

Tencent Cloud EdgeOne

Site Acceleration

Product Documentation



Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

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

Trademark Notice



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

Service Statement

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

Contents

Site Acceleration

Overview

Access Control

Token Authentication

Introduction to Token Authentication

Authentication Method A

Authentication Method B

Authentication Method C

Authentication Method D

Authentication Method V

Smart Acceleration

Cache Configuration

Overview

EdgeOne Cache Rules

Content Cache Rules

Cache Key Introduction

Vary Feature

Cache Configuration

Custom Cache Key

Node Cache TTL

Status Code Cache TTL

Browser Cache TTL

Offline Caching

Cache Prefresh

Clear and Preheat Cache

Cache Purge

URL Pre-Warming

Prefetch M3U8

How to improve the Cache Hit Rate of EdgeOne

File Optimization

Smart Compression

Network Optimization

HTTP/2

HTTP/3(QUIC)

Overview

Enable HTTP/3

QUIC SDK

SDK Overview

SDK Download and Integration

Sample Code

Android

iOS

API Documentation

Android

iOS

IPv6 Access

Maximum Upload Size

WebSocket

Origin-Pull Request Headers Carrying Client IP

Origin-Pull Request Headers Carrying Client IP Geo Location

gRPC

URL Rewrite

Access URL Redirection

Origin-Pull URL Rewrite

Modifying Header

Modifying HTTP Response Headers

Modifying HTTP Request Headers

Modify the response content

HTTP Response

Custom Error Page

Rule Engine

Overview

Rule Management

Variables

Supported Matching Types and Actions

Related References

Rule Engine Configuration Character Limit

Request and Response Actions

Processing order

Default HTTP Headers of Origin-Pull Requests

Default HTTP Response Headers

HTTP Restrictions

Speed limit for single connection download

Site Acceleration Overview

Last updated : 2024-09-25 11:08:51

Site acceleration service is specifically designed for internet content delivery acceleration services such as HTTP/HTTPS application layer protocols, especially suitable for the distribution of websites, online applications, streaming media, and other content. Site acceleration can help you achieve more efficient and stable content delivery through a wealth of functional configurations, such as cache optimization, file optimization, network optimization, etc., to improve the satisfaction of business users and enhance the competitiveness of your website, application, or other online services.

After accessing the EdgeOne security acceleration service, you can customize the following site acceleration services:

Rule type	Function	Usage Scenarios	Default Configuration
Access control	Token authentication	EdgeOne nodes authenticate client access to prevent unauthorized behavior.	Off
Smart acceleration		Smart acceleration for dynamic resource access requests to improve access speed.	Off
Cache configuration	Vary feature	Differentiate node cache content based on the Vary header in the origin response, and respond to corresponding resources based on the specified header content.	Globally disabled by default on the platform
	Custom Cache Key	Customize the cache key (Cache Key) of the file in the node to determine the caching rules and whether to hit the cache.	By default, the character cache is not ignored. For details, please refer to the introduction of Cache Key .
	Node cache TTL	Customize the cache time of the file in the node to determine whether the file should be cached in the EdgeOne node and the corresponding cache time.	Please refer to the default cache rules of EdgeOne nodes .
	Status code cache TTL	Customize the cache time of error status codes in EdgeOne nodes to reduce the pressure of origin-pull requests when the origin is abnormal.	Cache 404 for 10s, other exception status codes are not cached.
	Browser cache TTL	Control whether the browser caches files to reduce access request traffic.	By default, follow the origin Cache-Control.

	Offline cache	Control whether EdgeOne uses cached files (even if the file has expired) to respond to client requests when the origin is abnormal, ensuring normal service response when the origin is abnormal.	Enable
	Cache pre-refresh	Let EdgeOne nodes verify the validity of the file before the cache expires, ensuring that the latest resource files are cached in the node.	Off
File optimization	Smart compression	EdgeOne nodes perform Gzip compression or Brotli compression to reduce file transmission size.	Enable Gzip compression and Brotli compression
Media processing	Image processing	EdgeOne nodes automatically scale and convert image formats, eliminating the need for multiple sizes and formats of image copies on the origin, reducing image storage costs.	Off
Network optimization	HTTP /2	Require EdgeOne nodes to support HTTP/2 access.	Enable
	HTTP/3	Require EdgeOne nodes to support HTTP/3 access.	Off
	IPv6 access	Require EdgeOne nodes to support IPv6 access.	Off
	Maximum upload size	EdgeOne nodes limit the maximum file size that users can upload to prevent malicious uploads of large files and reduce transmission traffic.	Enable, maximum limit of 800MB
	WebSocket	Require EdgeOne nodes to support WebSocket transmission and configure timeout connection duration.	Off
	Client IP header	Specify the request header to carry the client IP when EdgeOne nodes origin-pull.	Off
	Client IP geographic location header	Specify the request header to carry the client IP geographic location when EdgeOne nodes origin-pull.	Off
	Enable gRPC	Require EdgeOne nodes to support gRPC	Off

		access.	
URL Rewrite	Rewrite access URL	Customize the redirection of the user's access URL address.	N/A
	Origin-pull URL rewrite	Customize the redirection of the user's origin-pull request URL address.	N/A
Modify headers	Modify HTTP node response headers	Customize the modification of HTTP headers carried by EdgeOne nodes when responding to user requests.	N/A
	Modify HTTP origin-pull request headers	Customize the modification of HTTP headers carried by EdgeOne nodes when origin-pulling.	N/A
Custom error page		Customize the modification of the error page displayed to the client when the origin responds with 4xx and 5xx status codes.	N/A

Access Control

Token Authentication

Introduction to Token Authentication

Last updated : 2025-04-01 14:45:21

Overview

Token authentication is a simple and reliable access control strategy that verifies URL access through authentication rules, effectively preventing malicious brushing of site resources. The usage of this function requires the cooperation of the client and EdgeOne. The client is responsible for initiating encrypted URL requests, and EdgeOne is responsible for verifying the legality of the URL based on pre-set rules.

Function principle

The implementation of Token authentication mainly consists of the following two parts:

Client: Initiate the authentication URL request based on the authentication rules (including authentication algorithm, key).

EdgeOne node: Verify the authentication information (md5 string + timestamp) in the authentication URL. When the verification is passed, the access request will be considered as a valid request, and the node will respond normally. If the verification fails, the node will reject the access and directly return 403.

Token authentication URL generation and verification tool

EdgeOne provides [a generation tool and verification tool](#) for Token authentication URLs. Developers can use this tool to quickly and accurately generate and verify anti-leeching URLs that meet the requirements.

Directions

1. Log in to [the EdgeOne console](#), In the left sidebar, click **Site List**. Within the Site List, click on the **Site** you wish to configure.
2. On the site details page, click **Site Acceleration** to enter the global configuration page for the site, then click the **Rule Engine** tab.

3. On the Rule Engine Management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, set the matching conditions that trigger this rule.
5. Click **Action** > **Select Box**, and select **Token authentication** in the pop-up operation list. The parameter configuration instructions are as follows:

Parameter	Description
Method	Currently, 5 authentication signature calculation methods are supported. Please choose the appropriate method based on the access URL format. For details, please refer to the authentication method .
Primary key (Required)	The primary password, consisting of 6-40 uppercase and lowercase English letters,numbers and special characters(Except " and \$).
Backup key (optional)	The secondary password, consisting of 6-40 uppercase and lowercase English letters,numbers and special characters(Except " and \$).
Authentication encryption string	An authentication parameter must be between 1-100 characters and contains letters, numbers and underscores. The parameter value will be authenticated by nodes.
Validity period	Validity period of the authentication URL (1-630720000 seconds). It determines whether a client request is valid: If the time "timestamp + validity period" is reached, the request is considered expired and a 403 is returned. If the current time does not exceed the "timestamp + valid duration" time, the request is not expired and continues to verify the md5 string.

Must-knows

1. After Authentication is passed, the node will automatically ignore the Authentication-related parameters in the URL to improve the Cache hit rate and reduce the amount of origin-pull.
2. The origin-pull request URL cannot contain any Chinese characters.

Authentication Method A

Last updated : 2024-12-19 11:00:19

Authentication URL format

```
http://Hostname/Filename?sign=timestamp-rand-uid-md5hash
```

Description of authentication fields

Field	Description
Hostname	Site Acceleration Domain.
Path	Resource access path, authentication requires prefixing with <code>/</code> .
sign	Authentication parameters name set by Definition.
timestamp	Decimal positive integer Unix timestamp (the total number of seconds from 00:00:00, January 1, 1970, UTC time, to now, independent of the timezone)
rand	0 - 100 characters, a random string consisting of uppercase and lowercase letters and numbers.
uid	User ID, currently unused, default is 0.
md5hash	A fixed length string of 32 bits calculated using the MD5 algorithm: Algorithm: MD5(Path-timestamp-rand-uid-key). Authentication Logic: If the request has not expired, the node compares this string value with the <code>md5hash</code> value carried in the request URL. If the values are the same, authentication passes, and the request is responded to; if the values are different, authentication fails, returning 403.

Configuration Samples

Assume the request `https://www.example.com/foo.jpg` uses Authentication Method A, configured as follows:

Get authentication parameters:

Path: `/foo.jpg` .

timestamp: The server generates the authentication URL timestamp as July 15, 2024, 15:27:17 (UTC+8), converted to a decimal (Unix timestamp) format value: `1721028437` .

rand: The generated random number is `Kv4cPTAAP5YTi` .

uid: `0` .

Key: `DvYmqE81E1F9R791H6lmht` .

md5hash:MD5(Path-timestamp-rand-uid-key)=

MD5(`/foo.jpg` - `1721028437` - `Kv4cPTAAP5YTi` - `0` - `DvYmqE81E1F9R791H6lmht`)=
`0fbdca749d7ab784750685347e42075c` .

Authentication URL generated by the client request

`https://www.example.com/foo.jpg?sign=1721028437-Kv4cPTAAP5YTi-0-0fbdca749d7ab784750685347e42075c`

Node Authentication

When the Node Server receives a request from the client via the encrypted URL, it extracts the timestamp parameter from the URL, adds the configured Effective duration of "1 second", and compares it to the current time:

1. exceed the "timestamp + effective duration" time, the request is not expired, continue to step 2.
2. The Node Server calculates the md5hash value using the obtained authentication parameters and compares it with the md5hash value carried in the request URL. If the values are the same, authentication passes and the request is responded to; if the values are different, authentication fails, returning 403.

Authentication Method B

Last updated : 2024-08-14 19:04:36

Format of Authentication URL

```
http://Hostname/timestamp/md5hash/Filename
```

Parameter Description

Field	Description
Hostname	Site acceleration domain name.
Path	Resource access path, which must start with <code>/</code> during authentication.
timestamp	UTC+8 time in the format of YYYYMMDDHHMM, e.g., 201807301000.
md5hash	<p>A string containing 32 characters calculated based on the MD5 algorithm: Algorithm: MD5 (key + timestamp + Path).</p> <p>Authentication logic: If the request has not expired, the node will compare this string value with the <code>md5hash</code> value carried in the request URL. If the values are the same, the request will pass the authentication, and a response will be made to the request; if the values are different, the authentication will fail, and 403 will be returned.</p>

Configuration Example

Assuming authentication method B is used for the requested URL `https://www.example.com/foo.jpg` , the configuration is as follows:

The screenshot shows the configuration interface for authentication parameters. It includes a matching rule section with 'URL Full' as the matching type, 'Is' as the operator, and 'https://www.example.com/foo.jpg' as the value. Below this is an action section with 'Token authentication' as the action, 'B' as the method, 'DvYmqE81E1F9R791H6lmht' as the primary key, and 'e7Tyty6ijy70xXKqckuxV7XXiu67x4G' as the backup key. A validity period of 1 second is also configured. A 'Generate/Verify Authentication URL' button is present.

Getting Authentication Parameters

Path: `/foo.jpg` .

timestamp: The time when the server generates the authentication URL is 15:33:50, July 15, 2024 (UTC+8), the decimal (YYYYMMDDHHmm) value of which is `202407151533` .

Key: `DvYmqE81E1F9R791H6lmht` .

md5hash: MD5 (key + timestamp + Path) = MD5 (`DvYmqE81E1F9R791H6lmht202407151533/foo.jpg`) = `d1f0b51c6894231fc12e054fcc7f0b3e` .

Authentication URL Generated by a Client Request

`https://www.example.com/202407151533/d1f0b51c6894231fc12e054fcc7f0b3e/foo.jpg` .

Node Authentication

When the node server receives a client request via the encrypted URL, it will parse the timestamp parameter in the URL, and compare the sum of it and the configured validity period of "1 second" with the current time:

1. If the current time is after the "timestamp + validity period" time, it indicates that the request has expired, and 403 will be returned directly. If the current time is before the "timestamp + validity period" time, it indicates that the request has not expired, and the node server will go to step 2.
2. The node server calculates the md5hash value based on the obtained authentication parameters and compares it with the md5hash value carried in the request URL. If the values are the same, the request will pass the authentication, and a response will be made to the request; if the values are different, the authentication will fail, and 403 will be returned.

Authentication Method C

Last updated : 2024-08-14 19:04:36

Format of Authentication URL

```
http://Hostname/md5hash/timestamp/Filename
```

Parameter Description

Field	Description
Hostname	Site acceleration domain name.
Path	Resource access path, which must start with <code>/</code> during authentication.
timestamp	A Unix timestamp in the hexadecimal format (the total number of seconds that have passed since 00:00:00, January 1, 1970 regardless of the time zone)
md5hash	<p>A string containing 32 characters calculated based on the MD5 algorithm: Algorithm: MD5 (key + Path + timestamp). Note: During calculation, the hexadecimal number identifier 0x of the hexadecimal timestamp must be filtered out.</p> <p>Authentication logic: If the request has not expired, the node will compare this string value with the <code>md5hash</code> value carried in the request URL. If the values are the same, the request will pass the authentication, and a response will be made to the request; if the values are different, the authentication will fail, and 403 will be returned.</p>

Configuration Example

Assuming authentication method C is used for the requested URL `https://www.example.com/foo.jpg` , the configuration is as follows:

The screenshot shows the configuration interface for a rule in Tencent Cloud EdgeOne. It is divided into two main sections: 'IF' (Condition) and 'Action'.

IF Section:

- Matching type:** URL Full
- Operator:** Is
- Value:** https://www.example.com/foo.jpg
- Buttons:** + And, + Or
- Toggle:** Ignore case (disabled)

Action Section:

- Action:** Token authentication
- Method:** C
- Primary key (Required):** DvYmqE81E1F9R791H6lmht
- Backup key (optional):** e77yry68jy7OoXqpcuxV700u67x4G
- Validity period:** 1 seconds
- Button:** Generate/Verify Authentication URL
- Buttons:** + Action, + IF

Getting Authentication Parameters

Path: `/foo.jpg` .

timestamp: The time when the server generates the authentication URL is 15:43:06, July 15, 2024 (UTC+8), the hexadecimal (Unix timestamp) value of which is `6694d30a` .

Key: `DvYmqE81E1F9R791H6lmht` .

md5hash: MD5 (key + Path + timestamp) = MD5 (`DvYmqE81E1F9R791H6lmht/foo.jpg6694d30a`) = `6688749e8906a726c12fe1be3aacd016` .

Authentication URL Generated by a Client Request

`https://www.example.com/6688749e8906a726c12fe1be3aacd016/6694d30a/foo.jpg` .

Node Authentication

When the node server receives a client request via the encrypted URL, it will parse the timestamp parameter in the URL, and compare the sum of it and the configured validity period of "1 second" with the current time:

1. If the current time is after the "timestamp + validity period" time, it indicates that the request has expired, and 403 will be returned directly. If the current time is before the "timestamp + validity period" time, it indicates that the request has not expired, and the node server will go to step 2.
2. The node server calculates the md5hash value based on the obtained authentication parameters and compares it with the md5hash value carried in the request URL. If the values are the same, the request will pass the authentication, and a response will be made to the request; if the values are different, the authentication will fail, and 403 will be returned.

Authentication Method D

Last updated : 2024-08-14 19:04:36

Format of Authentication URL

```
http://Hostname/Filename?sign=md5hash&t=timestamp
```

Parameter Description

Field	Description
Hostname	Site acceleration domain name.
Path	Resource access path, which must start with <code>/</code> during authentication.
sign	User-defined authentication parameter name.
t	User-defined timestamp parameter name
timestamp	Timestamp parameter. Format: Positive decimal integer Unix timestamp, which is the total number of seconds that have passed since 00:00:00, 1970.1.1 (UTC time) and is regardless of the time zone; or positive hexadecimal integer Unix timestamp, which is the total number of seconds that have passed since 00:00:00, 1970.1.1 (UTC time) and is regardless of the time zone.
md5hash	A string containing 32 characters calculated based on the MD5 algorithm: Algorithm: MD5 (key + Path + timestamp). Note: During calculation, the hexadecimal number identifier 0x of the hexadecimal timestamp must be filtered out. Authentication logic: If the request has not expired, the node will compare this string value with the <code>md5hash</code> value carried in the request URL. If the values are the same, the request will pass the authentication, and a response will be made to the request; if the values are different, the authentication will fail, and 403 will be returned.

Configuration Example

Assuming authentication method D is used for the requested URL `https://www.example.com/foo.jpg` , the configuration is as follows:

IF + Comment

Matching type: URL Full, Operator: Is, Value: https://www.example.com/foo.jpg

+ And + Or

Action: Token authentication, Method: D, Primary key (Required): DvYmqE81E1F9R791H6lmht, Backup key (optional): eT7ry6gy7OxXqckuxV7X0u67m4G

Authentication encryption string: sign, Authentication timestamp: t

Validity period: 1 seconds, Time format: Decimal timestamp

+ Action

+ IF

Getting Authentication Parameters

Path: `/foo.jpg` .

timestamp: The time when the server generates the authentication URL is 15:51:47, July 15, 2024 (UTC+8), the decimal (Unix timestamp) value of which is `1721029907` .

Key: `DvYmqE81E1F9R791H6lmht` .

md5hash: MD5 (key + Path + timestamp) = MD5 (`DvYmqE81E1F9R791H6lmht/foo.jpg1721029907`) = `cadcec4a04e67b9c2abf4b61c642a0dd` .

Authentication URL Generated by a Client Request

`https://www.example.com/foo.jpg?sign=cadcec4a04e67b9c2abf4b61c642a0dd&t=1721029907` .

Node Authentication

When the node server receives a client request via the encrypted URL, it will parse the timestamp parameter in the URL, and compare the sum of it and the configured validity period of "1 second" with the current time:

1. If the current time is after the "timestamp + validity period" time, it indicates that the request has expired, and 403 will be returned directly. If the current time is before the "timestamp + validity period" time, it indicates that the request has not expired, and the node server will go to step 2.
2. The node server calculates the md5hash value based on the obtained authentication parameters and compares it with the md5hash value carried in the request URL. If the values are the same, the request will pass the authentication, and a response will be made to the request; if the values are different, the authentication will fail, and 403 will be returned.

Authentication Method V

Last updated : 2025-04-01 14:43:34

Overview

In order to enhance permission control for video scenarios, EO has launched a solution to TypeV authentication. The features of this solution are as follows:

It supports specifying an expiration time in the URL, which cannot be used for a long time after being obtained by others;

It supports including a client IP in the signature calculation in the URL, which cannot be used after being obtained by others;

It supports specifying the preview duration in the URL to realize the preview feature;

It supports specifying a Referer blacklist/allowlist in the URL;

It supports specifying in the URL the start of video playback from a Unix timestamp to realize the pseudo-live streaming feature;

Developers use a `KEY` to sign the URL and include the signature in the URL. As long as a user's key is not leaked, other users cannot forge the encrypted URL;

The EdgeOne node checks the parameters and signature in the encrypted URL to control access requests. If a request fails to pass the check, a 403 response code will be returned.

Authentication Parameters

The following are the meaning and value description of each parameter in the authentication URL.

Parameter name	Required	Description
KEY	Yes	The key filled in when TypeV authentication is enabled. It should contain 6-40 characters, including uppercase and lowercase letters (a-Z), numbers (0-9), and special characters(Except <code>"</code> and <code>\$</code>). The random generation of it on the console is supported.
Path	Yes	The Path part of the original URL. If the original URL is <code>http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4</code> , the Path is <code>/dir1/dir2/myVideo.mp4</code> .
t	Yes	The expiration timestamp (in seconds) of the URL accessed, expressed in the lowercase hexadecimal format of Unix time.

		<p>Once expiration, the URL will no longer be valid, and a 403 response code will be returned. Considering the possible time difference between machines, the actual expiration time of the authentication URL is generally 5 minutes longer than the specified expiration time, that is, an additional tolerance time of 300 seconds is given.</p> <p>It is recommended that the expiration timestamp should not be too short, in order to ensure that users can complete downloads or video playback within the validity period.</p>
exper	No	<p>Preview duration in seconds, expressed in the decimal format. If the value is empty or 0, it means no preview (i.e., the full video is returned).</p> <p>The preview duration should not exceed the original video duration, as this may cause playback failures.</p>
us	No	<p>Link identifier, used to randomize an authentication URL and enhance the uniqueness of the link.</p> <p>It is recommended to specify a random 'us' value each time an authentication URL is generated.</p>
plive	No	<p>This parameter is in seconds, and is expressed in the lowercase hexadecimal format of Unix time. It enables the start of video playback from a specified time to realize the pseudo-live streaming feature.</p> <p>Example: If the value of plive is 669f9b40, it means that the resources corresponding to the encrypted request can be accessed only after 20:00:00, July 23, 2024.</p>
whref	No	<p>List of allowed domain names, which contain 1 to 10 items and are separated by half-width commas. No protocol name (http:// or https://) should be added to the beginning of a domain name. The domain name is in the form of exact match (for example, if abc.com is filled in, only abc.com can be matched and abc.com.cn cannot be matched), and wildcards (such as *.abc.com) are supported.</p>
bkref	No	<p>List of forbidden domain names, which contain 1 to 10 items and are separated by half-width commas. No protocol name (http:// or https://) should be added to the beginning of a domain name. The domain name is in the form of exact match (for example, if abc.com is filled in, only abc.com can be matched and abc.com.cn cannot be matched), and wildcards (such as *.abc.com) are supported.</p>
whip	No	<p>List of allowed client IPs, which contain 1 to 10 items and are separated by half-width commas. Client IPs can be IPs or IP segments, such as 192.168.0.0 or 192.168.0.0/24. Among others, 0.0.0.0/0 represents all IPv4 addresses, and ::/0 represents all IPv6 addresses.</p> <p>The client IP is obtained through the X-Forwarded-For header. If there are multiple header values, the first one is taken.</p>
bkip	No	<p>The list of forbidden client IPs, which contain 1 to 10 items and are separated by</p>

		half-width commas. Client IPs can be IPs or IP segments, such as 192.168.0.0 or 192.168.0.0/24. Among others, 0.0.0.0/0 represents all IPv4 addresses, and ::/0 represents all IPv6 addresses. The client IP is obtained through the X-Forwarded-For header . If there are multiple header values, the first one is taken.
sign	Yes	Hotlink protection signature, represented by a 40-character hexadecimal number. It is used to verify the validity of the authentication URL. If the signature verification fails, a 403 response code will be returned. For the calculation method, refer to the signature calculation formula .

Note:

For the [VOD origin server](#), all the above parameters are supported. For non-VOD origin servers, the `us` and `plive` parameters are not supported.

Signature Calculation Formula

```
sign = sha1(KEY + Path + t + plive + exper + us + whref + bkref + whip + bkip)
```

In the formula, `+` represents string concatenation, and optional parameters can be an empty string.

URL Generation Rules for Hotlink Protection

At the end of the original URL, hotlink protection parameters are added in the form of QueryString, as shown below:

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=[t]&exper=[exper]&us=[us]&
```

Note:

In the formula, `+` represents string concatenation, and optional parameters can be an empty string;

In the signature calculation, each parameter must strictly follow the order in the signature calculation formula. If the order is incorrect, an erroneous signature will be generated.

Configuration Examples

Assume the original URL is `http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4`. The developer has configured TypeV authentication, the generated Key is `24FEQmTzro4V5u3D5epW` and the generated random string is `72d4cd1101`. Below are scenarios for "validity time control of video playback URL",

"the same authentication URL allowing access by only one client IP" and "allowed video playback by client IP", which describe how to generate an authentication URL.

Example 1: Validity Time Control of Video Playback URL

If you need to generate an authentication URL for this video with an expiration time of 20:00:00, January 31, 2018 (Unix time: 1517400000), refer to the steps below.

Step 1: Determining Authentication Parameters

Parameter name	Value	Description
KEY	24FEQmTzro4V5u3D5epW	The key selected by the developer in enabling key hotlink protection
Dir	/dir1/dir2/myVideo.mp4	The Path part of the original URL
t	5a71afc0	The hexadecimal representation of the expiration timestamp 1517400000
us	72d4cd1101	A random string generated

Step 2: Calculating the Signature

```
sign = sha1('24FEQmTzro4V5u3D5epW/dir1/dir2/myVideo.mp45a71afc072d4cd1101') = '3
```

Step 3: Generating Authentication URL

The authentication parameters are concatenated into the original video URL's QueryString to get a video authentication URL:

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&us=72d4cd1101&sig
```

Example 2: The Same Authentication URL Allowing Access by Only One Client IP

If the same authentication URL requires access by only one client IP, refer to the steps below.

Step 1: Determining Authentication Parameters

Parameter name	Value	Description
KEY	24FEQmTzro4V5u3D5epW	The key selected by the developer in enabling key hotlink protection

Path	/dir1/dir2/myVideo.mp4	The Path part of the original URL
t	5a71afc0	The hexadecimal representation of the expiration timestamp 1517400000
us	72d4cd1101	A random string generated
whip	192.168.0.0	Allowed client IP

Step 2: Calculating the Signature

```
sign = sha1(24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc072d4cd1101192.168.0.0) = "c8cd89"
```

Step 3: Generating Authentication URL

The authentication parameters are concatenated into the original video URL's QueryString to get a video authentication URL:

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&us=72d4cd1101&whi
```

Example 3: Control of Allowed Playback Duration

If you need to generate a preview URL, with the preview duration being the first 5 minutes of the video (when the original video duration is greater than 5 minutes), refer to the steps below.

Step 1: Determining Authentication Parameters

Parameter name	Value	Description
KEY	24FEQmTzro4V5u3D5epW	The key selected by the developer in enabling key hotlink protection
Path	/dir1/dir2/myVideo.mp4	The Path part of the original URL
t	5a71afc0	The hexadecimal representation of the expiration timestamp 1517400000
exper	300	A preview of the first 5 minutes, i.e. 300 seconds
us	72d4cd1101	A random string generated

Step 2: Calculating the Signature

```
sign = sha1(24FEQmTzro4V5u3D5epW/dir1/dir2/myVideo.mp45a71afc030072d4cd1101) = "3a5"
```

Step 3: Generating Authentication URL

The authentication parameters are concatenated into the original video URL's QueryString to get a video authentication URL:

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&exper=300&us=72d4
```

Smart Acceleration

Last updated : 2025-05-26 16:53:00

Function Introduction

When your site provides dynamic content services or mixed dynamic and static content, the user's request for dynamic content needs an origin-pull request to respond to different resource content according to the user. At this time, due to the differences in the region and operator of the client, the network environment is complex and intricate. When accessing across regions and operators, it may cause slow access requests, high packet loss rates, and other situations for users.

Smart acceleration can adjust and optimize network paths in real-time. After enabling this function, EdgeOne will detect node network latency in real-time, select the best access path through intelligent algorithms, and dynamically adjust resource allocation and utilization according to the real-time network conditions, to help improve user experience and ensure business continuity.

Billing Description

This function is a value-added service. After enabling smart acceleration, the upstream traffic of client users and EdgeOne nodes will be included in the security acceleration traffic fee based on the original billing items, and value-added service fees will be charged according to the number of business requests. For details, please refer to the [Billing Overview](#).

Scenario 1: Enable smart acceleration for all domain names of the site

If your site is all mixed dynamic and static resources or pure dynamic resources, you need to enable smart acceleration for the whole connected site. Please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page.
3. Find the smart acceleration configuration card, which is off by default. Click the **switch** to configure on/off.

Scenario 2: Enable smart acceleration for specified domain names

If only a certain domain name under your site is pure dynamic resources or mixed dynamic and static resources, you need to enable smart acceleration separately for the specified domain name. Please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the Host matching type to match the requests of the specified domain name.
5. Click **Action** > **Select Box**, and in the pop-up action list, select the action as **smart acceleration**, and click the **switch** to turn on/off.
6. Click **Save and Publish** to complete the rule configuration.

Related Reference

What are static resources and what are dynamic resources?

Static resources: When a user accesses a resource multiple times, the same content is returned. For example: images, videos, software installation packages, compressed files, CSS, JavaScript files, and other content that does not change frequently.

Dynamic resources: When a user accesses a resource multiple times, different content is returned. That is, content that needs real-time updates, user interaction, and other dynamic content. For example: API interfaces, `jsp`, `asp`, `php`, `perl`, and `cgi` format files, etc.

Cache Configuration Overview

Last updated : 2025-03-17 17:40:55

After your site is connected to EdgeOne, EdgeOne edge nodes will decide whether to cache the resource files of the client request response based on the cache configuration rules. After the edge node caches the file, when other users initiate the same file request, it can be directly responded by the EdgeOne edge node, effectively avoiding the long link origin-pull situation and responding to the latest file request at a faster speed.

You can customize your site cache according to the following usage scenarios:

Rule Type	Usage Scenario	Function
Custom EdgeOne node caching rules	There are various types of file resources under the site/domain, and it is necessary to customize the caching time of various resources in the node to ensure that users access the latest files while reducing the origin-pull request volume. The site/domain contains dynamic resources, and it is necessary to avoid caching the content of these resources.	EdgeOne Node Cache TTL
	When requesting the same path file, different URLs carry different parameters, request headers, etc., which will point to different files. When requesting the same path file, the parameters, request headers, etc. carried by the URL do not affect the file version and need to point to the same cache file.	Custom Cache Key
cache control when origin response exception	When the origin response is abnormal, it is necessary to protect the origin from further damage while normally responding to customer requests.	Status code caching TTL Offline Caching
Control browser caching	To further improve the web page loading speed and reduce traffic consumption, allow the browser to cache static resources files for a certain period of time.	Browser Cache TTL
Clear the cache	When the cached files in the node have expired or illegal resources are cached, clear the cached resources in the node.	Cache Purge
URL Pre-Warming	When the domain access is just completed or the files are updated, it is necessary to cache the files in advance to the EdgeOne node to improve the acceleration effect and reduce the origin-pull volume during peak periods.	URL Pre-Warming

Cache Prefresh	For files with continuous user access, it is necessary to ensure that the files have continuous cache in the node to avoid concentrated origin-pull after the cache expires. The cache pre-refresh can be used to verify the validity of the files and refresh the cache time.	Cache Prefresh
----------------	--	--------------------------------

If you need to learn more about cache rules, you can refer to the following:

- [Learn about EdgeOne's content caching rules](#)
- [Learn about the role of Cache Key](#)
- [Learn about the role of Vary feature](#)

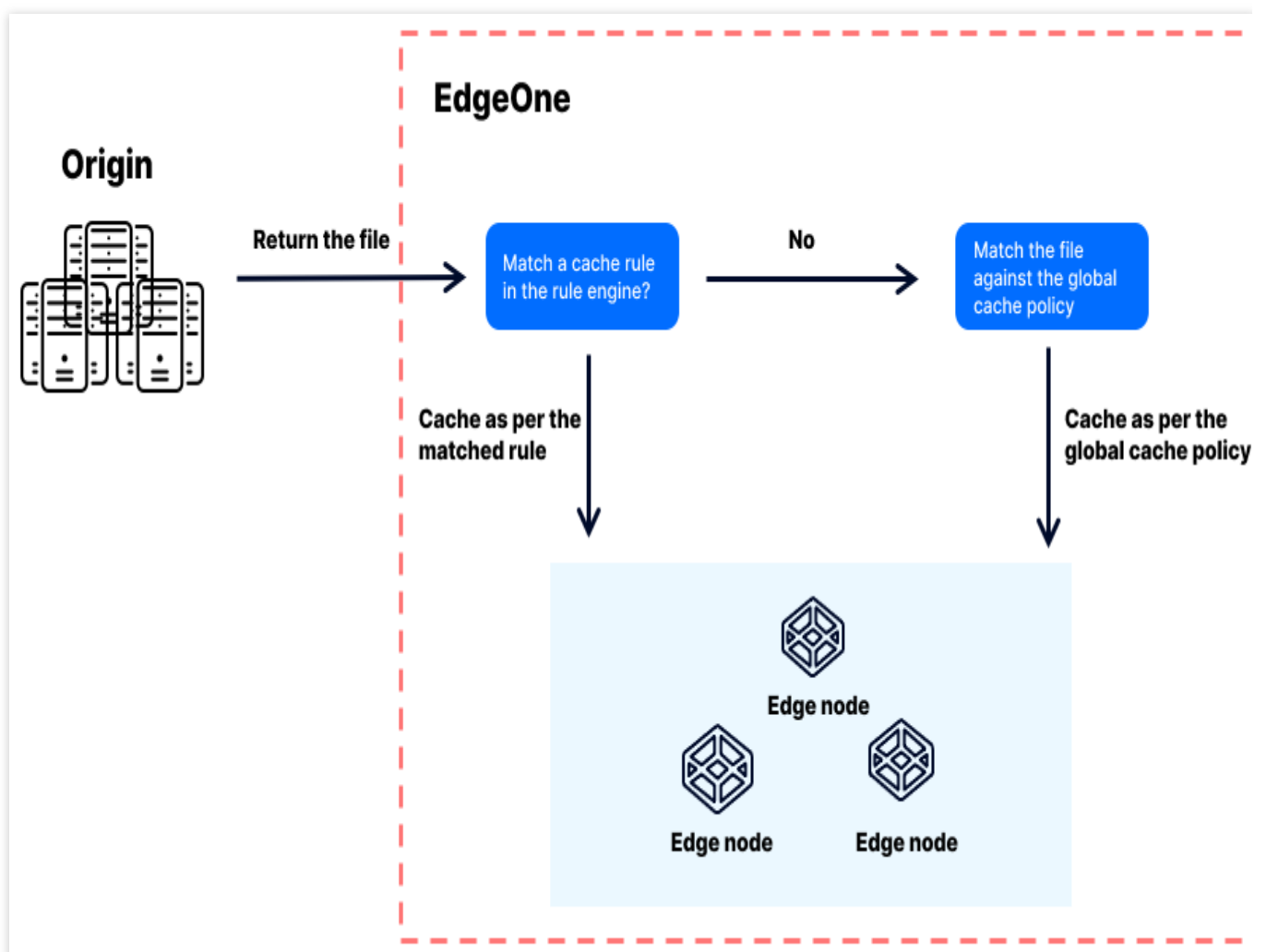
EdgeOne Cache Rules

Content Cache Rules

Last updated : 2023-05-25 15:19:39

Overview

After a client initiates an HTTP request to an EdgeOne node, if the requested file is not found in the cache, EdgeOne initiates a request to the origin to obtain the latest version of the file. After the origin returns the requested file, EdgeOne caches the file based on the default cache policy and your custom rules (See [Node Cache TTL](#). Cache rules take effect in the following order:



Note:

A cache rule takes effect only if the origin returns 200 or 206. If the origin returns 404, the node caches the status code for 10s, and other status codes are not cached.

1. Rules configured in the rule engine are first matches the file against the cache rules in the rule engine from top to bottom. If the file matches a cache rule in the rule engine, it is cached as per that rule.
2. If the file does not match any rule in the rule engine, it is cached as per the global cache policy specified in **Site Acceleration**. EdgeOne uses the global cache policy as the default cache policy. You can modify the default cache policy as needed.

Cache Rules

EdgeOne supports the following three types of cache policies:

Default cache policy: The default cache policy of EdgeOne. The default cache policy determines the cache time of a file on a node based on the `Cache-Control` header and other caching-related headers of the HTTP response.

No-cache policy: Rules are set in the rule engine to specify not to cache specified files or not to cache any files of the site. A no-cache policy is applicable to dynamic or frequently updated files.

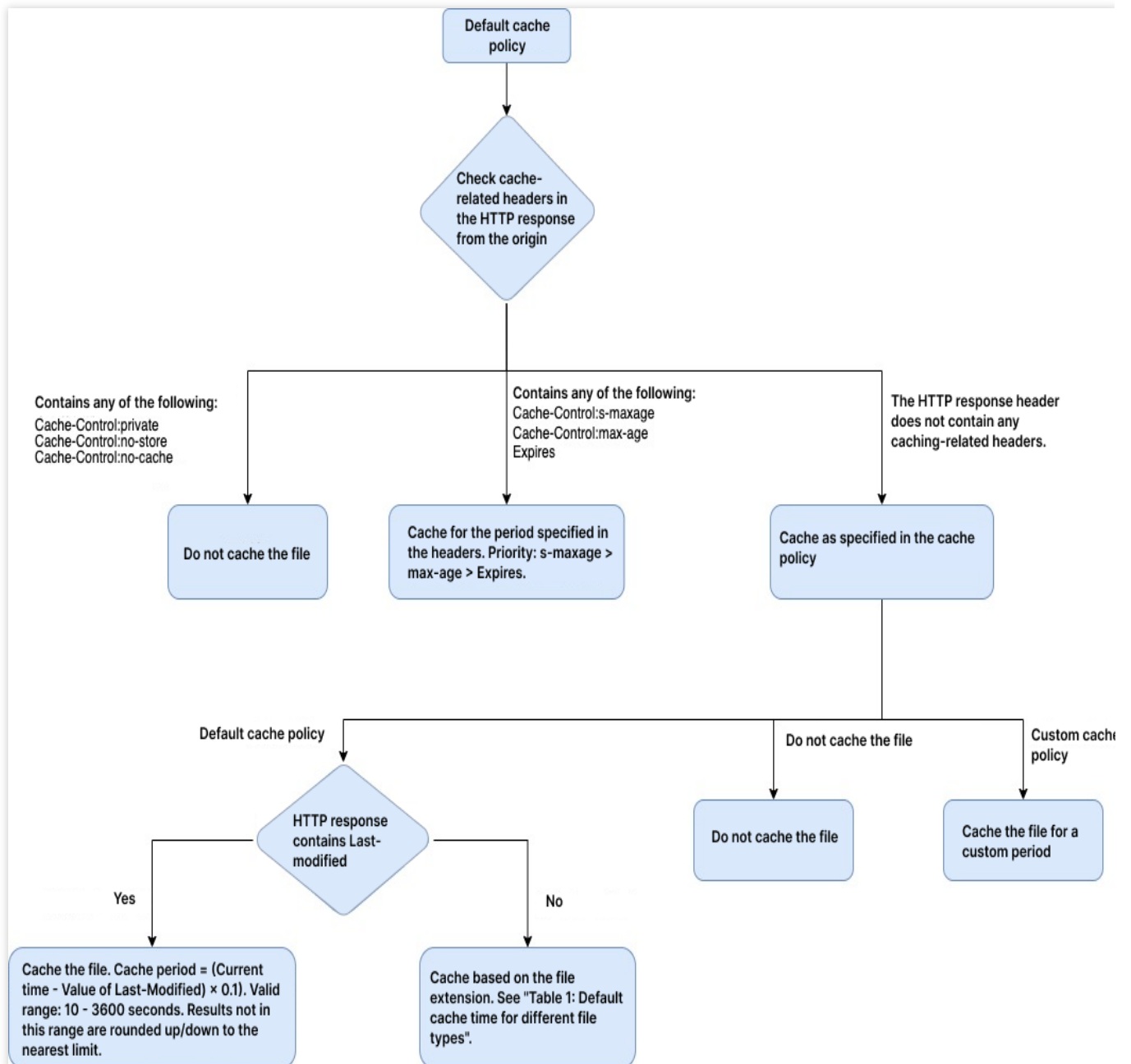
Custom cache policy: A custom cache policy allows you to cache files for a custom period of time.

Note:

EdgeOne supports cold file eviction. If a file cached in an EdgeOne node has not been requested over a long period of time, EdgeOne may remove it from the node cache before the specified cache time expires.

Default cache policy

The following figure describes the default cache policy of EdgeOne:



The default cache policy allows an EdgeOne node to control the caching of the file based on the following cache rules:

1. When the HTTP response contains any of the following not-to-cache headers, the file is not cached:

`Cache-Control:private`

`Cache-Control:no-store`

`Cache-Control:no-cache`

2. When the HTTP response header contains any of the following to-cache headers, the file is cached for the period of time specified in the header:

`Cache-Control:s-maxage`

`Cache-Control:max-age`

`Expires`

If more than one of the preceding response headers exists at the same time, their precedence is as follows: s-maxage > max-age > Expires. The file is cached for the period of time specified by the header with the highest priority.

3. When the HTTP response does not contain any of the preceding caching-related headers, the caching action specified in the rule is performed:

Default cache policy:

If the HTTP response contains the `Last-Modified` header, the cache time is calculated by this formula: (Current time - Value of `Last-Modified`) \times 0.1. If the result ranges from 10 to 3,600 seconds, the result is taken as the cache time. If the result is less than 10 seconds, 10 seconds is taken as the cache time. If the result is greater than 3,600 seconds, 3,600 seconds is taken as the cache time.

If the HTTP response does not contain the `Last-Modified` header, the cache time of a file is determined based on the default cache rules and the file extension. The following table describes the cache time of files with different extensions:

Table 1: Default cache time for different file types

File Type		Extension	Cache Time
Dynamic files		php, aspx, asp, jsp, do, dwr, cgi, fcgi, action, ashx, axd, and json	Do not cache the file
Static files	Images	jpg, png, jpeg, webp, gif, heif, heic, kpg, and ico	2 hours
	Audio/Video	mp4, mp3, m3u8, ts, m4a, avi, m4s, and ogg	
	Webpages	html, js, and css	
	Packages	zip, 7z, tar, br, gz, rar, and bz2	
	Documents	doc, docx, xls, xlsx, pdf, ppt, and pptx	
	Applications	apk, exe, and bin	
	Others	vsv, iso, jar, swf, chunk, and atlas	
Other files		N/A	Do not cache the file

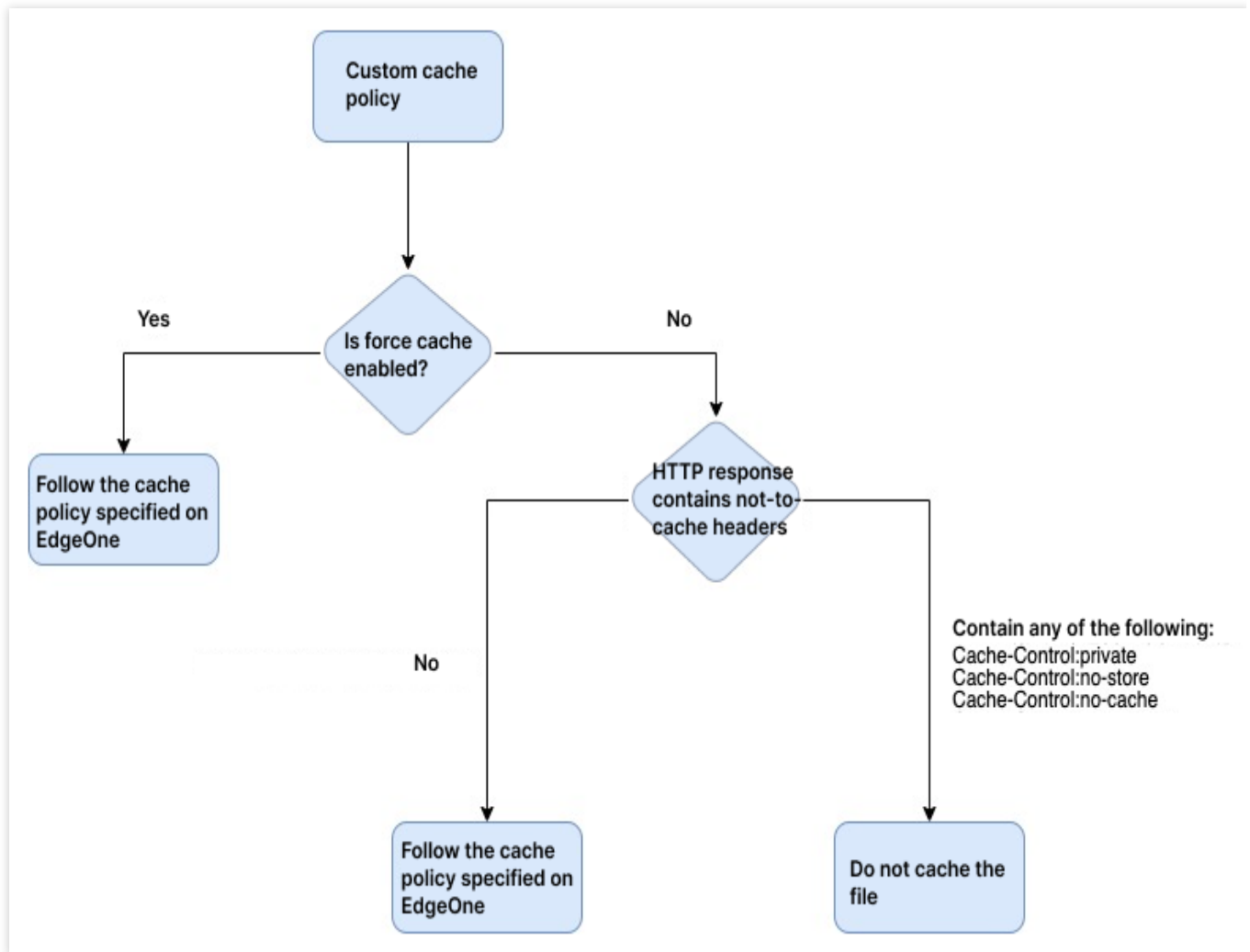
No Cache: If the HTTP response does not contain any of the preceding caching-related headers, the file is not cached.

Custom cache policy: If the HTTP response does not contain any of the preceding caching-related headers, the file is cached for the cache time specified in the custom rule.

No-cache policy

If the no-cache policy is set for the EdgeOne rule engine or the entire site, the file is not cached regardless of whether the HTTP response contains the `Cache-Control` header or other caching-related headers.

Custom cache policy



A custom cache policy allows you to cache a file for a custom period of time, and enable or disable the force cache feature.

Enable force cache: Force cache is enabled by default. If force cache is enabled, EdgeOne caches the file for the custom period of time, regardless of whether the HTTP response contains the `Cache-Control` header or other caching-related headers.

Disable force cache: After you disable force cache, if the HTTP response contains any of the following not-to-cache headers, the file is not cached:

`Cache-Control:private`

`Cache-Control:no-store`

`Cache-Control:no-cache`

If the HTTP response does not contain any of the preceding headers, EdgeOne caches the file for the custom period of time.

Learn More

[Node Cache TTL](#)

[Cache Purge](#)

[URL Pre-Warming](#)

Cache Key Introduction

Last updated : 2023-08-07 16:30:58

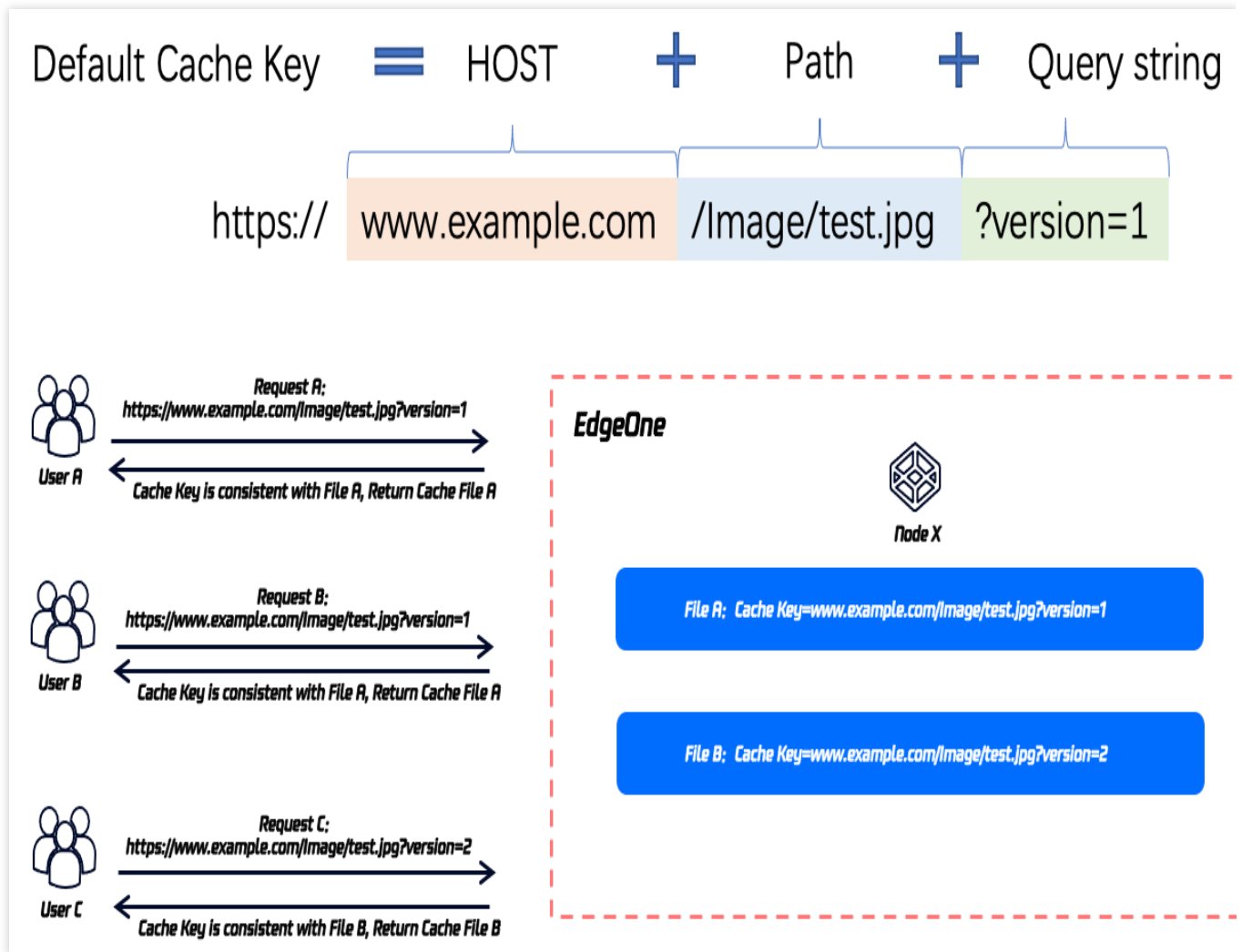
What is Cache Key

Cache Key is used to determine whether the file resources accessed by users hit the edge caching content of EdgeOne, and it is the unique identifier of cached resources within the node. Cache hits can help your site: Reduce origin-pull requests and lower the bandwidth consumption of the origin.

Improve the access request speed of users.

The working principle of Cache Key is as follows:

When a file is cached in the EdgeOne edge node, the node will generate a corresponding Cache Key identifier for the file according to the Cache Key rules. By default, the Cache Key is calculated based on the client request URL and query string. When other clients initiate an HTTP request to the edge node, the node will compare the HTTP request with the Cache Key of all cached resources in the node according to the Cache Key calculation rule. If they are consistent, the corresponding cached resources will be directly responded to the client, which is a cache hit.



Cache Key's application scenarios

Cache Key is used for EdgeOne nodes to establish multi-version caching content for different versions of files. Even if the client requests through the same path, the correct user file can be responded through the calculation rules of Cache Key.

You can understand how to correctly configure Cache Key to help you correctly match the requested cache files and reduce the origin-pull rate through the following scenario examples.

For example, User A and User B have the following requests:

Request A: `https://www.example.com/Image/test.jpg?version=1&time=1651752743`

Request B: `https://www.example.com/Image/test.jpg?version=2&time=1651758319`

Scenario 1: The file paths accessed by users are completely identical, but there are version differentiations according to `version` parameter carried in the query string. The above requests are two different images. In this case,

`version` parameter that affects the file version should be retained in the Cache Key to ensure that the node can correctly cache and respond to the corresponding file content.

Scenario 2: The content of the query string in the user's access URL does not affect the file content. The files corresponding to the above requests are the same and do not affect the file version. In this case, all query strings should be ignored in the Cache Key calculation to improve the hit rate of files in the node and reduce the origin-pull requests.

Customizable Cache Key content and effective rules

EdgeOne allows users to customize Cache Key rules, supports configuring query strings, HTTP request headers, or cookies to distinguish caches. You can learn how to configure custom Cache Key rules through [Custom Cache Key](#).

Note :

1. The calculation of Cache Key is based on the HTTP request content initiated by the client to the node. The origin URL rewriting, origin follow redirect, origin HTTP request header, and Rewrite access URL do not affect the calculation of Cache Key.
2. When Token authentication is configured in EdgeOne, the authentication content will not participate in the Cache Key calculation.
3. When image processing parameters are enabled in EdgeOne, the image processing parameters carried in the request will participate in the Cache Key calculation by default.

Query string

The query string refers to the string parameters after the " ? " in the request URL (including one or multiple parameters, separated by " & "), for example, `color=blue&size=large` in

```
https://www.example.com/images/example.jpg?color=blue&size=large .
```

EdgeOne supports customizing the retention of specified query string content to distinguish caches.

Note :

When retaining the query string as the Cache Key, if the position order of the parameters changes, the Cache Key will also change.

For example, when clients initiate the following requests separately:

Request A: `https://www.example.com/Image/test.jpg?version=1&type=a`

Request B: `https://www.example.com/Image/test.jpg?version=2&type=a`

Request C: `https://www.example.com/Image/test.jpg?type=a&version=1`

Configuration	Cache behavior
Query string configuration is to retain all	The parameter content carried by Request A and Request B is different, corresponding to different cache versions. The parameter content of

	Request A and Request C is consistent, but the order is different, corresponding to different cache versions.
Query string configuration is to ignore all	Requests A, B, and C all correspond to the same cache version.
Query string configuration is to retain the specified parameter Type	The parameter order and content of Request A and Request B are completely identical, corresponding to the same cache version; the parameter content of Request B and Request C is the same, but the order is different, corresponding to different cache versions.
Query string configuration is to ignore the specified parameter Type	The remaining parameter content of Request A and Request B is not consistent, corresponding to different cache versions; the remaining parameter content of Request A and Request C is consistent, but the order is not consistent, corresponding to different cache versions.

HTTP request header

EdgeOne supports adding specified HTTP request headers to the Cache Key calculation. EdgeOne edge nodes will establish different cache versions based on the request header content. The order change of the request header does not affect the calculation of the Cache Key.

For example, specify the HTTP request header `User-Agent` to be included in the Cache Key calculation. The URL and parameter content of the following Request A and Request B are consistent, but the `User-Agent` header content is not consistent, corresponding to different cache versions.

Request A: `https://www.example.com/Image/test.jpg?version=1&type=a` , with `User-Agent : chrome`

Request B: `https://www.example.com/Image/test.jpg?version=1&type=a` , with `User-Agent : ios`

Cookie

EdgeOne supports adding specified parameters in the Cookie to the Cache Key calculation, distinguishing cache versions based on Cookie parameters and content. When multiple parameters in the Cookie participate in the Cache Key calculation, the order change of the parameters does not affect the Cache Key calculation.

For example, specify the `User` parameter in the Cookie to be included in the Cache Key calculation. The parameter content of the following Request A and Request B is the same, corresponding to the same cache, and the parameter content of Request A and Request C is different, corresponding to different cache versions.

Request A: `https://www.example.com/Image/test.jpg?version=1&type=a` , with `Cookie: User=A; ID=1` .

Request B: `https://www.example.com/Image/test.jpg?version=1&type=a` , with `Cookie: User=A; ID=2` .

Request C: `https://www.example.com/Image/test.jpg?version=1&type=a` , with `Cookie:`
`User=B; ID=1` .

Learn more

[Learn about EdgeOne content caching rules](#)

[How to configure custom Cache Key](#)

[How to configure node caching rules](#)

[How to purge node cache resources](#)

Vary Feature

Last updated : 2024-08-26 16:28:37

Support for Vary

EdgeOne supports the Vary feature, which is currently disabled by default. To enable it, please [contact us](#). Once the feature is enabled, you can simply add a Vary header to the response header in the origin response without any configuration. For more information about the standards of a Vary header, see [Vary](#).

Note:

If your domain name was created before 00:00:00 on June 13, 2024, the Vary feature was enabled and supported for the domain name by default.

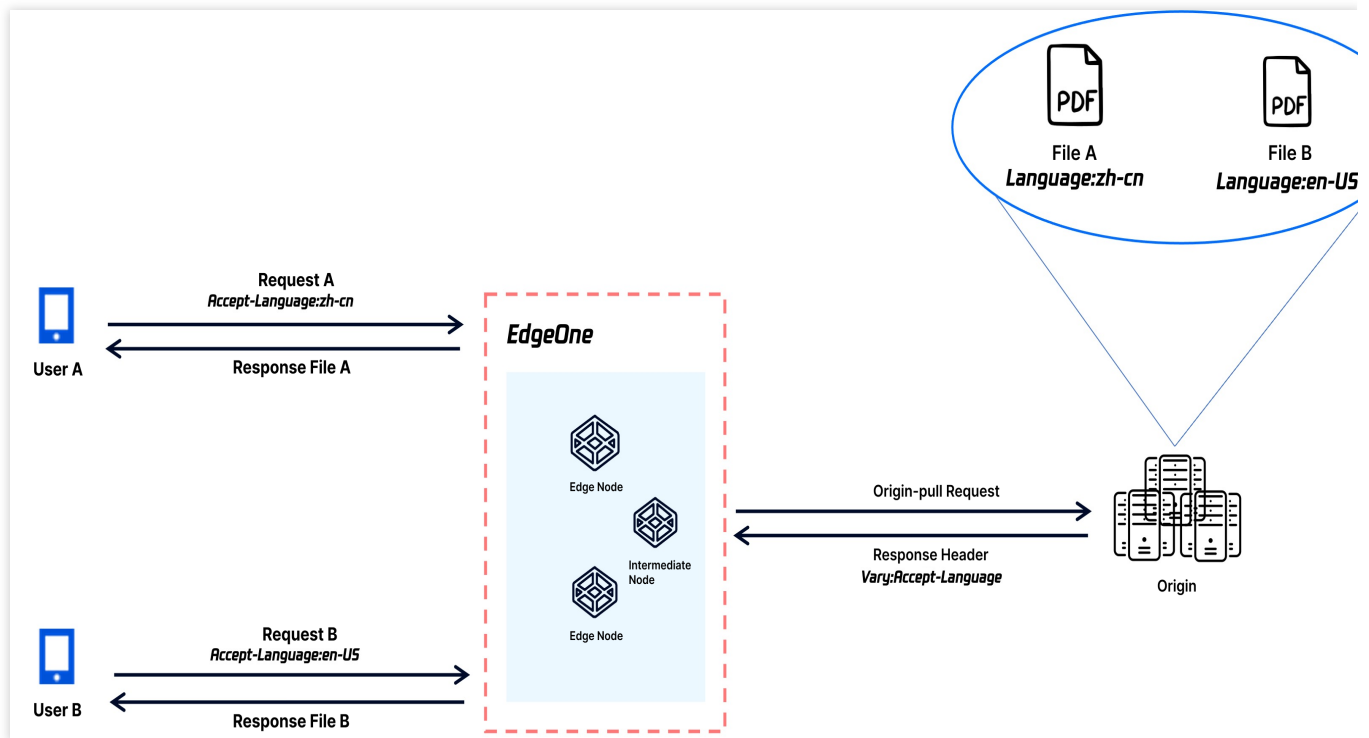
What Is the Vary Feature?

Vary is an HTTP response header newly added in HTTP/1.1. When a client uses the same URL to initiate requests to an origin server, if the origin server has response files of different versions, requested resources may be cached by an intermediary cache system, such as the browser cache and content distribution network (CDN) cache, and the origin server may fail to respond to requests by scenario. To prevent this condition, the origin server can use a Vary header in the HTTP response to notify the intermediary cache system of the specific request header for distinguishing the version of the cached content.

For example, if the client requests are all targeting at `https://www.example.com/test.pdf`, and the origin server uses `Vary: Accept-Language` in the HTTP response header to distinguish the client language, EdgeOne will generate caches of different versions based on the `Accept-Language` content specified in the client requests.

When User A initiates a request with the URL `https://www.example.com/test.pdf`, and the request contains the request header `Accept-Language: zh-cn`, EdgeOne responds to the request with File A.

When User B initiates a request with the URL `https://www.example.com/test.pdf`, and the request contains the request header `Accept-Language: en-US`, EdgeOne responds to the request with File B.

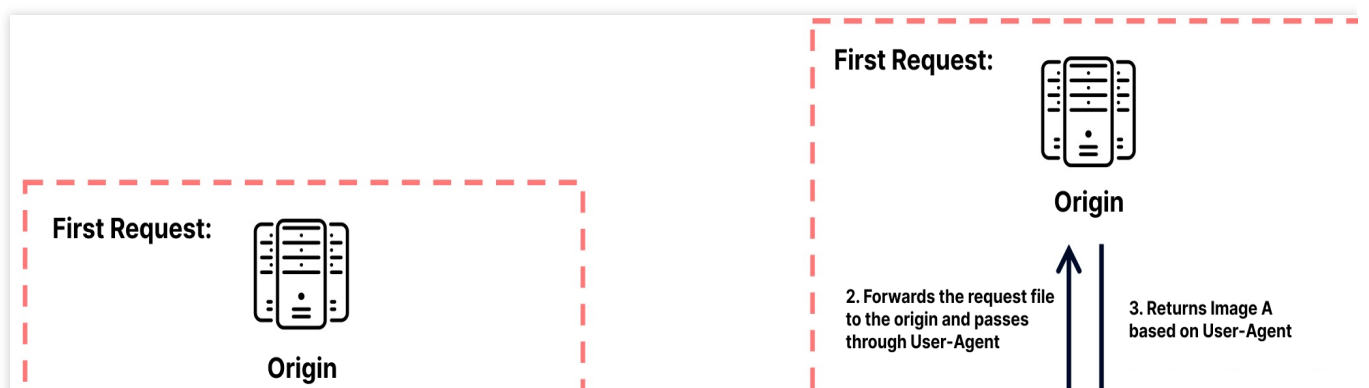


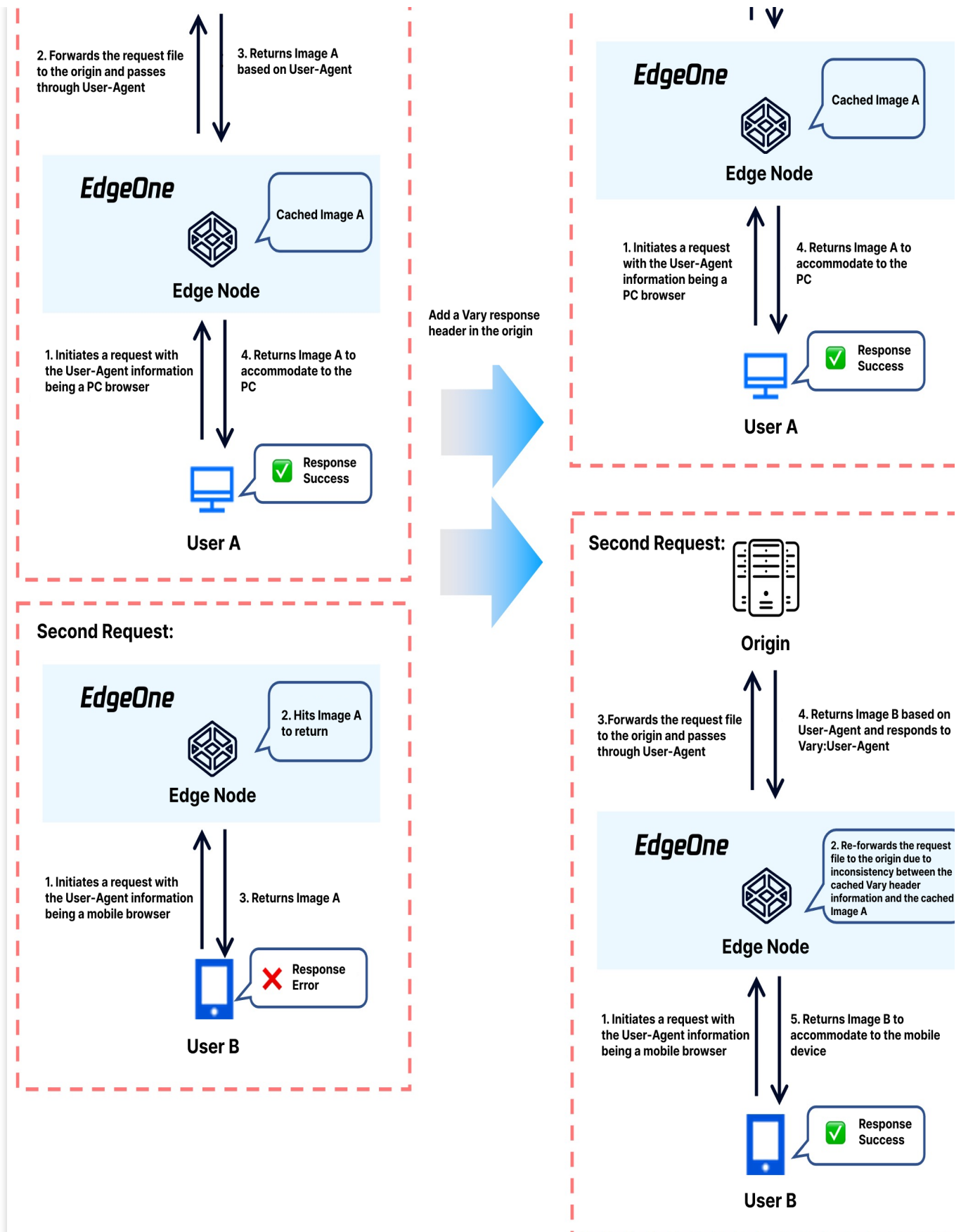
Application Scenarios of Vary

You can flexibly use the Vary header to control the files requiring different cached file versions according to the HTTP request header and solve issues in the following scenarios:

Scenario I: Distinguishing response files by client type when the request URLs are the same

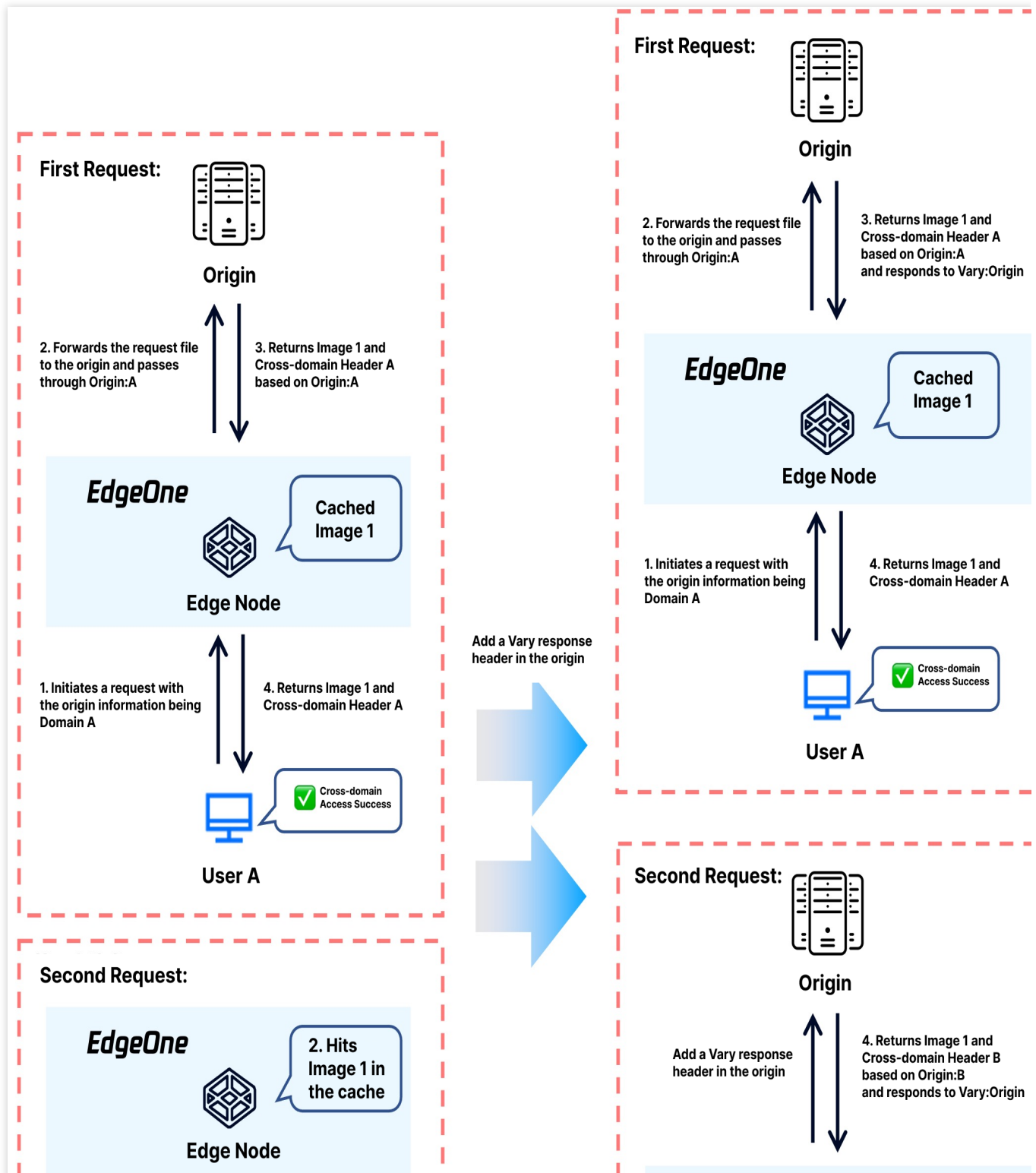
When a website serves user access from both PCs and mobile devices at the same URL for user convenience, due to the different screen resolutions between PCs and mobile devices, the frontend needs to adapt the image content to the access source. In this case, the `User-Agent` header carried in the user requests is used to distinguish the access source type, ensuring requests from PCs are responded with Image A and requests from mobile devices are responded with Image B. In this case, the origin server can add the `Vary:User-Agent` header to the responses to instruct Edge nodes to distinguish the cached versions according to the `User-Agent` header in the user requests.

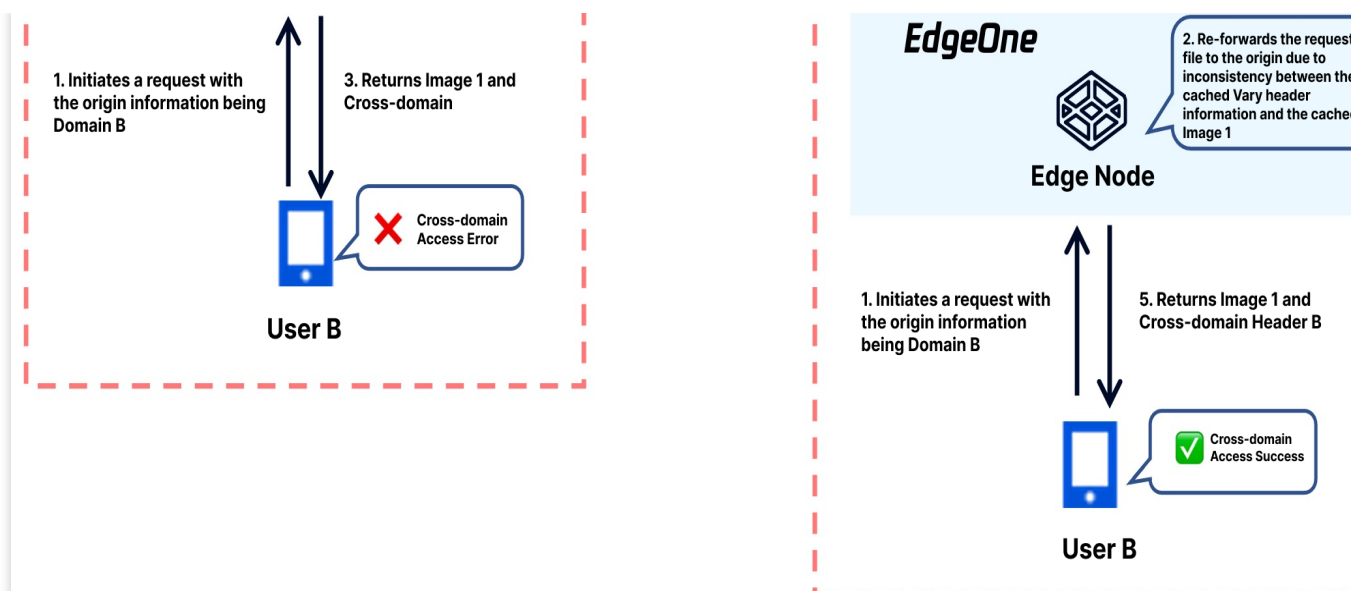




Scenario II: Responding to cross-domain access according to the access source when the request URLs are the same

When the same image file in a page cache is referenced by both Domain A and Domain B, to avoid cross-domain errors, `Access-Control-Allow-Origin` is usually added to the response file in the origin server to allow cross-domain access. However, when the current file is cached, both cross-domain headers may be cached or a cross-domain access error may occur. In this case, the origin server can add the `Vary:origin` header to instruct Edge nodes to distinguish the cached versions according to the `Origin` header in the user requests, while responding the `Access-Control-Allow-Origin` cross-domain header.





Cache Configuration

Custom Cache Key

Last updated : 2024-08-26 11:38:41

Feature Introduction

When you need to point the Request URL of the same path to different files based on request parameters, cookies, or HTTP request headers, or point the Request URL with different parameters to the same file, the custom Cache Key supports customizing the Cache Key identification of resources in the node, including concatenating query strings, concatenating HTTP headers or Cookie information, etc., so that the Request URL can correctly obtain the corresponding cached resources according to different scenarios. You can learn what a Cache Key is through the [Cache Key Introduction](#).

Usage Scenarios

Scenario One: The file paths accessed by users are exactly the same, but there will be version differences based on the carried query strings, HTTP request headers, and Cookie contents. The cache key of this type of file can be adjusted by customizing the Cache Key.

Scenario Two: The content of the query string in the user's accessed URL does not affect the file content, and the files corresponding to the above requests are consistent and do not affect the file version. The cache key of this type of file can be adjusted by customizing the Cache Key.

Directions

Scenario One: Configure custom Cache Key for all domain names of the site

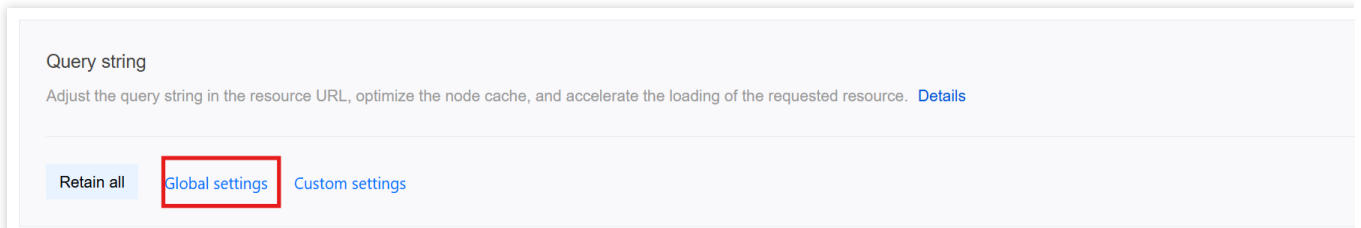
If you need to configure a custom Cache Key for the entire connected site, or as a site-level fallback configuration, please refer to the following steps:

1. Log in to [the EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the Site Global Configuration page. In the right-hand navigation bar, click **Cache Configuration**.

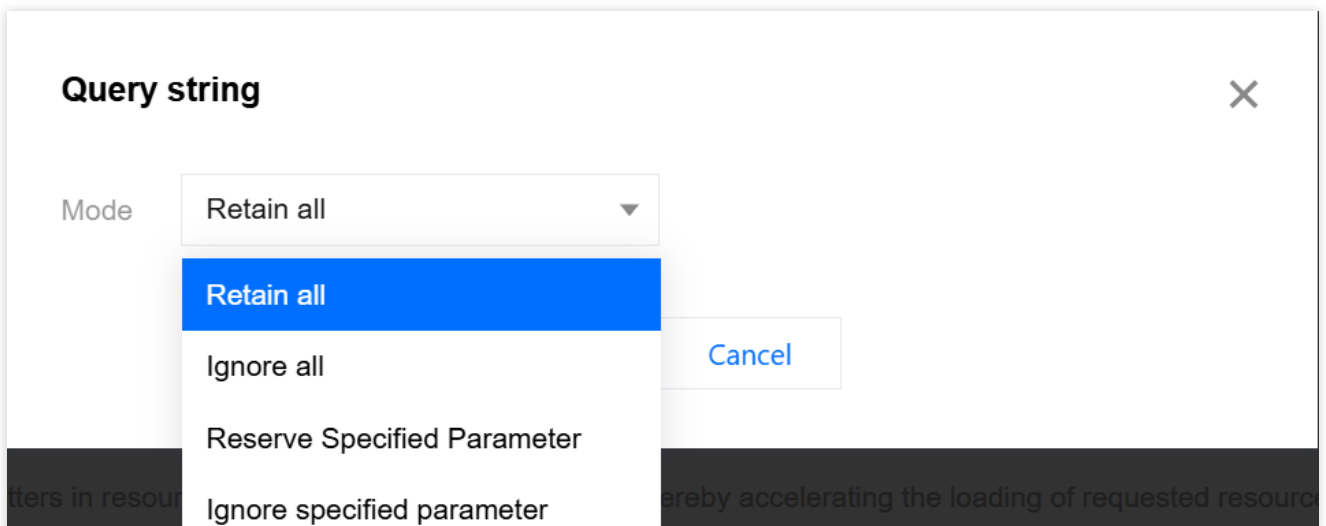
Note :

Global configuration can only configure query strings and case-insensitive. For more comprehensive custom Cache Key configuration options, please refer to the [configuration steps in Scenario Two's rule engine](#).

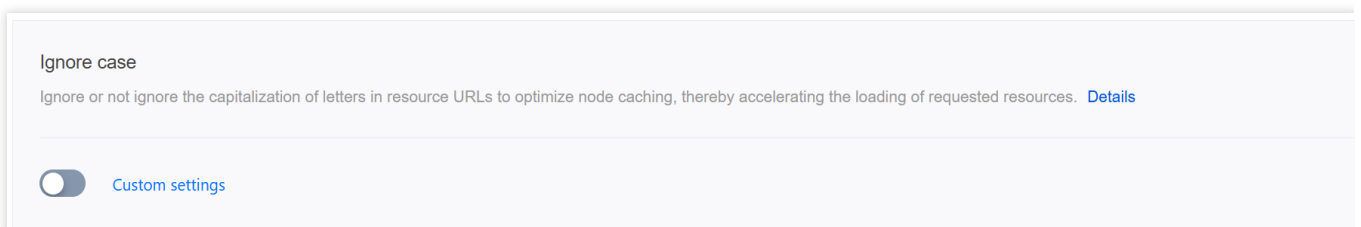
On the cache configuration page, locate the Query string card and click **Global settings** to proceed with the configuration.



The default configuration is to retain all, that is, to retain all query parameters of the original Request URL as the Cache Key. Other options are available: a. Ignore all: Ignore the entire query string; b. Reserve Specified Parameter: Only retain the specified parameters in the query string; c. Ignore specified parameter: Only ignore the specified parameters in the query string.



On the cache configuration page, find the Case-Insensitive Card. The default configuration is disabling the Ignore case. Even if the URL's content is the same, but the letter case is different, it will be regarded as a different Cache Key. Click the **Global Enable** switch to turn on the Ignore case, then different letter cases will be regarded as the same Cache Key.



Scenario Two: Configure custom Cache Key for specific domain names, paths, or file extensions, etc.

If you need to configure a custom Cache Key rule for the `www.example.com` domain under the site `example.com` to ignore all query strings, concatenate the HTTP request header `My-Client-Header`, and

use the parameters `name1` and `name2` in the Cookie as the Cache Key, you can refer to the following steps for configuration:

Directions

1. Log in to [the EdgeOne console](#), and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select Host as the matching type and configure it as `www.example.com`.
5. Click on the **Action**, and in the pop-up operation list, select the operation as **Custom Cache Key**;
6. Click on Add under the Type to add the custom Cache Key type. In this example scenario, add Query String, HTTP Request Header, and Cookie for configuration and fill in the corresponding content. The complete rule configuration is as follows:

7. Click **Save and Publish** to complete the rule configuration.

Effective Example

After the configuration is completed, the Cache Key is composed of URL+My-Client-Header+Cookie: Ignore all query strings, concatenate `My-Client-Header`, and retain the specified parameters in the Cookie.

Then Client A request:

URL: `https://www.example.com/path/demo.jpg?key1=value1&key2=value2`

HTTP request header: Contains `My-Client-Header: fruit`

Cookie: `name1=yummy;name2=tasty;name3=strawberry`

And Client B request:

URL: `http://www.example.com/path/demo.JPG?key1=value1&key2=value2&key3=value3`

HTTP request header: Contains `My-Client-Header:fruit`

Cookie: `name1=yummy;name2=tasty;name3=blueberry`

And Client C request:

URL: `http://www.example.com/path/demo.JPG?`

`key1=value1&key2=value2&key3=value3&key4=value4`

HTTP request header: Contains `My-Client-Header:sea`

Cookie: `name1=yummy;name2=tasty;name3=fish`

Requests A and B will hit the same cached resource, while C will hit another cached resource.

Related Reference

Description of supported header names:

Header Type	Description
Custom Header	Custom Headers. Name: 1 - 100 characters, consisting of digits 0 - 9, letters a - z, A - Z, and the special character -. Value: 1 - 1000 characters, Chinese is not supported.
Preset Header	Aggregated headers based on client User-Agent information: Client Device Type: <code>EO-Client-Device</code> Values: <code>Mobile</code> , <code>Desktop</code> , <code>SmartTV</code> , <code>Tablet</code> , or <code>Others</code> Client Operating System: <code>EO-Client-OS</code> Values: <code>Android</code> , <code>iOS</code> , <code>Windows</code> , <code>MacOS</code> , <code>Linux</code> , or <code>Others</code> Client Browser Type: <code>EO-Client-Browser</code> Values: <code>Chrome</code> , <code>Safari</code> , <code>Firefox</code> , <code>IE</code> , or <code>Others</code>

Node Cache TTL

Last updated : 2025-05-07 09:42:13

Feature Introduction

Node Cache TTL is used to determine whether resources are cached in EdgeOne nodes and the cache duration within the nodes. When users request expired or uncached files, the node will not directly respond to the user's request, but will go back to the origin to obtain the latest resources for response, and decide whether to cache them in EdgeOne according to the cache rules. Caching files and allowing user requests to hit can help you:

Reduce the number of origin-pull requests and lower the bandwidth consumption of the origin.

Improve the speed of user access requests.

You can customize the cache time for different resources according to your business needs, optimize the cache strategy for different resources, and improve the loading speed of requested resources. For more information on cache instructions, please view [EdgeOne Content Cache Rules](#).

Note :

1. If the resources at the origin are updated and you need to update the node cache immediately, you can use the [Cache Purge](#) function to actively purge the unexpired old cache, ensuring that subsequent requests can obtain the latest resources from the origin.
2. Please do not cache dynamic resources in edge nodes to avoid users accessing incorrect content.
3. After the file is cached in the EdgeOne node, the platform has a hot and cold elimination mechanism. If the current cached file has not been requested for a long time, it may be deleted from the node cache before the maximum cache time is reached.

Directions

Scenario 1: Configure Node Cache TTL for all domain names of the site

If you need to configure the same Node Cache TTL for the entire connected site, or as a site-wide fallback configuration, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the Site Global Configuration page. In the right-hand navigation bar, click **Cache Configuration**.
3. Click on the Global settings to configure. For detailed configuration instructions, please refer to [EdgeOne Content Cache Rules](#).

Default configuration: Follow origin `Cache-Control` , and follow [EdgeOne default cache policy](#) when the origin has no `Cache-Control` .

Scenario 2: Configure Node Cache TTL for specified domain names, paths, or file extensions, etc.

If you need to configure different Node Cache TTLs for different domain names, paths, or file extensions, such as not caching files with `php/jsp/asp/aspx` extensions under the `www.example.com` domain, and caching files with `jpg/png/gif/bmp/svg/webp` extensions for 30 days, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, first select the matching type as HOST, with the value of `www.example.com` as the outermost matching condition, and click on **Add IF**.
5. In the newly added IF condition, select the matching type as File extension, add `php/jsp/asp/aspx` extensions, click on the **Action**, and in the pop-up operation list, select the operation as **Node Cache TTL**, and configure it as uncached.
6. Repeat the above steps, add another IF condition, add `jpg/png/gif/bmp/svg/webp` extensions, and configure it as cached for 30 days.
7. Click **Save and Publish** to complete the rule configuration.

Related Reference

[How do I tell whether user access has hit the EdgeOne cache?](#)

Status Code Cache TTL

Last updated : 2025-05-07 09:41:17

Function Introduction

When EdgeOne retrieves resources from the origin, if the origin successfully responds with the resources, EdgeOne will respond to the client request and cache it in EdgeOne for direct response next time. If the origin responds with an exception status code such as 4xx or 5xx, EdgeOne cannot obtain the resources, and the next request will still trigger a follow origin, which may put significant pressure on the origin. By configuring the status code cache TTL, EdgeOne can directly respond with the exception status code within the cache time, instead of triggering a follow origin for all requests, which can mitigate the pressure on the origin and improve the response speed.

Currently, the following status codes can be configured:

4xx: 400, 401, 403, 404, 405, 407, 414.

5xx: 500, 501, 502, 503, 504, 509, 514.

Note :

EdgeOne caches the 404 status code by default for 10 seconds.

The prerequisite for the status code cache to take effect is that the resource can be cached in the node according to the node cache TTL configuration. If the resource is not cached in the node, the status code cache will not be triggered.

Directions

Scenario 1: Configure status code cache TTL for all domain names of a site

If you need to configure the status code cache TTL for the whole connected site, or as a fallback configuration for the site level, please refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.
3. On the rule engine page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as All.
5. Click the **Action**, and in the pop-up operation list, select the operation as status code cache, and configure the corresponding cache status code and cache time.

6. Click **save and publish** to complete the rule configuration.

Scenario 2: Configure status code cache TTL for specified domain names, paths, or file extensions

If you need to configure different status code cache TTL for different domain names, paths, or file extensions, for example, configure the status code cache TTL for the `www.example.com` domain under the `example.com` site, please refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.
3. On the rule engine page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select Host as the matching type and configure it as `www.example.com`.
5. Click the **Action**, and in the pop-up operation list, select the operation as **status code cache**, and configure the corresponding cache status code and cache time.
6. Click **save and publish** to complete the rule configuration.

Browser Cache TTL

Last updated : 2024-08-26 11:38:41

Feature Introduction

The Client browser cache TTL is the cache duration of resources in the browser, which by default follows the origin's `Cache-Control` headers. You can control the cache duration of resources in the browser by configuring EdgeOne's browser cache TTL without modifying the origin configuration.

EdgeOne implements browser cache TTL by setting the `Cache-Control` response headers when responding to clients. The following configurations are supported:

Follow Origin: Follow the origin's Cache-Control; if the origin does not have Cache-Control, no changes will be made;
No Cache: Regardless of whether the origin carries Cache-Control, the browser will be controlled not to cache files;
Custom Time: Regardless of whether the origin carries Cache-Control, the max-age will be modified to the specified cache time.

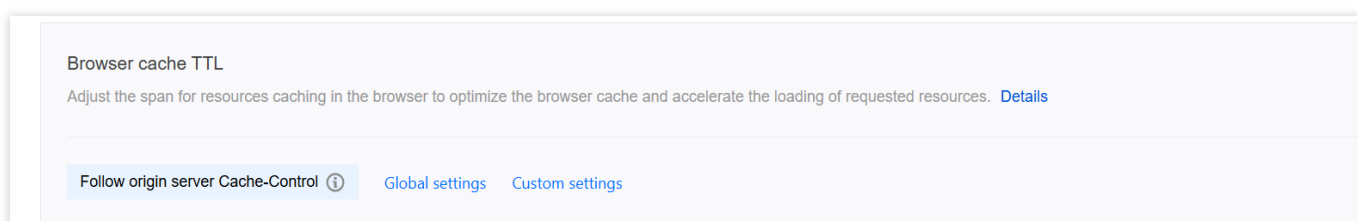
You can control the cache duration of resources in the browser by adjusting the browser cache TTL, optimizing the cache strategy for different resources, and improving the loading speed of requested resources.

Directions

Scenario One: Configure browser cache TTL for all domains of the site

If you need to configure the same browser cache TTL for the whole connected site, or as a site-wide fallback configuration, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the Site Details page, click **Site Acceleration** to enter the Global Configuration page. In the right-hand navigation bar, click **Cache Configuration**.
3. Find the Browser Cache TTL card, click **Global Site Settings** to modify it.

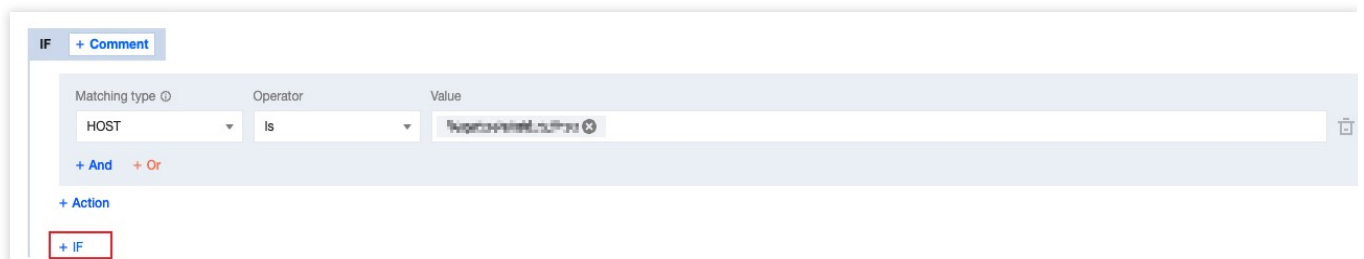


Default Configuration: Supports following the origin's `Cache-Control` ; if the origin does not have `Cache-Control` , no caching will be done.

Scenario Two: Configure browser cache TTL for specific domains, paths, or file extensions, etc.

If you need to configure different browser cache TTLs for different domains, paths, or file extensions, such as not caching files with `php/jsp/asp/aspx` extensions for the `www.example.com` domain, and caching files with `jpg/png/gif/bmp/svg/webp` extensions for 1 hour, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. In the rule editing page, first select the matching type as HOST, with the value of `www.example.com` as the outermost matching condition, and then click `Add IF;`



5. In the newly added IF condition, select the matching type as File extension, add `php/jsp/asp/aspx` extensions, click on **Action**, and in the pop-up operation list, select the operation as **browser cache TTL**, and configure it as No Cache.
6. Repeat the above steps, add another IF condition, add `jpg/png/gif/bmp/svg/webp` extensions, and configure it as cache for 1 hour.
7. After the configuration, the rule is as follows, click **Save and Publish** to complete the rule configuration.

IF

+ Comment

Matching type

HOST

Operator

Is

Value

+ And

+ Or

+ Action

IF

+ Comment

Matching type

File extension

Operator

Is

Value

php jsp asp aspx

Ignore case

+ And

+ Or

Action

Browser cache TTL

Behavior

Do not cache

+ Add

ELSE IF

Matching type

File extension

Operator

Is

Value

jpg png gif bmp svg webp

Ignore case

+ And

+ Or

Action

Browser cache TTL

Behavior

Custom TTL

Time

1

hours

+ Add

+ IF

Offline Caching

Last updated : 2024-08-26 11:38:41

Function Introduction

By default, if EdgeOne cannot establish a connection with the origin when following the origin to obtain resources, it will respond with an error code. After enabling offline caching, when EdgeOne cannot establish a connection with the origin, it can use the resources cached in EdgeOne (even if the resources have expired) until the origin recovers the connection. This can effectively ensure the availability and continuity of the business and improve the user experience.

Note :

If there is no cache available in EdgeOne, it will respond with an error code.

Usage Scenarios

Unstable origin: If your origin server is prone to failures or instability, enabling offline caching can provide a better user experience during origin failures. Even if the cached resources have expired, the service can still be provided to users, avoiding the situation where users cannot access the site when the origin fails.

Critical business assurance: For some critical businesses, you may want to ensure that users can still access key content on the website or application when the origin has issues. Enabling offline caching can ensure that users can still access critical resources when the origin fails, ensuring business continuity.

Avoid sudden traffic impact: In some cases, the origin may be subject to sudden traffic surges, causing server overload or crashes. Enabling offline caching can continue to provide services to users during origin failures, mitigate the pressure on the origin, and help the origin recover to normal operation.

Directions

Scenario 1: Configure offline caching for all domain names of the site

If you need to enable/disable offline caching for the whole connected site, or as a site-level fallback configuration, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the Site Global Configuration page. In the right-hand navigation bar, click **Cache Configuration**.
3. Find the offline caching card and click **Switch** to enable it.

Offline cache

When your origin server fails, you cannot obtain resources through origin-pull. But after enabling permanent cache, you can use the cached resources in the node (even if the resources have expired) until the origin server recovers. [Details](#)



Custom settings

Default state: Enabled. If disabled, when the origin fails, i.e., it cannot follow the origin to obtain resources normally, the node will pass the origin response to the client request.

Scenario 2: Configure offline caching for specific domain names, paths, or file extensions, etc.

If you need to configure different offline caching for different domain names, paths, or file extensions, etc., for example, enable offline caching for the `www.example.com` domain under the `example.com` site, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the Rule Editing page, select Host as the matching type and configure it as `www.example.com`.
5. Click on the **Action**, and in the pop-up operation list, select the operation as **Offline Cache** and turn on the switch.

The screenshot shows the 'Rule Engine' configuration interface. It features a table with columns for 'Matching type', 'Operator', and 'Value'. The first row has 'HOST' as the matching type, 'is' as the operator, and 'www.example.com' as the value. Below this, there is a section for 'Action' with a toggle switch for 'Offline cache' which is currently turned on. The interface also includes buttons for '+ Comment', '+ And', '+ Or', '+ Action', and '+ IF'.

6. Click **Save and Publish** to complete the rule configuration.

Cache Prefresh

Last updated : 2024-08-26 11:38:41

Feature Introduction

After the cache resources expire within the EdgeOne node, EdgeOne will follow the origin to obtain the latest resource files when receiving the corresponding client requests, which may cause a large increase in origin-pull requests during peak periods. The cache pre-refresh function can verify the validity of cache resources before they expire, without waiting for expiration, which helps maintain the real-time nature of resources and respond to requests more quickly. The cache pre-refresh time can be configured according to the percentage of the file cache TTL.

Usage Scenarios

Since the cache pre-refresh function can verify the validity of resources in advance, it is suggested to use it in scenarios where content needs to be frequently updated or user experience is highly demanded:

High Real-time Requirements: For content that needs to be updated quickly, such as news, event pages, etc., customers hope that users can obtain the latest resources when requesting. By enabling the cache pre-refresh function, the node verifies and updates the cache before the resources expire, ensuring that users can obtain relatively new resources when accessing, thus avoiding additional waiting time when users request and improving user experience.

Reduce Origin-pull Pressure: For some hotspot resources, a large number of origin-pull requests may be triggered after expiration. Enabling the cache pre-refresh function can advance these origin-pull requests, reducing the concentration of a large number of origin-pull requests when resources expire, thereby reducing origin-pull pressure.

Directions

Scenario One: Configure cache pre-refresh for all domain names of the site

If you need to configure the same cache pre-refresh for the whole connected site, or as a site-level fallback configuration, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the Site Global Configuration page. In the right-hand navigation bar, click **Cache Configuration**.
3. Locate the Cache Pre-refresh card, click **Switch**, and enter the percentage value for the pre-refresh time in the pop-up confirmation box.

Cache prerefresh

Validate cached resources via origin-pull before expiry to speed up your site. [Details](#)



Custom settings

Configuration state: Default is enabled, can be turned off by clicking the slider.

Pre-refresh Time: The percentage of the node cache TTL, can enter an integer between 1-99. Default is 90%.

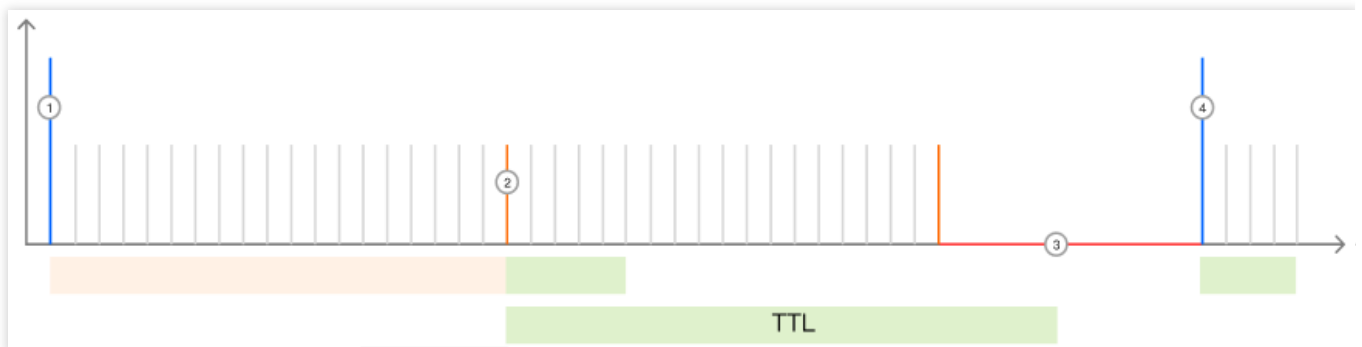
Scenario Two: Configure cache pre-refresh for specific domain names, paths, or file extensions, etc.

If you need to configure different cache pre-refresh for different domain names, paths, or file extensions, etc., for example: Configure a more advanced pre-refresh time - 60% for the `www.example.com` domain under the `example.com` site. Please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select Host as the matching type and configure it as `www.example.com`.
5. Click on the **Action**, and in the pop-up operation list, select the operation as cache pre-refresh, and configure it as 60% of the TTL.
6. The complete configuration is shown below, click **Save and Publish** to complete the rule configuration.

The screenshot shows the Rule Engine configuration interface. It features a table with columns for Matching type, Operator, and Value. The first row is configured with 'HOST' as the Matching type, 'Is' as the Operator, and 'www.example.com' as the Value. Below this, there are buttons for '+ And' and '+ Or'. The second row is for the Action, with 'Cache prerefresh' selected. It includes a 'Prefresh interval' section with a dropdown for 'TTL', a minus sign, the value '60', a plus sign, and a percentage sign. To the right of this is an 'On/Off' toggle switch, which is currently turned on. At the bottom, there are buttons for '+ Action' and '+ IF'.

Attachment: Functional Principle



Assuming that the specified image `test.jpg` has a node cache TTL of 10 seconds and a cache pre-refresh time of 80% of the TTL (i.e., 8 seconds), then:

1. When the node receives the client request for the first time, the current node does not cache the file, and it will follow the origin to pull the resource and cache it in the node, with a cache TTL of 10 seconds. Within 0-7 seconds, if the client request is received again, the node will directly provide the resource from the cache and normally respond to the client request;
2. When the `test.jpg` cached in the node reaches the pre-refresh time, between the 8th and 10th seconds, if the client request is received, the node will still normally respond to the client request, but at the same time, it will asynchronously follow the origin to verify whether the cache resource is valid;
If the resource is valid, the cache TTL of the resource on the node will be updated and reset to 10 seconds;
If the resource is invalid, the latest valid resource will be obtained from the origin to the node, and the node cache TTL will be reset to 10 seconds;
3. If no client requests are received after the file exceeds the node cache TTL, the cache will exceed the cache.
4. When the node receives a Client request next time, the node will send an origin-pull request to the origin to verify whether the resources are valid. If the file is updated, the latest file will be pulled, otherwise, the file will be cached again and the Cache time will be refreshed to 10 seconds.

Clear and Preheat Cache

Cache Purge

Last updated : 2025-01-10 17:07:00

Overview

When your resource content is cached to the EdgeOne edge node, during the cache validity period, users accessing the resource will be directly responded by the EdgeOne edge node without triggering a return to the origin. If your origin site updates the resource content at this time, in order to prevent users from still accessing the old resource files, you can manually clear the cached resources in all edge nodes by using the Cache Purge function. After the cache is cleared, when users access the resource, EdgeOne will follow the origin to obtain the latest resource for response.

Quota Description

Different billing plans have different quotas, please refer to: [Comparison of EdgeOne Plans](#).

Use Cases

You may need to use this function in the following scenarios:

Content update: When you have updated some resources on the origin, but the cache on EdgeOne has not expired, you may want to let the client user see the latest content immediately.

Error fix: If there are some erroneous contents cached on EdgeOne from your origin, to avoid business risks, you need to purge these erroneous caches immediately to ensure that EdgeOne no longer provides the wrong content.

Testing and debugging: When developing and debugging a website, you may need to frequently modify and test the content. To ensure that you see the latest modified content rather than the cached old version, you can use the clear the cache function.

Emergency response: In some emergency situations, such as being attacked or releasing sensitive information, you need to remove the relevant content from EdgeOne immediately.

Cache Rules adjustment: When you adjust or optimize the EdgeOne Cache Rules, you may need to purge the existing cache to ensure that the new Rules are Effective immediately.

Support Type

EdgeOne clear the cache support based on various types of purging, details as follows:

Type	Details
URL	Match the node cache resources of the URL, for example, <code>https://www.example.com/path/foo.jpg</code> .
Directory	Match the node cache resources of the directory, for example, <code>https://www.example.com/path/</code> .
Hostname	Match the node cache resources of the Hostname, for example, <code>www.example.com</code> . Do not support submitting URLs in the format of <code>*.test.com</code> , that is, the domain name cannot contain wildcards, and the corresponding subdomains need to be specified.
Cache-Tag	<p>Cache-Tag refresh is a method for quickly refreshing node caches. It allows you to refresh relevant content based on specific cache tags. In your website or application, you need to set cache tags for relevant content, which can be achieved by adding the Cache-Tag header in the HTTP response from the origin server.</p> <p>After setting the cache tag, you can match the tag value of the <code>Cache-Tag</code> header in the HTTP response to clear the cache, for example, <code>Cache-Tag: pic</code>. When submitting a refresh, you can enter <code>pic</code> in the <code>Tag(s)</code> field to refresh all resources with the tag. Only applicable to the Enterprise plan.</p> <p><code>Cache-Tag</code> usage instructions:</p> <p>The maximum header size is 6 KB.</p> <p>Multiple tags are separated by ",", a single tag does not exceed 128 characters, and the tag limitation is 1,000.</p> <p>Tags are case-insensitive, that is, <code>Tag1</code> and <code>tag1</code> will be recognized as the same tag.</p>
All Cache	<p>All cache resources of the site on the node.</p> <p>If a wildcard domain name (e.g., <code>*.foo.example.com</code>) is connected to the current site (<code>example.com</code>), it is unable to take effect against all caches under the wildcard domain name. You need to submit separate tasks to clear the cache for each specific subdomain.</p>

EdgeOne Cache Purge is divided into direct deletion and mark as expired methods, as follows:

URL Type and Cache-Tag Type are set to "directly delete" by default, which means directly deleting the cache content. When a user sends a request for resources, EdgeOne will immediately origin-pull the latest resources, increasing the number of origin-pull requests within a short time and weakening the acceleration effect. If a large amount of content is submitted, the origin will be under greater pressure.

Other purge types are set to "Mark as Invalid" by default, which means that the cache will not be directly deleted, but marked as expired. If the cache node has `Last-Modified` and `Etag` headers, the next time a user requests the resource, the node will carry `If-None-Match` and `If-Modified-Since` headers for origin-pull verification whether the resource has been updated. The response of 304 or 200 is determined by the origin, generally speaking:

If there is no update - the origin returns 304 (Not Modified), then the node continues to use the cache to respond, effectively saving bandwidth;

If there is an update - the origin returns 200 (OK), then the node collects the latest resources from the origin and compares the Last-Modified and `Content-Length` headers of the cached resources and the new resources. If either header value is different, the new resource will overwrite the expired cache on the node.

Note:

EdgeOne nodes differentiate cache based on the URL before and after encoding by default. Therefore, when refreshing, you need to submit them separately. If the URL contains special characters, then when submitting the URL refresh in the console:

`https://example.com/d/default_avatar.png?x-oss-process=image/resize,w_600,1_800`, the cache corresponding to the URL before encoding will be refreshed;

`https://example.com/d/default_avatar.png?x-oss-process=image%2Fresize%2Cw_600%2C1_800`, the cache corresponding to the URL after encoding will be refreshed.

A maximum of the first 10,000 records can be displayed for cache purge history query. If more records are needed, it is recommended to narrow down the query scope or click Export Records at the bottom of the list. Currently, up to 500,000 records can be exported.

Directions

Scenario One: Clear cache by entering content

If you have a small amount of content to clear and it is convenient to enter the content directly in the input box, you can follow these steps:

1. Log in to the [EdgeOne console](#), and in the left sidebar, click **Purge Cache**.
2. On the cache purge page, select the corresponding site, choose the resource type to be cleared, enter the corresponding resource content, and click **OK**.

Purge Cache History

• Purge cached resources in the node. After purging, you need to obtain the latest resources by origin-pull for access. [Learn more](#)

• After purging the cached resources in the node, users need to obtain resources by origin-pull for access. But the increase of origin-pull requests weakens the acceleration effect and places pressures on the origin server.

Site

Content type

Content ☒ Manual input ☐ Upload file

Example: <https://www.example.jpg> (one per line)

Remaining quota per request: 0

OK

3. Switch to the History Records tab to view the history records of the specified time range (within the last month) and purge type.

Scenario Two: Clear cache by uploading a file for batch import

If you have a large amount of content to clear or have already placed the content in a file, you can choose to upload the file:

1. Log in to the [EdgeOne console](#), and in the left sidebar, click **Purge Cache**.
2. On the cache purge page, select the corresponding site, choose the resource type to be cleared, choose the "Upload file" method, and after uploading, click **OK**.

Purge Cache

History

- Purge cached resources in the node. After purging, you need to obtain the latest resources by origin-pull for access. [Learn more](#)
- After purging the cached resources in the node, users need to obtain resources by origin-pull for access. But the increase of origin-pull requests weakens the acceleration effect and places pressures on the origin server.

Site

Content type

URL

Content

Manual input

Upload file

Example: [https://www/example.jpg](https://www.example.jpg) (one per line)

Upload or drag it here

Upload a TXT file within 10 MB

OK

3. Switch to the History Records tab to view the history records of the specified time range (within the last month) and purge type.

Related References

[How long does it take for cache purge and cache pre-warming to take effect after submitting content?](#)

URL Pre-Warming

Last updated : 2024-09-18 15:26:11

Function Introduction

When a business releases new resources, the client's first request for these resources may encounter a situation where there is no cache on EdgeOne, resulting in an inability to respond immediately and the need to follow the origin to obtain. The cache pre-warming function allows resources to be cached on EdgeOne in advance. In this way, even if the client requests for the first time, it can be directly responded from the cache of EdgeOne without the need to follow the origin. The implementation of cache pre-warming is to submit the URLs that need to be pre-warmed, and then cache the resources that match these URLs from the origin to EdgeOne in advance, thereby improving the acceleration effect and mitigating the pressure on the origin.

Quota Description

Different billing plans have different quotas, please refer to: [Comparison of EdgeOne Plans](#).

Usage Scenarios

You may need to use this function in the following scenarios:

Newly released content: When your business releases new content or updates existing content, you want to ensure that this content is immediately available on EdgeOne, so that client users can access the latest content for the first time, reducing the delay when accessing for the first time. For example, before the game business officially releases a new version of the installation package or upgrade package, the installation package resources can be preheated to EdgeOne. After the official release, users can directly obtain the installation package resources from the node when requesting to download these installation packages, improving the download speed.

Large-scale event operations: Before large-scale events, you want to ensure that the key resources of the event have been cached on EdgeOne, which helps to ensure that when the event starts, client users can quickly access the required content, reducing the delay and congestion caused by high traffic.

Expected traffic peaks: If you expect a significant increase in website traffic during a specific period (e.g., holiday promotions, news releases, etc.), you can use the cache pre-warming function to ensure that key resources have already been cached on the edge nodes. This helps to disperse the pressure of origin-pull requests during peak periods and improve the access speed of client users.

Note:

When preheating resources, simulated requests will be made to retrieve the corresponding resources from the origin. If there are many preheating tasks submitted, more origin-pull requests will be generated, and the bandwidth of the origin will increase.

If the preheated resources conflict with the node cache, that is, if EdgeOne has cached identical resources and they have not expired, they will still be valid and will not be overwritten by the preheated resources. If the identical resources have changed, you can purge the corresponding node cache before preheating.

A maximum of the first 10,000 records can be displayed for cache pre-warming history query. If more records are needed, it is recommended to narrow down the query scope or click Export Records at the bottom of the list. Currently, up to 500,000 records can be exported.

Directions

Scenario One: Prefetch cache by inputting content

If you have less content to preheat and it is convenient to input the content directly in the input box, please follow the steps below:

1. Log in to the [EdgeOne console](#), In the left sidebar menu, click **Prefetch URLs**.
2. On the Prefetch URLs page, select the corresponding site, enter the respective resource content, and click **OK**.

Prefetch URLs History

Cache the resources matching the URL(s) from the origin server to the node in advance. The node directly responds to users' requests, thereby improving the acceleration and relieving the pressure on the origin server. [Learn more](#)

Site

URL ☒ Manual input ☐ Upload file

Example: `https://www.example.jpg` (one per line)(Note: Only complete URLs can be submitted. Directories such as "https://www.example/" are not supported.)

Remaining quota per request: 0

Remaining daily quota: 0

OK

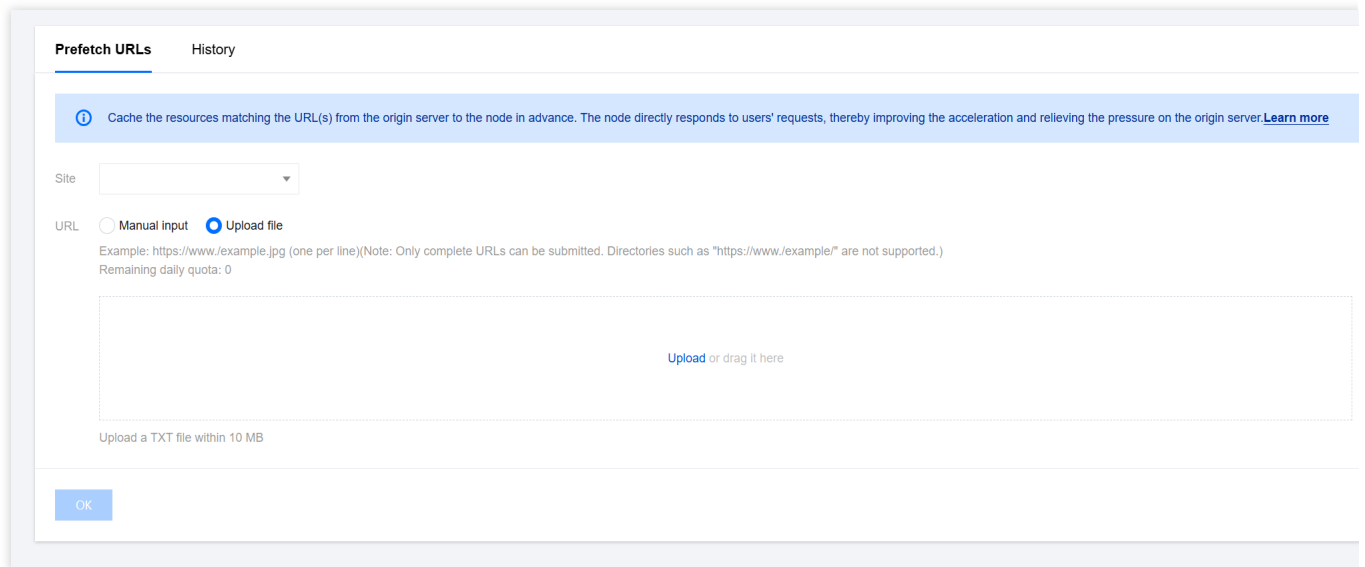
3. Switch to the History Record tab to view the history records within a specified time range (within one month).

Scenario Two: Batch import Prefetch cache content by uploading files

If you have more content to preheat or have already placed the content in a file, you can choose to upload the file:

1. Log in to the [EdgeOne console](#), In the left sidebar menu, click **Prefetch URLs**.

2. On the Prefetch URLs page, select the corresponding site, choose the "Upload File" method, and after uploading, click **OK**.



The screenshot shows the 'Prefetch URLs' tab in the Tencent Cloud EdgeOne console. At the top, there's a blue banner with an information icon and text: 'Cache the resources matching the URL(s) from the origin server to the node in advance. The node directly responds to users' requests, thereby improving the acceleration and relieving the pressure on the origin server. [Learn more](#)'. Below this, there's a 'Site' dropdown menu. Under the 'URL' section, there are two radio buttons: 'Manual input' (unselected) and 'Upload file' (selected). Below the radio buttons, there's a text input area with a dashed border. Inside this area, there's a blue link that says 'Upload or drag it here'. Below the input area, there's a small text label: 'Upload a TXT file within 10 MB'. At the bottom left of the form, there's a blue button labeled 'OK'.

3. Switch to the History Record tab to view the history records within a specified time range (within one month).

Related References

[How long does it take for cache purge and cache pre-warming to take effect after submitting content?](#)

Prefetch M3U8

Last updated : 2025-05-30 14:32:58

This article will provide a detailed introduction on how to achieve M3U8 pre-warming through EdgeOne, systematically bypassing the cache penetration issue during the premiere phase, ensuring that user requests can directly obtain complete resources from the nodes, thereby improving access quality.

Note:

This capability is a whitelisted function. Please [contact us](#) if you wish to use it.

Background Introduction

For streaming media services using M3U8 segmentation protocols like HLS/DASH, cache pre-warming can effectively resolve performance bottlenecks during the premiere phase. In traditional distribution models, users must fetch the M3U8 index file and associated TS segment resources from the origin station step by step during their first visit. If there is no cache on the edge nodes, it will lead to significant premiere delays and playback stuttering. The core principle of M3U8 pre-warming is to retrieve all TS segment resources associated with the M3U8 index file for pre-warming.

Core Value

Reduce Access Latency: User requests are directly obtained from the nearest EdgeOne node without returning to the origin station.

Increase Playback Success Rate: Reduce video interruptions caused by network fluctuations.

Optimize Origin Station Costs: Hot resources with high frequency can reduce bandwidth pressure on the origin station through EdgeOne's caching capability.

Steps

Example Scenario

Assuming you are a video vendor who has integrated the site domain `www.example.com` into EdgeOne acceleration and due to popular series updates, you expect that submitting the M3U8 resource of a film will automatically pre-warm the associated TS resources to EdgeOne.

Step 1: Create a Pre-warming Task for M3U8

Note:

The M3U8 description file must be accessible and describe the segment paths according to industry standards. Supported formats are as follows:

Assuming the request URL is:

```
https://www.example.com/c8679239vodtranssgp1500031474/5fac87c91397757892217228202/addedp.1505647.m3u8 .
```

The content format of this URL is relative, such as 1505647_0_0.ts, and EdgeOne will concatenate it into the following TS URL:

```
https://www.example.com/c8679239vodtranssgp1500031474/5fac87c91397757892217228202/1505647_0_0.ts .
```

The recursive parsing depth of the M3U8 description file should not exceed 3 layers.

The parsed segment count will accumulate the daily pre-warming quota, and once it exceeds the quota, it will be handled silently without further warming.

Call the [CreatePrefetchTask](#) API, where:

1. The Targets field: the value is the URL of the M3U8, such as

```
https://www.example.com/c8679239vodtranssgp1500031474/5fac87c91397757892217228202/addedp.1505647.m3u8 .
```

2. The PrefetchMediaSegments field, value is on. The value explanation is:

on: Pre-warm the M3U8 description file while recursively parsing and pre-warming the TS resources described in the file.

off: If not filled, the default value is off. Only the submitted M3U8 description file will be pre-warmed.

3. Other parameter fields can be filled as needed.

Step 2: Query the Status of the M3U8 Pre-warming Task

Call the [CreatePrefetchTask](#) API, you can check based on the job-id returned when creating the pre-warming task or query through the target. The explanations of the task status are as follows:

processing: In progress.

success: Successful.

failed: Failed.

timeout: Timeout.

invalid: Invalid.

Calling Example

curl

Golang

```
curl -X POST https://teo.tencentcloudapi.com -H "Authorization: TC3-HMAC-SHA256 Credential=*****/2025-02-19/teo/tc3_request, SignedHeaders=content-type;host, Signature=9ec53d3ba8d4049c219052b0a2275ff3a30d3429d6295ae4c799c74d32c8f015" -H
```

```
"Content-Type: application/json" -H "Host: teo.tencentcloudapi.com" -H "X-TC-Action: CreatePrefetchTask" -H "X-TC-Timestamp: 1739965395" -H "X-TC-Version: 2022-09-01" -H "X-TC-Language: zh-CN" -d '{"ZoneId": "zone-xxx", "Targets": ["https://www.example.com/c8679239vodtranssgp1500031474/5fac87c91397757892217228202/adp.1505647.m3u8"]}'
```

```
package main

import (
    "fmt"

    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/errors"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/profile"
    teo "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/teo/v20220901"
)

func main() {
    // Instantiate a credential object. You need to pass in the Tencent Cloud a
    // Code leakage may result in the exposure of your SecretId and SecretKey, thre
    // Keys can be obtained from the official console at https://consoleintl.cloud
    credential := common.NewCredential(
        "SecretId",
        "SecretKey",
    )
    // Instantiate a client option, optional, can be skipped if there are no sp
    cpf := profile.NewClientProfile()
    cpf.HttpProfile.Endpoint = "teo.tencentcloudapi.com"
    // Instantiate the client object for the requested product. For domestic si
    client, _ := teo.NewClient(credential, "ap-guangzhou", cpf)

    // Instantiate a request object; each interface corresponds to a request ob
    request := teo.NewCreatePrefetchTaskRequest()
    request.ZoneId = common.StringPtr("zone-364ni75dvzva")
    request.Targets = common.StringPtrs([]string{"https://www.example.com/1.m3u
    request.PrefetchMediaSegments = common.StringPtr("on")

    // The returned resp is an instance of CreatePrefetchTaskResponse, correspo
    response, err := client.CreatePrefetchTask(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        fmt.Printf("An API error has returned: %s", err)
        return
    }
    if err != nil {
        panic(err)
    }
    // Output JSON format string response
    fmt.Printf("%s", response.ToJsonString())
}
```

```
}
```

How to improve the Cache Hit Rate of EdgeOne

Last updated : 2024-07-01 09:40:56

This article introduces how to reasonably use various configurations on EdgeOne, combined with your actual business scenarios for tuning, and improve the cache hit rate of files within the site.

Background introduction

When your site is connected to EdgeOne and acceleration is enabled, when users access static resources such as images and videos, EdgeOne will cache the corresponding static files in the edge nodes. When other users initiate repeated requests, the edge nodes will directly respond to the requests, avoiding origin-pull requests.

If the cache hit rate is too low, it will cause a large number of user requests to go back to the origin, which will bring a lot of processing pressure to the origin site, reduce the user's access experience, and the site acceleration effect. You can optimize and improve the cache hit rate through the following configuration tuning.

Note :

If you need to view the current cache hit analysis, you can view it on the console through Data **Analysis > Cache Analysis**. For details, please refer to [Cache Analysis](#).

Optimization methods

1. Adjust the node cache TTL configuration

The configuration of the node cache TTL will directly affect whether EdgeOne caches the specified file resources and the corresponding cache time. If the file caching strategy is not cached or the cache time on the node is short, it will cause users to access without hitting the cache, frequently going back to the origin, and reducing the access experience.

By default, EdgeOne has enabled default caching rules for global sites. You can view the [EdgeOne content caching rules](#) to understand how EdgeOne's caching rules take effect. In order to improve the cache hit rate, it is recommended that you configure caching rules separately in the rule engine based on file extensions.

Suggested configuration

It is recommended that you configure personalized caching rules for different file types in different scenarios:

1. Files that are not updated frequently, such as download resources, video files, etc., it is recommended to configure a custom cache time on EdgeOne, with a cache time of 30 days or longer, and force caching through EdgeOne nodes; common download resources and video file formats are as follows:

Audio and video	mp4;mp3;ts;m4a;avi;m4s;ogg;mkv;mov;flv;rm;rmvb;swf;wav;wmv;rmi;aac
Compressed package	rar;7z;zip;gzip;dmg;gz;ios;tar;jar;br;bz2
Document	doc;docx;xls;xlsx;pdf;ppt;pptx
Application	apk;exe;bin
Others	vsv;iso;jar;swf;chunk;atlas

2. Frequently updated file content, such as image content, if the cache time is too long, it may cause users to access expired content due to cache hits. Therefore, it is recommended to configure a custom cache time on EdgeOne, with a general cache time configured according to business needs, which can be set between 1-7 days. Common image content formats are as follows:

Images	jpg/png/jpeg/webp/gif/heif/heic/kpg;ico
Web pages	html;htm;shtml;hml;js

3. Dynamic files, such as php, json files, etc., if cached, will cause users to access content that cannot be correctly responded to. Therefore, it is recommended to configure them separately on EdgeOne as not cached. Common dynamic file formats are as follows:

Dynamic resources	php;aspx;asp;jsp;do;dwr;cgi;fcgi;action;ashx;axd;json
-------------------	---

Optimization example

When you view the resource hit rate in the [cache analysis](#), you can view the specific file extensions on the right to see which types of resources have a large number of misses. For example, in the current cache distribution, there are many `.mp4` format files that have not hit the cache.

If you follow EdgeOne's default caching rules completely, the problem is that the response file failed to respond to the Cache-Control header. According to the default caching rules, the `.mp4` file has a cache time of 2 hours on the node. Because the cache time is short, this file will frequently go back to the origin. If you need to cache this file, you can go to the rule engine, add a new rule, set the node cache TTL to custom cache for 30 days when the file extension is equal to mp4, and enable forced caching, that is, ignoring the CC header of the origin response, and the node forcibly caches the file. For detailed operation steps, please refer to: [Node cache TTL](#).

The screenshot shows the 'IF' configuration panel in the Tencent Cloud EdgeOne console. It includes a '+ Comment' button and a list of conditions and actions.

Matching type	Operator	Value
HOST	Is	[redacted]
File extension	Is	mp4

Buttons: + And, + Or

Action	Behavior	Time	Force cache
EdgeOne Node Cache ...	Custom TTL	30 days	Enabled

Buttons: + Action, + IF

2. Customize the cache key Cache Key to point the same type of request to a cache file

By default, EdgeOne will generate a unique identifier for the cache key based on the user's access Request URL and query string, which serves as the cache key for the file. When there are the same requests, the edge node will compare whether the request is consistent with the cache key in the cache to determine whether the cache is hit. If the URL carries dynamic parameter content that does not affect the file version, such as user identification ID, multiple caches will be established based on the different parameters, resulting in a decrease in cache hit rate. You can optimize this by customizing the cache key Cache Key.

Suggested configuration

When some parameters in the request URL do not affect the file version, it is recommended to improve the cache hit rate by retaining or ignoring the specified parameter content.

Optimization example

The current request URL is: `https://image.example.com/test.jpg?`

`version=1.1&token=1234567890`, where the parameter `version=1.1` will affect the content of the image, and `token=1234567890` will not affect it. To improve the hit rate of the cache, you can ignore the token parameter in the custom Cache Key. For detailed operations, please refer to: [Custom Cache Key](#).

The screenshot displays the configuration interface for an IF rule in the Tencent Cloud EdgeOne console. The interface is divided into two main sections: 'IF' (blue header) and 'Action' (green header). In the 'IF' section, the 'Matching type' is set to 'HOST', the 'Operator' is 'Is', and the 'Value' is a placeholder for a domain. Below this, there are buttons for '+ And' and '+ Or'. The 'Action' section contains a 'Custom cache key' input field. Below this, there are three dropdown menus: 'Type' (set to 'Query string'), 'Mode' (set to 'Ignore specified param'), and 'Parameter' (set to 'token'). There are also buttons for '+ Add', '+ Action', and '+ IF'.

3. URL Pre-Warming

URL Pre-Warming allows EdgeOne to cache files to edge nodes in advance. When users access the files, they can directly hit the cache, reducing the concurrent follow origin for the first time and improving the cache hit rate. When you add a new site to EdgeOne or release new popular resources, it is suggested to pre-warm the cache in advance. For detailed operations, please refer to: [URL Pre-Warming](#).

4. Enable Cache Pre-Refresh

If your files are mainly popular files, and you need to ensure that the files can continuously have cache in the node, you can enable cache pre-refresh. Before the cache of the file in the node expires, when a user requests a file in the node, it will verify with the origin whether the file has been updated. If not, the cache time of the file in the node will be refreshed. For detailed operations, please refer to: [Cache Pre-Refresh](#).

5. Make Reasonable Use of Vary Mechanism

When the origin responds with a Vary header, the CDN will cache the content based on the specified content in the Vary header. For a detailed explanation of the Vary principle, please see the [Vary feature](#). If the current file does not need to be controlled by the Vary header to cache different versions, it is suggested that you avoid responding to this header in the origin response to reduce the number of cache versions created and improve the cache hit rate.

Learn more

[How to determine whether a user's request hits the EdgeOne node cache](#)

File Optimization

Smart Compression

Last updated : 2025-05-07 09:43:05

Function Introduction

EdgeOne enables Gzip or Brotli compression globally by default. When the client request header carries `Accept-Encoding: br, gzip` or `Accept-Encoding: br` or `Accept-Encoding: gzip`, the node will intelligently compress files based on their `Content-Type`. Compressed files can effectively reduce the size of resources and speed up content transmission. Enabling smart compression can help you:

- 1. Improve user experience:** By reducing resource size, web page loading speed can be significantly improved, providing a better user experience. Especially for websites with a large amount of CSS, JavaScript, and other resources, enabling compression can greatly reduce loading time.
- 2. Save traffic:** Compressed resources will consume less network traffic, which will help reduce operational costs.

Directions

Note :

By default, you do not need to modify this configuration. If in some scenarios, such as: the current client will perform MD5 verification on the file or the current client does not support parsing the specified compressed file, and you want the current site to use only Brotli compression or Gzip compression, or not to compress at all, you can follow the steps below.

Scenario 1: Enable/Disable Smart Compression for All Domain Names of the Site

If you need to enable/disable smart compression for the whole connected site, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **File Optimization** in the right sidebar.
3. Find the smart compression configuration card, which is enabled by default. Click the **switch** to configure enable/disable.

Scenario 2: Enable/Disable Smart Compression for Specific Domain Names

If you only need to enable/disable smart compression for specific domain names, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.

2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the Host matching type to match requests for specific domain names.
5. Click **Action > Select Box**, and in the pop-up operation list, select the operation as Smart Compression, and click the **switch** to enable/disable Gzip or Brotli compression.
6. Click **Save and Publish** to complete the rule configuration.

Related References

Smart Compression Effective Rules

1. Smart compression supports file size range:256B - 30MB。
2. Smart compression is synchronous compression, compressing files while fetching them from the origin. When the node first requests a compressed file, it can directly respond with the compressed file.
3. Smart compression compresses files based on `Content-Type` by default, supporting the following types:

```
text/html
text/xml
text/plain
text/css
text/javascript
application/json
application/javascript
application/x-javascript
application/rss+xml
application/xmltext
image/svg+xml
image/tiff
text/richtext
text/x-script
text/x-component
text/x-java-source
text/x-markdown
text/js
image/x-icon
image/vnd.microsoft.icon
application/x-perl
application/x-httpd-cgi
application/xml
application/xml+rss
application/vnd.api+json
```

```
application/x-protobuf
multipart/bag
multipart/mixed
application/xhtml+xml
font/ttf
font/otf
font/x-woff
application/vnd.ms-fontobject
application/ttf
application/x-ttf
application/otf
application/x-otf
application/truetype
application/opentype
application/x-opentype
application/font-woff
application/eot
application/font
application/font-sfnt
application/wasm
application/javascript-binast
application/manifest+json
application/ld+json
```

4. If you have both Gzip and Brotli compression enabled at the same time, and the client request header `Accept-Encoding` carries both `br` and `gzip`:

If the node already has cached content, it will respond according to the following rules:

If the node has both Brotli and gzip compressed cache content, it will prioritize responding with Brotli compression.

If the node only has Brotli compressed cache content, it will prioritize responding with Brotli compression.

If the node only has gzip compressed cache content, it will prioritize responding with Gzip compression.

If the node does not have cached content, it will prioritize responding with Brotli compression.

5. When only Brotli compression is enabled, if the request compression header is `gzip`, the compression will not take effect, and the original resources will be returned; when only Gzip compression is enabled, if the request compression header is `br`, the compression will not take effect, and the original resources will be returned.

6. If the origin server has enabled compression and the server carries response headers: `Content-Encoding`, the smart compression function will no longer be effective.

Request Example

Not Enabled Smart Compression

First request for gzip compressed file, not hit node cache, fetch the original file from the origin and cache it to the node, EdgeOne responds with the original file:

Enabled Smart Compression

First request for gzip compressed file, not hit node cache, fetch the file from the origin, node synchronously compresses and caches the compressed file, EdgeOne responds with the compressed file:

Smart compression supports streaming compression, and if the request does not hit the node cache, it will respond in a chunked manner after fetching the file from the origin.

Subsequent requests, hit the node cache of gzip compressed file, the node directly responds with the compressed file.

Network Optimization

HTTP/2

Last updated : 2025-03-21 11:53:10

What is HTTP/2?

EdgeOne supports clients to initiate requests using the HTTP/2 protocol. HTTP/2 (i.e., HTTP 2.0, the second version of the Hypertext Transfer Protocol) is the second major version of the HTTP protocol, which can effectively reduce network latency and improve site page loading speed.

Note :

1. If the client request does not use HTTP/2, EdgeOne is compatible with HTTP 1.x protocol access.
2. For configuring access requests, please refer to this document. If you need to configure HTTP/2 to follow the origin, please refer to [HTTP/2 origin-pull](#).
3. Starting from November 23, 2023, for security reasons (for details, see [Protection against DDoS attacks targeting HTTP/2 protocol vulnerabilities](#)), the HTTP/2 will be disabled by default for incremental sites, and users may enable it as needed.

Prerequisites

The access domain name of the current site has been configured with an SSL certificate. For how to configure an SSL certificate, please refer to [Certificate Configuration](#).

Directions

Scenario 1: Modify HTTP/2 support for all domain names of the site

If you need to enable or disable HTTP/2 for the whole connected site, please follow the steps below:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the HTTP/2 configuration card. This protocol is disabled by default. Toggle the **switch** to enable it.

HTTP/2

HTTP/2 (HTTP2.0) requests are supported to accelerate sites and improve web performance. [Details](#)

Note: Please configure the HTTPS certificate first to enable the protocol.



[Custom settings](#)

Scenario 2: Enable HTTP/2 for a specified domain name

If you only need to enable or disable HTTP/2 for a specified domain name, please follow the steps below:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the Host matching type to match requests for the specified domain name.
5. Click **Action**, and in the pop-up operation list, select the operation as **HTTP/2**. Click the switch to enable/disable.

The screenshot shows the 'Rule Engine' configuration interface. At the top, there is a tab labeled 'IF' with a '+ Comment' button. Below this, the rule configuration is divided into two main sections: 'Matching type' and 'Action'.

Matching type section:

- Matching type:** A dropdown menu set to 'HOST'.
- Operator:** A dropdown menu set to 'Is'.
- Value:** A text input field containing a domain name, with a small icon to the right for clearing or editing.

Below the matching type section, there are two buttons: '+ And' and '+ Or'.

Action section:

- Action:** A dropdown menu set to 'HTTP/2'.
- On/Off:** A toggle switch that is currently turned 'On' (blue).

At the bottom of the action section, there are two buttons: '+ Action' and '+ IF'.

6. Click **Save and Publish** to complete the rule configuration.

HTTP/3(QUIC)

Overview

Last updated : 2023-12-15 09:59:07

What is HTTP/3

[HTTP/3](#) (HTTP over QUIC) is the next-generation Internet transmission protocol, designed to solve the head-of-line blocking issues in HTTP/2. Instead of being based on TCP, HTTP/3 uses the QUIC protocol, which is based on the UDP protocol. Compared to HTTP/1.1 and HTTP/2, HTTP/3 provides faster connection establishment and data transmission speeds, supports multiplexing for simultaneous transmission of multiple requests and responses, and has made improvements in areas such as congestion control, header compression, etc., significantly reducing latency and waiting time, and improving website performance and user experience.

EdgeOne's support for HTTP/3

EdgeOne now supports both HTTP/3 and QUIC protocols, with the following versions of the QUIC protocol supported:

Standard	Supported Versions
Google QUIC (gQUIC)	Q039、Q043、Q046、Q050
IETF QUIC (iQUIC)	draft-27、draft-29、 RFC 9000

How to use HTTP/3 or QUIC access in your App

If your users need to access through an App, the App needs to integrate support for HTTP/3 or QUIC protocols. EdgeOne provides a QUIC SDK to help you quickly complete the integration. For details, please refer to [the QUIC SDK download and integration example](#).

If your site is mainly a website, and the user's current browser supports the QUIC protocol and has enabled QUIC protocol access, your site only needs to [enable HTTP/3 \(QUIC\)](#) on EdgeOne to support it. You can check whether the current browser supports HTTP/3 by [viewing if the browser supports HTTP/3](#).

Enable HTTP/3

Last updated : 2025-05-07 09:43:53

This article introduces how to enable support for the HTTP/3 protocol within the EdgeOne platform.

Note :

1. If you need to enable HTTP/3 support, the current access domain must have an [HTTPS certificate configured](#) to take effect.
2. If both HTTP/2 and HTTP/3 are enabled at the same time, the actual client requests will use either HTTP/2 or HTTP/3 based on the client's request.

Billing Instructions

HTTP/3 is a billed feature, and the specific fee standard can be referred to: [VAU Fee \(Pay-as-You-Go\)](#).

Scenario 1: Enable support for the HTTP/3 protocol for all domain names of the site

If you need to enable support for the HTTP/3 protocol for all domain names, please follow the steps below:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the HTTP/3 (QUIC) configuration card. This protocol is disabled by default. Toggle the **switch** to enable it.

Off state (default): does not support HTTP/3 requests.

Activated state: supports HTTP/3 requests and uses HTTP/3 to accelerate site requests.

Scenario 2: Enable support for the HTTP/3 protocol for specified domain names

If you need to enable support for the HTTP/3 protocol for different domain names, please follow the steps below:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.

3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. In the rule editing page, select the Host matching type to match requests for specified domain names.
5. Click **Action** > **Selection Box**, and in the pop-up operation list, select the operation as HTTP/3, and switch the switch to On.
6. Click **Save and Publish** to complete the rule configuration.

QUIC SDK

SDK Overview

Last updated : 2023-06-29 11:21:46

Tencent Cloud EdgeOne QUIC SDK is a QUIC-based development toolkit that provides easy-to-use APIs for developers to integrate QUIC into their applications more quickly, enabling stable, high-quality network transfer. Supported platforms include Android and iOS.

About QUIC

QUIC (Quick UDP Internet Connections) is a new general-purpose, secure, and multiplexing transport layer network protocol. The standard HTTP/3 protocol is implemented based on QUIC, which supports 0-RTT connections, non-HOL blocking multiplexing and easy implementation of user-mode congestion control to transfer more data with a lower bandwidth, enabling high-quality data transfer even under poor network conditions with a high packet loss rate and network latency. It also supports connection migration that can guarantee an uninterrupted connection even if the network of a mobile device is switched frequently.

Must-Knows

EdgeOne QUIC SDK is free of charge during beta testing.

Supported Versions

EdgeOne QUIC SDK supports IETF QUIC and Google QUIC. See below for details:

Standard	Supported Version
Google QUIC (gQUIC)	QO43, QO46, QO50, QO51
IETF QUIC (iQUIC)	draft-29, RFC 9000

SDK Download and Integration

Last updated : 2024-05-08 21:35:03

Downloading SDK

Tencent Cloud EdgeOne QUIC SDK supports iOS and Android operating systems.



Android [Download ZIP](#)



iOS [Download ZIP](#)

Directions

Access Android SDK

Access iOS SDK

1. Environment requirements:

Android Studio 2.0+

Android 5.0 (SDK API level 21) or above

2. Download `TQUIC_Android_SDK.zip` and compress the package. Copy the `AAR` file to the project directory (app/libs).

3. Include sdk and okhttp dependencies in the build.gradle file.

```
dependencies {  
    ...  
    implementation fileTree(dir: 'libs', include: ['*.aar']) //Add the *.aar  
    file.  
    implementation 'com.squareup.okhttp3:okhttp:3.11.0' //Add the okhttp  
    dependencies.  
}
```

4. Configure permissions in `AndroidManifest.xml` . The QUIC SDK requires the following permissions:

```
<uses-permission android:name="android.permission.INTERNET" />
```

1. Environment requirements:

Xcode 10.0+

iPhone or iPad on iOS 10.0 or above

A valid developer signature for your project

2. Download `TQUIC_iOS_SDK.zip` and compress the package. Copy the `framework` file to the project directory.

3. Import `TQUICiOS.framework` and `Tquic.framework` to XCode.

Note

`Tquic.framework` is a dynamic library and needs to be embedded and signed.

4. Add `-ObjC, -l"c++"` to the compilation option **Other Linker Flags**.

Sample Code

Android

Last updated : 2023-06-29 11:20:56

The following code shows how to make QUIC requests with the Android client. For details of the API description, see [Android APIs](#).

Creating GET Requests

```
//Create QuicClient and initialize QUIC configuration. It is recommended to use QuicClient
QuicClient quicClient = new QuicClient.Builder()
    .setCongestionType(QuicClient.CONGESTION_TYPE_BBR) //Use BBR
    .setConnectTimeoutMillis(6 * 1000) //Configure connection
    .build();

//Create QuicRequest and specify the request URL.
String url="";
QuicRequest request = new QuicRequest.Builder(url).get().build();

//Execute the request asynchronously and get the result. See %!s(<nil>) for instructions
quicClient.newCall(request).enqueue(new QuicCallback() {
    @Override
    public void onResponse(QuicCall call, QuicResponse response) throws IOException {
        //When the request is executed successfully, it returns the response data
        ResponseBody body = response.body();
        if(body != null) {
            String res = body.string();
        }
    }

    @Override
    public void onFailed(QuicCall call, int errorCode, String error) {
        //When the request fails to be executed, it returns the error message.
    }
});
```

Creating POST Requests

```
//Create QuicClient and initialize QUIC configuration. It is recommended to use QuicClient
QuicClient quicClient = new QuicClient.Builder()
    .setCongestionType(QuicClient.CONGESTION_TYPE_BBR) //Use BBR
    .setConnectTimeoutMillis(3 * 1000) //Configure connection timeout
    .build();

//Construct body data.
String body="your body string";
RequestBody requestBody = RequestBody.create(MediaType.parse("application/json"), body);

//Create QuicRequest.
String url="";
QuicRequest request = new QuicRequest.Builder(url).post(requestBody).build();

//Execute the request asynchronously and get the result. See %!s(<nil>) for instructions.
quicClient.newCall(request).enqueue(new QuicCallback() {
    @Override
    public void onResponse(QuicCall call, QuicResponse response) throws IOException {
        //When the request is executed successfully, it returns the response data
        ResponseBody body = response.body();
        if(body != null) {
            String res = body.string();
        }
    }

    @Override
    public void onFailed(QuicCall call, int errorCode, String error) {
        //When the request fails to be executed, it returns the error message.
    }
});
```

Canceling Requests

```
...
//Create QuicCall. See %!s(<nil>) for instructions.
QuicCall quicCall = quicClient.newCall(request);

// Initiate a request.
...

//Cancel the request using the cancel method via QuicCall.
quicCall.cancel();
```


iOS

Last updated : 2023-06-29 11:20:56

The following code shows how to create QUIC requests with the iOS client. For details of the API description, see [iOS APIs](#).

Creating GET Requests

```
// Session configuration
TQUICURLSessionConfiguration *quicSessionConfiguration = [TQUICURLSessionConfigurat
// Congestion control algorithm
quicSessionConfiguration.congestionType = TQUICCongestionTypeBBR;
// Connection timeout
quicSessionConfiguration.connectTimeoutMillis = 6 * 1000;
// Create SessionManager. See %!s(<nil>) for instructions.
TQUICHTTPSessionManager *quicSessionManager = [[TQUICHTTPSessionManager alloc] init

// (Optional) Serialize via TQUICHTTPRequestSerializer/TQUICHTTPResponseSerializer.

// Initiate a GET request.
[quicSessionManager GET:@"url"
    parameters:nil
    headers:nil
    timeoutInterval:0
    downloadProgress:nil
    success:^(TQUICURLSessionTask * _Nonnull task, id _Nullable respon
// When the request is executed successfully, it returns the response data.
// Get the response headers. Since this is an HTTP response, it can be conv
NSDictionary *headers = [(NSHTTPURLResponse *)task.response allHeaderFields
// Get the response body.
id body = responseObject;

} failure:^(TQUICURLSessionTask * _Nullable task, NSError * _Nonnull e
// When the request fails to be executed, it returns the error message.
NSInteger errorCode = error.code;

}];
```

Creating POST Requests


```
// Session configuration
TQUICURLSessionConfiguration *quicSessionConfiguration = [TQUICURLSessionConfigurat
// Congestion control algorithm
quicSessionConfiguration.congestionType = TQUICCongestionTypeBBR;
// Connection timeout
quicSessionConfiguration.connectTimeoutMillis = 6 * 1000;
// Create SessionManager. See %!s(<nil>) for instructions.
TQUICHTTPSessionManager *quicSessionManager = [[TQUICHTTPSessionManager alloc] init

// (Optional) Serialize via TQUICHTTPRequestSerializer/TQUICHTTPResponseSerializer.

// Construct body data.
NSData *bodyData;
// Initiate a POST request.
[quicSessionManager POST:@"url"
    body:bodyData
    headers:nil
    timeoutInterval:timeInterval
    uploadProgress:^(NSProgress * _Nonnull uploadProgress) {

    } success:^(TQUICURLSessionTask * _Nonnull task, id _Nullable
// When the request is executed successfully, it returns the re
// Get the response body.
id body = responseObject;

    } failure:^(TQUICURLSessionTask * _Nullable task, NSError * _No
// When the request fails to be executed, it returns the error
NSInteger errorCode = error.code;

}];
```

Canceling Requests

```
// Create SessionManager.

...
// Initiate a request via SessionDataTask. See %!s(<nil>) for instructions.
TQUICURLSessionDataTask *dataTask =
[quicSessionManager dataTaskWithRequest:request
    uploadProgress:nil
    downloadProgress:nil
    completionHandler:^(NSURLResponse * _Nonnull response, id _

// Run callback after the request has finished.
```

```
    }];  
  
    //Send the request.  
    [dataTask resume];  
  
    //Cancel the request.  
    [dataTask cancel];
```

API Documentation

Android

Last updated : 2025-03-24 17:18:58

API Overview

API	Description
QuicClient	The main function of QUIC, used to create QuicCall instances and QUIC configuration and get the version number etc.
QuicCall	Manage QUIC requests.
QuicRequest	Encapsulate requests.
QuicResponse	Encapsulate responses.
QuicCallback	Execute callbacks.
QuicNetStats	Get information about QUIC network status.

QuicClient

The main function of QUIC, used to create QuicCall instances and QUIC configuration and get the version number etc.

API	Description
newCall	Create a QuicCall instance.
Builder	Construct QUIC configuration .
getVersion	Get the SDK version number.

newCall

Create a QuicCall instance for each QUIC request.

```
QuicCall newCall(QuicRequest request)
```

Parameter	Description
request	The request to be encapsulated. See QuicRequest .

getVersion

A static method that can get the SDK version number.

```
String getVersion()
```

Builder

QUIC configuration APIs

API	Description
setQuicVersion	Set the QUIC version.
setCongestionType	Set the congestion control algorithm.
setConnectTimeoutMillis	Set the connection timeout.
setTotalTimeoutMillis	Set the total timeout for the request.
setIdleTimeoutMillis	Set the idle connection timeout.
setSupportIpv6	Whether to support IPv6.
build	Create QuicClient .

setQuicVersion

Set the QUIC version.

```
Builder setQuicVersion(int quicVersion)
```

Parameter	Description
quicVersion	<p>Set the QUIC version.</p> <p>Supported versions: Q043, Q046, Q050, Q051, draft-29, RFC-V1 (RFC 9000).</p> <p>Values:</p> <ul style="list-style-type: none">QuicClient.QUIC_VERSION_Q43 (default)QuicClient.QUIC_VERSION_Q46 audio/video proxyQuicClient.QUIC_VERSION_Q50 audio/video proxyQuicClient.QUIC_VERSION_Q51 audio/video proxyQuicClient.QUIC_VERSION_IETF_DRAFT_29 audio/video proxyQuicClient.QUIC_VERSION_IETF_RFC_V1 audio/video proxy

setCongestionType

Set the congestion control algorithm.

```
Builder setCongestionType(int congestionType)
```

Parameter	Description
congestionType	Supported congestion control algorithms: CubicBytes, RenoBytes, BBR, PCC, GCC. Values: QuicClient.CONGESTION_TYPE_BBR (default) QuicClient.CONGESTION_TYPE_RENO_BYTES QuicClient.CONGESTION_TYPE_BBR QuicClient.CONGESTION_TYPE_PCC QuicClient.CONGESTION_TYPE_GCC

setConnectTimeoutMillis

Set the connection timeout.

```
Builder setConnectTimeoutMillis(int connectTimeoutMillis)
```

Parameter	Description
connectTimeoutMillis	Set the connection timeout in milliseconds. Default value: 60000.

setTotalTimeoutMillis

Set the total timeout that covers data read and write.

```
Builder setTotalTimeoutMillis(int totalTimeoutMillis)
```

Parameter	Description
totalTimeoutMillis	Set the total timeout in milliseconds. Default value: 0 (no timeout).

setIdleTimeoutMillis

Set the idle connection timeout. When the timeout expires, connections are closed and cannot be reused.

```
Builder setIdleTimeoutMillis(int idleTimeoutMillis)
```

Parameter	Description
-----------	-------------

idleTimeoutMillis	The idle connection timeout in milliseconds. Default value: 90000.
-------------------	---

setSupportIpv6

Whether to support IPv6.

```
Builder setSupportIpv6(boolean supportIpv6)
```

Parameter	Description
supportIpv6 audio/video proxy	Whether to support IPv6. Values: <code>true</code> , <code>false</code> . Default value: <code>false</code> .

build

Create QuicClient.

```
QuicClient build()
```

QuicCall

Manage QUIC requests.

API	Description
enqueue	Initiate an asynchronous QUIC network request.
cancel	Cancel requests.
getQuicNetStats	Get information about QUIC network status. For more details, see QuicNetStats .

enqueue

Add an asynchronous QUIC request to the queue. You can get response from the callback function.

```
void enqueue(QuicCallback callback)
```

Parameter	Description
callback	Return response from the callback function. For more details, see QuicCallback .

cancel

Cancel requests.

```
void cancel()
```

getQuicNetStats

Get information about QUIC network status.

```
QuicNetStats getQuicNetStats()
```

QuicRequest

Request parameter

API	Description
Builder	Construct a request parameter.

Builder

API	Description
setUrl	Set the request URL.
setIp	Set the request IP.
addHeader	Add the request header
get	Set the request type to GET.
post	Set the request type to POST.
method	Set other request parameters.
build	Create a QuicRequest object.

setUrl

Set the request URL.

```
Builder setUrl(String url)
```

Parameter	Description
-----------	-------------

url	(Required) The request URL.
-----	-----------------------------

setIp

Set the request IP address.

```
Builder setIp(String ip)
```

Parameter	Description
ip	(Optional) If you do not use DNS resolution, set the IP address that the hostname resolves to.

addHeader

Add the request header in a key-value format.

```
Builder addHeader(String key, String value)
```

Parameter	Description
key	Key of the header.
value	Value of the header.

get

Set the request type to GET.

```
Builder get()
```

post

Set the request type to POST.

```
Builder post(RequestBody body)
```

Parameter	Description
body	Body data.

method

Construct DELETE, PUT and other requests.

```
Builder method(String method, RequestBody body)
```


Parameter	Description
method	Supported request methods: PUT, DELETE, HEAD, PATCH.
body	Body data.

build

Create QuicRequest

```
QuicRequest build()
```

QuicResponse

Response information

API	Description
getStatusCode	Get the response status code.
getHeaders	Get the response headers.
getContentType	Get Content-Type.
getContentLength	Get Content-Length.
body	Get the response body.

getStatusCode

Get the response status code.

```
int getStatusCode()
```

getHeaders

Get a list of response headers.

```
List<String> getHeaders()
```

getContentType

Get the content type from the response.

```
void getContentType(String contentType)
```

Parameter	Description
contentType	Get the content type.

getContentLength

Get the content length.

```
void getContentLength(long contentLength)
```

Parameter	Description
contentLength	Get the content length.

body

Get the body of the response.

```
ResponseBody body()
```

QuicCallback

Execute callbacks.

API	Description
onResponse	The callback succeeded.
onFailed	The callback failed.

onResponse

The callback function to execute when the request has succeeded.

```
void onResponse(QuicCall call, QuicResponse response) throws IOException
```

Parameter	Description
call	Manage the QUIC request. For more details, see QuicCall .
response	Return the QUIC response. For more details, see QuicResponse .

onFailed

The callback function to execute when the request has failed.

```
void onFailed(QuicCall call, int errorCode, String errorMsg)
```

Parameter	Description
call	Manage the QUIC request. For more details, see QuicCall .
errorCode	The error code.
errorMsg	The error message.

QuicNetStats

Get information about QUIC network status.

API	Description
isValid	Whether the value of status is valid.
isQuic	Whether it is a QUIC request.
is0rtt	Whether it is a 0-RTT connection.
isConnReuse	Whether the connection is reused.
getConnectMs	Get the connection duration in milliseconds.
getDnsMs	Get the DNS duration in milliseconds.
getDnsCode	Get the DNS error code.
getTtfbMs	Get the time taken for the first byte to be received in milliseconds.
getCompleteMs	Get the time taken for the request to be completed in milliseconds. The time taken by the connection is not included.
getSrttMs	Get the average round-trip time in milliseconds.
getPacketsSent	Get the number of packets sent in bytes.
getPacketsRetransmitted	Get the number of packets retransmitted in bytes.
getBytesSent	Get the number of bytes sent.

getBytesRetransmitted	Get the number of bytes retransmitted.
getPacketsLost	Get the number of packets lost in bytes.
getPacketsReceived	Get the number of packets received in bytes.
getBytesReceived	Get the number of bytes received.
getStreamBytesReceived	Get the number of bytes received within the stream.

iOS

Last updated : 2023-07-26 15:32:06

API Overview

API	Description
TQUICHTTPSessionManager	Session management APIs
TQUICURLSessionConfiguration	QUIC request configuration
TQUICURLSessionDataTask	Task management APIs
TQUICURLRequestSerialization	Request serialization APIs
TQUICURLResponseSerialization	Response serialization APIs

TQUICHTTPSessionManager

API	Description
manager	A static method to construct a TQUICHTTPSessionManager instance.
initWithSessionConfiguration	Create a TQUICHTTPSessionManager instance with the specified configuration. For detailed configuration, see TQUICURLSessionConfiguration
initWithBaseURL	Create a TQUICHTTPSessionManager instance with the specified URL.
GET	Initiate a GET request.
POST	Initiate a POST request.

manger

Create a TQUICHTTPSessionManager instance and pass the default QUIC configuration.

```
+ (instancetype)manager
```

initWithSessionConfiguration

Create a TQUICHTTPSessionManager instance and pass the QUIC configuration.

```
- (instancetype)initWithSessionConfiguration:(nullable %!s(<nil>) *)configuration
```

Parameter	Description
configuration	QUIC request configuration. For details, see TQUICURLSessionConfiguration .

initWithBaseURL

Create a TQUICHTTPSessionManager instance and pass the specified URL.

```
- (instancetype)initWithBaseURL:(nullable NSURL *)baseURL;

- (instancetype)initWithBaseURL:(nullable NSURL *)baseURL

    sessionConfiguration:(nullable %!s(<nil>) *)configuration;
```

Parameter	Description
baseURL	The request domain name.
configuration	QUIC request configuration. For details, see TQUICURLSessionConfiguration .

GET

Create a GET request.

```
- (nullable TQUICURLSessionDataTask *)GET:(NSString *)URLString

    parameters:(nullable id)parameters

    headers:(nullable NSDictionary <NSString *, NSString *)headers

    timeoutInterval:(NSTimeInterval)timeoutInterval

    downloadProgress:(nullable %!s(<nil>))downloadProgress

    success:(nullable %!s(<nil>))success
```

failure:(nullable %!s(<nil>))failure	
Parameter	Description
URLString	The request URL.
parameters	The request parameters.
headers	The request headers.
timeoutInterval	The connection timeout.
downloadProgress	The callback for download progress.
success	The callback for request success.
failure	The callback for request failure.

POST

Create a POST Request.

<pre>- (nullable TQUICURLSessionDataTask *)POST:(NSString *)URLString parameters:(nullable id)parameters headers:(nullable NSDictionary <NSString *, NSString *>)headers timeoutInterval:(NSTimeInterval)timeoutInterval uploadProgress:(nullable %!s(<nil>))uploadProgress success:(nullable %!s(<nil>))success failure:(nullable TQUICURLSessionTask)failure</pre>	
Parameter	Description
URLString	The request URL.
parameters	The request parameters.
headers	The request headers.
timeoutInterval	The connection timeout.
uploadProgress	The callback to be executed to get upload progress.

success	The callback to be executed upon request success.
failure	The callback to be executed upon request failure.

TQUICURLSessionTaskSuccess

Execute the callback for task success.

```
typedef void (^TQUICURLSessionTaskSuccess)(%!s(<nil>) *task, id _Nullable responseObject)
```

Parameter	Description
task	The request task.
responseObject	The response data.

TQUICURLSessionTaskFailure

Execute the callback for task failure.

```
typedef void (^TQUICURLSessionTaskFailure)(%!s(<nil>) * _Nullable task, NSError *error)
```

Parameter	Description
task	The request task.
error	The error message.

TQUICURLSessionManager

Session management APIs

API	Description
initWithSessionConfiguration	Create an instance with the specified configuration.
dataTaskWithRequest	Initiate requests with the NSURLRequest parameter.
setTaskDidFinishCollectingMetricsBlock	Set the callback to collect statistics.
setTaskDidReceiveResponseBlock	Set the callback block to receive request response.

initWithSessionConfiguration

Create a `TQUICURLSessionManager` instance and pass the QUIC configuration.

```
- (instancetype)initWithSessionConfiguration:(nullable %!s(<nil>) *)configuration
```

Parameter	Description
configuration	QUIC request configuration.

dataTaskWithRequest

Initiate requests with the `NSURLRequest` parameter.

```
- (TQUICURLSessionDataTask *)dataTaskWithRequest:(NSURLRequest *)request
        uploadProgress:(nullable %!s(<nil>))uploadProgres
        downloadProgress:(nullable %!s(<nil>))downloadProgr
        completionHandler:(nullable %!s(<nil>))completionHan
```

Parameter	Description
request	For detailed configuration, see <code>NSURLRequest</code> .
uploadProgress	The callback to be executed to get upload progress.
downloadProgress	The callback to be executed to get download progress.
completionHandler	The callback to be executed upon request completion.

setTaskDidFinishCollectingMetricsBlock

Set the callback to collect statistics.

```
- (void)setTaskDidFinishCollectingMetricsBlock:(nullable %!s(<nil>))block
```

Parameter	Description
block	Execute the callback block to collect statistics when the request is complete.

setTaskDidReceiveResponseBlock

Set the callback block to receive request response.

```
- (void)setTaskDidReceiveResponseBlock:(nullable %!s(<nil>))block
```

Parameter	Description

block

Execute the callback block to receive request response.

TQUICURLSessionTaskDidReceiveResponseBlock

Execute the callback block to receive session task response.

```
typedef void (^TQUICURLSessionTaskDidReceiveResponseBlock) (NSError<nil>) *session, NSError<nil> *task, NSData<nil> *response)
```

Parameter	Description
session	The class that manages sessions.
task	The class that manages tasks.
response	The response result.

TQUICURLSessionTaskDownloadProgressBlock

Execute the callback block to get download progress.

```
typedef void (^TQUICURLSessionTaskDownloadProgressBlock) (NSProgress *downloadProgress)
```

Parameter	Description
downloadProgress	The download progress.

TQUICURLSessionTaskUploadProgressBlock

Execute the callback block to get upload progress.

```
typedef void (^TQUICURLSessionTaskUploadProgressBlock) (NSProgress *uploadProgress)
```

Parameter	Description
uploadProgress	The upload progress.

QUICURLSessionTaskCompletionHandler

Execute the callback block upon session completion.

```
typedef void (^TQUICURLSessionTaskCompletionHandler)(NSURLResponse *response, id re
```

Parameter	Description
response	For details about the response result, see NSURLResponse.
responseObject	The response data.
error	The error message.

TQUICURLSessionTaskDidFinishCollectingMetricsBlock

Execute the callback block to collect statistics upon session completion.

```
typedef void (^TQUICURLSessionTaskDidFinishCollectingMetricsBlock)(TQUICURLSession
```

Parameter	Description
session	Session management.
task	Task management.
metrics	The network statistics.

TQUICURLSessionConfiguration

QUIC request configuration.

Member variable	Description
quicVersion	<p>Set the QUIC version.</p> <p>Supported versions: 043, Q046, Q050, Q051, draft-29, RFC-V1 (RFC 9000).</p> <p>Values:</p> <ul style="list-style-type: none">TQUICVersionQ043 (default)TQUICVersion046 audio/video proxyTQUICVersion050 audio/video proxyTQUICVersion051 audio/video proxy

	TQUICVersionDraft29 audio/video proxy TQUICVersionRFCV1 audio/video proxy
congestionType	Supported congestion algorithms: CubicBytes, RenoBytes, BBR, PCC, GCC. Values: TQUICCongestionTypeBBR (default) TQUICCongestionTypeCubicBytes TQUICCongestionTypeRenoBytes TQUICCongestionTypePCC TQUICCongestionTypeGCC
connectTimeoutMillis	The connection timeout in milliseconds. Default value: 60000.
idleTimeoutMillis	The idle connection timeout in milliseconds. This setting will affect connection reuse. Default value: 90000.
ipv6Enabled audio/video proxy	Whether to support IPv6. Values: YES, NO (default).
dnsParser	The custom DNS parser. For details, see TQUICDNSParserDelegate.

TQUICDNSParserDelegate

Implement a custom DNS parsing.

```
- (NSString *)lookup:(NSString *)hostName
```

Parameter	Description
hostName	The domain name, which is called back to resolve an IP address.

TQUICURLSession

Task management APIs

API	Description
sessionWithConfiguration	Create an instance with the specified configuration.
sharedSession	Create an instance with the default configuration.

[dataTaskWithRequest](#)

Create a task with the request parameters.

sessionWithConfiguration

Create an instance with the specified configuration.

```
+ (instancetype)sessionWithConfiguration:(%!s(<nil>) *) configuration
```

Parameter	Description
configuration	The QUIC request configuration.

sharedSession

Create an instance with the default configuration.

```
+ (instancetype)sharedSession
```

dataTaskWithRequest

Create a task based on the request parameters.

```
- (nullable %!s(<nil>) *) dataTaskWithRequest:(NSURLRequest *) request
```

Parameter	Description
request	The request parameters. For more details, see NSURLRequest.

TQUICURLSessionTask

Task management APIs.

API	Description
resume	Start a request task.
cancel	Cancel a request task.

resume

Start a request task.

```
- (void) resume
```

cancel

Cancel a request task.

```
- (void)cancel
```

TQUICURLSessionDataTask

Manage request tasks. This API inherits the [TQUICURLSessionTask](#) class.

API	Description
resume	Start a request task.
cancel	Cancel a request task.

resume

Start a request task.

```
- (void)resume
```

cancel

Cancel a request task.

```
- (void)cancel
```

TQUICURLRequestSerialization

API	Description
TQUICHTTPRequestSerializer	Serialize HTTP request parameters.
TQUICJSONRequestSerializer	Serialize JSON request parameters.

TQUICHTTPRequestSerializer

Request serialization APIs

API	Description
serializer	Implement instantiation.
setValue	Set a value for the header field.
valueForHTTPHeaderField	Return the value that corresponds to the header field.
requestWithMethod	Create NSMutableURLRequest.
multipartFormRequestWithMethod	Create NSMutableURLRequest to transfer streaming data.

serializer

Create a TQUICHttpRequestSerializer instance.

```
+ (instancetype)serializer
```

setValue

Set a value for the header field.

```
- (void)setValue:(nullable NSString *)value forHTTPHeaderField:(NSString *)field
```

Parameter	Description
value	The value of the header field.
field	The name of the header field.

valueForHTTPHeaderField

Return the value that corresponds to the header field.

```
- (nullable NSString *)valueForHTTPHeaderField:(NSString *)field
```

Parameter	Description
field	The name of the header field.

requestWithMethod

Create NSMutableURLRequest.

```
- (nullable NSMutableURLRequest *)requestWithMethod:(NSString *)method
```

```
URLString:(NSString *)URLString

parameters:(nullable id)parameters

error:(NSError * _Nullable __autorele
```

Parameter	Description
method	Set the HTTP request method, such as GET, POST, PUT, DELETE, HEAD and PATCH.
URLString	The request URL.
parameters	The request parameters.
error	The error message.

multipartFormRequestWithMethod

```
- (NSMutableURLRequest *)multipartFormRequestWithMethod:(NSString *)method

URLString:(NSString *)URLString

parameters:(nullable NSDictionary <NSString *, NSString *>)parameters

constructingBodyWithBlock:(nullable void (^)(id <#nullable string>))constructingBodyWithBlock

error:(NSError * _Nullable __autorele
```

Parameter	Description
method	Set the HTTP request method, such as GET, POST, PUT, DELETE, HEAD and PATCH.
URLString	The request URL.
parameters	The request parameters.
constructingBodyWithBlock	Construct the body using the block.
error	The error message.

TQUICJSONRequestSerializer

API	Description
-----	-------------

[serializerWithOptions](#)

Create an instance.

serializerWithOptions

Create a TQUICJSONRequestSerializer instance with the JSON serialization options.

```
+ (instancetype)serializerWithOptions:(NSJSONWritingOptions)writingOptions
```

Parameter	Description
writingOptions	Create an instance with the JSON serialization options.

TQUICMultipartFormData

Multipart form data.

API	Description
appendPartWithFileURL	Upload the form data with the specified file URL.
appendPartWithInputStream	Upload the form data with the input stream.
appendPartWithFileData	Upload form data using multipart.
appendPartWithFormData	Add data parts.
appendPartWithHeaders	Add the header fields and body data to the form.
throttleBandwidthWithPacketSize	The bandwidth limit for uploads.

appendPartWithFileURL

Upload the form data with the specified file URL.

```
- (BOOL)appendPartWithFileURL:(NSURL *)fileURL
    name:(NSString *)name
    error:(NSError * _Nullable __autoreleasing *)error
```

Parameter	Description
fileURL	Add the file URL to the form.
name	The name of the associated file.
error	The error message.

appendPartWithInputStream

Upload data with the specified input stream.

```
- (void)appendPartWithInputStream:(nullable NSInputStream *)inputStream
    name:(NSString *)name
   fileName:(NSString *)fileName
    length:(int64_t)length
   mimeType:(NSString *)mimeType
```

Parameter	Description
inputStream	The input stream.
name	The name of the input stream.
fileName	The name of the file associated with the input stream.
length	The length of the stream in bytes.
mimeType	The MIME type, such as image/jpeg. For more details, see HTTP specifications.

appendPartWithFileData

Upload the form data using multipart.

```
- (void)appendPartWithFileData:(NSData *)data
    name:(NSString *)name
   fileName:(NSString *)fileName
   mimeType:(NSString *)mimeType
```

Parameter	Description
data	The data to be added to the form.
name	The name of the associated data.
fileName	The name of the associated file.
mimeType	The MIME type, such as image/jpeg. For more details, see HTTP specifications.

appendPartWithFormData

Add data parts.

```
- (void)appendPartWithFormData:(NSData *)data
                           name:(NSString *)name
```

Parameter	Description
data	The data to be added to the form.
name	The name of the associated data.

appendPartWithHeaders

Add the header fields and body data to the form.

```
- (void)appendPartWithHeaders:(nullable NSDictionary <NSString *, NSString *> *)headers
                           body:(NSData *)body;
```

Parameter	Description
headers	The headers to be added.
body	The body data to be added.

throttleBandwidthWithPacketSize

The bandwidth limit for uploads.

```
- (void)throttleBandwidthWithPacketSize:(NSUInteger)numberOfBytes
                                       delay:(NSTimeInterval)delay
```

Parameter	Description
numberOfBytes	The maximum size of a packet in bytes. Default value: 16 KB.
delay	The read delay for a packet. Default value: 0 seconds.

TQUICURLResponseSerialization

Response serialization APIs

API	Description
-----	-------------

TQUICHTTPResponseSerializer	Serialize HTTP response data.
TQUICJSONResponseSerializer	Serialize JSON response data.

TQUICHTTPResponseSerializer

Response serialization APIs

API	Description
serializer	Implement instantiation.
validateResponse	Validate the response data.

serializer

Create a TQUICHTTPResponseSerializer instance.

```
+ (instancetype)serializer
```

validateResponse

Validate the response data.

```
- (BOOL)validateResponse:(nullable NSHTTPURLResponse *)response  
  
    data:(nullable NSData *)data  
  
    error:(NSError * _Nullable __autoreleasing *)error
```

Parameter	Description
response	The response result to be validated. For more details, see NSHTTPURLResponse.
data	The returned data.
error	The validation error.

TQUICJSONResponseSerializer

API	Description
serializerWithOptions	Create an instance.

serializerWithOptions

Create an instance with the JSON serialization options.

```
+ (instancetype)serializerWithOptions:(NSJSONReadingOptions)readingOptions
```

Parameter	Description
serializerWithOptions	Create an instance with the JSON serialization options.

TQUICURLSessionTaskMetrics

The QUIC network metrics collected for a session.

Property	Description
transactionMetrics	An array of metrics for each request during the session. For more details, see TQUICURLSessionTaskTransactionMetrics .
taskInterval	The time taken between when the task is created and when the task is completed.
redirectCount	Number of redirects.

TQUICURLSessionTaskTransactionMetrics

The QUIC network metrics collected for a request.

Property	Description
isValid	Whether the value of status is valid.
isQuic	Whether it is a QUIC request.
is0rtt	Whether it is a 0-RTT connection.
isConnReuse	Whether the connection is reused.
connectMillis	Get the connection duration in milliseconds.
dnsMillis	Get the DNS duration in milliseconds.
dnsCode	Get the DNS error code.
ttfbMillis	Get the time taken for the first byte to be received in milliseconds.
completeMillis	Get the time taken for the request to be completed in milliseconds. The time taken by the connection is not included.

srttMicros	Get the average round-trip time in milliseconds.
packetsSent	Get the number of packets sent in bytes.
packetsRetransmitted	Get the number of packets retransmitted in bytes.
bytesSent	Get the number of bytes sent.
bytesRetransmitted	Get the number of bytes retransmitted.
packetsLost	Get the number of packets lost in bytes.
packetsReceived	Get the number of packets received in bytes.
bytesReceived	Get the number of bytes received.
streamBytesReceived	Get the number of bytes received within the stream.

IPv6 Access

Last updated : 2024-10-28 15:34:17

Function Introduction

EdgeOne supports one-click enable of IPv6 access, allowing IPv6 clients to access nodes using the IPv6 protocol.

Note :

Currently, the majority of EdgeOne nodes support IPv6 access. You can contact us to confirm the specific resource coverage.

Usage Scenarios

IPv6-only network environment: Some regions and organizations may already be using IPv6-only network environments, and devices in these network environments may not be able to directly access IPv4-based services. By enabling IPv6 access, you can ensure that these clients can normally access your accelerated resources.

Dual-stack network environment: For dual-stack network environments that support both IPv4 and IPv6, clients can automatically select whether to use IPv4 or IPv6 protocol to access accelerated resources based on network conditions. In some cases, IPv6 connections may be faster than IPv4, so enabling IPv6 access can help improve the access performance of these clients.

Future network compatibility: As IPv4 address resources gradually become exhausted, more and more networks and devices will adopt IPv6. By enabling IPv6 access, you can ensure that your acceleration service remains compatible with these emerging networks and devices in the future.

Policy and compliance requirements: Some regions or industries may require IPv6 support in services to meet policy or compliance requirements. In this case, enabling IPv6 access can help you meet these requirements.

Directions

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the IPv6 Access configuration card. This protocol is disabled by default. Toggle the **switch** to enable it.

IPv6 access

EdgeOne nodes support IPv6 access. [Details](#)



[Custom settings](#)

Maximum Upload Size

Last updated : 2024-08-28 22:04:03

Function Introduction

The maximum upload size is the maximum value of data that a client user can upload in a single request. If the set limit is exceeded, EdgeOne will respond to the client with a 413 (Request Entity Too Large).

Note :

1. Only EdgeOne Enterprise and Standard plans support disabling the upload size limit.
2. The maximum upload size limit is enabled by default, with a limit of 800MB.

Usage Scenarios

1. **Large file upload:** For businesses involving large file uploads, such as online video platforms, large-scale game distribution, and big data analysis, you can increase the upload size limit or disable it directly to allow your large files to be uploaded smoothly.
2. **Defend against malicious uploads:** For businesses with frequent user interactions, such as social media, forums or blogs, you may encounter malicious users uploading excessively large files. You can enable the size limit and reduce the limit to prevent large files from being uploaded to the origin, thereby alleviating the pressure on the origin, preventing potential security risks, and improving the user experience.
3. **Save traffic:** For traffic-sensitive businesses, such as online education, online meetings, and API services, you may want to save traffic by limiting the size of uploaded files. You can enable the size limit and adjust the limit to a smaller value to reduce unnecessary transmission traffic and lower traffic costs.

You can flexibly configure the "maximum upload size" to meet the needs of various business scenarios.

Directions

Scenario 1: Configure the maximum upload size for all domain names of the site

If you need to configure the same maximum upload size for the whole connected site or as a site-level fallback configuration, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. In the site details page, click **Site Acceleration** to enter the Site Global Configuration Page. In the right sidebar, click **Network Optimization**.
3. Find the **maximum upload size** configuration card and click **Global Settings**.

Maximum upload size

The maximum data size that a user can upload in a single request. [Details](#)

800MB

[Global settings](#)

[Custom settings](#)

4. In the pop-up window, check **Enable Size Limit** and modify the limit value. Click **Save** to make it effective.

Maximum upload size



Only Enterprise and Standard plan support disabling the size limit. After enabling the size limit, the maximum configurable size is 800 MB.

Size Limit ☒ **Enable**

Maximum

Save

Cancel

Scenario 2: Configure the maximum upload size for specified domain names, paths, or file extensions, etc.

If you need to configure different maximum upload sizes for different domain names, paths, or file extensions, etc., for example, configure the maximum upload size for the `www.example.com` domain under the `example.com` site to be 20 MB, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. In the rule editing page, enter the rule name, select Host as the matching type, and configure it as `www.example.com`.
5. Click **Action**, and in the pop-up operation list, select Maximum Upload Size as the operation, and click **Enable Size Limit**, with a limit value of 20 MB.

IF + Comment

Matching type	Operator	Value
HOST	Is	192.168.1.1

+ And + Or

Action	Size Limit	Maximum
Maximum upload size	<input checked="" type="checkbox"/>	20 MB

+ Action

+ IF

6. Click **Save and Publish** to complete the rule configuration.

WebSocket

Last updated : 2024-10-28 15:34:17

Function Introduction

EdgeOne supports WebSocket protocol access, which enables the server to actively push data to the client. WebSocket protocol is a persistent protocol based on TCP, which implements full-duplex communication between the client and the server, allowing the server to actively send information to the client. Before the WebSocket protocol, Web Apps that implemented client-server duplex communication had to constantly send HTTP requests for inquiries, which led to increased service costs and inefficiency. Due to the advantages of full-duplex communication, WebSocket is widely used in social subscription, collaborative office, market updates, interactive live streaming, online education, Internet of Things, and other scenarios, which can better save server resources and bandwidth, and achieve more real-time communication.

Note:

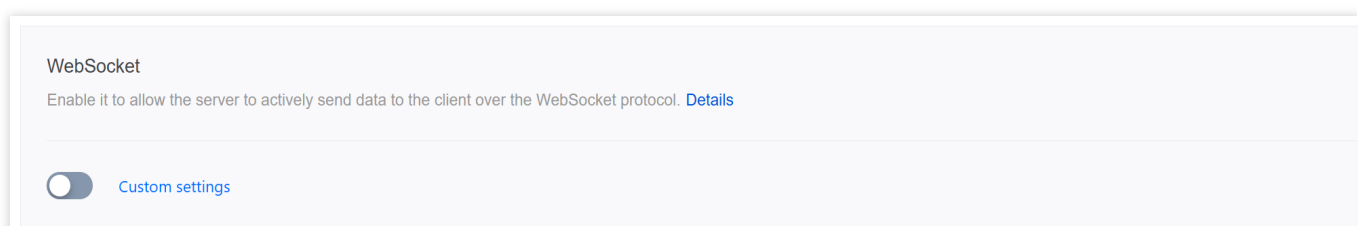
1. Currently, only HTTP/1.1-based WebSocket is supported, and HTTP/2 WebSocket is not supported.
2. Maximum connection timeout duration supported: 300 seconds.

Directions

Scenario 1: Configure WebSocket for all domain names of the site

If you need to enable/disable WebSocket for the whole connected site, or as a site-level fallback configuration, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the **WebSocket** configuration card and toggle the **switch** to enable the WebSocket feature.



Activated: By default, WebSocket protocol is not supported. When enabled, WebSocket protocol is supported.

Maximum connection timeout duration: If no data is sent or received within the timeout period, the connection will be disconnected.

Scenario 2: Configure WebSocket for specified domain names

If you need to configure WebSocket for different domain names, for example, configure WebSocket for the `www.example.com` domain under the `example.com` site, please refer to the following steps:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as Host and configure it as `www.example.com`.
5. Click **Action** > **choice box** and select the action as WebSocket in the dropdown action list. Then toggle the switch to enable it and configure the maximum connection timeout duration.

Note:

Maximum connection duration: Can be configured between 1-300 seconds.

The screenshot displays the 'Rule Engine' configuration page. It features two main sections: 'IF' (Conditions) and 'Action'.
In the 'IF' section, there is a table with columns: 'Matching type', 'Operator', and 'Value'. The first row shows 'HOST' as the matching type, 'Is' as the operator, and 'www.example.com' as the value. Below this table are buttons for '+ And' and '+ Or' to add more conditions.
In the 'Action' section, there is a table with columns: 'Action', 'On/Off', and 'Maximum connection timeout'. The first row shows 'WebSocket' as the action, a toggle switch set to 'On', and '300 seconds' as the maximum connection timeout. Below this table are buttons for '+ Action' and '+ IF' to add more actions or conditions.

6. Click **Save and Publish** to complete the rule configuration.

Origin-Pull Request Headers Carrying Client IP

Last updated : 2024-10-24 10:34:23

Overview

If you need to collect the client IP information within the origin and distinguish it by service provider source, EdgeOne supports placing the client IP information in a custom HTTP header and returning it to the business origin when the X-Forwarded-For header or other standard headers are not applicable.

Note:

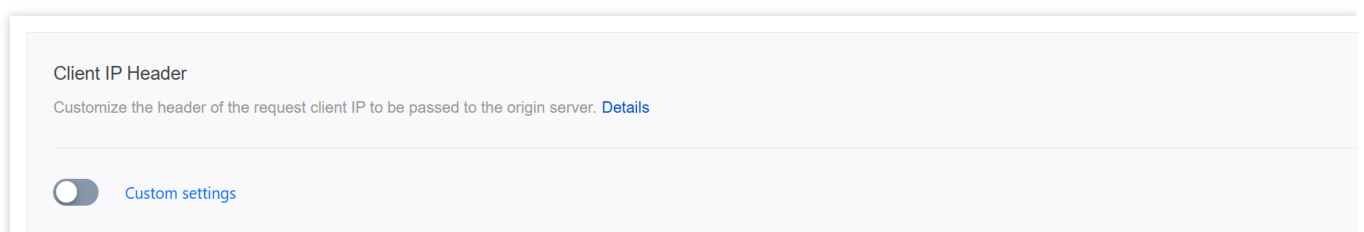
For default headers carried in EdgeOne origin-pull requests, see [Default HTTP Response Headers](#). For modification of the origin-pull request header, see [Modifying HTTP Origin-Pull Request Headers](#).

Directions

Scenario 1: Origin-Pull Request Headers Carrying the Client IP Information for All Domain Names of the Site

If you need to set a custom origin-pull request header carrying the client IP information for the entire access site, you can follow the steps below:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the client IP header configuration card. This feature is disabled by default. Toggle the **switch** to enable it.



4. In the pop-up window, set a custom header name such as `EO-Client-IP` and click **Save** to apply it.

Scenario 2: Origin-Pull Request Headers Carrying the Client IP Information for a Specified Domain Name

If you need to set a custom origin-pull request header carrying the client IP information only for a specified domain name, you can follow the steps below:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST to match requests for the specified domain name.
5. Click **Action** > **choice box** and select the action as **Client IP Header** in the dropdown action list. Toggle the **switch** to enable it and enter a header name such as `EO-Client-IP` .

The screenshot displays the 'IF' configuration section of the Rule Engine. It includes a table for matching rules with columns for Matching type, Operator, and Value. Below the table are buttons for '+ And' and '+ Or'. The 'Action' section shows a dropdown menu with 'Client IP Header' selected, a toggle switch for 'On/Off' that is currently on, and a text input for 'Header name' containing 'EO-Client-IP'. There are also buttons for '+ Action' and '+ IF'.

6. Click **Save and Publish** to complete the rule configuration.

Origin-Pull Request Headers Carrying Client IP Geo Location

Last updated : 2024-10-24 10:35:32

Overview

If you need to collect the client geo location information within the origin for responding with different content based on the client geo location and analyzing regional distribution of users, you can set a custom origin-pull request header that carries the client IP geo location information back to the business origin.

Note:

The client IP geo location header is currently displayed at the country/region level. Its value is a two-letter country/region code. For details, see [ISO 3166-1 alpha-2](#).

IPv6 address resolution is not currently supported.

For default headers carried in EdgeOne origin-pull requests, see [Default HTTP Origin-Pull Request Headers](#). For modification of the origin-pull request header, see [Modifying HTTP Origin-Pull Request Headers](#).

Directions

Scenario 1: Origin-Pull Request Headers Carrying the Client Geo Location Information for All Domain Names of the Site

If you need to set a custom origin-pull request header carrying the client geo location information for the entire access site, you can follow the steps below:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the client IP geo location configuration card. This feature is disabled by default. Toggle the **switch** to enable it.

Client IP location Header

The custom header carries the client IP geolocation information (two-letter country code: ISO 3166-1 alpha-2 codes) back to the origin. [Details](#)

Note: IPv6 is not currently supported



Custom settings

4. In the pop-up window, set a custom header name such as `EO-Client-IPCountry` and click **Save**.

Scenario 2: Origin-Pull Request Headers Carrying the Client Geo Location Information for a Specified Domain Name

If you need to set a custom origin-pull request header carrying the client geo location information only for a specified domain name, you can follow the steps below:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST to match requests for the specified domain name.
5. Click **Action > choice box** and select the action as **Client IP location Header** in the dropdown action list. Toggle the **switch** to enable it and enter a header name such as `EO-Client-IPCountry`.

The screenshot displays the 'IF' configuration panel in the Tencent Cloud EdgeOne console. It includes a '+ Comment' button at the top. Below, there are two main sections: 'Matching type' and 'Action'. The 'Matching type' section has a dropdown menu set to 'HOST', an 'Operator' dropdown set to 'Is', and an empty 'Value' input field. Below this are '+ And' and '+ Or' buttons. The 'Action' section has a dropdown menu set to 'Client IP location Header', a toggle switch that is currently turned on, and a 'Header name' input field containing the text 'EO-Client-IPCountry'. At the bottom of the panel, there are '+ Action' and '+ IF' buttons.

6. Click **Save and Publish** to complete the rule configuration.

gRPC

Last updated : 2024-10-28 15:34:17

EdgeOne Support Status for gRPC

EdgeOne supports enabling gRPC protocol support within the console (default is off). Once enabled, it can simultaneously support HTTP/HTTPS/gRPC protocols, automatically adapting to the user's request protocol, i.e., using HTTP protocol for HTTP requests and gRPC protocol for gRPC requests.

Note :

1. Currently, only Simple RPC and Server-side streaming RPC modes are supported;
2. gRPC is implemented based on end-to-end HTTP/2, so please make sure that [HTTP/2 access](#) and [HTTP/2 origin-pull](#) are enabled when enabling gRPC.

What is gRPC?

gRPC (gRPC Remote Procedure Calls) is an open-source remote procedure call system initiated by Google. The system is designed based on the HTTP/2 standard and features bidirectional streaming, flow control, header compression, and multiplexed requests on a single TCP connection.

Directions

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page. Then click **Network Optimization** in the right sidebar.
3. Locate the gRPC card and toggle the **switch** to enable gRPC support.

gRPC

Provide gRPC support for HTTP/HTTPS/gRPC requests. [Details](#)

Note that only simple RPC and server-side streaming RPC are available.



URL Rewrite

Access URL Redirection

Last updated : 2024-09-10 11:06:45

Overview

EdgeOne Nodes redirect the client requests request URL to the target URL by responding to 3XX status codes. This feature can change the URL redirection that originally needed to be generated and returned by the origin server in your business scenario to be directly constructed and returned by the EdgeOne edge nodes, reducing the network latency of following the origin-pull and the load of the origin server generating the URL redirection, and improving the access performance of the client.

Use Cases

The following are common Applicable Scenarios for Rewrite access URL:

Migration of website or reconstruction: When a website undergoes migration or reconstruction, the URL structure may change. To maintain the validity of old links, you can use URL redirect to redirect old URLs to new URLs, ensuring that users and Search Engines can smoothly access the new resources.

Geographical location or Device Type orientation: Based on the user's geographical location or device type, you can use URL redirect to guide users to different resources or pages. For example, provide optimized mobile pages for mobile device users, or provide different language versions of pages based on the user's region.

Temporary maintenance or activity page: When a website is undergoing temporary maintenance or hosting a specific event, you can use URL redirect to guide users to the maintenance notification page or activity page, thereby improving the user Experience.

Directions

Scenario One: Domain Services Require a 302 Redirection to a Temporary Maintenance Page

If you require temporary maintenance for the domain business `www.example.com` under your `example.com` site, and want to redirect all requests under the domain to

`https://www.example.com/public/waitingpage/index.html` via a 302 redirection, you can refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the **Matching type** as HOST is `www.example.com`.
5. Click on **Action**, and in the pop-up action list, select the Action as **Redirect access URL**.
6. Configure the rules for Redirect access URL. Select the Target protocol as HTTPS, the Target hostname as Follow request, and the Destination Path as `/public/waitingpage/index.html`.
7. Click **Save and Publish** to complete the configuration. The complete rule configuration is as shown below:

Scenario Two: Migration of Origin Server Resource Directory, Client Request URL Must Remain Unchanged

If you need to migrate all jpg image resources from the `test` directory to the `newtest` directory under the `www.example.com` domain of your `example.com` site, but the client request URL needs to remain unchanged, that is, the `test` directory is still accessed. You may refer to the following steps:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the **Matching type** as HOST is `www.example.com`.
5. Click on **Action**, and in the pop-up action list, select the action as **Redirect access URL**.
6. Configure the rules for Redirect access URL. Select the Target protocol as Follow request, the Target hostname as Follow request. And you can configure the Destination Path as Replace by regex, input the Regular Expression as `/test/([^/]*).jpg` to match the destination path, and replace it with `/newtest/$1.jpg`.
7. Click **Save and Publish** to complete the configuration. The complete rule configuration is as shown below:

IF + Comment

Matching type: HOST, Operator: Is, Value: [input field]

+ And + Or

Action: Redirect access URL

Target protocol: Follow request, Target hostname: Follow request

Destination Path: Replace by regex, Regular Expression: /test/([^/]*).jpg, Replaced with: /newtest/\$1.jpg

Query string: [checked], Status code: 302

+ Action + IF

Relevant References

The explanations for each configuration item of the redirect access URL are as follows:

Configuration Item	Description
Target Protocol	The request protocol of the target redirect address, default to follow the request, can support specifying the jump to HTTP/HTTPS protocol.
Target Hostname	The Hostname part of the target redirect address, default to follow the request, support modifying to a custom domain. For example, www.example.com.
Destination Path	<p>The path part of the target redirect address, providing three mode options:</p> <p>Follow request: Default configuration, follow the request path.</p> <p>Custom: Customize a complete path, replace the original request path with the target path. For example, <code>/download</code>.</p> <p>Regular Expression: Support matching and replacing paths through Google RE2 Regex. At the same time, it supports using <code>\$num</code> to refer to the Regex capture group, where <code>num</code> represents the group number, up to \$9.</p> <p>For example: Currently, we hope to replace the path <code>/old-path/1234</code> with <code>/new-path/1234</code>, we can configure the Regex expression as <code>^/old-path/(\\d+)\$</code>, and the replacement path can be configured as <code>/new-path/\$1</code>, where <code>\$1</code> refers to the first capture group in the Regex expression, that is, the number part of the path.</p>
Query String	Whether to carry the original query string to the target URL, default to enable, that is, carry the original query string after redirecting.
Status Code	Select the response status code for the redirect: 302 (default), 301, 303, and 307.

Origin-Pull URL Rewrite

Last updated : 2025-06-03 16:53:49

Overview

When the client sends the request to the EdgeOne node, and the request fails to hit the node cache and needs to be returned to the origin, it supports the redirection of the request to the destination URL of the origin server according to the rules set by origin-pull URL rewrite. This feature does not affect the node cache.

Use cases

In certain cases, the URL accessed by the client has been published and should not be modified, but the origin server has changed its URL for certain reasons; or the URL accessed by the client differs from that on the origin server for SEO. Then, you can set origin-pull URL rewrite rules, so that the node can rewrite the origin-pull URL to the actual resource URL on the origin server without changing the URL accessed by the client.

Directions

Scenario: The client request URL is inconsistent with the corresponding origin server resource path

For example, when a client requests the URL path `https://www.example.com/online/index.html` from the site domain `www.example.com`, and the file directory has been changed, it is necessary to remove the directory prefix `/online` during the origin-pull to access the corresponding file resources. The following steps can be referred to:

1. Log in to the [EdgeOne console](#) and click Site List in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the **Matching type** as HOST is `www.example.com`.
5. Click on **Action**, and in the pop-up action list, select the action as Rewrite origin-pull URL.
6. Select the **Type** as Remove path prefix, with the path prefix being `/online`, as configured below:
7. Click **Save and Publish** to complete the configuration.

Relevant References

The explanations for each configuration item of the origin-pull URL rewrite are as follows:

Type	Description
Add path prefix	<p>Add specified path prefix to request URL Path. The path prefix refers to the first directory after the domain name. For example, if the request URL is <code>https://www.example.com/path0/index.html</code> and the added path prefix is <code>/prefix</code>, then the resulting rewritten URL will be <code>https://www.example.com/prefix/path0/index.html</code>.</p>
Remove path prefix	<p>Remove the specified path prefix from the request URL. The path prefix refers to the first directory after the domain name and only supports exact matching.</p> <p>For example, if the request URL is <code>https://www.example.com/path0/path1/index.html</code> and the specified path prefix to remove is <code>/path0</code>, the rewritten URL will be <code>https://www.example.com/path1/index.html</code>.</p> <p>For instance, if the request URL is <code>https://www.example.com/path000/path1/index.html</code> and the specified path prefix to remove is <code>/path0</code>, it will not match, and the rewriting rule will not take effect.</p>
Replace full path	<p>Replace the complete request URL. For example, if the request URL is <code>https://www.example.com/path0/index.html</code> and the path to be replaced is <code>/new/page.html</code>, the rewritten URL will be <code>https://www.example.com/new/page.html</code>.</p>
Regular replacement	<p>Supports matching and replacing paths using Google RE2 regular expressions. It also allows referencing regular expression capture groups using <code>\$num</code>, where <code>num</code> represents the group number, such as <code>\$1</code>. The referenced group number must not exceed the total number of capture groups in the regular expression, and you must ensure that the replaced path starts with a <code>/</code>.</p> <p>For example:</p> <p>If you want to replace the path <code>/old-path/1234</code> with <code>/new-path/1234</code>, you can configure the regular expression as <code>^/old-path/(\\d+)\$</code> and the replacement path as <code>/new-path/\$1</code>, where <code>\$1</code> refers to the first capture group in the regular expression, i.e., the numeric part of the path.</p>

Modifying Header

Modifying HTTP Response Headers

Last updated : 2025-05-07 09:44:48

Overview

Support customization/adding/deleting HTTP node response headers (responding to the customer's direction), modifying HTTP node response headers will not affect the node cache.

Note :

EdgeOne has automatically carried some response headers by default, and you don't need to configure them. For details, please refer to: [Default HTTP Response Headers](#).

Scenario 1: Cross-Domain Header Response Only Allows Specified Domain Names to Access Page Resources

If your business scenario involves cross-domain access and the resources of the current business domain name `www.example.com` only allow the pages from `example.com` and `site.com` to access the acceleration domain name, you can refer to the following steps.

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.

3. On the rule engine page, click **Create rule** and select **Add blank rule**.

4. On the rule editing page, select the matching type as HOST equals `www.example.com`.

At the same time, select the matching type as HTTP request header Origin equals `*.example.com` and `*.site.com`.

5. Click the **Action checkbox** and select **Modify HTTP nodes response header** in the pop-up operation list.

6. Select the type as **Set** and the header name as `Access-Control-Allow-Origin`, and set the header value to `${http.request.headers["Origin"]}`.

7. Click **Save and publish** to complete the rule configuration.

Scenario 2: Cross-Domain Header Response Supports All Domain Names to Access Page Resources

If your business scenario involves cross-domain access and the resources of the current business domain name `www.example.com` allow all pages to access the acceleration domain name, you can refer to the following steps.

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.
3. On the rule engine page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST equals `www.example.com`.
5. Click the **Action**, and in the pop-up operation list, select the operation to **modify the HTTP node response header**.
6. Select the type as set, and set the `Access-Control-Allow-Origin` as `*`.
7. Click **Save and publish** to complete the rule configuration.

Related References

Supported Types Description:

Type	Description
Set	Change the value of the specified header parameter to the set value, and the header is unique. Note: If the specified header does not exist, the header will be added.
Add	Add the specified header. Note: If the header already exists, it will still be added and will not overwrite the existing header.
Delete	Delete the specified header.

Supported Header Types Description:

Header Type	Description
Custom	Supports modifying custom header content, fill in the description as follows: Name: 1 - 100 characters, consisting of numbers 0 - 9, characters a - z, A - Z, and special symbols -. Value: Supports 1 - 1000 characters, does not support Chinese.

Specify Header	<p>Supports modifying the following specified headers:</p> <p>Access-Control-Allow-Origin: Used to specify the source (domain name) allowed to access resources, must contain http:// or https://. Supports setting Wildcard <code>*</code>, i.e., allowing all domain requests.</p> <p>Access-Control-Allow-Methods: Used to set the HTTP request methods allowed for cross-domain access, such as POST, GET, OPTIONS.</p> <p>Access-Control-Max-Age: Specifies how many seconds the preflight request result is valid, in seconds.</p> <p>Content-Disposition: Activates the browser's download pop-up window and can set the default download file name. For example: Content-Disposition: attachment;filename=FileName.txt</p> <p>Content-Language: Defines the language code used by the page. For example: Content-Language: zh-CN</p>
----------------	--

Limitations

In the same Modify HTTP Request Header operation, multiple different types of operations can be added, up to 30, and the execution order is from top to bottom.

Some standard headers are not supported for configuration, as follows:

```
Date
Cache-Control(Only deletion is not supported)
CDN-Loop
xx-script-xxx
EO-Inner-xxx
EO-Cache-Status
EO-LOG-UUID
EO-Debug-xxx
Content-Length
Content-Range
```

Modifying HTTP Request Headers

Last updated : 2025-05-07 09:45:39

Overview

When Edgeone nodes origin-pull, if the origin needs to obtain some specific information through the HTTP request header for business logic judgment or data analysis, such as: client device type, acceleration service provider. This can be achieved by customizing/adding/deleting HTTP origin-pull request headers (node origin-pull direction), where the header value supports variables, please refer to EdgeOne preset variables for details.

Note :

EdgeOne supports carrying `X-Forwarded-For` and `X-Forwarded-Proto` by default for origin-pull, and you do not need to configure it again, please see: [Default HTTP Headers of Origin-Pull Requests](#).

Directions

Scenario 1: Carry device type information to the origin when origin-pulling

For example: If you want the current domain name `www.example.com` to aggregate the `User-Agent` header value carried by the client request into device type-related headers and pass them to the origin when origin-pulling, you can refer to the following steps for configuration:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as Host equals `www.example.com`.
5. Click on the **Action**, and in the pop-up operation list, select the operation to modify the HTTP origin-pull request header.
6. Select the type as Add, the preset header with the header name `EO-Client-Device`, and the configuration result is as follows:
7. Click **Save and Publish** to complete the rule configuration.

Scenario 2: Pass the acceleration domain name to the origin through a custom header when origin-pulling

For example: If your current domain name `www.example.com` has been configured to origin-pull Host for other domain names, and you want to pass the acceleration domain name to the origin through a custom header

`Tencent-Acceleration-Domain`, you can refer to the following steps for configuration:

1. Log in to the [EdgeOne console](#) and click **Site List** in the left sidebar. In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as Host equals `www.example.com`.
5. Click on the **Action**, and in the pop-up operation list, select the operation to modify the HTTP origin-pull request header.
6. Select the type as Add, the header name as `Tencent-Acceleration-Domain`, and the header value as `${http.request.host}`, and the configuration result is as follows.
7. Click **Save and Publish** to complete the rule configuration.

Related Reference

The types supported by modifying the HTTP origin-pull request header are described as follows:

Type	Description
Set	Change the value of the specified header parameter to the set value, and the header is unique. Note: If the specified header does not exist, the header will be added.
Add	Add the specified header. Note: If the header already exists, it will still be added and will not overwrite the existing header.
Delete	Delete the specified header.

Supported header name description:

Header Type	Description
Custom	Custom header. Name: 1 - 100 characters, consisting of numbers 0 - 9, characters a - z, A - Z, and special symbol -. Value: 1 - 1000 characters, Chinese not supported.
Preset	Headers aggregated based on client User-Agent information:

Header	<div>Client Device Type: EO-Client-Device</div> <div>Value: Mobile , Desktop , SmartTV , Tablet or Others</div> <div>Client Operating System: EO-Client-OS</div> <div>Value: Android , iOS , Windows , MacOS , Linux or Others</div> <div>Client Browser Type: EO-Client-Browser</div> <div>Value: Chrome , Safari , Firefox , IE or Others</div>
--------	---

Limitations

In the same Modify HTTP Request Header operation, multiple different types of operations can be added, up to 30, and the execution order is from top to bottom.

Some standard headers are not supported for configuration, as follows:

```
EO-Connecting-IP
X-Forwarded-For
Expect
EO-LOG-UUID
X-Tencent-Ua
```

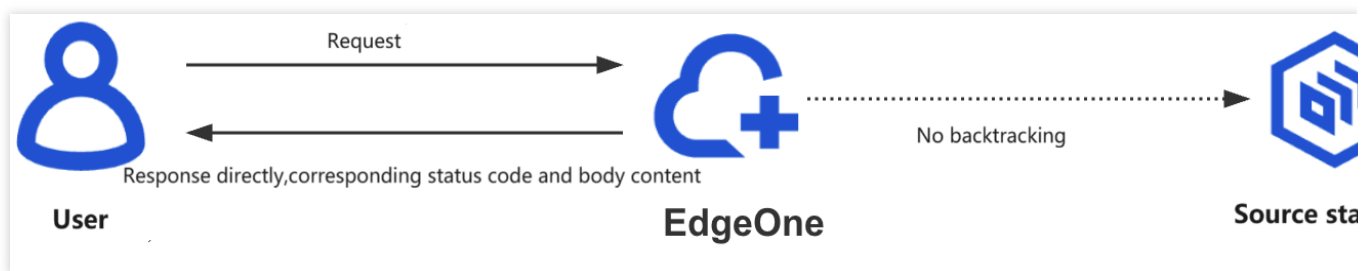
Modify the response content

HTTP Response

Last updated : 2024-08-26 16:59:43

Overview

The HTTP response feature is supported by the rule engine. When the corresponding conditions are matched, the EdgeOne node directly responds with the specified status code and page content.



Feature Description

Response status code: supports 2XX, 4XX, and 5XX, excluding 499, 514, 101, 301, 302, 303, 509, and 520-599.

Response page: supports the Content-Type options including text/html, application/json, text/plain, and text/xml.

Use Cases

The following are some common use cases for HTTP responses:

Configuring the Flash cross-domain policy file: The crossdomain.xml file is used to specify the domain names or IP addresses that can access the website resources.

Basic access control: such as IP allowlist/blocklist, Referer allowlist/blocklist, UA allowlist/blocklist, and region access control.

Temporary maintenance or activity page: When the website is under temporary maintenance or hosting specific activities, the HTTP response feature can be used to guide users to the maintenance notification page or activity page, thereby enhancing user experience.

Preparations

Go to the [Custom Response Page](#) to create a necessary custom page for the `HTTP response action`.

Scenario 1: The Domain Name Requires Configuring the crossdomain.xml File

If your business at the `www.example.com` domain name under the `example.com` site needs to control cross-domain access through the crossdomain.xml file, you can refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.
3. On the rule engine page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST and set its value to equal `www.example.com`.
5. Click the **choice box** below **Action** and select the action as **HTTP Response** in the pop-up action list.
6. Configure the response status code to 200 and select the created crossdomain.xml file from the dropdown list for the response page.
7. The complete rule configuration is shown below. Click **Save and publish** to finish the rule configuration.

The screenshot displays the 'Rule Engine' configuration interface. It is divided into two main sections: 'IF' (Condition) and 'Action'.

- IF Section:** Contains a matching rule with the following fields:
 - Matching type: `HOST`
 - Operator: `Is`
 - Value: `www.example.com`
- Action Section:** Contains the configuration for the selected action:
 - Action: `HTTP Response`
 - Response status code: `200`
 - Response page: `crossdomain`

Buttons for '+ And', '+ Or', '+ Action', and '+ IF' are visible at the bottom of the configuration area.

Scenario 2: The Domain Name Requires Configuring a Referer Allowlist

If your business at the `www.example.com` domain name under the `example.com` site only allows access by requests with `https://www.example.com/` as Referer, and directly denies other requests with 403, you can refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.

2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.
3. On the rule engine page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST and set its value to equal `www.example.com` . Additionally, select the matching type as HTTP Request Header, select the header name as Referer, and set the header value to equal `https://www.example.com/` .
5. Click the **choice box** below **Action** and select the action as **HTTP Response** in the pop-up action list.
6. Configure the response status code to 403 and select the response page from the dropdown list. If no page is available, click **Create page** to create a page first and then reference it.
7. The complete rule configuration is shown below. Click **Save and publish** to finish the rule configuration.

The screenshot shows the 'Rule Engine' configuration page. It has two main sections: 'IF' (conditions) and 'Action'.

IF Section:

- Condition 1: Matching type 'HOST', Operator 'Is', Value 'www.example.com'.
- Condition 2: Matching type 'HTTP Request Header', Header name 'Referer', Operator 'Is not', Header value 'https://www.example.com'.
- Logic: '+ And' (both conditions must be true).

Action Section:

- Action: 'HTTP Response'.
- Response status code: '403'.
- Response page: 'Custom-pages1'.

Buttons for '+ Comment', '+ And', '+ Or', '+ Action', and '+ IF' are visible.


Scenario 3: The Domain Name Requires Configuring a UA Blocklist

The resources of your business at the `www.example.com` domain name under the `example.com` site are maliciously crawled by Google crawlers, causing a sudden bandwidth increase of the domain name and severely affecting the bills. Through analysis, it is found that the crawler requests contain `spider` in the User-Agent. To block such requests, you can refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and then click the **site** you want to configure in the site list.
2. On the site details page, click **Site Acceleration** to enter the global configuration page. Then click the **Rule Engine** tab.
3. On the rule engine page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST and set its value to equal `www.example.com` . Additionally, select the matching type as HTTP Request Header, select the header name as User-Agent, and set the header value to match the regular expression `*spider*` .
5. Click the **choice box** below **Action** and select the action as **HTTP Response** in the pop-up action list.

6. Configure the response status code to 403 and select the response page from the dropdown list. If no page is available, click **Create page** to create a page first and then reference it.
7. The complete rule configuration is shown below. Click **Save and publish** to finish the rule configuration.

IF [+ Comment](#)

Matching type ⓘ	Operator	Value		
HOST	Is			
Matching type ⓘ	Header name	Operator	Header value	
HTTP Request Header	User-Agent	Is	*spider*	

[+ And](#) [+ Or](#)

Action ⓘ

HTTP Response

Response status code	Response page
403	Custom-pages1

[+ Action](#)

[+ IF](#)

Custom Error Page

Last updated : 2024-08-26 11:10:08

Overview

EdgeOne provides support to redirect to a specified custom page and return a 302 status code when the origin server responds with specific error statuses. This helps your site inform users of the current website status by the customized error page, avoiding uncertainty regarding the specific cause and resolution when a request error occurs.

Note

This feature is the redirection of origin-pull error status. It does not support redirection of status codes generated by access control policies such as token authentication or web protection rules.

Directions

For instance, your self-built e-commerce service website provides online payment capability through `shop.example.com` . During peak user traffic, the original server could potentially become overloaded, responding with a 503 status code. In order to avoid losing users, I hope to customize an error page `https://www.example.com/error.html` to guide users to other e-commerce platforms. You can follow the steps:

1. Log in to the [TencentCloud EdgeOne console](#) and click **Site List** in the left sidebar.In the site list, click the target **Site**.
2. On the site details page, click **Site Acceleration** to enter the global site configuration page, then click the **Rule Engine** tab.
3. On the rule engine management page, click **Create rule** and select **Add blank rule**.
4. On the rule editing page, select the matching type as HOST equal to `shop.example.com` .
5. Click **Action**, and select **Custom Error Page** in the pop-up action list.
6. To configure the custom error page, select the Status code as 503 and enter `https://www.example.com/error.html` for the Page URL. The relevant configuration items are described as follows:

Configuration Item	Description
Status code	Specifies the error status code returned by the origin: 4XX:400, 403, 404, 405, 414, 416, 451 5XX:500, 501, 502, 503, 504
Page URL	Designate the error page address, for

instance: `https://www.example.com/custom-page.html`

7. The complete rule configuration is as shown below. Click **Save and publish**, the rule configuration will be completed.

IF

+ Comment

Matching type ⓘOperatorValue

HOST

Is

+ And

+ Or

Action ⓘ

Custom Error Page

Status codePage URL ⓘ

503

https://www.example.com/error.html

+ Add

+ Action

+ IF

Rule Engine

Overview

Last updated : 2024-08-01 21:32:16

Overview

The rule engine is designed to meet more flexible and fine-grained business requirements through a rich rule language. You can customize the match type as needed and apply it to the corresponding operations. Compared to the configuration of site acceleration, the priority of the rule engine is higher, meaning that the custom policies created by the rule engine will override the configurations of site acceleration.

Use Cases

Provide custom configurations based on different conditions (subdomain name, path and file extension) when site-level configuration in **Site Acceleration** cannot meet your needs.

Provide basic features (caching and HTTPS) and acceleration features (custom cache key, URL rewrite and HTTP header modification).

Key Terms

Term	Description
Rule	It defines specific types of requests and the applicable operations.
Conditional Expression	It defines the logics that identify the requests. The followings are supported. IF Note 1 ELSE IF ELSE
Matching Condition	It defines the criteria that identifies the requests. The followings are included. Matching type Operator Value
And/Or	Logical AND/OR, which can link multiple conditions.
Action	A wide range of feature configurations that can be applied to hit requests.

Note:

Note 1:

An IF statement can be nested inside another IF statement, indicating that the nested one will be executed only after the other is met.

Rule Priorities

Range	Description
Site Acceleration vs Rule Engine	If the same operation is configured for both site acceleration and the rule engine, the rule engine has a higher priority and is the final effective configuration.
Single rule in the rule engine	<p>If there exist nested IF conditions within an IF statement, the execution of the embedded IF statement necessitates the fulfillment of the outermost IF condition.</p> <p>In the event of multiple coequal IF conditions, they are executed in relative order from top to bottom. That is, if multiple rules are matched simultaneously, the operations of the lower rules will supersede those of the upper rules.</p> <p>In the event that IF, Else IF, and Else coexist, upon satisfying any one of the IF or Else IF conditions, the corresponding operation will be executed and concluded, precluding further matching of other rules under the current IF condition. If none are met, operations will be executed in accordance with the Else rule.</p>
Multiple rules in the rule engine	<p>The rules are executed in relative order, from top to bottom.</p> <p>Note: You can place general or coarse-grained rules at the top as the default configuration and request-specific or finer-grained rules at the bottom.</p>

Note:

There are two scenarios with special execution:

[Token authentication](#) will be executed first no matter where it is placed. If a request hits two rules, token authentication will be executed first, as other operations will be performed only after authentication is passed.

For operations with redirect logic, such as URL redirection and forced HTTPS, their execution method is Break. This means that if the same request encounters both a redirect operation and other operations, the other operations below will not be executed after the redirect operation is executed.

Example of Rule Priorities

Example One: Nested IF Conditions within IF Matches

The current user's node cache TTL rule configuration is as follows, with multiple nested IF conditions present.

IF

+ Comment

Matching type

HOST

Operator

Is

Value

test.example.com

Matching type

URL Path

Operator

Is

Value

/example/*

Ignore case

+ And

+ Or

Action

EdgeOne Node Cache ...

Behavior

Do not cache

+ Action

IF

+ Comment

Matching type

File extension

Operator

Is

Value

jpgpng

Ignore case

+ And

+ Or

Action

EdgeOne Node Cache ...

Behavior

Custom TTL

Time

-

10

+

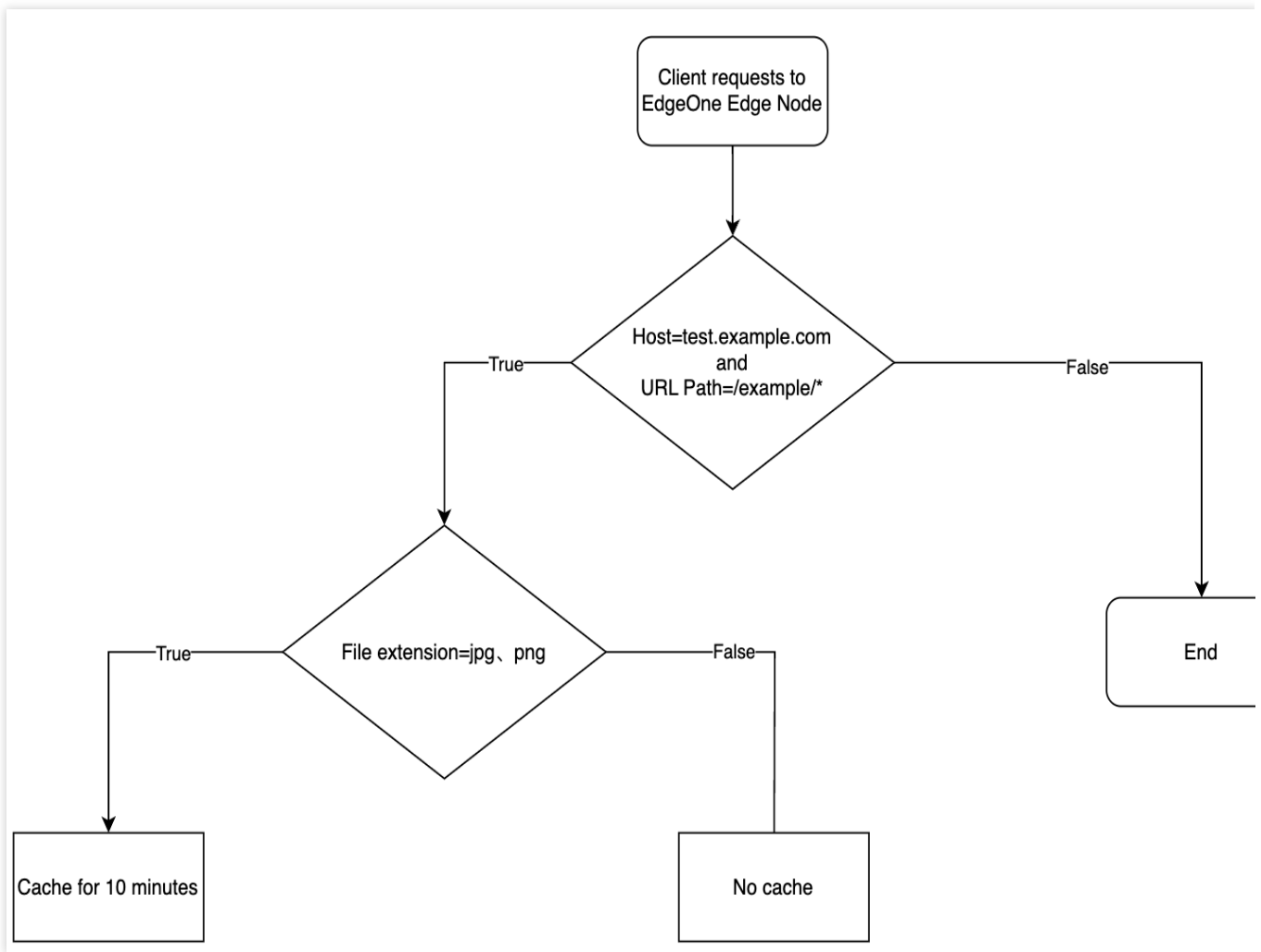
minutes

Force cache

+ Add

+ IF

The caching behavior of the user-requested URL is activated as follows:



When the request URL is: `https://test.example.com/example/1.jpg` , the file is cached for a duration of 10 minutes.

When the request URL is: `https://test.example.com/example/1.mp4` , the file is not subjected to caching.

When the request URL is: `https://test.example.com/video/1.jpg` , it does not conform to the stipulated rule.

Example Two: IF Condition Contains Multiple Parallel Else IF Matches

The current user's node cache TTL rule configuration is depicted below, with multiple coequal Else IF conditions present.

IF [+ Comment](#)

Matching type ⓘ

Operator

Value

HOST

Is

test.example.com ✕

+ And + Or

+ Action

IF [+ Comment](#) ✕

Matching type ⓘ

Operator

Value

File extension

Is

gif ✕ png ✕ bmp ✕ jpeg ✕ jpg ✕

+ And + Or

Ignore case

Action ⓘ

Behavior

Time

Force cache ⓘ

EdgeOne Node Cache ...

Custom TTL

-

7

+

days

☒

+ Add ▼

ELSE IF ✕

Matching type ⓘ

Operator

Value

File extension

Is

aspx ✕ jsp ✕ php ✕ asp ✕

+ And + Or

Ignore case

Action ⓘ

Behavior

EdgeOne Node Cache ...

Do not cache

+ Add ▼

ELSE IF ✕

Matching type ⓘ

Operator

Value

URL Path

Is

/admin/* ✕

+ And + Or

Ignore case

Action ⓘ

Behavior

EdgeOne Node Cache ...

Do not cache

+ Add ▼

ELSE ✕

Action ⓘ

Behavior

No Cache-Control

EdgeOne Node Cache ...

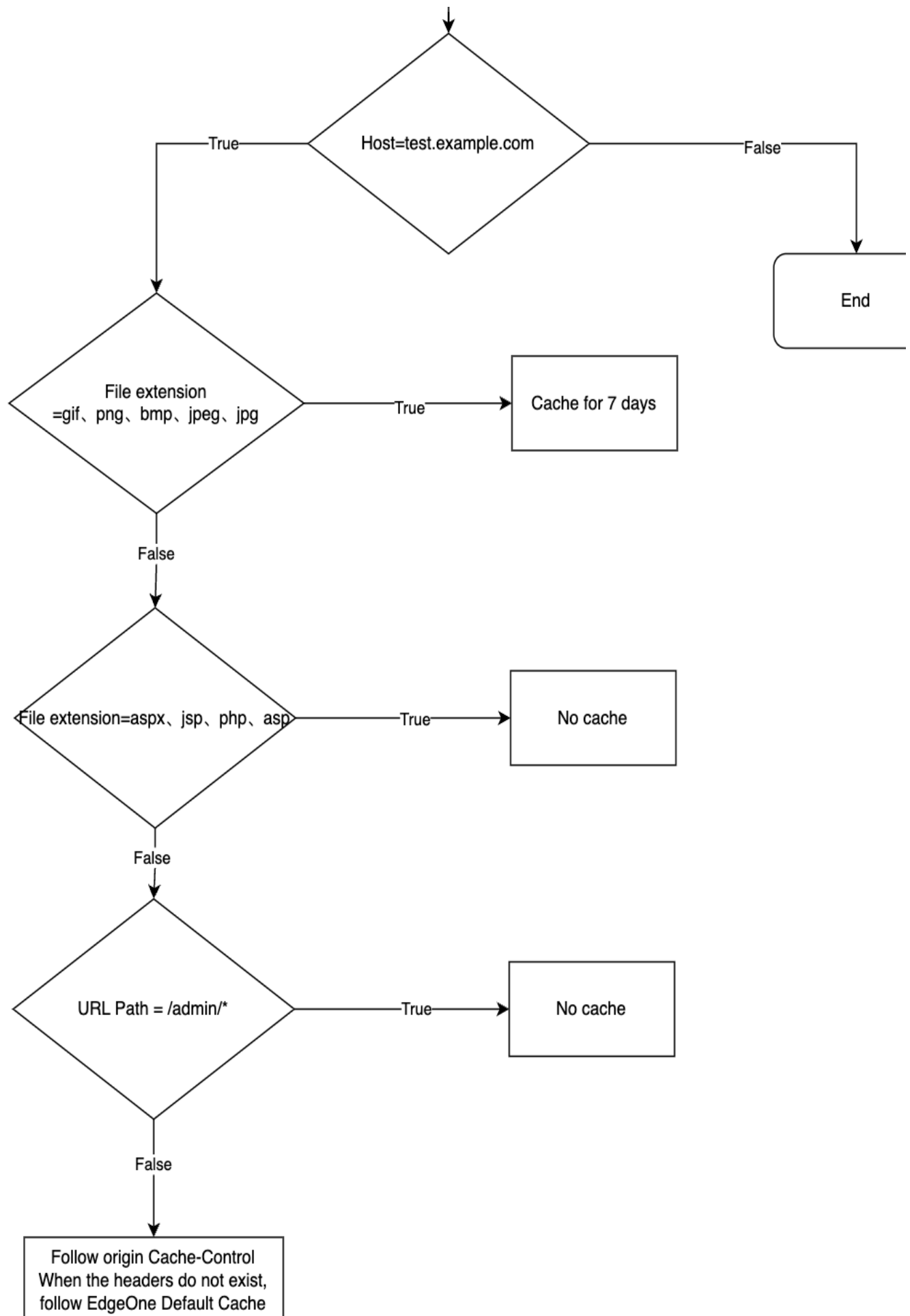
Follow origin server Car

Default cache policy

+ Action

The caching behavior of the user's requested URL will take effect as follows:

Client requests to
EdgeOne Edge Node



Rules

When the request URL is: `https://test.example.com/image/1.jpg` , the file is cached for a duration of 7 days.

When the request URL is: `https://test.example.com/index/1.jsp` , the file is not subjected to caching.

When the request URL is: `https://test.example.com/admin/1.php` , caching is not implemented.

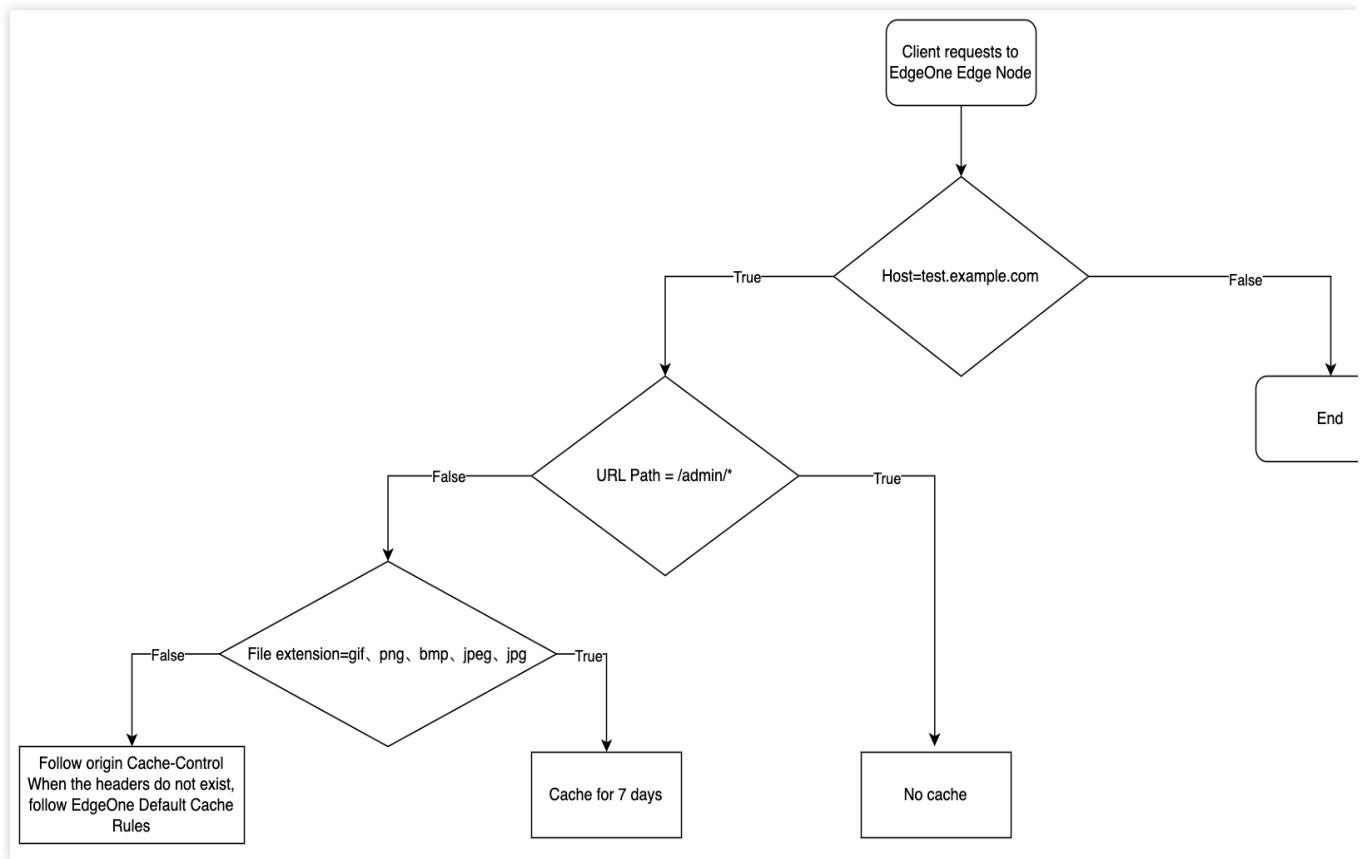
Example Three: Multiple Peer-Level IF Condition Matching

The current user's node cache TTL rule configuration is as follows. In the presence of multiple peer IF conditions, the effectiveness priority sequence of the subsequent conditions is the highest.

The screenshot displays the 'Rules' configuration interface for Tencent Cloud EdgeOne. It shows three IF conditions stacked vertically, each with its own Action and Behavior settings.

- Condition 1:**
 - Matching type: HOST
 - Operator: Is
 - Value: test.example.com
 - Action: EdgeOne Node Cache ...
 - Behavior: Follow origin server Cac
 - No Cache-Control: Default cache policy
- Condition 2:**
 - Matching type: File extension
 - Operator: Is
 - Value: gif, png, bmp, jpeg, jpg
 - Action: EdgeOne Node Cache ...
 - Behavior: Custom TTL
 - Time: 7 days
 - Force cache: ☒
- Condition 3:**
 - Matching type: URL Path
 - Operator: Is
 - Value: /admin/*
 - Action: EdgeOne Node Cache ...
 - Behavior: Do not cache

The caching behavior of the user's requested URL is activated as follows:



When the request URL is: `https://test.example.com/image/1.jpg` , the file is cached for a duration of 7 days.

When the request URL is: `https://test.example.com/admin/1.php` , the file is not subjected to caching.

When the request URL is: `https://test.exampel.com/admin/1.jpg` , the file is not subjected to caching.

When the request URL is: `https://test.exampel.com/index/1.txt` , the file adheres to the source site's Cache-Control header settings. In the absence of such a header, it complies with the default caching policy of EdgeOne.

Rule Management

Last updated : 2025-05-07 09:38:07

The console supports a series of icons and buttons to manage rules, for example, sorting, copying, enabling, and disabling rules, as follows.

Icon/Button	Description
	Drags a rule up or down.
	Edits a rule.
	Creates the same rule as the copied rule.
	Deletes a rule.
	Searches for a rule by rule name or keyword.
	Rule status Enable: Publishes a rule to the production environment. Disable: Saves a rule but does not publish it to the production environment.
	Saves a rule but does not publish it to the production environment.
	Saves and publishes a rule to the production environment.
	If a single rule is complex and has multiple IF statements, you can add comments to them. Then, the rule navigation will be automatically generated on the right of the rule content to simplify viewing and locating.

--	--

Variables

Last updated : 2025-04-25 17:01:58

Introduction

The variables of the rule engine allow you to dynamically extract and process data within request. These variables can not only store static values but also use for specific fields or information in the request, the value of which may change when processing each request. For example: the `http.request.host` variable, which can extract the `hostname` in each HTTP request. This capability enables the rule engine to handle more complex business logic.

Content

Name	Type	Description	Example
<code>http.request.scheme</code>	String	Client request protocol	http https
<code>http.request.zone</code>	String	Site name	example.com
<code>http.request.zoneid</code>	String	Site ID	zone-2c2r77pc3796
<code>http.request.host</code>	String	Hostname in the client request URI	www.example.com
<code>http.request.full_uri</code>	String	Full URI of the client request (not including #fragment)	https://www.example.org/articles/index?section=539061&expand=comments
<code>http.request.method</code>	String	Client request HTTP method	GET
<code>http.request.uri</code>	String	Client request URI path and query string	/articles/index?section=539061&expand=comments
<code>http.request.uri.path</code>	String	Client request URI path	/articles/index
<code>http.request.file_extension</code>	String	File extension of the client request file	jpg
<code>http.request.filename</code>	String	Filename of the client request file	bot.txt

http.request.uri.query	String	The whole query string of the client request, not including the <code>?</code> separator	section=539061&expand=comments
http.request.headers["key"]	String	The header value of the specified header name "key" in the client request, "key" can be replaced with your specified name. If the specified header name appears multiple times, the variable value will be the last one.	Client request URL: <code>https://developer.mozilla.org</code> , with headers <code>-H 'key: 123' -H 'key: 456'</code> , then the variable value obtained is <code>456</code>
http.request.uri.args["key"]	String	The parameter value of the specified parameter name "key" in the client query string, "key" can be replaced with your specified name. If the specified header name appears multiple times, the variable value will be the last one.	Client request URL: <code>https://developer.mozilla.org?key=123&key=456</code> , then the variable value obtained is <code>456</code>
http.request.version	String	The version of the HTTP protocol used in the client request	HTTP/1.0 HTTP/1.1 HTTP/2 HTTP/3
http.request.ip	String	Client TCP IP address, for example: <code>1.1.1.1</code>	93.184.216.34
http.request.ip.port	String	Client Port	1028
http.request.ip.city	String	City associated with the client IP address	San Francisco
http.request.ip.continent	String	Continent code associated with the client IP address	AF: Africa AS: Asia EU: Europe

			NA: North America SA: South America OC: Oceania AN: Antarctica
http.request.ip.country	String	2-letter country code in ISO 3166-1 Alpha 2 format associated with the client IP address	GB, see more in ISO 3166-1 Alpha 2

Use Case

1. The custom origin-pull request header carries the information of the country where the client IP address is located back to the origin.

IF

HOST Is ☐

Modify HTTP origin-pull request header

Type: Add

Header name: Tencent-Client-Country

Header value: \${http.request.ip.country}

2. Custom origin-pull request headers allow the origin server to collect and analyze which domains have been accelerated by Tencent's EdgeOne.

IF

HOST Is ☐

Modify HTTP origin-pull request header

Type: Add

Header name: Tencent-Acceleration-Domain-Name

Header value: \${http.request.host}

3. Custom Cross-Origin Request Policy: Allows cross-origin requests from domains specified in the Origin header of the request.

IF

HOST Is ☐

HTTP Request Header

Origin

Exist

Modify HTTP nodes response header

Type: Set

Header name: Access-Control-Allow-Origin

Header value: \${http.request.headers["Origin"]}

Supported Matching Types and Actions

Last updated : 2025-01-22 09:53:15

Supported Matching Types

Supported matching types are listed in the following tables.

Note

1. URL Path and URL Full support wildcard match. If the **URL Path** is `/foo/*/bar` , both `/foo/example/bar` and `/foo/demo/bar` are valid values.
2. URL Path, URL Full, query string, file extension,file name and HTTP request header support enabling ignoring case (it is disabled by default).

Type	Description	Sample values
HOST	Request Host	<code>www.example.com</code>
URL Path	Request URL path	<p>If you need to match the <code>/example/foo/bar</code> path, then select the operator Equal to and enter the value <code>/example/foo/bar</code> .</p> <p>If you need to match the <code>/example</code> directory and all files under the directory, then select the operator Equal to and enter the value <code>/example/*</code> .</p>
URL Full	Full request URL, including the protocol, domain name, URL path, and query string.	<p>If you need to fully match the <code>https://www.example.com/foo.jpg</code> path, then select the operator Equal to and enter the value <code>https://www.example.com/foo.jpg</code> .</p> <p>If you need to match a path consisting of <code>https://www.example.com/foo.jpg</code> and a query string, then select the operator Regular expression matching and enter the value <code>https://www.example.com/foo\\.jpg(\$ \\?)</code> .</p> <p>If you need to match a path consisting of <code>https://www.example.com/foo.jpg</code> and the query string <code>test</code> , then select the operator Regular expression matching and enter the value <code>https://www.example.com/foo\\.jpg(\$ \\?.*test=)</code> .</p>
Query string	Query string in the request URL	Parameter name: key Parameter value: value

File extension	File extension (file extension) of the request content	jpg, png, css
File name	File name of the request content	foo.txt
HTTP request header	HTTP request header	HTTP request header name: name HTTP request header value: value
Client geo location	Country/region of the client IP	United States
Request protocol	Requested protocol type by the client	HTTPS or HTTP
Client IP address	Client IP of the request	You can enter an IP address or IP address range. For example: IPv4: 192.168.0.0 or 192.168.0.0/24 , where 0.0.0.0/0 matches all IPv4 addresses. IPv6: 2001:db8:1234::1 or 2001:db8::/32 , where ::/0 matches all IPv6 addresses.
Request Method	Client request method	The supported methods are as follows, multi-option. GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT, OPTIONS, PATCH, COPY, LOCK, MKCOL, MOVE, PROPFIND, PROPPATCH, UNLOCK.
All	Any site request	N/A

Operators

Type	Description
Equal to	The request is equal to a specified value (value of the matching type).
Not equal to	The request is not equal to a specified value (value of the matching type).
Exist	A specified value exists in the request (HTTP header name or query parameter name).
Not exist	A specified value does not exist in the request (HTTP header name or query parameter name).
Regular expression matching	URL Path and URL Full support Google RE2 regular expression matching.

Supported Actions

Actions refer to a series of feature configurations performed after the requests hit the conditions. The supported actions and matching types are listed in the following tables.

Cache configuration

Action	Description	Supported Matching Types
Node Cache TTL	By configuring the cache TTL, you can optimize node cache to improve resource loading and update resources in a timely manner.	HOST URL FULL URL Path File name File extension
Browser Cache TTL	By adjusting the cache period of resources in browsers, you can optimize the browser cache and increase the loading speed of the requested resources.	HOST URL FULL URL Path File name File extension Query string Client geo location Client IP Request method
Custom Cache Key	A cache key can be customized to suit your needs by setting the query string, HTTP header and URL case, so that requested resources can be loaded faster.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP
Status Code Cache TTL	You can specify a TTL period for origin response status codes, allowing the node to directly respond with non-2XX codes.	HOST URL FULL URL Path File name File extension Query string
Cache Prefresh	Cached resources are validated via origin-pull before expiration, so that your	HOST URL FULL URL Path

	site can respond to requests more rapidly.	File name File extension
Offline Cache	After offline caching is enabled, when your origin fails, and resources cannot be pulled through origin-pull normally, resources cached on nodes (even expired resources) can be used until the origin recovers.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP

Network optimization

Action	Description	Supported Matching Types
HTTP/2	HTTP/2 (HTTP 2.0) requests are supported to accelerate sites and improve the web performance.	HOST
HTTP/3 (QUIC)	HTTP/3 (QUIC) requests are supported. HTTP/3 (QUIC) is used to accelerate site requests and improve data transfer efficiency and security.	HOST
WebSocket	EdgeOne supports the WebSocket protocol that allows the server to proactively send data to the client.	HOST Request method
Maximum Upload Size	The maximum upload size is the maximum data volume that can be uploaded in a single client request. You can restrict it to improve the data transfer efficiency and optimize the network transfer.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Request method
Smart Compression	Smart Compression can automatically compress the resources to Gzip/Brotli files to reduce the files size and shorten the resource loading time.	HOST Request method
Smart Acceleration	Smart acceleration refers to smart dynamic routing acceleration. After this	HOST

	<p>feature is enabled, it will detect the node network latency in real time and use the smart algorithm to select the optimal transfer path, so as to handle both static and dynamic client requests more quickly, stably, and securely.</p> <p>Smart dynamic routing minimizes problems such as high network latency, connection errors, and request failures.</p>	
HTTP/2 Origin-Pull	Request origin-pull over the HTTP/2 protocol is supported.	HOST Request method
Upstream Timeout	<p>You can reasonably set the origin-pull request timeout based on the network link conditions and the data processing capability of the origin server to ensure normal origin-pull as requested. The origin-pull timeout is defined as follows. If there is no data response from the origin server after a node initiates an origin-pull request, no matter how long the duration is, the node considers it a timeout and actively disconnects from the origin server.</p>	HOST

HTTPS optimization

Action	Description	Supported Matching Types
Forced HTTPS	You can use 301 or 302 redirect to redirect HTTP client requests to HTTPS requests and send them to EdgeOne.	HOST
HSTS Configuration	Force clients such as browsers to establish connections to edge nodes over HTTPS for global website encryption.	HOST
SSL/TLS Security Configuration	Configure the protocol version and Cipher suite that are allowed to use when the client shakes hands with the edge server TLS as needed.	HOST
OCSP Stapling	Pre-cached OCSP responses are sent at the time of TLS handshake to improve the efficiency.	HOST

Origin-Pull HTTPS	You can specify the protocol that EdgeOne uses in the origin-pull request.	HOST
-----------------------------------	--	------

Modifying HTTP header

Action	Description	Supported Matching Types
Modifying HTTP Response Headers	You can customize, add, and delete headers in HTTP responses from nodes to clients.	HOST URL FULL URL Path File name File extension Query string Client geo location Client IP Request method
Client IP Header	The custom header can carry the real client IP to the origin.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request method
Client IP Geographical Location	The custom header can carry the geographical location information of the client IP to the origin.	HOST Client geo location Client IP Request method
Modifying HTTP Origin-Pull Request Headers	You can customize, add, and delete headers in HTTP origin-pull requests from nodes to the origin.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request method
Host Header Rewriting	Host header rewriting enables you to rewrite the host header to the actual origin	HOST URL FULL

	domain when the origin domain is different from the acceleration domain in the load balancing task.	URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request protocol Request method
--	---	--

Advanced configuration

Action	Description	Supported Matching Types
Access URL Redirection	A node redirects the URL requested by the client to the destination URL based on the response status code.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request method
Token Authentication	As an access control policy, token authentication supports creating rules to validate access and filter out unauthorized access requests. This effectively prevents your site resources from being maliciously hotlinked and thus protects your business.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Request method
Modify Origin	Configuration of primary and secondary sources, separate Path, and separate region Rules for complex origin-pull strategies.	HOST + Any matching type below : URL Path Client country/region HTTP Request Header Query string File extension Request method
Origin-Pull URL Rewrite	Based on specified rules, this feature rewrites the user request URL received by the node to the destination URL when the	HOST URL FULL URL Path File name

	node sends the request to the origin server, which doesn't affect the node cache key.	File extension Query string HTTP Request Header Client geo location Client IP Request method
Controlling Origin-pull Requests	You can specify which part of the query string and Cookie to be included in the request when it's forwarded to the origin.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request method
Redirect Following During Origin-Pull	When origin-pull is requested, the redirect will be based on the 302/301 status code of the origin server, and you can specify the maximum number of redirects (which is 3 by default and can be 1-5).	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request method
Custom Error Page	You can redirect requests to a custom error page for specific error status codes returned by the origin. The redirection is performed when a 302 is returned.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Client IP Request method
Range GETs	Range GETs can be enabled to reduce the consumption of large file origin-pulls and response time.	HOST URL FULL URL Path File name File extension Query string

		HTTP Request Header Request method
HTTP response	The HTTP response feature is supported by the rule engine. When the corresponding conditions are met, the EdgeOne node directly responds with the specified status codes and page content.	HOST URL FULL URL Path File name File extension Query string HTTP Request Header Client geo location Request protocol Client IP Request method All (any site request)

Related References

Rule Engine Configuration Character Limit

Last updated : 2025-04-01 15:01:11

The current Rule Engine partially matches conditions and operations involving HTTP header names / parameter names. The character range supported by EdgeOne is: letters (a - z, A - Z), digits (0 - 9), and some special characters. Details of the supported special characters are as follows:

character	Header name	Parameter Name
	Match condition: HTTP request header Operations: custom Cache Key "HTTP headers" Modify the HTTP node response header Modify the HTTP back-to-origin request header client IP header Client IP Geolocation Header	Match condition: Query string Operations: Token authentication "authentication encryption string parameter name" Token authentication "timestamp parameter name for authentication"
(space)	×	×
.	✓	✓
"	×	×
#	✓	✓
\$	✓	×
%	✓	✓
&	✓	×
'	×	×
(×	×
)	×	×
*	✓	×
+	✓	×

,	×	✓
-	✓	✓
.	✓	✓
/	×	✓
:	×	✓
;	×	✓
<	×	✓
=	×	×
>	×	✓
?	×	×
@	×	✓
[×	×
\\	×	×
]	×	×
^	✓	×
—	✓	✓
`	×	✓
{	×	×
	✓	×
}	×	×
~	✓	✓

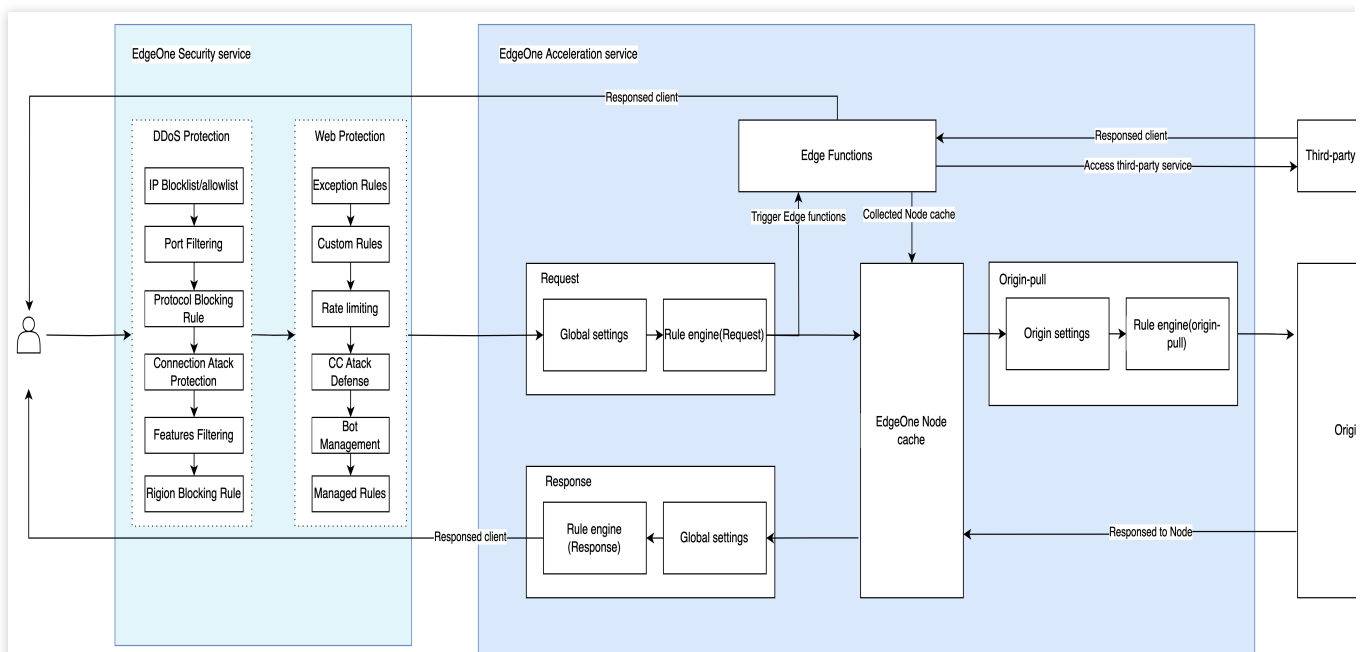
Request and Response Actions

Processing order

Last updated : 2024-05-28 14:03:42

This article introduces the processing order of various module rules configured in the platform after the client initiates a request to EdgeOne, helping users understand the effective order and impact of the rules to ensure that the configured rules work as expected.

Processing order



After the user initiates a request, the request will be processed in the above order. For example, if the user configures the modification of HTTP headers in both the rule engine and the Edge functions, the final result will be based on the processing of the Edge functions since they are processed later.

1. Multiple rules may be triggered in the security service module, and the relevant processing order is as follows:

The DDoS protection request processing order only applies to users who have purchased the exclusive DDoS protection with L4 proxy. For details, please refer to the [DDoS protection overview](#).

The Web protection module contains the bot management module. For the request processing order within the Web protection module, please refer to the [Web Protection request processing order](#). For the rule application order within the bot management module, please refer to the bot management overview.

If a request triggers a security policy in the EdgeOne security protection module, and the action is interception, discard and blacklist, or IP blocking, the request will be rejected.

2. The rule priority in the rule engine is higher than the global site setting rules, and the rule application order in the rule engine is that the lower rules have higher priority. For details, please refer to the [rule engine overview](#).
3. When a request triggers an Edge function rule, the request will be processed by the Edge functions. Edge functions can access third-party services, EdgeOne cache content, or return to the client's origin through sub-requests, or directly respond to client requests.
4. When requests to the EdgeOne node Cache, if the current node does not have Cache, it will continue to origin-pull. If the node hits the Cache, it will not Trigger the subsequent origin-pull rules and directly Return the corresponding resources to the user.

Default HTTP Headers of Origin-Pull Requests

Last updated : 2025-05-26 15:42:29

Overview

EdgeOne passes through all client request headers to the origin, and the origin-pull request carries the default custom request headers of EdgeOne. To modify the HTTP headers of origin-pull requests, see [Modifying HTTP Headers of Origin-Pull Requests](#).

Default HTTP headers of origin-pull requests

EdgeOne adds the following default HTTP headers to origin-pull requests.

EO-Connecting-IP

The `EO-Connecting-IP` header records the IP that of the request initiator. If the request is not forwarded by any proxy, the IP address in the header is the real IP address of the client. Otherwise, it's the proxy IP.

X-Forwarded-For

This header records the proxy IP and real client IP. If a client request is forwarded to an EdgeOne node after multiple hops, the header records the real client IP and the IP of proxy before the EdgeOne node. The value of the header is determined based on the following rules:

If a request sent to an EdgeOne node carries the `X-Forwarded-For` header that has recorded the client IP address, EdgeOne appends the IP address of the proxy before the EdgeOne node to the header. For example, if the client request carries `X-Forwarded-For: 192.168.1.1` (where `192.168.1.1` is the client IP) and is forwarded to an EdgeOne node via the proxy `10.1.1.1`, the header of the origin-pull request is `X-Forwarded-For: 192.168.1.1,10.1.1.1`.

If the request does not carry the `X-Forwarded-For` header, EdgeOne adds this header to the origin-pull request, taking the proxy IP of the EdgeOne node as the value. In this case, it's the same as the `EO-Connecting-IP` header.

For more information, see [X-Forwarded-For](#).

X-Forwarded-Proto

The `X-Forwarded-Proto` header records the HTTP protocol used by the client to initiate the request. Valid values:

```
X-Forwarded-Proto: http
```

```
X-Forwarded-Proto: https
```

```
X-Forwarded-Proto: quic
```

For more information, see [X-Forwarded-Proto](#).

CDN-Loop

It records how many times a client request passes an EdgeOne node. The count is added by one every time the request passes the node. When the count reaches 16, the EdgeOne node denies the request and returns the 423 status code.

Example: `CDN-Loop: TencentEdgeOne; loops=3` .

EO-LOG-UUID

It carries the unique identifier of the request. When an access exception occurs, you can locate the problem by querying logs with this UUID.

Example: `EO-LOG-UUID: 4105283880544427145` .

EO-Bot-Tag

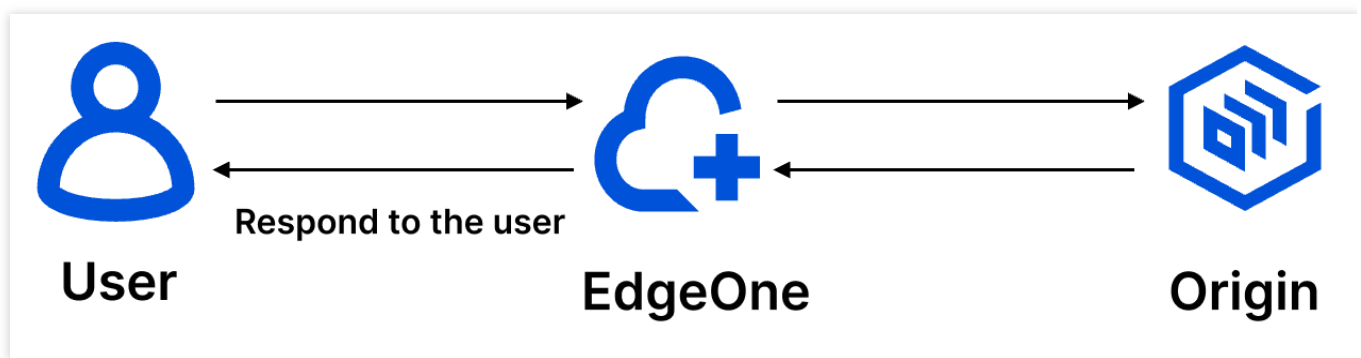
When Bot management is enabled, the platform automatically appends an HTTP request header EO-Bot-Tag to the origin requests, containing a JSON string that includes the bot tag recognition results for the requesting client, assisting the origin site in log analysis and security protection policy linkage. For details, please refer to [Get Bot management tag via HTTP Headers of origin-pull requests](#).

Default HTTP Response Headers

Last updated : 2024-05-08 21:20:09

Overview

This document describes default response headers that EdgeOne passes to the client. These headers follow the origin unless there are custom HTTP headers.



Default HTTP response headers

EdgeOne adds the following default HTTP response headers to responses to the client.

EO-Cache-Status

`EO-Cache-Status` is used to indicate whether the current Client-initiated requests hit the Cache or not, with the following values:

`EO-Cache-Status: HIT` : Indicates that the requested resources hit the Cache on the EdgeOne node and the Cache is not expired, directly responded by the node to the user requests.

`EO-Cache-Status: MISS` : Indicates that the requested resources did not hit the Cache on the EdgeOne node, or hit the Cache but the Cache is expired, the node performs origin-pull verification, the origin file is updated and responded with a 200 status code, the node needs to origin-pull the resources.

`EO-Cache-Status: RefreshHit` : Indicates that the requested resources hit the Cache on the EdgeOne node, but the Cache is expired, the node performs origin-pull verification, the origin file is not updated and responded with a 304 status code, the node continues to use the Cache to respond to user requests.

Server

This header indicates the server name. The header value depends on the service that the Web Server is built on. By default, this header follows the origin response (if exists). Otherwise, the EdgeOne node adds this header with the value of `Server:TencentEdgeOne`. For more information, see [Server](#).

Values of `Server` header for different origin types:

Cloud Object Storage (COS) origin: `Server: tencent-cos`

Cloud Virtual Machine (CVM) origin: `Server: nginx`, `Server: apache`, `Server: tomcat`, or `Server: Microsoft-IIS`

Cloud Load Balancer (CLB) origin: `Server: openresty`

Date

The value of the `Date` header is the current time of the EdgeOne node server. For more information, please refer to [Date](#).

For instance: `Date: Sat, 07 Jan 2023 14:15:52 GMT`.

Connection

This header indicates how a persistent connection is handled during the communication between the client and server. By default, this header follows the origin response (if exists). Otherwise, EdgeOne sets the header based on the following rules:

HTTP/2 and QUIC requests: This header is not added.

HTTP 1.0 requests w/o KeepAlive option: The header is set to `Connection:close`.

****Origin response does not contain `content-length` and `transfer-encoding` **: This header is set to `Connection:close`.**

In other cases, the header is set to `Connection:keepalive`.

For more information, see [Connection](#).

Alt-Svc

`Alt-Svc` stands for `Alternative-Service`. It lists the alternative access methods of the site. This header is commonly used for backward compatibility with previous protocols after updating to a new one, such as QUIC. If you enable HTTP/3 (QUIC) for a domain name, this header is added to the HTTP response header by default. For more information, see [Alt-Svc](#).

EO-LOG-UUID

It carries the unique identifier of the request. When an access exception occurs, you can locate the problem by querying logs with this UUID.

Example: `EO-LOG-UUID: 4105283880544427145`.

HTTP Restrictions

Last updated : 2025-01-24 15:09:12

This document will introduce the restrictions of EdgeOne on HTTP requests/responses in various dimensions and the response behaviors once these restrictions are exceeded.

Restriction Item	Description
Request header length (key + value)	The total size of the request header name and header value is restricted to 128 KB. If exceeded, the EdgeOne node will respond with a 413 status code.
Response header length (key + value)	The total size of the response header name and header value is restricted to 128 KB. If exceeded, the EdgeOne node will respond with a 413 status code.
Number of HTTP request headers	The total number of HTTP request headers is 256. HTTP/1.1 requests: If exceeded, the EdgeOne node will respond with a 400 status code. HTTP/2.0 requests: If exceeded, the EdgeOne node will send goaway and close the stream.
Number of HTTP response headers	The total number of HTTP response headers is 256. HTTP/1.1 requests: If exceeded, the EdgeOne node will respond with a 400 status code. HTTP/2.0 requests: If exceeded, the EdgeOne node will send goaway and close the stream.
Size of the header key or value for HTTP/2 requests	The size of a single header name or value is restricted to 32 KB. If exceeded, the EdgeOne node will send goaway and close the stream.
Maximum number of requests per HTTP/2 stream	Maximum number of requests per stream is 1000, no limit on origin requests.
Maximum concurrency per HTTP/2 stream	Maximum concurrency per stream is 128
Request URL length	It is currently unrestricted, but will be restricted to 8192 B in the future.
Request body length	Upper limit of UInt64. For the POST request upload body length limit, please refer to Maximum Upload Size .
Response body length	Upper limit of UInt64.
Cache size	It is currently unrestricted, but will be restricted to 30 GB in the future.

Request method	The supported methods only include GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT, OPTIONS, PATCH, COPY, LOCK, MKCOL, MOVE, PROPFIND, PROPPATCH, and UNLOCK. For requests with other methods, the EdgeOne node will directly respond with a 400 status code.
----------------	--

Speed limit for single connection download

Last updated : 2025-04-30 17:46:49

Feature Overview

The release of new game versions and the downloading of software can lead to sudden spikes in bandwidth and costs. EdgeOne offers Speed limit for single connection download, which allows for rate limiting of requests between nodes and clients on the downlink, thus helping to control the bandwidth peak of accelerated domains. However, this may also impact the user experience of file downloads, so a reasonable throttling value should be set according to business characteristics.

Feature Description

1. Supported throttling modes:

Speed limit for entire download process: Throttling is applied according to the set value throughout the entire process starting from the response to the client.

Speed limit starts after a specific byte of full-speed download: No throttling occurs before reaching the specified byte; after responding with the specified byte, throttling is applied according to the set value.

Speed limit starts after a specific time of full-speed download: No throttling occurs before the specified time; after the specified time, throttling is applied according to the set value.

2. Supported throttling capabilities:

Field	Meaning
Set the start byte for rate limiting	Configuring to 0 means throttling throughout the entire download process; otherwise, throttling starts from the specified byte and there is no throttling before that byte. Note: Supports filling in constants or Variables . If you fill in <code>\${http.request.uri.args["length"]}</code> , the node will extract the value of the length parameter from the URL to start throttling. Example: Setting throttling to start from 1024 bytes with a throttling value of 1000 KB/s means no throttling from 0-1024 KB, and after 1024 KB, the speed will maintain at 1000 KB/s.
Set the start time for rate limiting in seconds	Configuring to 0 means throttling throughout the entire download process; otherwise, throttling starts from the specified duration (timed from the beginning of the response data to the client). Note: Supports filling in constants or Variables . If you fill in <code>\${http.request.uri.args["time"]}</code> , the node will extract the value of the time

	<p>parameter from the URL to start throttling.</p> <p>Example: Setting throttling to start from 2 seconds with a throttling value of 1000 KB/s means no throttling from 0-2 s, and after 2 s, the speed will maintain at 1000 KB/s.</p>
Speed Limit Value	<p>The node will control the response speed to the client according to the set value.</p> <p>Note: Supports filling in constants or Variables. If you fill in the variable <code>\${http.request.uri.args["rate"]}</code>, EO will extract the rate parameter value from the request URL for throttling.</p> <p>Example: If you fill in the constant 2048, the throttling value will be 2048 KB/s; if you fill in the variable <code>\${http.request.uri.args["rate"]}</code>, with a URL like <code>http://example.com/download/test.zip?rate=1024</code>, the throttling value will be 1024 KB/s.</p>

Note:

The specified duration throttling starts timing from when response data is sent to the client and may be influenced by factors such as access path, client reception speed, and edge node TCP buffer, so the actual start time for throttling may have some discrepancies with the configured time in the console.

The variable capabilities for setting bytes to start throttling and seconds to start throttling are currently in grayscale release. If you need to use them, please [contact us](#).

The grayscale release for starting throttling after a specific time at full-speed download is ongoing. For usage, please [contact us](#).

Operation Steps

Scenario 1: Throttling a specific domain to a specific throttling value

If you want all requests to be limited to a download speed of `500 KB/s` for the domain `www.example.com` under the site `example.com`, refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and click the target **site** in the site list that needs to be configured.
2. In the site details page, click **Site Acceleration**, go to the global configuration page, and click on the **Rules Engine** tab.
3. On the rules engine page, click **Create Rule**, select **New Blank Rule**, and proceed to the new rule editing page.
4. In the rule editing page, select matching type as HOST equals `www.example.com`.
5. Click **Action** > **Select Box**, and from the operation list, choose **Speed limit for single connection download**.
6. Select the mode as `Full-Process Download Throttling`, and fill in the throttling value as 500.
7. The complete rule configuration is shown below. Click **Save and Publish** to complete the rule configuration.

Scenario 2: Throttling a specific domain based on the variable value in the request query string

If you want the domain `www.example.com` under the site `example.com` to dynamically throttle based on the values of the query parameters in the request URL, where `speed` is the throttling value and `byte` is the size at which throttling begins, refer to the following steps:

1. Log in to the [EdgeOne console](#), click **Site List** in the left sidebar, and click the target **site** in the site list that needs to be configured.
2. In the site details page, click **Site Acceleration**, go to the global configuration page, and click on the **Rules Engine** tab.
3. On the rules engine page, click **Create Rule**, select **New Blank Rule**, and proceed to the new rule editing page.
4. In the rule editing page, select matching type as HOST equals `www.example.com`.
5. Click **Action > Select Box**, and from the operation list, choose **Speed limit for single connection download**.
6. Select the mode as `Speed limit starts after a specific byte of full-speed download`, set `Set the start byte for rate limiting` as `${http.request.uri.args["byte"]}`, and set `Speed Limit Value` as `${http.request.uri.args["speed"]}`.
7. The complete rule configuration is shown below. Click **Save and Publish** to complete the rule configuration.