the current request back to the
origin.
  event.passThroughOnException();
  event.respondWith(handleEvent(event));
});

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|IEMobile|MQQBrowser|JUC|Fennec|wOSBrowser|
BrowserNG|WebOS|Symbian|Windows Phone)/i;

  if(regex.test(userAgent)) {
```

```
    return false;
  }

  return true;
}
```

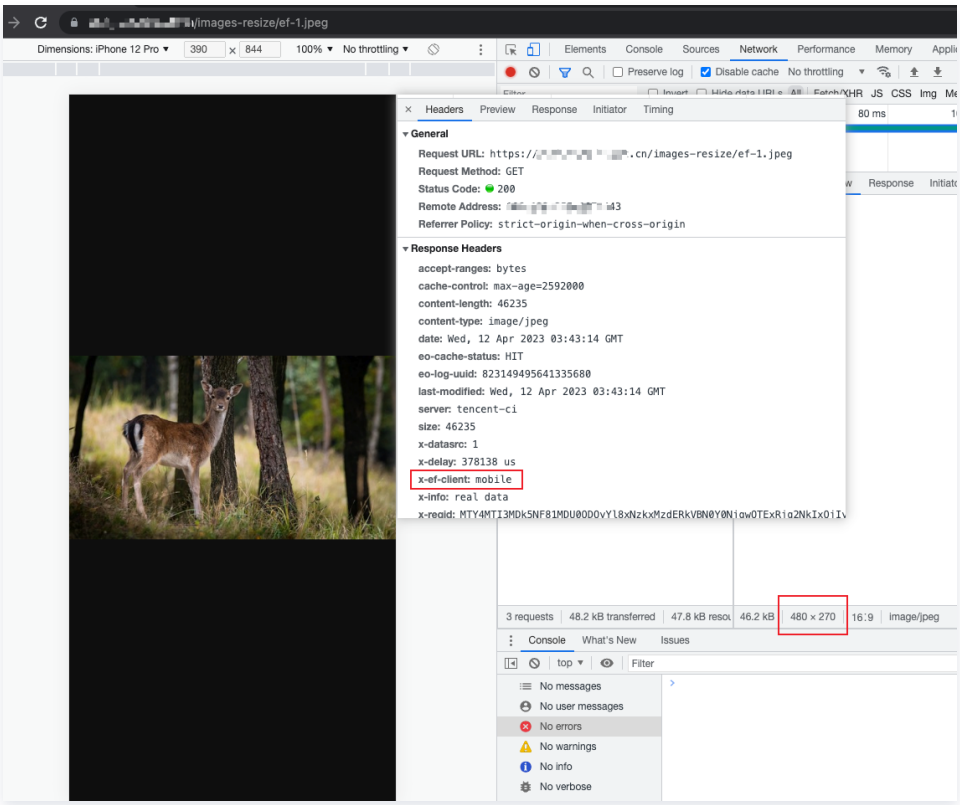
Sample Preview

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

Best Practice
Adaptive Image Format Conversion via Edge Functions

Last updated: 2023-09-07 15:21:28

This document elucidates the process of utilizing edge functions to automatically determine the image file format to be returned, based on the `User-Agent` header carried in the client request, without the need to alter the original client request URL. This process automatically triggers the conversion of the image format.

Background

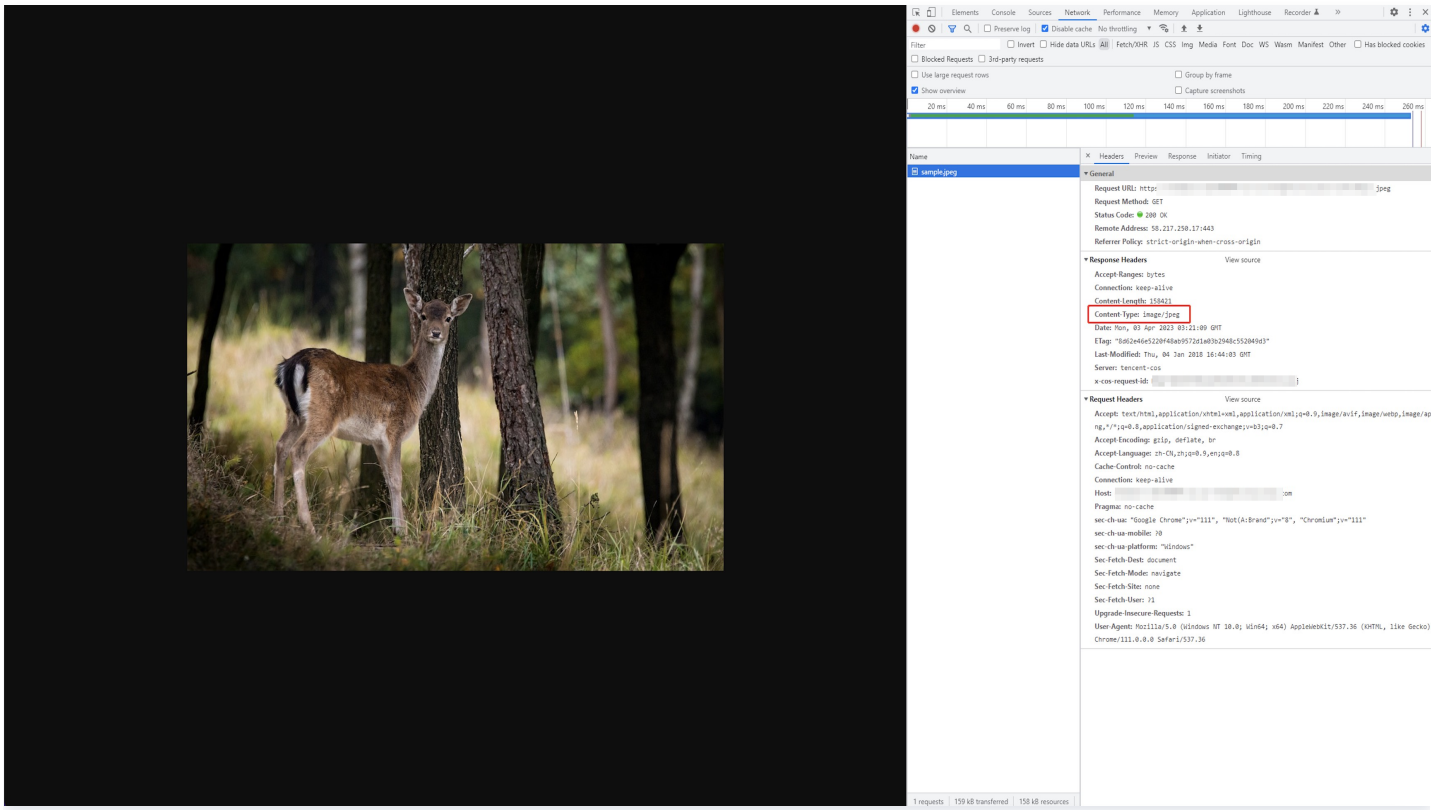
For platforms predominantly featuring a vast array of images, such as news media, e-commerce platforms, forums, etc., it is imperative to adapt the image file format in accordance with the type of browser. This ensures the return of a browser-compatible image format, while simultaneously compressing the image to its maximum extent to conserve data traffic. For instance:

- Return webp images for Chrome, Opera, Firefox and Edge browsers.
- Return jp2 images for the Safari browser.
- Return jxr images for the Internet Explorer browser.
- Return webp images for all the other browsers.

Edge functions offer flexible image processing capabilities, enabling automatic image format conversion by EdgeOne's edge functions without the need to modify the original client request URL. This is achieved by adaptively responding to the specified image format based on the client's `User-Agent` information. This allows you to provide users with the best format images adaptively, reducing data traffic consumption without the need to alter your business logic. If you wish to actively control the triggering of image format conversion in the request URL, you may also consider using the Image Processing capability.

Use Cases

The current site is `example.com`, with all image content stored under the path `http://image.example.com/image/`. It is required to adapt all images under this path to respond with the optimal image format based on the client's browser type. The original image request URL for testing is `https://image.example.com/image/test.jpg`. The image format can be viewed after accessing the following:



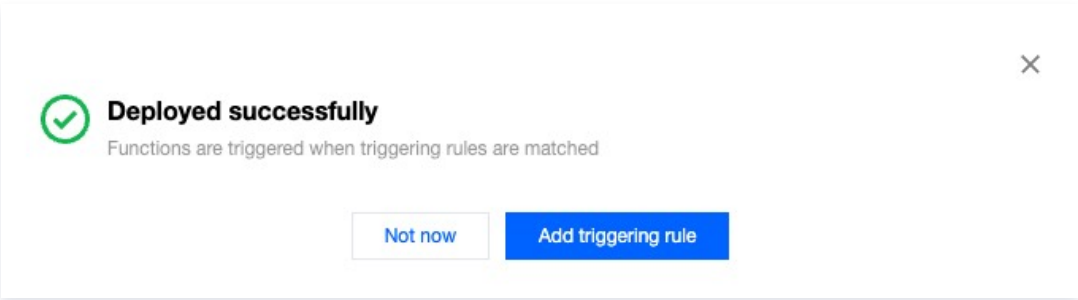
Instructions

1. Log in to the EdgeOne console. Click the target site in the site list to display second-level menus for site management.
2. In the left navigation bar, click on **Edge Functions > Function Management.**
3. On the function management page, click on **Create New Function.**
4. On the function creation page, enter the function name, description and codes. See below for the sample codes:

```
// Browser utilizes 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 the current request back to the
  event.passThroughOnException();
  event.respondWith(handleEvent(event));
});

async function handleEvent(event) {
  const { request } = event;
  const userAgent = request.headers.get('user-agent');
```

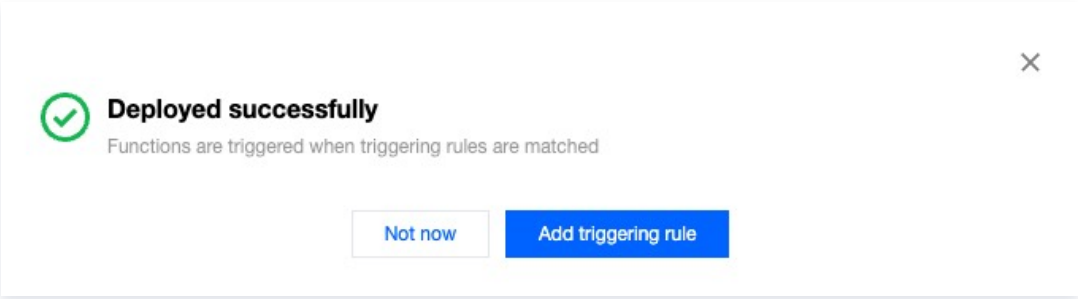
```
  const bs = getBroswer(userAgent);
  const format = broswerFormat[bs];

  // No need to 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. Upon completion of function editing, click **Create and deploy function**. Once the function is deployed, you can directly click **Add trigger rule** to configure the triggering rules for the function.



6. Within the function trigger rules, configure the conditions that will activate this function. Depending on the current scenario requirements, you can set up two trigger conditions, which will be initiated using an AND logic.
   - 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** to activate the rule.
8. You can verify the effectiveness of the edge function in two ways:

Curl Request Testing

You can run a curl command with the specified User-Agent to test.

Testing Chrome and other browsers

To test on Google Chrome on a 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`.

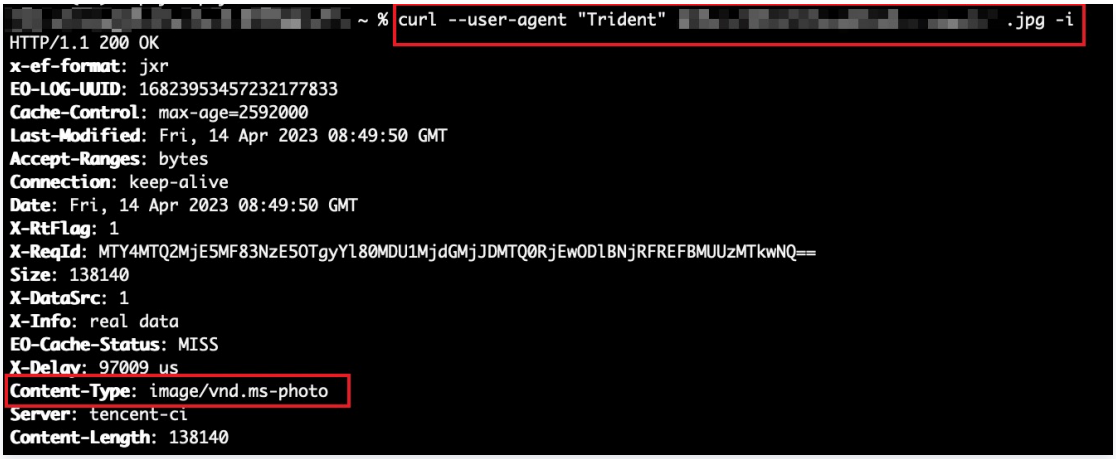
```
~ % curl --user-agent "chrome" h..jpg -i
HTTP/1.1 200 OK
x-ef-format: webp
EO-LOG-UUID: 3816446099087859674
Cache-Control: max-age=2592000
Last-Modified: Fri, 14 Apr 2023 08:14:28 GMT
Accept-Ranges: bytes
Connection: keep-alive
Date: Fri, 14 Apr 2023 08:14:28 GMT
X-RtFlag: 1
X-ReqId: MTY4MTQ2MDA2Nl83NzE5OTgyYl85QkZERTQ5OEFDQjE0N0Q4ODBBM0ZDMTQ5QjU4NzI4MA==
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
```

Testing the Safari Browser

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`.

Testing Internet Explorer

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`.

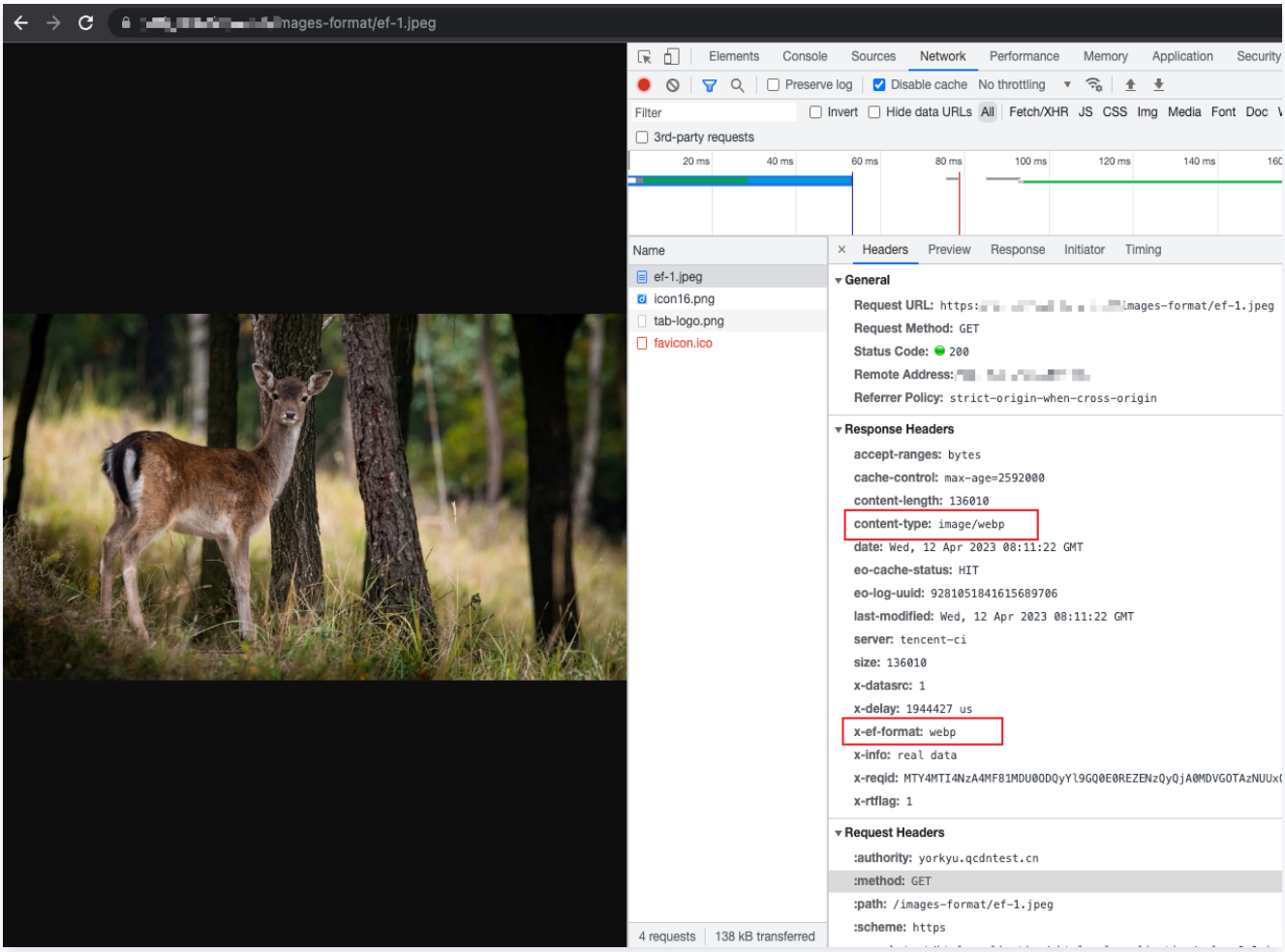


Browser Access Testing

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.

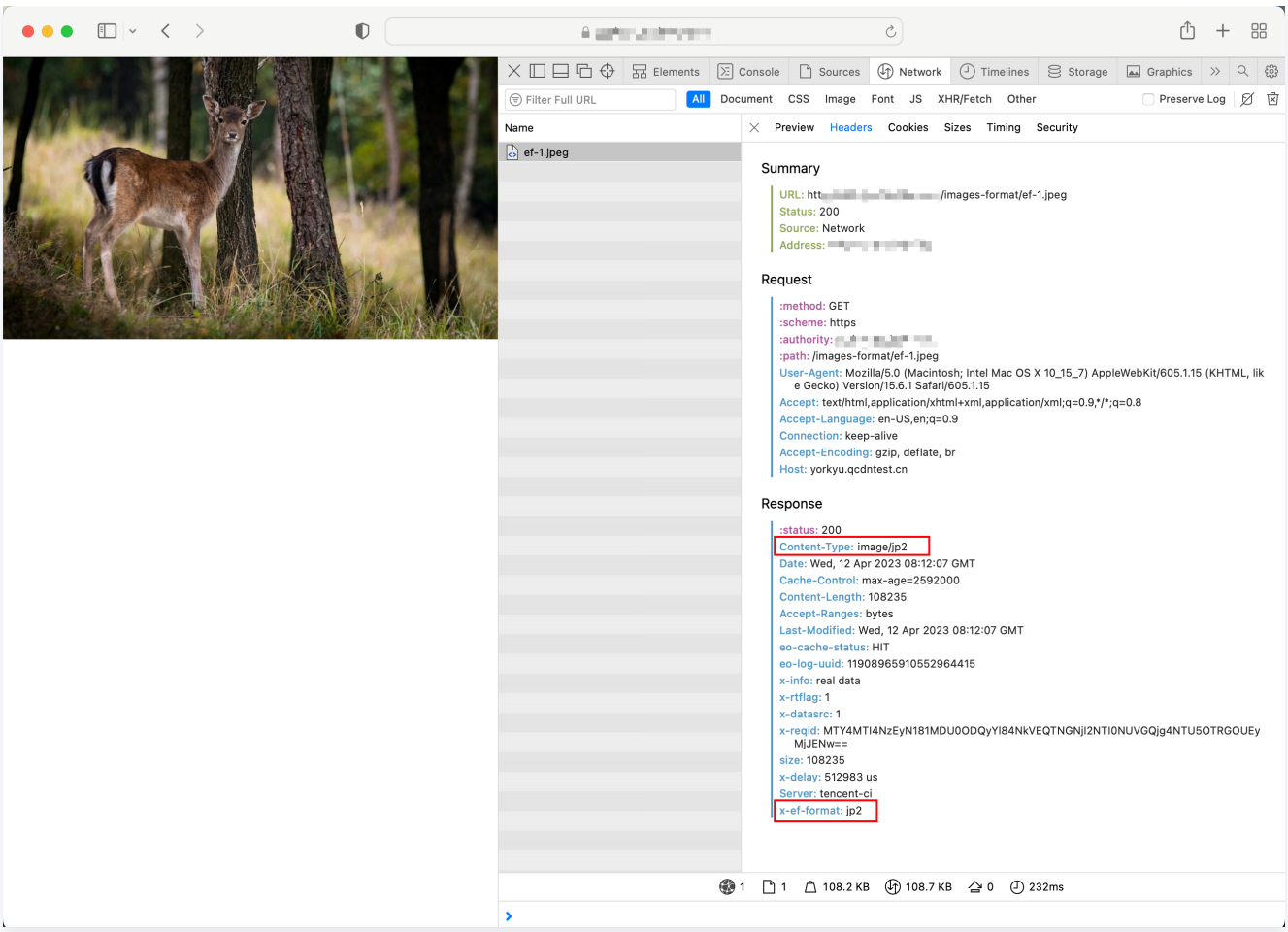
Testing Chrome and other browsers

Access the test image address `https://image.example.com/image/test.jpg` with Chrome browser. The responded image should be in webp format.



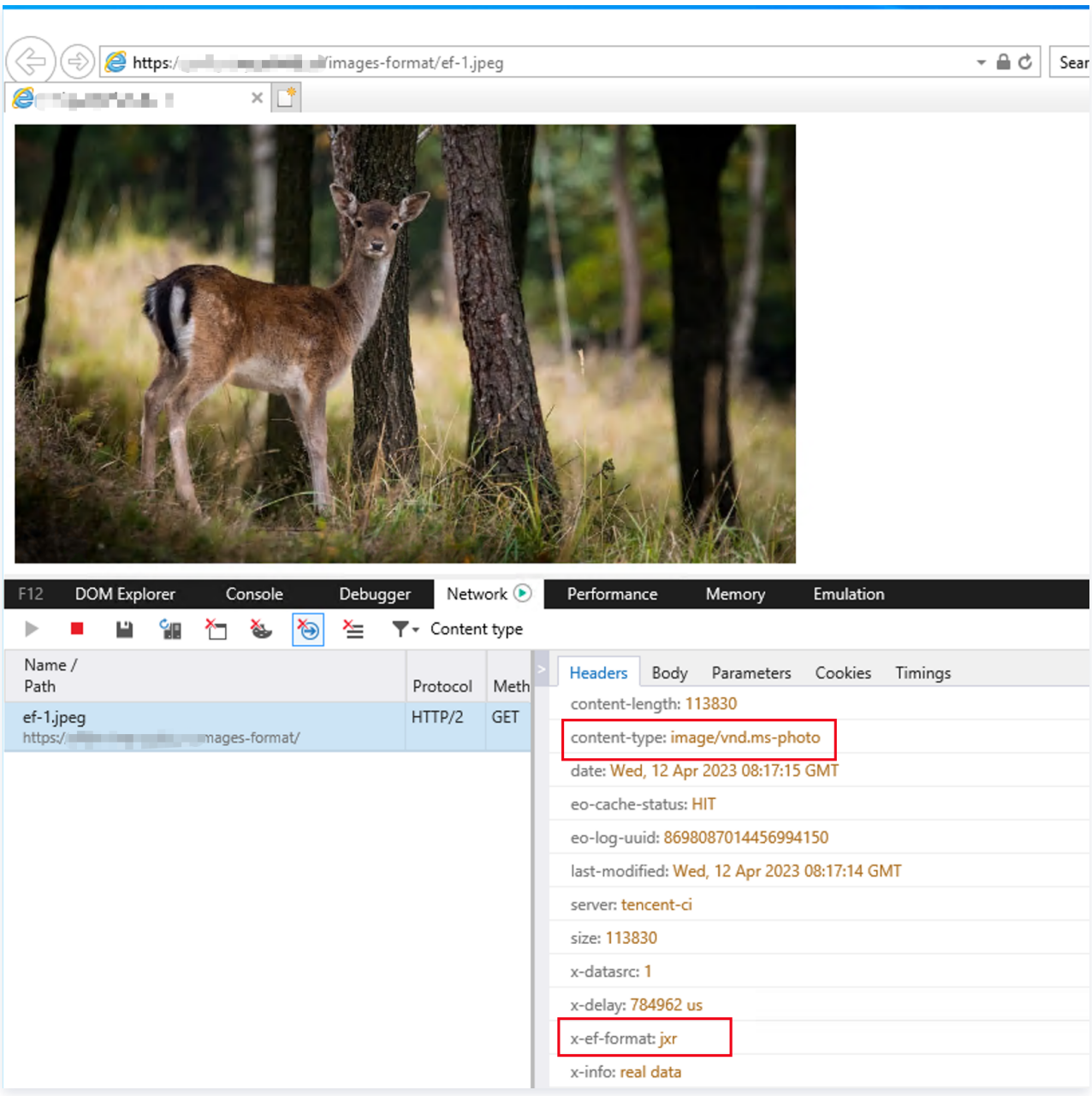
Testing the Safari Browser

Access the test image address `https://image.example.com/image/test.jpg` with Safari browser. The responded image should be in jp2 format.



Testing Internet Explorer

Access the test image address `https://image.example.com/image/test.jpg` with Safari browser. The responded image should be in jxr format.



Learn more

- Sample function: Image auto-adaptation
- Implementing Image Scaling through Site Acceleration