Tencent Cloud EdgeOne
Site Acceleration

Contents

Site Acceleration

Access Control

Token Authentication

Last updated：2023-09-07 15:02:38

Overview

Token authentication serves as an access control policy, enabling the configuration of authentication rules for access validation and the filtration of unauthorized access requests. It effectively safeguards your site resources from malicious exploitation, thereby protecting your business content.

How does Token Authentication achieve access control?

When a client user initiates a request, the access request URL must generate an authentication URL in accordance with the authentication rules. Only when the authentication information in the authentication URL (for example, a timestamp) passes the node verification, the access request is considered legitimate and the node responds normally. If the verification fails, the node rejects the access and directly returns a 403 error.

Instructions

1. Log in to the EdgeOne console and select **Rule Engine** from the left sidebar.
2. On the Rule Engine page, select the desired site and configure the Token Authentication rules as needed. For guidance on using the Rule Engine, please refer to Rule Engine.

Configuration items	Note
Authentication Method	Currently, four types of authentication signature calculation methods are supported. Please choose the appropriate method based on the format of the access URL. For more details, refer to Authentication Methods.
Primary authentication key	A primary authentication key must be between 6-40 characters and contains letters and numbers.
Secondary authentication key	A secondary authentication key must be between 6-40 characters and contains letters and numbers.
Authentication	The authentication parameter name, the node will verify the value

parameters	corresponding to this parameter name. It consists of 1 – 100 characters, which can be uppercase or lowercase letters, numbers, or underscores.
Validity period	Configure the validity period of the authentication URL in seconds (1 – 630720000) to determine whether the client's access request has expired: 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 + validity period", the request is not expired and the verification continues.

Authentication Method

Method A

Authentication URL format

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

Field description

Parameter	Note
Hostname	Accelerated Domain Name.
Filename	For the resource access path, the Filename during authentication must begin with `/`.
sign	Specify the custom name for the authentication parameter.
timestamp	Timestamp parameter Format: A positive integer Unix timestamp in decimal format, representing the total number of seconds elapsed from 00:00:00 on January 1, 1970, UTC time, to the present moment. Its definition is independent of the time zone.
rand	A random string of 0–100 characters, composed of uppercase and lowercase letters and numbers.
uid	User ID, currently unused, is set to 0 by default.
md5hash	The fixed-length 32-character string calculated using the MD5 algorithm: • Algorithm: MD5(/Filename-timestamp-rand-uid-key). • Authentication logic: If the request has not expired, the node compares

	this string value with the `md5hash` value carried in the request URL: if the two values are identical, the authentication is successful and the request is responded to; if the two values are different, the authentication fails and a 403 error is returned.

Option B

Authentication URL format

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

Description

Parameter	Note
Hostname	Accelerated Domain Name.
Filename	For the resource access path, the Filename during authentication must begin with `/`.
timestamp	Timestamp parameter. Format: YYYYMMDDHHMM, UTC+8 time, for example, 201807301000.
md5hash	The fixed-length 32-character string calculated using the MD5 algorithm: • Algorithm: MD5 (Key + timestamp + /Filename). • Authentication logic: If the request has not expired, the node compares this string value with the `md5hash` value carried in the request URL: if the two values are identical, the authentication is successful and the request is responded to; if the two values are different, the authentication fails and a 403 error is returned.

Method C

Authentication URL Format

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

Description

Parameter	Note

Hostname	Accelerated Domain Name.
Filename	For the resource access path, the Filename during authentication must begin with `/`.
timestamp	Timestamp parameter. Format: A hexadecimal positive integer Unix timestamp, representing the total number of seconds from 00:00:00 on January 1, 1970, UTC time, to the present. Its definition is independent of the time zone.
md5hash	The fixed-length 32-character string calculated using the MD5 algorithm: • Algorithm: MD5 (Key + /Filename + timestamp). Note: When calculating, the hexadecimal timestamp should be filtered to remove the hexadecimal number identifier 0x. • Authentication logic: If the request has not expired, the node compares this string value with the `md5hash` value carried in the request URL: if the two values are identical, the authentication is successful and the request is responded to; if the two values are different, the authentication fails and a 403 error is returned.

Option D

Authentication URL format

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

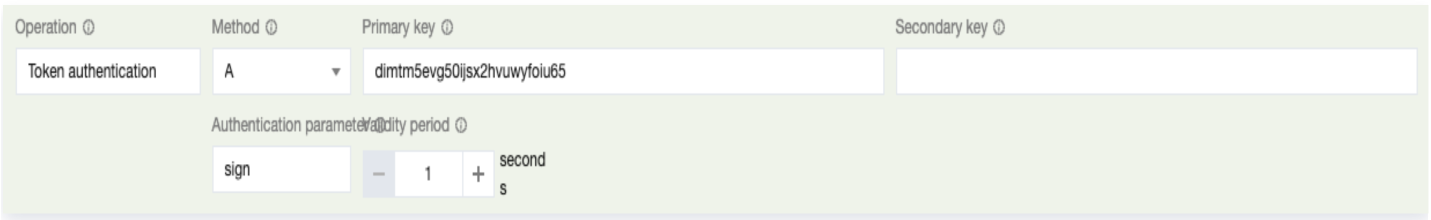
Description

Parameter	Note
Hostname	Accelerated Domain Name.
Filename	For the resource access path, the Filename during authentication must begin with `/`.
sign	Specify the custom name for the authentication parameter.
t	Custom Timestamp Parameter Name
timestamp	Timestamp parameter. Format: Decimal integer Unix timestamp, which is the total number of seconds from 00:00:00 on January 1, 1970, UTC time, to the present. Its

	definition is independent of the time zone; or hexadecimal integer Unix timestamp, which is the total number of seconds from 00:00:00 on January 1, 1970, UTC time, to the present. Its definition is also independent of the time zone.
md5hash	The fixed-length 32-character string calculated using the MD5 algorithm: • Algorithm: MD5 (Key + /Filename + timestamp). Note: When calculating, the hexadecimal timestamp should be filtered to remove the hexadecimal number identifier 0x. • Authentication logic: If the request has not expired, the node compares this string value with the `md5hash` value carried in the request URL: if the two values are identical, the authentication is successful and the request is responded to; if the two values are different, the authentication fails and a 403 error is returned.

Parameter Configuration Sample Code

Assuming the request `http://www.example.com/test.jpg` complies with authentication method A, it can be configured as follows:



Operation: Token authentication
Method: A
Primary key: dimtm5evg50ijsx2hvuwyfoiu65
Secondary key:
Authentication parameter: sign
Validity period: 1 seconds

Obtain authentication parameters:

- /Filename: `/foo.jpg` 。
- Timestamp: The server generates the authentication URL at 10:30:32 on March 15, 2022 (UTC+8), which converts to the decimal integer value of `1647311432` .
- rand: Generate a random number `J0ehJ1Gegyia2nD2HstLvw` .
- uid: `0` 。
- Key: `3C9mxSGzc8ZadmGNzE` .
- md5hash: MD5(/Filename-timestamp-rand-uid-key) = MD5(`/foo.jpg` - `1647311432` - `J0ehJ1Gegyia2nD2HstLvw` - `0` - `3C9mxSGzc8ZadmGNzE`) = ecce3150cbdaac83b116d937777ca77f.

Generating an authentication URL

```
http://www.example.com/foo.jpg?sign=1647311432-J0ehJ1Gegyia2nD2HstLvw-0-
ecce3150cbdaac83b116d937777ca77f
```

。

Node Authentication

After the request is initiated via an encrypted URL, the node parses the value of the "timestamp" parameter from the URL to determine whether the request is valid:

1. If the time "timestamp + validity period" is reached, the request is considered expired and a 403 is returned.
2. If the time "timestamp + validity period" is not reached, the request is considered valid and will be authenticated.
3. 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 two values match, the authentication is successful and the request is responded to; if the two values differ, the authentication fails and a 403 error is returned.

Supports and Limits

1. If the authentication succeeds, the authentication parameters in the request URL will be ignored during origin-pull and used as the cache key to increase the cache hit rate.
2. Upon successful authentication, if the node cache is not hit, the process will continue to pull from the origin. The actual origin-pull URL will be consistent with the authentication URL format, retaining the authentication parameters. The origin server can choose to ignore or re-verify these parameters as needed, or use the Origin-pull Request Parameter Settings to configure the origin-pull to ignore relevant authentication parameters.
3. The origin-pull request URL cannot contain any Chinese characters.

Smart Acceleration

Last updated：2023-09-07 15:02:44

Overview

When your site provides services for **pure dynamic content** or a **mix of dynamic and static content**, user requests for dynamic content may need to be sourced from different resources based on user responses. This can lead to complex network environments due to regional and operator differences. As a result, cross-regional and cross-operator access can lead to slow user access requests and high packet loss rates.
Smart acceleration can adjust and optimize network paths in real time. Once this feature is enabled, EdgeOne will detect node network latency in real time, select the optimal access path through intelligent algorithms, and dynamically adjust resource allocation and utilization based on real-time network conditions, thereby enhancing user access experience and ensuring business continuity.

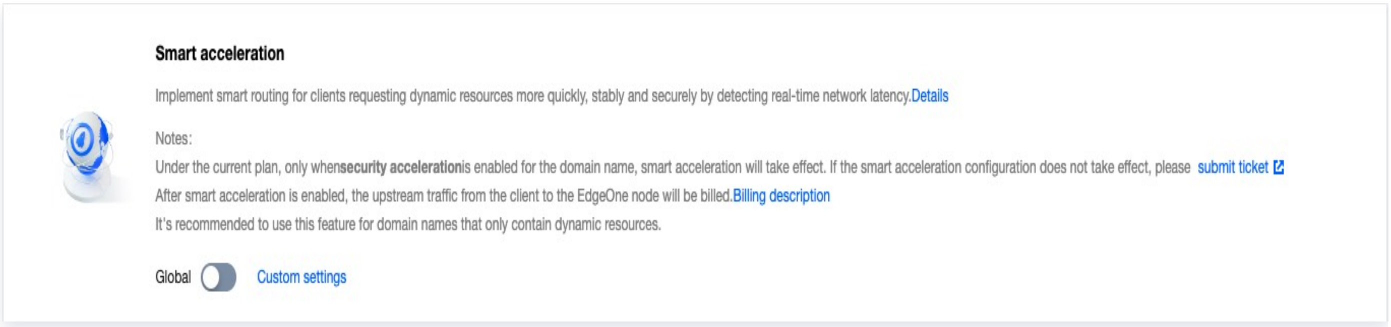
Billing

This is a paid service. After enabling smart acceleration, in addition to the original billing items, the upstream traffic from the client to the EdgeOne node will also be included in the secure acceleration traffic fee. Additionally, a value-added service fee will be charged based on the number of business requests. For more details, please refer to Billing Overview.

Scenario One: Smart acceleration is enabled for all domain names on the site.

If your site consists entirely of dynamic resources or a combination of dynamic and static resources, and you need to enable smart acceleration for the entire access site, please refer to the following steps:

1. Log in to the EdgeOne console. In the left sidebar, click **Site List**. Within the site list, click the **site** that needs to be configured to enter the site details page.
2. On the site details page, click **Site Acceleration** > **Smart Acceleration** to enter the Smart Acceleration details page.
3. Locate the Smart Acceleration configuration card, which is off by default. You can toggle the **switch** to enable or disable it.


Smart acceleration

Implement smart routing for clients requesting dynamic resources more quickly, stably and securely by detecting real-time network latency.Details

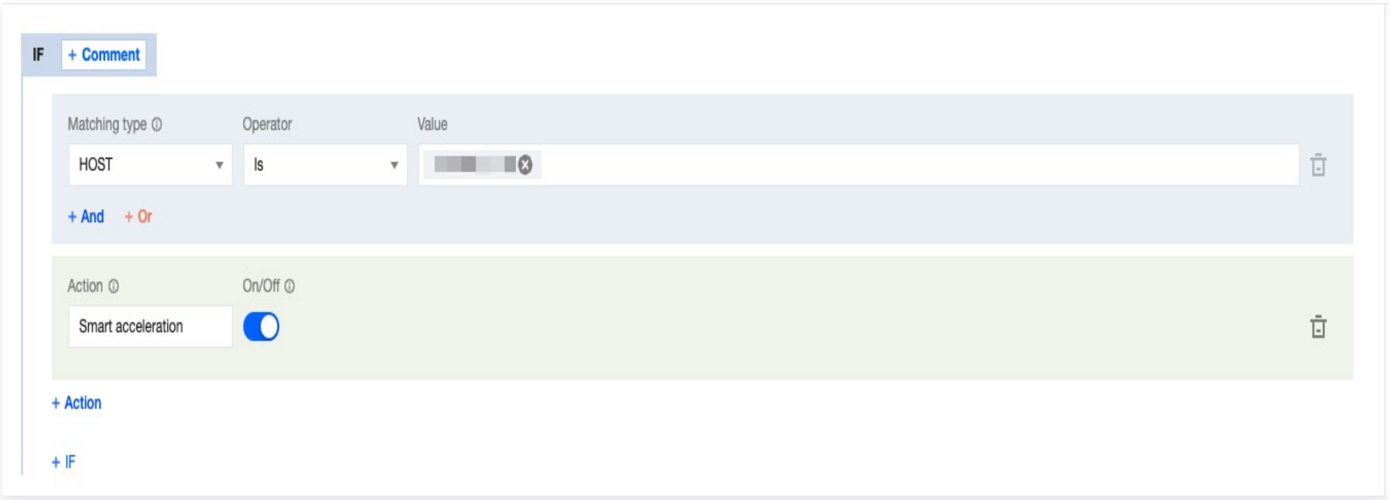
Notes:
Under the current plan, only when **security acceleration** is enabled for the domain name, smart acceleration will take effect. If the smart acceleration configuration does not take effect, please submit ticket
After smart acceleration is enabled, the upstream traffic from the client to the EdgeOne node will be billed.Billing description
It's recommended to use this feature for domain names that only contain dynamic resources.

Global Custom settings

Scenario Two: Enable smart acceleration for a specific domain name.

If only a specific domain under your site consists of purely dynamic resources or a combination of dynamic and static resources, and you need to enable smart acceleration for that specific domain, please refer to the following steps:

1. Log in to the Edge Security Acceleration Platform Console . In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page.
4. On the page that appears, select **HOST** from **Matching type** and specify an operator and a value to match the requests of specified domain names.
5. Click on **Operation** > **Selection Box**. In the pop-up operation list, select **Smart Acceleration**. Click on the **Switch** to enable/disable.



Matching type	Operator	Value
HOST	Is	[redacted]

Action	On/Off
Smart acceleration	On

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

References

What are dynamic and static resources?

- Static resource: When a user repeatedly accesses a particular resource and the content returned remains **consistent**. Examples include images, videos, software installation packages, compressed files, CSS, JavaScript files, and other content that does not change frequently.
- Dynamic resource: When a user accesses a particular resource multiple times and the content returned **varies** each time. This includes content that needs to be updated in real time based on user requests, user interactions, etc. Examples include API interfaces, files in `JSP`, `ASP`, `PHP`, Perl, and `CGI` formats.

Cache Configuration Overview

Last updated：2023-09-07 15:02:50

Upon integration with EdgeOne, the edge nodes will determine whether to cache the resource files in response to client requests based on the configured caching rules. Once a file is cached on an edge node, it can directly respond to identical file requests from other users. This effectively circumvents the need for long-link back-to-source scenarios, enabling faster response times for the most recent file requests.

You can customize your site's cache based on the following use cases:

Type	Use Cases	SDK
Customize EdgeOne Node Cache Rules	• The site/domain contains a variety of file resource types, necessitating the customization of cache durations for each resource type within the node. This ensures that users access the most recent files while reducing the volume of back-to-source requests. • The site/domain contains dynamic resources, and it is necessary to prevent these resources from being cached.	Node Cache TTL
	• When requesting the same path file, different URLs carrying parameters, request headers, and other content will point to different files; • When requesting the same path file, parameters carried by the URL, request headers, etc., do not affect the file version and should point to the same cached file.	Custom Cache Key
Cache Control Under Origin Anomalies	In the event of an abnormal response from the origin server, it is crucial to safeguard the server from further damage while still responding to client requests in a timely manner.	• Status Code Cache TTL • Offline

		Cache
Browser Cache Control	Desire to further enhance webpage loading speed, reduce traffic consumption, and allow the browser to cache static resource files for a certain duration.	Browser Cache TTL
Cache Purging	Purge cached resources within the node when the cached files have expired or non-compliant resources have been cached.	Cache Purge
Cache Prefetching	When a domain has just been integrated or a file updated, it is necessary to pre-cache the file within the EdgeOne node to enhance acceleration and reduce back-to-source traffic during peak times.	Cache Prefetching
Cache File Prefresh	For files with continuous user access, it is essential to ensure persistent caching within the node to prevent concentrated back-to-source after cache expiration. The validity of the file and the refresh of the cache time can be verified through cache prefresh.	Cache Prefresh

For further understanding of cache rules, you may refer to the following:

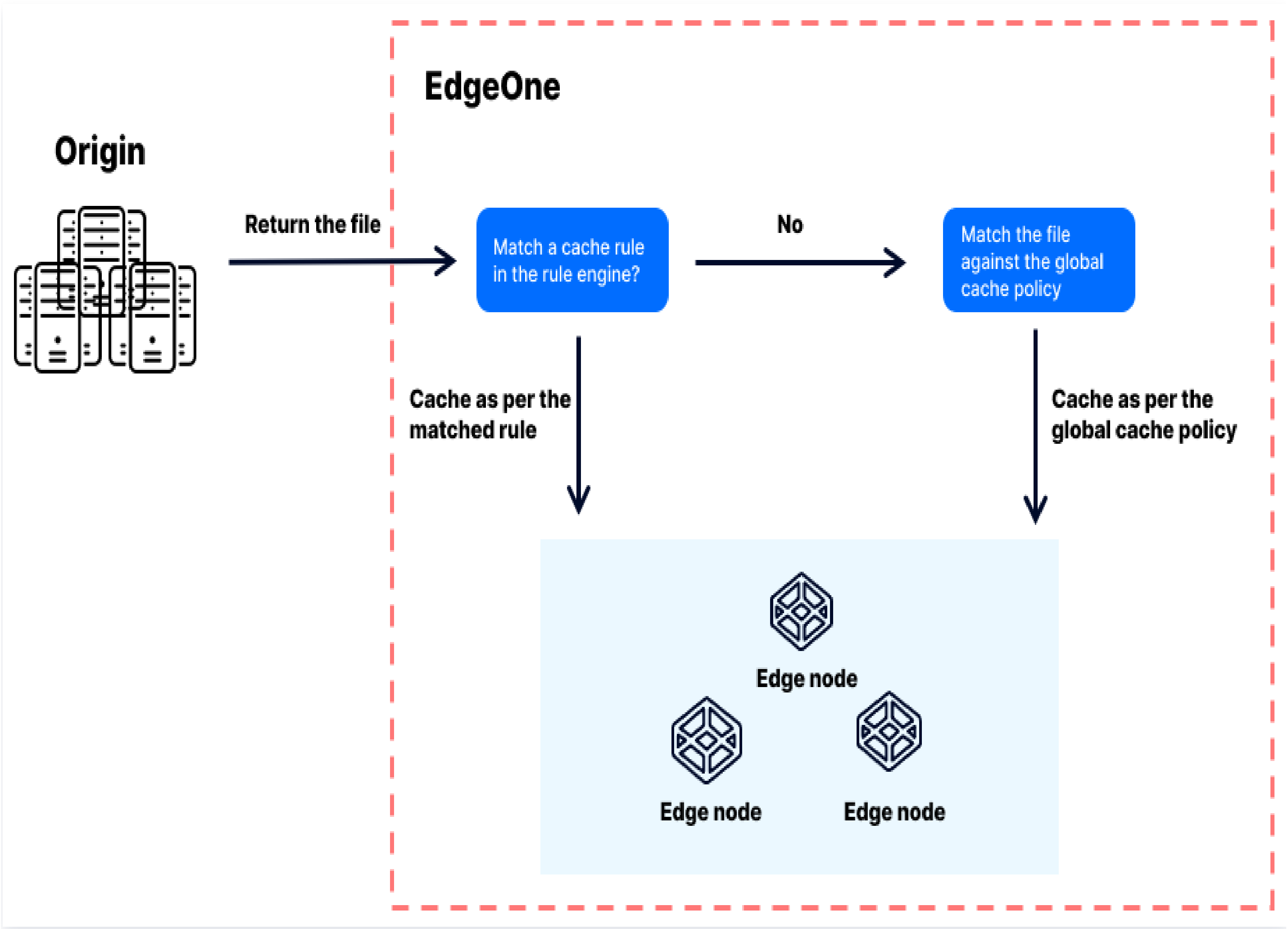
- Understand the content caching rules of EdgeOne
- Understanding the Function of Cache Keys
- Understanding the Function of the Vary Feature

EdgeOne Cache Rules
Content Cache Rules

Last updated：2023-09-07 15:02:59

Overview

Upon receiving an HTTP request from a client, an EdgeOne edge node will determine whether the current file hits the cache. If it does not, the node will request the latest file from the origin server. After the origin server correctly responds with the file, EdgeOne will cache the file according to the user's cache rules and the platform's default cache policy. You can learn how to customize your file cache rules by viewing How to Configure Cache Rules. After configuring the cache rules, they will take effect in the following order:


EdgeOne
Origin
Return the file
Match a cache rule in the rule engine?
No
Match the file against the global cache policy
Cache as per the matched rule
Cache as per the global cache policy
Edge node
Edge node
Edge node

⚠ **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

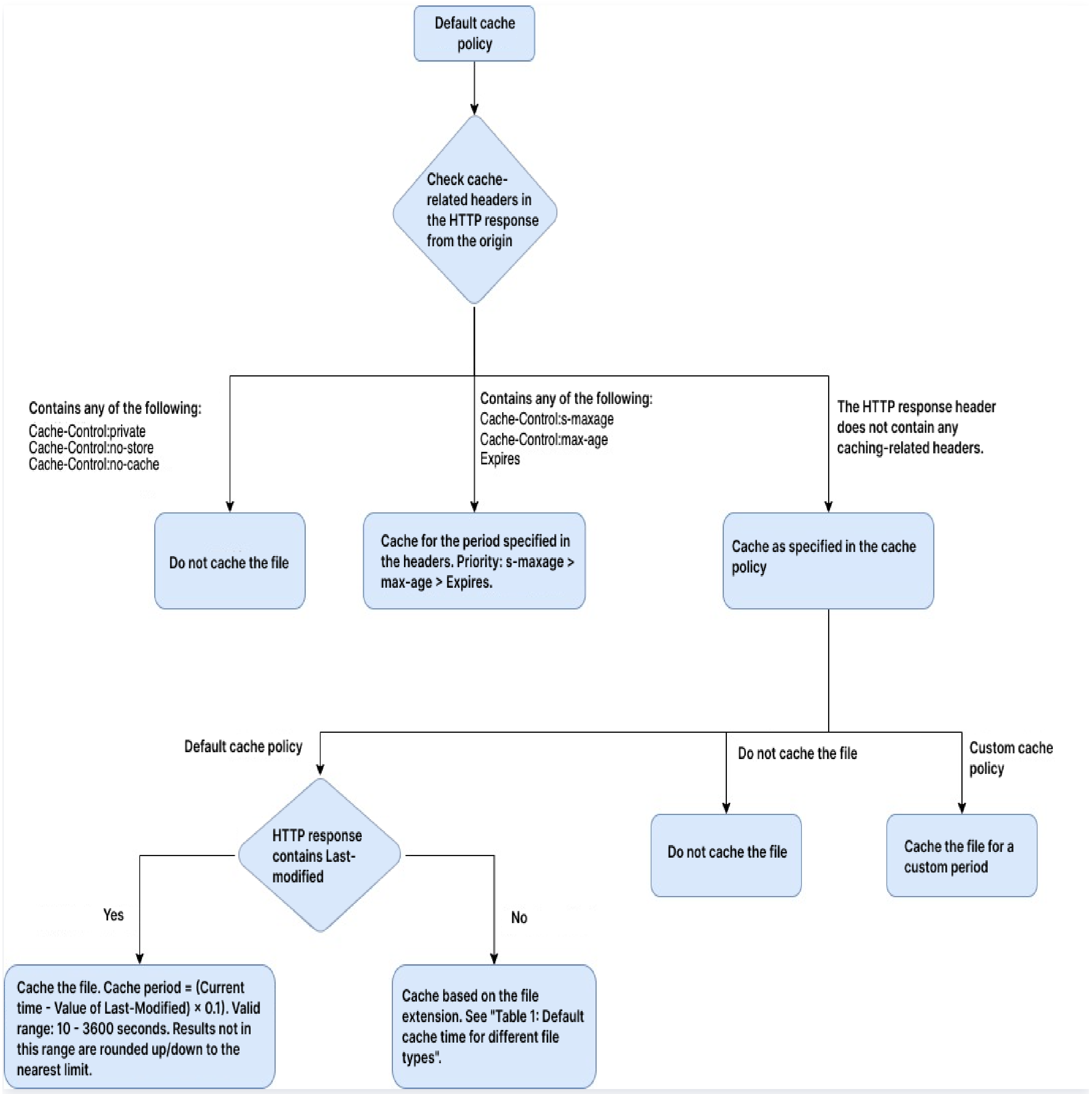
EdgeOne supports the following three types of cache policies:

- **Default Cache Policy:** Adheres to the EdgeOne default cache policy, determining the cache duration of a file within a node based on the `Cache-Control` and other cache headers in the HTTP response.
- **No-cache policy:** Rules are set in the rule engine to specify not to cache certain files or not to cache any files globally. This policy is suitable for dynamic files or files with frequently updated content.
- **Custom cache policy:** Files are cached according to the custom cache duration.

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:


Default cache policy
Check cache-related headers in the HTTP response from the origin
Contains any of the following:
Cache-Control:private
Cache-Control:no-store
Cache-Control:no-cache
Contains any of the following:
Cache-Control:s-maxage
Cache-Control:max-age
Expires
The HTTP response header does not contain any caching-related headers.
Do not cache the file
Cache for the period specified in the headers. Priority: s-maxage > max-age > Expires.
Cache as specified in the cache policy
Default cache policy
Do not cache the file
Custom cache policy
HTTP response contains Last-modified
Do not cache the file
Cache the file for a custom period
Yes
No
Cache the file. Cache period = (Current time - Value of Last-Modified) × 0.1). Valid range: 10 - 3600 seconds. Results not in this range are rounded up/down to the nearest limit.
Cache based on the file extension. See "Table 1: Default cache time for different file types".

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`) × 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		Suffix	Cache period
Dynamic files		php、aspx、asp、jsp、do、dwr、cgi、fcgi、action、ashx、axd、json	Do not cache
Static files	Image	jpg、png、jpeg、webp、gif、heif、heic、kpg、ico	2 hours
	Audio/Video	mp4、mp3、m3u8、ts、m4a、avi、m4s、ogg	
	Webpages	html、js、css	
	Packages	zip、7z、tar、br、gz、rar、bz2	
	Docu	doc、docx、xls、xlsx、pdf、ppt、pptx	

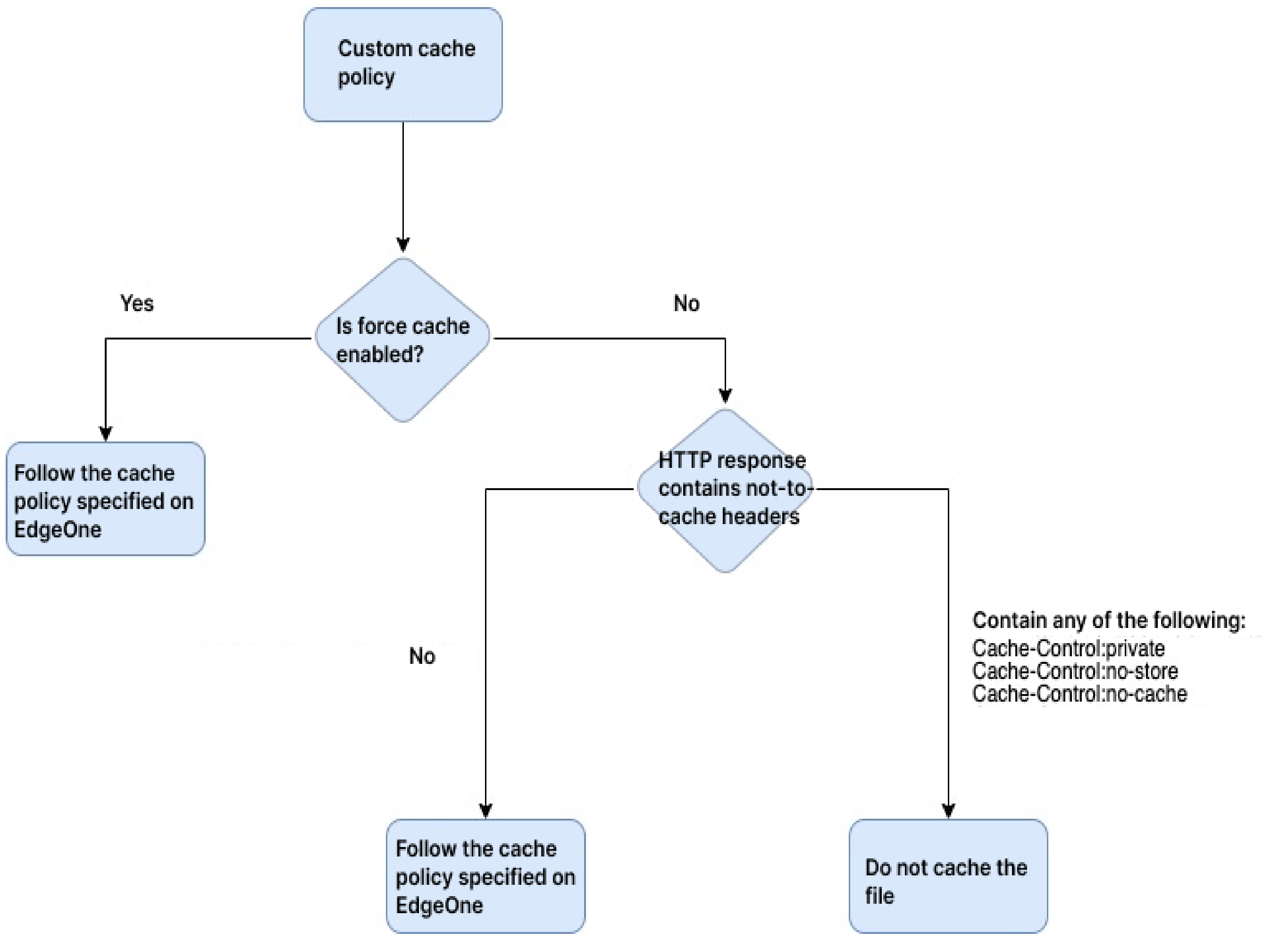
	ments		
	Applications	apk、exe、bin	
	Others	vsv、iso、jar、swf、chunk、atlas	
Other Files		N/A	Do not cache

- 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


Custom cache policy
Yes
Is force cache enabled?
No
Follow the cache policy specified on EdgeOne
HTTP response contains not-to-cache headers
No
Contain any of the following:
Cache-Control:private
Cache-Control:no-store
Cache-Control:no-cache
Follow the cache policy specified on EdgeOne
Do not cache the file

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: By default, force cache is enabled. Regardless of whether the origin server carries the `Cache-Control` or other cache headers, the corresponding header configurations are ignored. The file is cached according to the custom cache time configured within the EdgeOne platform.
- 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

- How to Configure Node Cache Rules

- How to Purge Cached Files within a Node
- How to Prefetch Hot Files to Nodes

Cache Key Introduction

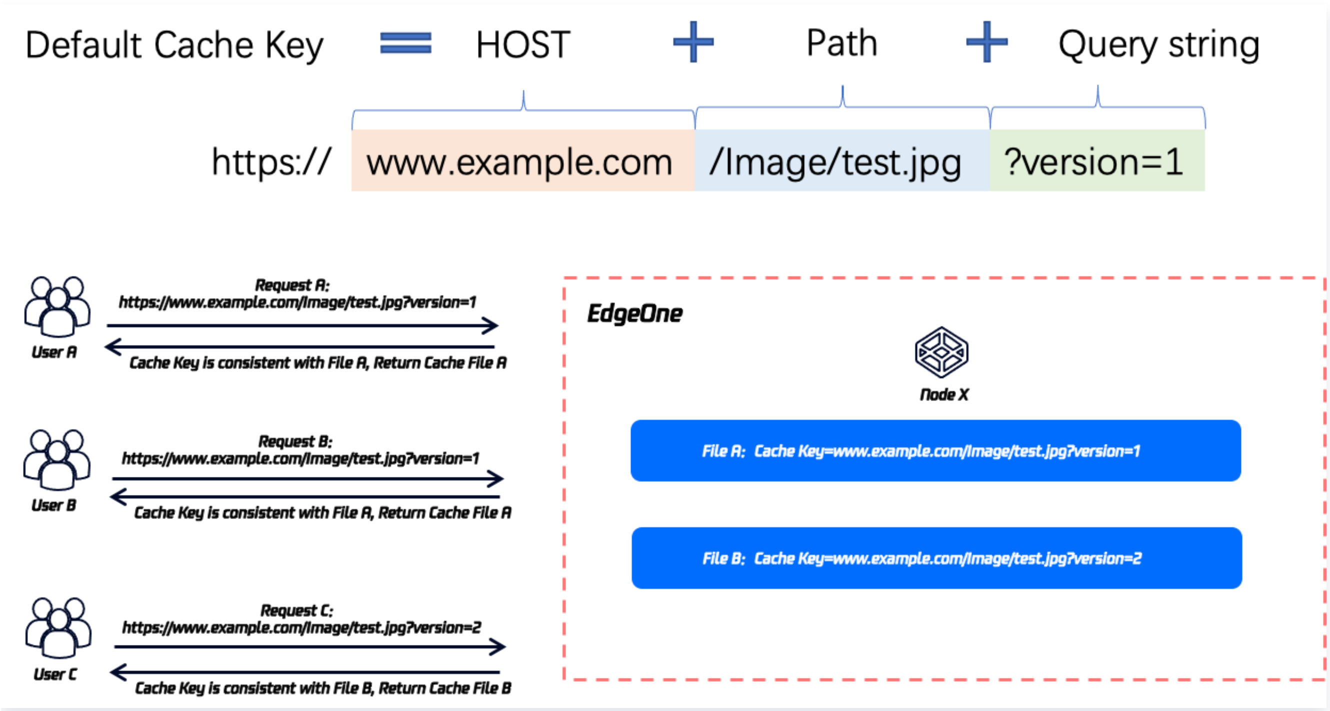
Last updated：2023-09-08 10:30:38

What is a Cache Key?

A Cache Key is used to determine whether the file resources accessed by users hit the EdgeOne edge cache content. It is the unique identifier of cached resources within a node. Cache hits can assist your site in the following ways:

- Reducing the volume of origin-pull requests, thereby decreasing the bandwidth consumption of the origin server.
- Enhancing the speed of user access requests.

When a file is cached in an EdgeOne edge node, the node generates a corresponding cache key identifier for the file based on the cache key rules. By default, the cache key is calculated using the client request URL and query string. When other clients initiate HTTP requests to the edge node, the node compares the HTTP request with the cache keys (Cache Key) of all cached resources in the node based on the Cache Key calculation rules. If they match, the corresponding cached resource is directly responded to the client, resulting in a cache hit.


Default Cache Key = HOST + Path + Query string
https:// www.example.com /Image/test.jpg ?version=1
Request A: https://www.example.com/Image/test.jpg?version=1
User A
Cache Key is consistent with File A, Return Cache File A
Request B: https://www.example.com/Image/test.jpg?version=1
User B
Cache Key is consistent with File A, Return Cache File A
Request C: https://www.example.com/Image/test.jpg?version=2
User C
Cache Key is consistent with File B, Return Cache File B
EdgeOne
Node X
File A: Cache Key=www.example.com/Image/test.jpg?version=1
File B: Cache Key=www.example.com/Image/test.jpg?version=2

Use Cases for the Cache Key

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

You can understand how to correctly configure the Cache Key to help you accurately match the requested cache files and simultaneously reduce the origin-pull rate, through the following scenario examples.
For instance, User A and User B have the following requests respectively:

- 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 I:** The file paths accessed by the user are identical, but the `version` parameter carried in the query string differs, resulting in version differentiation. The above requests correspond to two different images. In this case, the `version` parameter that affects the file version should be retained in the Cache Key to ensure that the node can correctly cache and respond with the corresponding file content.
- **Scenario Two:** The content of the query string in the URL accessed by the user does not affect the file content at all. The files corresponding to the above requests are consistent and do not affect the file version. In this case, all query strings should be ignored in the calculation of the Cache Key to improve the hit rate of the file within the node and reduce the request to origin.

Permits the customization of Cache Key content and its effective rules.

EdgeOne allows users to customize Cache Key rules, supporting the configuration of query strings, HTTP request headers, or cookies to differentiate caches. You can learn how to configure custom Cache Key rules through Customizing Cache Key.

Note:

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

Query String

A query string refers to the string parameters following the `?` in the request URL (containing one or more parameters, separated by `&`). For instance, in `https://www.example.com/images/example.jpg?color=blue&amp;size=large`, `color=blue&amp;size=large` is the query string.

EdgeOne supports custom retention of specified query string content to differentiate caches.

Note:
When retaining the query string as a Cache Key, any alteration in the order of parameters will result in a change to the Cache Key.

For instance, when the client initiates the following requests:

- 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	Validity
The query string is configured to retain all.	Request A and Request B carry different parameter contents, corresponding to different cache versions. Request A and Request C carry identical parameter contents, but in a different order, also corresponding to different cache versions.
The configuration of the query string is set to ignore all.	Requests A, B, and C all correspond to the same cache version.
The query string is configured to retain the specified parameter `Type`.	Requests A and B, with identical parameter order and content, correspond to the same cache version; whereas requests B and C, with the same parameter content but different order, correspond to different cache versions.
The query string is configured to ignore the specified parameter `Type`.	Request A and Request B have inconsistent remaining parameter content, corresponding to different cache versions; Request A and Request C have consistent remaining parameter content, but in a different order, corresponding to different cache versions.

HTTP request header

EdgeOne supports the inclusion of specified HTTP request headers in the calculation of the Cache Key. EdgeOne edge nodes will establish different cache versions based on the content

of the request headers. Changes in the order of request headers do not affect the calculation of the Cache Key.
For instance, if the HTTP request header `User-Agent` is included in the Cache Key calculation, and requests A and B have identical URLs and parameter content, but different `User-Agent` header content, they will correspond to different cache versions.

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

Cookie

EdgeOne supports the inclusion of specified parameters within the Cookie into the calculation of the Cache Key, distinguishing cache versions based on Cookie parameters and content. When multiple parameters within the Cookie participate in the Cache Key calculation, changes in the order of parameters do not affect the calculation of the Cache Key.
For instance, when the Cookie parameter `User` is included in the Cache Key calculation, Request A and Request B, having the same parameter content, correspond to the same cache. However, Request A and Request C, with different parameter content, correspond to different versions of the cache.

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

Learn more

- Understanding the content caching rules of EdgeOne
- How to Configure a Custom Cache Key
- How to Configure Node Cache Rules
- How to Purge Node Cache Resources

Vary Feature

Last updated：2023-09-07 15:03:12

Support for Vary

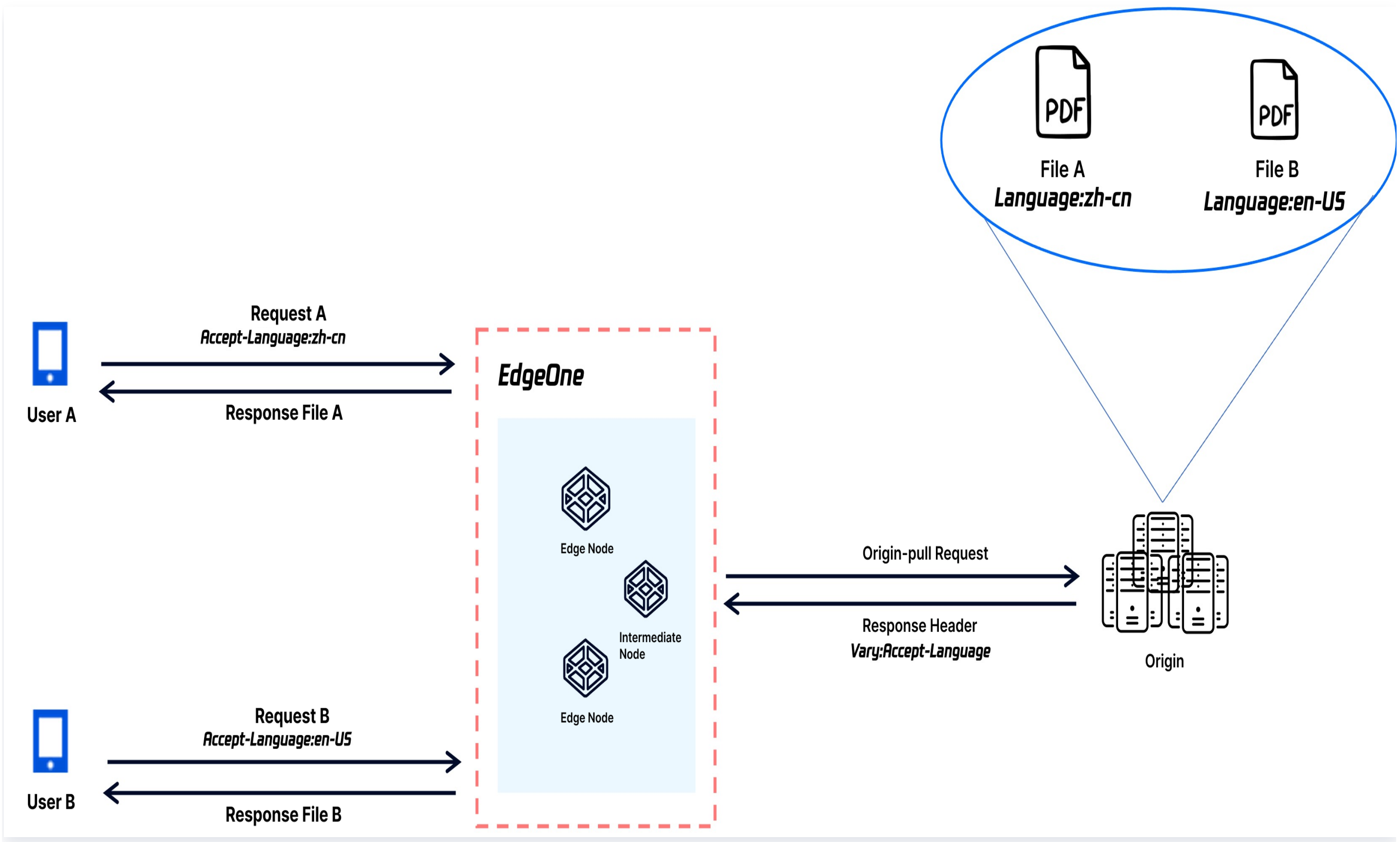
EdgeOne now fully supports the Vary feature across all platforms by default. There is no need for any configuration on your part. Simply add a Vary header to the response header when responding from the origin as needed. For more information on the standards of a Vary header, please refer to Vary.

What Is the Vary Feature?

Vary is an HTTP response header introduced after HTTP/1.1. When a client sends a request to the origin server using the same URL, if the origin server has different versions of file content, it may be cached by an intermediate caching system (such as browser cache or Content Delivery Network, CDN), making it impossible to distinguish the scenario response files. Therefore, the origin server can add a Vary header to the HTTP response to inform the intermediate caching system which request header to use to differentiate the cached version content.

For instance, when all client requests are for `https://www.example.com/test.pdf`, the origin server, in order to respond with different versions of the file based on the client's language, adds `Vary: Accept-Language` to the HTTP response header. EdgeOne, when establishing a cache, will create different versions of the cache based on the `Accept-Language` content requested by the client.

- 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.

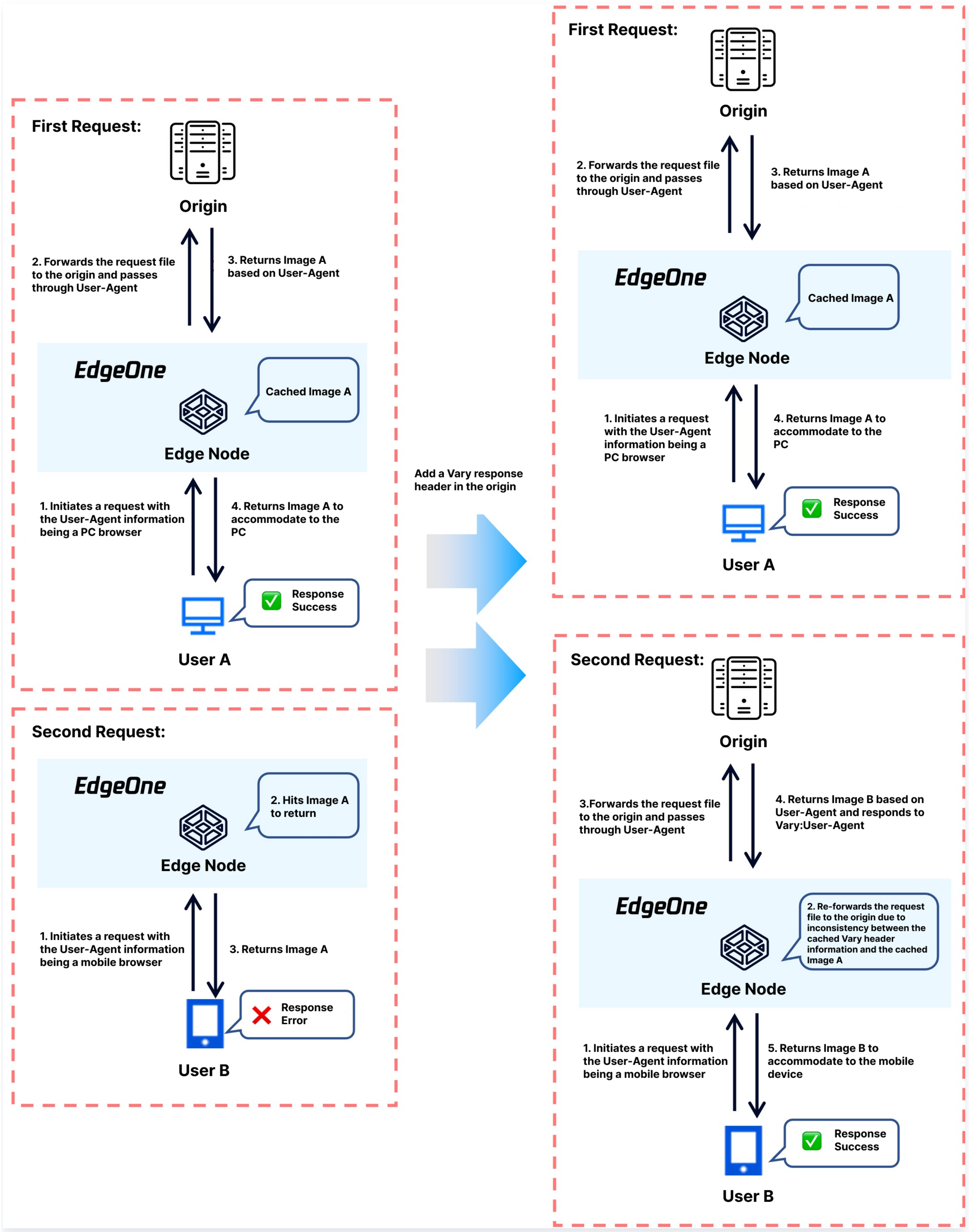


Application Scenarios of Vary

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:

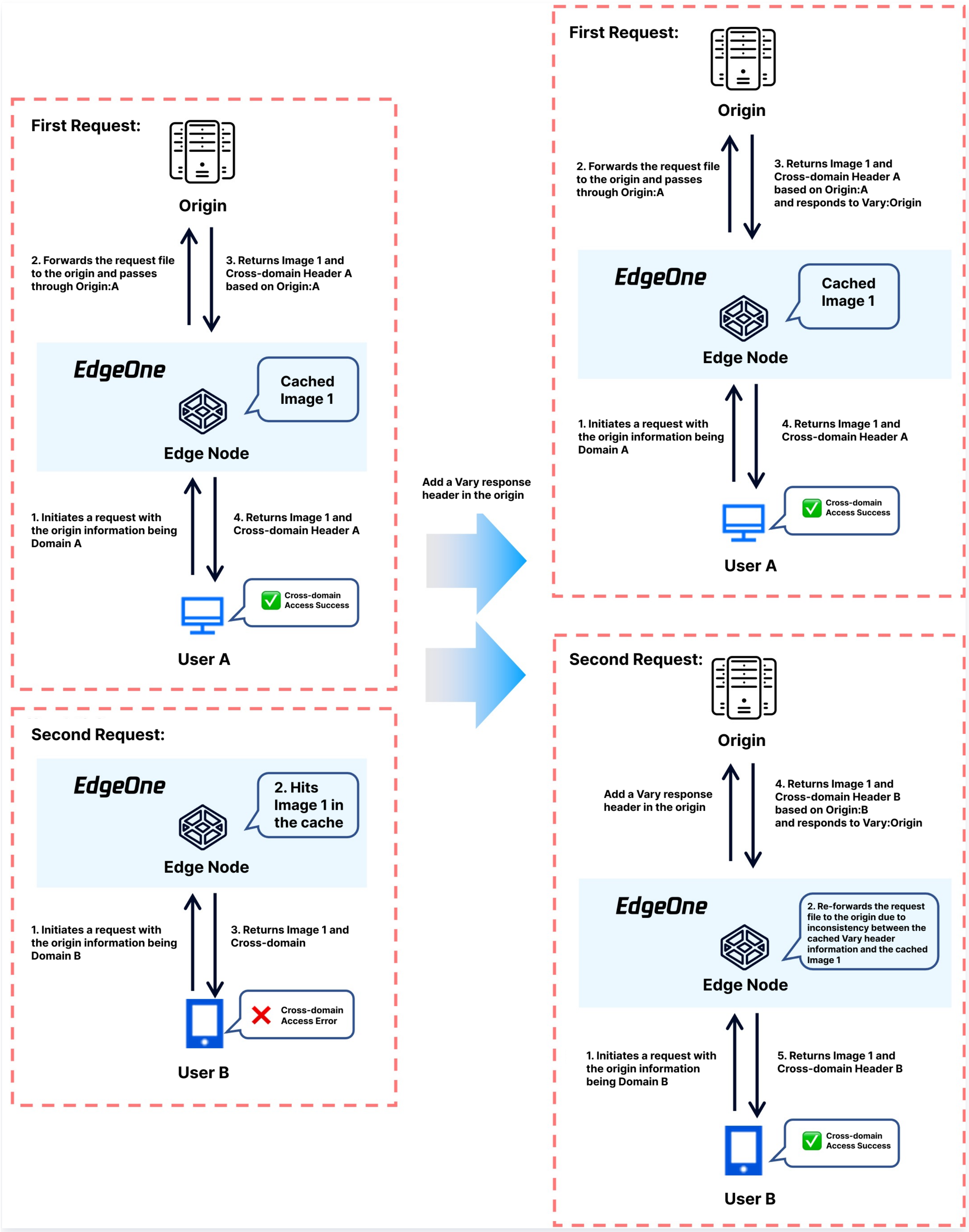
Scenario I: Distinguishing response files by client type when the request URLs are the same

Suppose a website has both PC and mobile pages. For user convenience, the resources for both the PC and mobile versions of the site are accessed via the same URL. However, due to differences in screen resolution, the front end needs to adapt the content to the mobile version, resulting in inconsistent image resolution. At this point, it is necessary to distinguish the response images based on the `User-Agent` header carried in the user's request: Image A is adapted for the PC version, and Image B is adapted for the mobile version. To prevent the response image content from being cached and causing the mobile version to display the same content as the PC version, the origin server can add `Vary:User-Agent` when responding with an image, instructing the node to differentiate the cache based on the user's request `User-Agent` header.


First Request:
Origin
2. Forwards the request file to the origin and passes through User-Agent
3. Returns Image A based on User-Agent
EdgeOne
Cached Image A
Edge Node
1. Initiates a request with the User-Agent information being a PC browser
4. Returns Image A to accommodate to the PC
Response Success
User A
Second Request:
EdgeOne
2. Hits Image A to return
Edge Node
1. Initiates a request with the User-Agent information being a mobile browser
3. Returns Image A
Response Error
User B
Add a Vary response header in the origin
First Request:
Origin
2. Forwards the request file to the origin and passes through User-Agent
3. Returns Image A based on User-Agent
EdgeOne
Cached Image A
Edge Node
1. Initiates a request with the User-Agent information being a PC browser
4. Returns Image A to accommodate to the PC
Response Success
User A
Second Request:
Origin
3.Forwards the request file to the origin and passes through User-Agent
4. Returns Image B based on User-Agent and responds to Vary:User-Agent
EdgeOne
2. Re-forwards the request file to the origin due to inconsistency between the cached Vary header information and the cached image A
Edge Node
1. Initiates a request with the User-Agent information being a mobile browser
5. Returns Image B to accommodate to the mobile device
Response Success
User B

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

Suppose there is an image file on the current page that is referenced by both domain A and domain B. To avoid cross-domain errors, it is usually necessary to add `Access-Control-Allow-Origin` to the file when responding from the origin to allow cross-domain access. However, if the current file is cached, the cross-domain header may also be cached, leading to cross-domain access errors. In this case, the origin can add a `Vary: Origin` header while responding with the `Access-Control-Allow-Origin` cross-domain header, to specify that the node should differentiate the cache based on the user's request `Origin` header.


First Request:
Origin
2. Forwards the request file to the origin and passes through Origin:A
3. Returns Image 1 and Cross-domain Header A based on Origin:A
EdgeOne
Cached Image 1
Edge Node
1. Initiates a request with the origin information being Domain A
4. Returns Image 1 and Cross-domain Header A
Cross-domain Access Success
User A
Second Request:
EdgeOne
2. Hits Image 1 in the cache
Edge Node
1. Initiates a request with the origin information being Domain B
3. Returns Image 1 and Cross-domain
Cross-domain Access Error
User B
Add a Vary response header in the origin
First Request:
Origin
2. Forwards the request file to the origin and passes through Origin:A
3. Returns Image 1 and Cross-domain Header A based on Origin:A and responds to Vary:Origin
EdgeOne
Cached Image 1
Edge Node
1. Initiates a request with the origin information being Domain A
4. Returns Image 1 and Cross-domain Header A
Cross-domain Access Success
User A
Second Request:
Origin
Add a Vary response header in the origin
4. Returns Image 1 and Cross-domain Header B based on Origin:B and responds to Vary:Origin
EdgeOne
2. Re-forwards the request file to the origin due to inconsistency between the cached Vary header information and the cached Image 1
Edge Node
1. Initiates a request with the origin information being Domain B
5. Returns Image 1 and Cross-domain Header B
Cross-domain Access Success
User B

Cache Configuration
Custom Cache Key

Last updated：2023-09-07 15:03:19

Overview

When you need to direct requests from the same path URL to different files based on request parameters, cookies, or HTTP request headers, or when you want to direct requests from URLs with different parameters to the same file, the custom Cache Key allows you to customize the Cache Key identifier for resources cached within the node. This includes concatenating query strings, HTTP headers, or Cookie information, so that the request URL can correctly obtain the corresponding cached resources according to different scenarios. You can learn about what a Cache Key is through the Cache Key Introduction.

Use Cases

Scenario I: The file paths accessed by the user are identical, but there will be version distinctions based on the carried query strings, HTTP request headers, and Cookie content. The cache key for this type of file can be adjusted through customizing the Cache Key to differentiate file caching.
Scenario II: The content of the query string in the URL accessed by the user does not affect the file content at all. The files corresponding to the above requests are consistent and do not affect the file version. The cache key for this type of file can be adjusted through customizing the Cache Key to make requests hit the same file cache.

Instructions

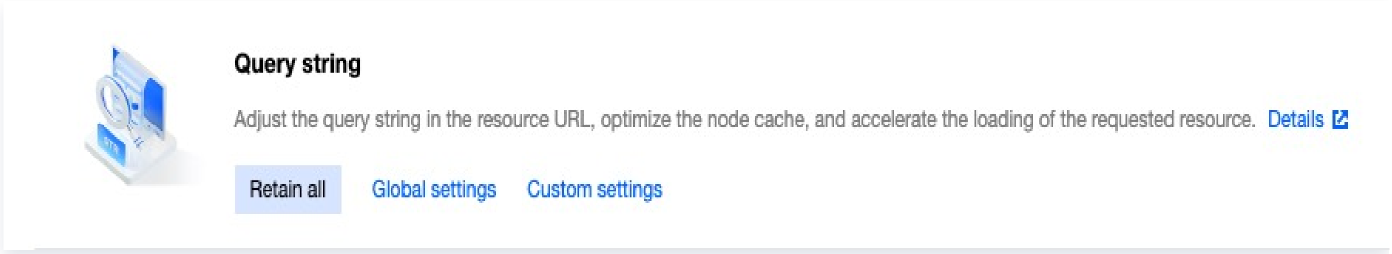
Scenario 1: Configuring Custom Cache Key for All Domain Names of the Site

To configure a custom Cache Key for all domain names used to access a site, or as a fallback configuration at the site level, refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Site Acceleration > Cache Configuration**, find the Query String card, and click on the global site settings to configure.

Note:

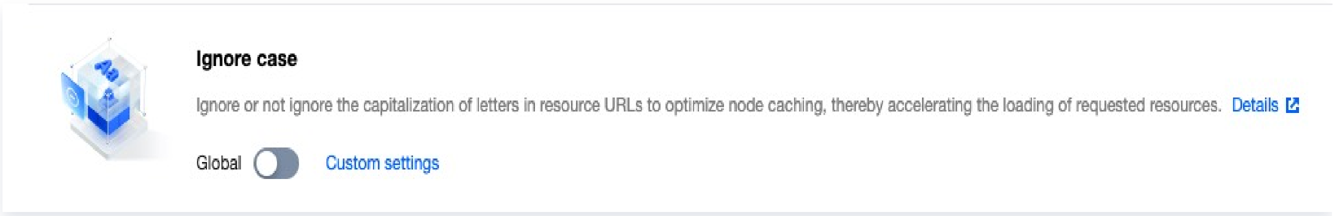
> Global configuration only allows for the configuration of **Query String** and **Case Ignoring**. For more extensive custom Cache Key configuration options, please refer to the configuration steps in Scenario II's rule engine.



Query string

Adjust the query string in the resource URL, optimize the node cache, and accelerate the loading of the requested resource. Details

- The default configuration is to retain all, that is, to keep all query parameters of the original request URL as the Cache Key. Other options are supported: a. Ignore all: Ignore the entire query string; b. Retain specified parameters: Only retain the specified parameters in the query string; c. Ignore specified parameters: Only ignore the specified parameters in the query string.



Ignore case

Ignore or not ignore the capitalization of letters in resource URLs to optimize node caching, thereby accelerating the loading of requested resources. Details

- By default, case sensitivity is not ignored. Even if the URL content is the same, different letter cases are considered as different Cache Keys. When case insensitivity is enabled, different letter cases will be considered as the same Cache Key.

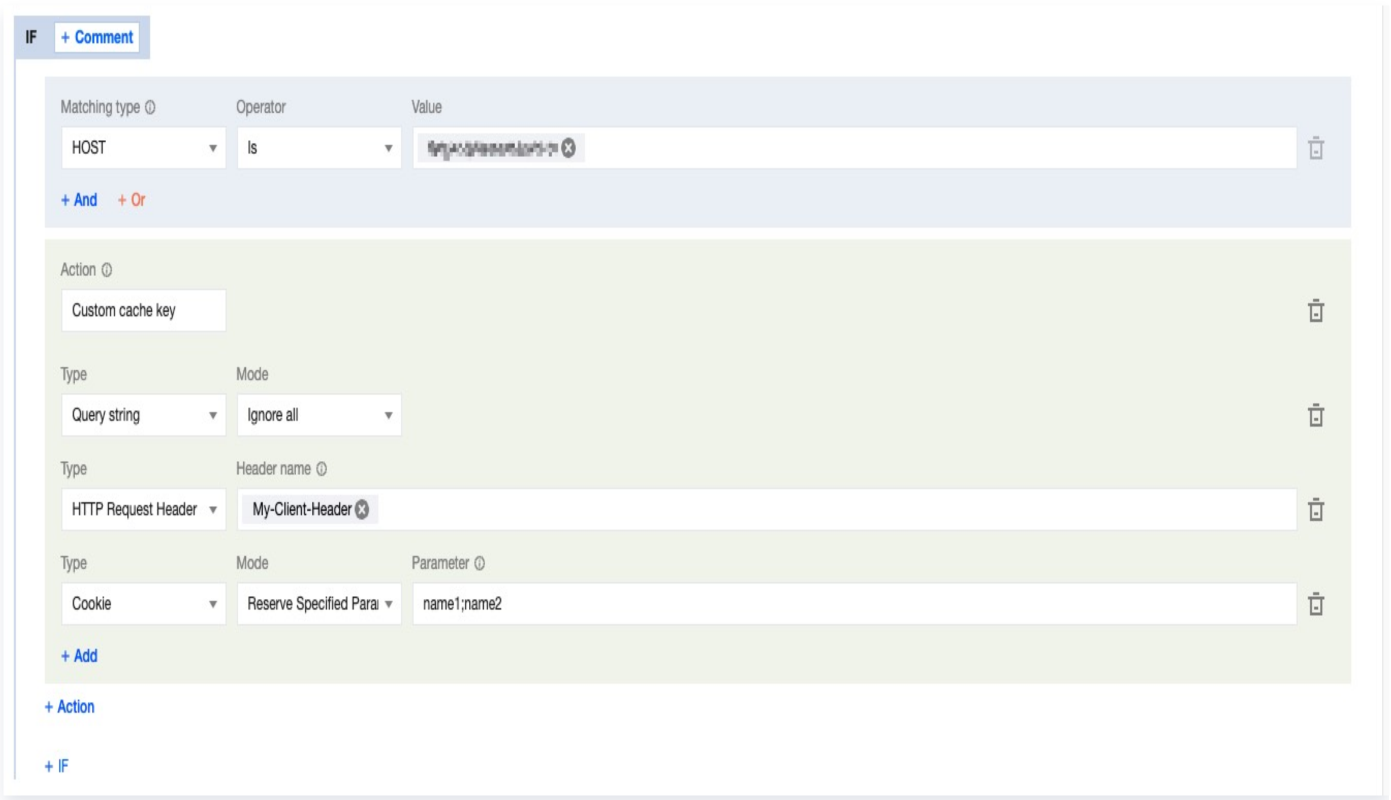
Scenario 2: Configuring Custom Cache Key for Specific Domain Names, Paths, or File Extensions Request Granularity

If you need to configure a custom Cache Key rule for the domain `www.example.com` under the site `example.com` to ignore all query strings, and use the HTTP request header `My-Client-Header` and the parameters `name1` , `name2` in the Cookie as the Cache Key, you can refer to the following steps for configuration:

Instructions

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page. On the rule editing page, select Host as the matching type and configure it as `www.example.com` .

4. Click **Operation,** and in the pop-up operation list, select **Customize Cache Key**.
5. Click **Add** below the type to add a custom Cache Key type. In this example scenario, add query strings, HTTP request headers, and Cookies, and fill in the corresponding content. The complete rule configuration is as follows:



Matching type: HOST | Operator: Is | Value: [redacted]

Action: Custom cache key

Type: Query string | Mode: Ignore all

Type: HTTP Request Header | Header name: My-Client-Header

Type: Cookie | Mode: Reserve Specified Parai | Parameter: name1;name2

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

Effective Examples

Upon configuration completion, the Cache Key is composed of URL+My-Client-Header+Cookie: all query strings are ignored, and the `My-Client-Header` and the Cookie after retaining specified parameters are concatenated.
Request A from the client:
URL: `https://www.example.com/path/demo.jpg?key1=value1&key2=value2` 。
HTTP Request Header: Includes `My-Client-Header:fruit` .
Cookie: name1=yummy;name2=tasty;name3=strawberry。
Request B from the client:
URL: `http://www.example.com/path/demo.JPG?key1=value1&key2=value2&key3=value3` 。
HTTP Request Header: Includes `My-Client-Header:fruit` .
Cookie: name1=yummy;name2=tasty;name3=blueberry。
Request C from the client:
URL:
`http://www.example.com/path/demo.JPG?key1=value1&key2=value2&key3=value3&key4=value4`
。

HTTP Request Header: Includes `My-Client-Header:sea` .
Cookie：name1=yummy;name2=tasty;name3=fish。
Requests A and B will hit the same cached resource, and request C will hit another.

Node cache TTL

Last updated：2023-09-07 15:03:24

Overview

Node Cache TTL is used to determine whether resources are cached in EdgeOne nodes and the duration of such caching. When users request files that are either expired or not cached, the node will not directly respond to the user's request. Instead, it will fetch the latest resources from the origin server for response, and decide whether to cache it in EdgeOne based on the caching rules. Caching files and enabling user requests to hit the cache can assist you in:

- Reducing the volume of origin-pull requests, thereby decreasing the bandwidth consumption of the origin server.
- Enhancing the speed of user access requests.

Based on your business needs, you can customize the cache duration for different resources, optimize the caching strategy for various resources, and improve the loading speed of requested resources. For more information on caching, please refer to EdgeOne Content Caching Rules.

Note:

1. If the resources on the origin server are updated and you need to immediately refresh the node cache, you can use the Clear Cache function to actively remove the unexpired old cache from the node, ensuring that subsequent requests can access the latest resources from the origin server.
2. Please refrain from caching dynamic resources within edge nodes to prevent users from accessing incorrect content.
3. 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.

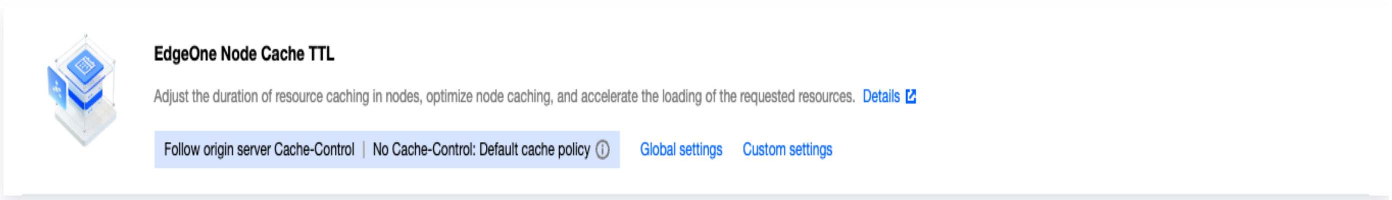
Instructions

Scenario 1: Configuring Node Cache TTL for All Domain Names of a Site

To configure the same Node Cache TTL for all domain names used to access a site, or as a fallback configuration at the site level, refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on the site list and select the site you wish to configure from within the list.

2. On the site details page, click on **Site Acceleration** > **Cache Configuration** and locate the Node Cache TTL card.
3. Click on **Global Site Settings** to configure. For detailed configuration instructions, please refer to EdgeOne Content Caching Rules.



EdgeOne Node Cache TTL

Adjust the duration of resource caching in nodes, optimize node caching, and accelerate the loading of the requested resources. Details

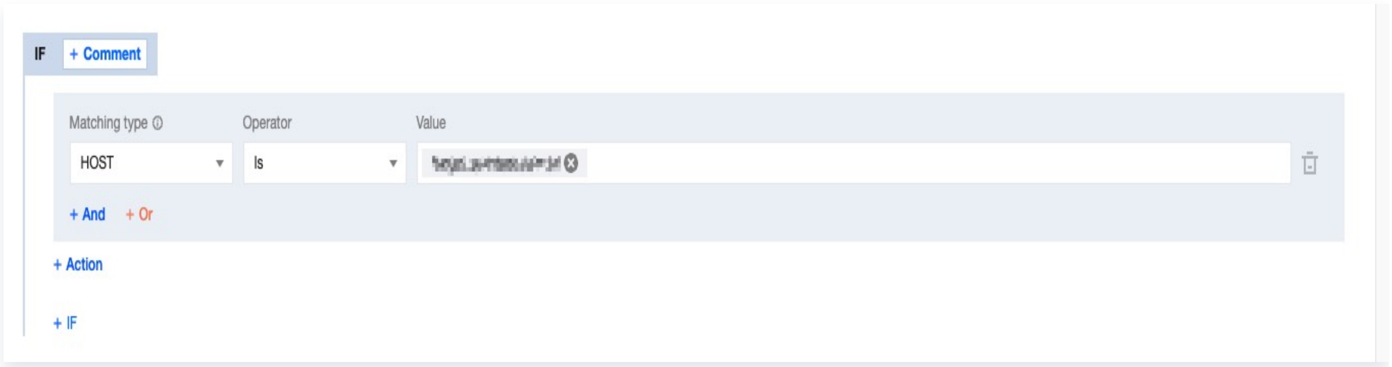
Follow origin server Cache-Control | No Cache-Control: Default cache policy

- Default Configuration: Adheres to the origin server's `Cache-Control`. If the origin server does not have a `Cache-Control`, it follows the EdgeOne default caching policy.

Scenario 2: Configuring Node Cache TTL for specific domain names, paths, or file extensions request granularity.

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

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page.
4. On the rule editing page, first select the matching type as HOST and the value as `www.example.com` as the outermost matching condition, then click **Add IF**.

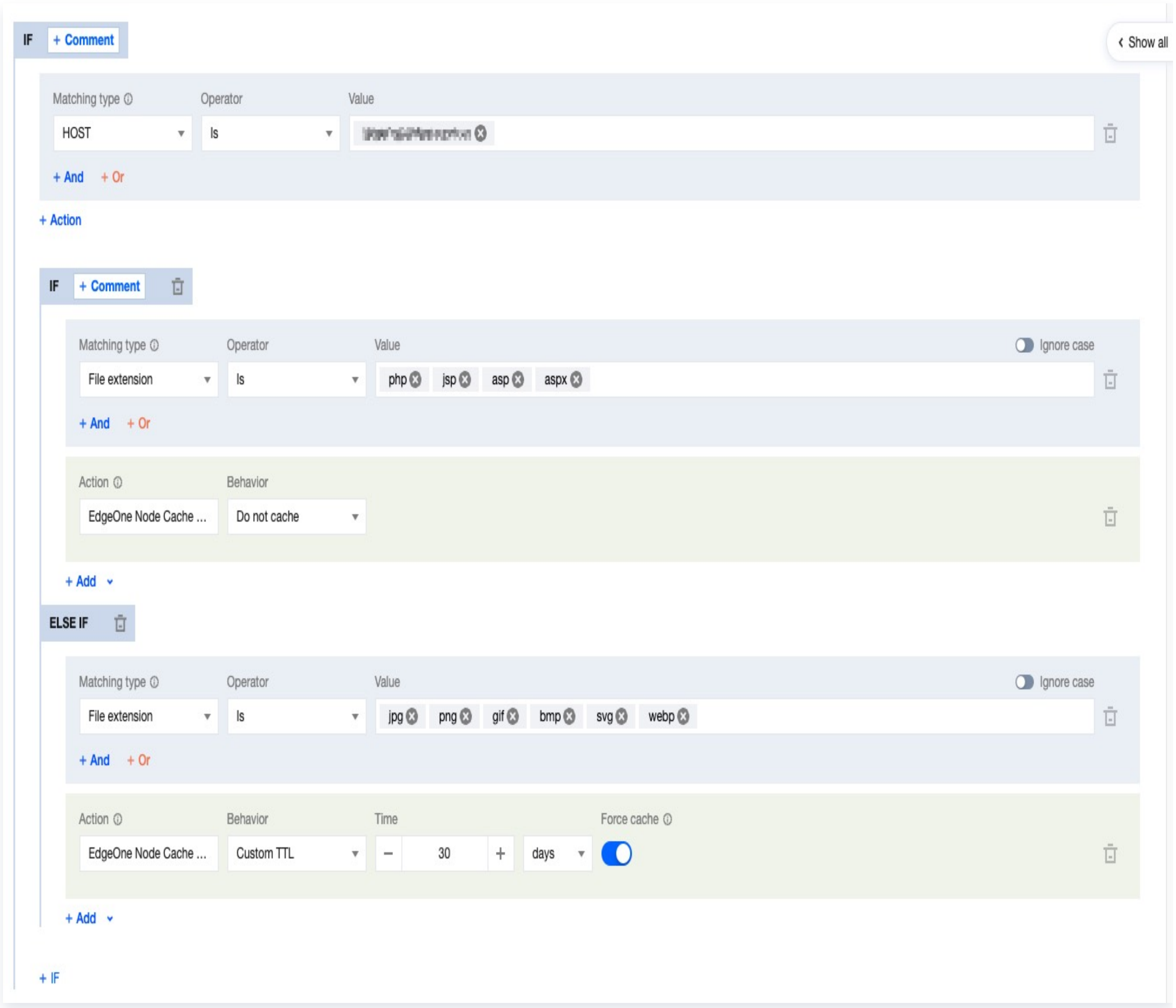


IF

Matching type	Operator	Value
HOST	Is	[illegible]

5. Within the newly added IF condition, select the match type as file extension, add the `php/jsp/asp/aspx` extensions, and click **Action**. In the pop-up action list, select the action as **Node Cache TTL,** and configure it to **No Cache.**
6. Repeat the above steps to add another IF condition, include the `jpg/png/gif/bmp/svg/webp` suffix, and configure it to cache for 30 days.

7. After configuring the rules as shown below, click **Save and Publish** to complete the rule configuration.



IF + Comment

Matching type: HOST Operator: Is Value: [redacted]

+ And + Or

+ Action

IF + Comment

Matching type: File extension Operator: Is Value: php jsp asp aspx Ignore case

+ And + Or

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

+ Add

+ IF

Status Code Cache TTL

Last updated：2023-09-07 15:03:30

Overview

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 within EdgeOne for direct response next time. If the origin responds with abnormal status codes such as 4xx or 5xx, and EdgeOne cannot obtain the resources, the next request will still trigger a return to the origin, which may put significant pressure on the origin. By configuring the status code cache TTL, EdgeOne can directly respond with the abnormal status code within the cache time, rather than triggering a return to the origin for all requests. This can alleviate the pressure on the origin and improve response speed.
The following status codes can currently be configured:

- **4xx：** 400、401、403、404、405、407、414。
- **5xx：** 500、501、502、503、504、509、514。

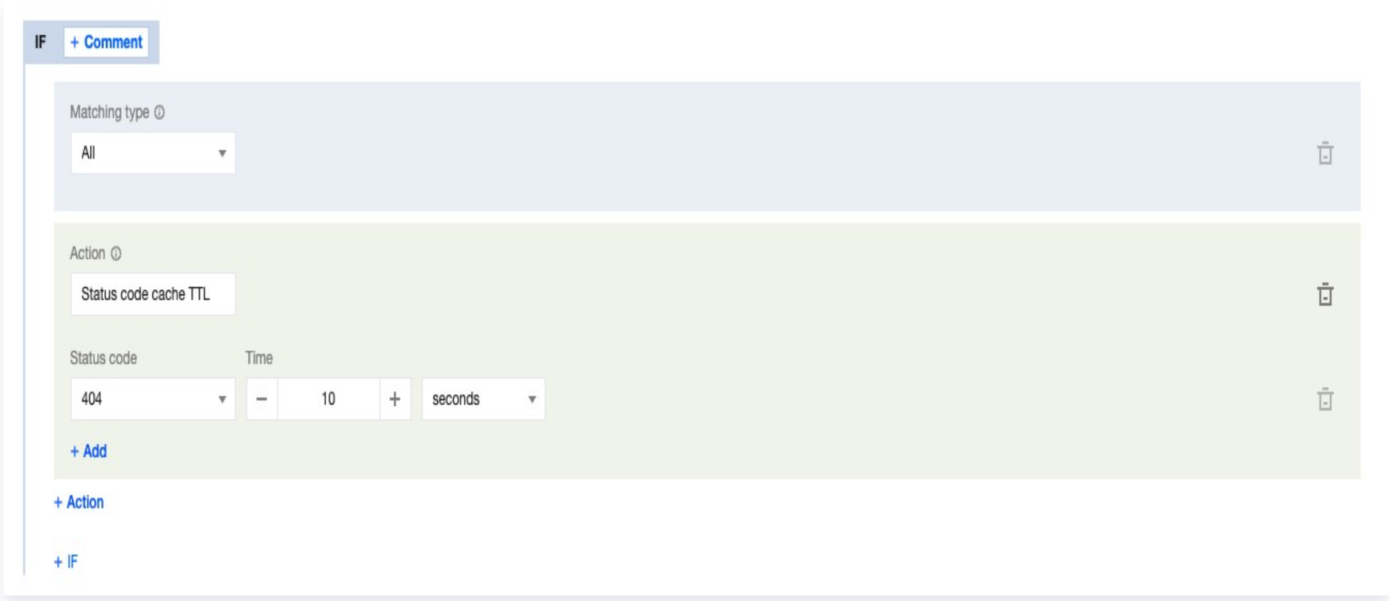
> **Note:**
> By default, EdgeOne caches the 404 status code for 10 seconds.

Instructions

Scenario 1: Configuring Status Code Cache TTL for All Domain Names of a Site

To configure the status code cache TTL for all domain names used to access a site, or as a fallback configuration at the site level, refer to the following steps:

1. Log in to the Edge Security Acceleration Platform Console . In the left-hand menu, click on **Site List.** Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to enter the new rule editing page. On the rule editing page, select the match type as 'All' (any site request).
4. Click on **Operation**, in the pop-up operation list, select **Status Code Cache**, and configure the corresponding cache status code and cache time.

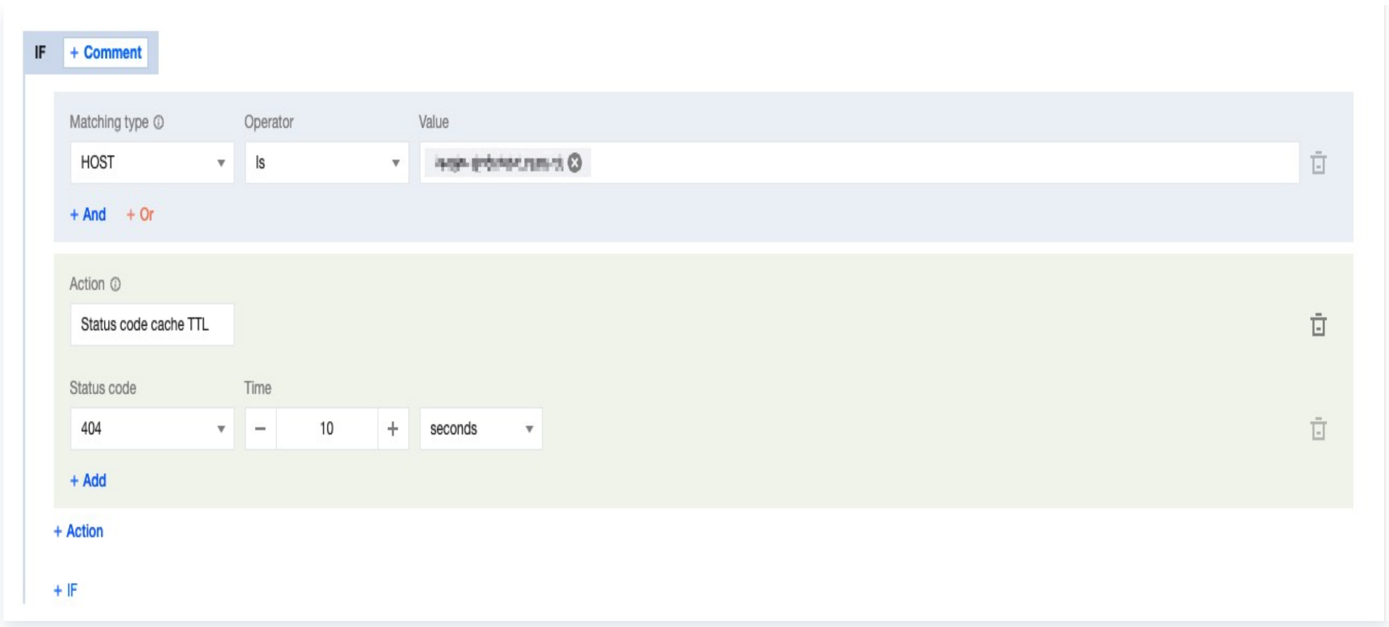


5. Click "Save and Publish" to complete the configuration of this rule.

Scenario 2: Configuring Status Code Cache TTL for Specific Domains, Paths, or File Extensions

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

1. Log in to the Edge Security Acceleration Platform Console . In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page. On the rule editing page, select Host as the matching type and configure it as `www.example.com` .
4. Click **Operation**, in the pop-up operation list, select **Status Code Cache** and configure the corresponding cache status code and cache time.



IF + Comment

Matching type: HOST | Operator: Is | Value: [redacted]

+ And + Or

Action: Status code cache TTL

Status code: 404 | Time: 10 seconds

+ Add

+ Action

+ IF

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

Browser Cache TTL

Last updated：2023-09-07 15:04:39

Overview

The client browser cache TTL refers to the duration of resource caching in the browser. By default, it follows the `Cache-Control` header of the origin server. You can control the duration of resource caching in the browser by configuring the browser cache TTL of EdgeOne, without modifying the configuration of the origin server.
EdgeOne implements browser cache TTL by setting the `Cache-Control` response header when responding to the client. The following configurations are supported:

- **Follow origin server:** Adhere to the origin server's `Cache-Control`. If the origin server does not have a `Cache-Control`, no changes will be made.
- **No-cache policy:** Regardless of whether the origin server carries the `Cache-Control`, the browser is controlled not to cache the file.
- **Custom time:** Regardless of whether the origin server carries the `Cache-Control`, the `max-age` will be modified to the specified cache time.

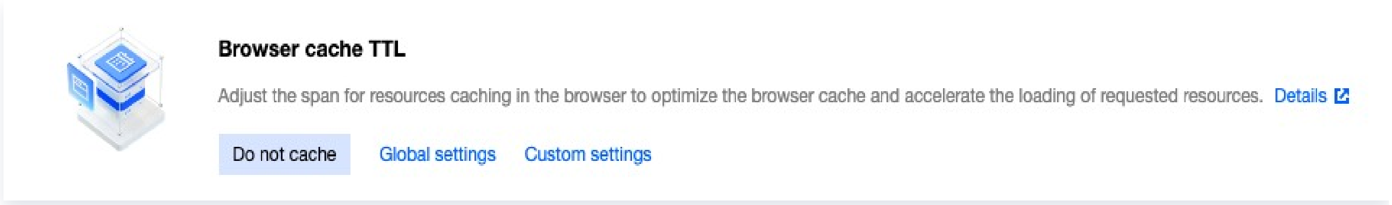
You can control the caching duration of resources in the browser by adjusting the browser cache TTL, optimize the caching strategy for different resources, and enhance the loading speed of requested resources.

Instructions

Scenario 1: Configuring Browser Cache TTL for All Domain Names of a Site

If you need to configure the same browser cache TTL for all domain names used to access a site, or as a fallback configuration at the site level, refer to the following steps:

1. Log in to the EdgeOne console. In the left menu bar, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. Within the site details page, click on **Site Acceleration > Cache Configuration**, locate the browser cache TTL card, and click on **Global Site Settings** to make modifications.


Browser cache TTL

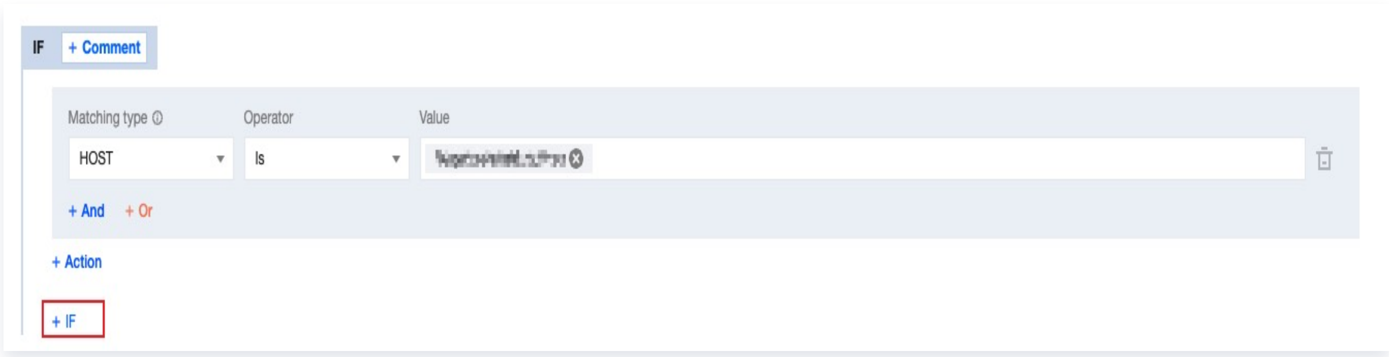
Adjust the span for resources caching in the browser to optimize the browser cache and accelerate the loading of requested resources. Details

- Default configuration: Supports following the `Cache-Control` of the origin server. If the origin server does not have a `Cache-Control`, caching is not performed.

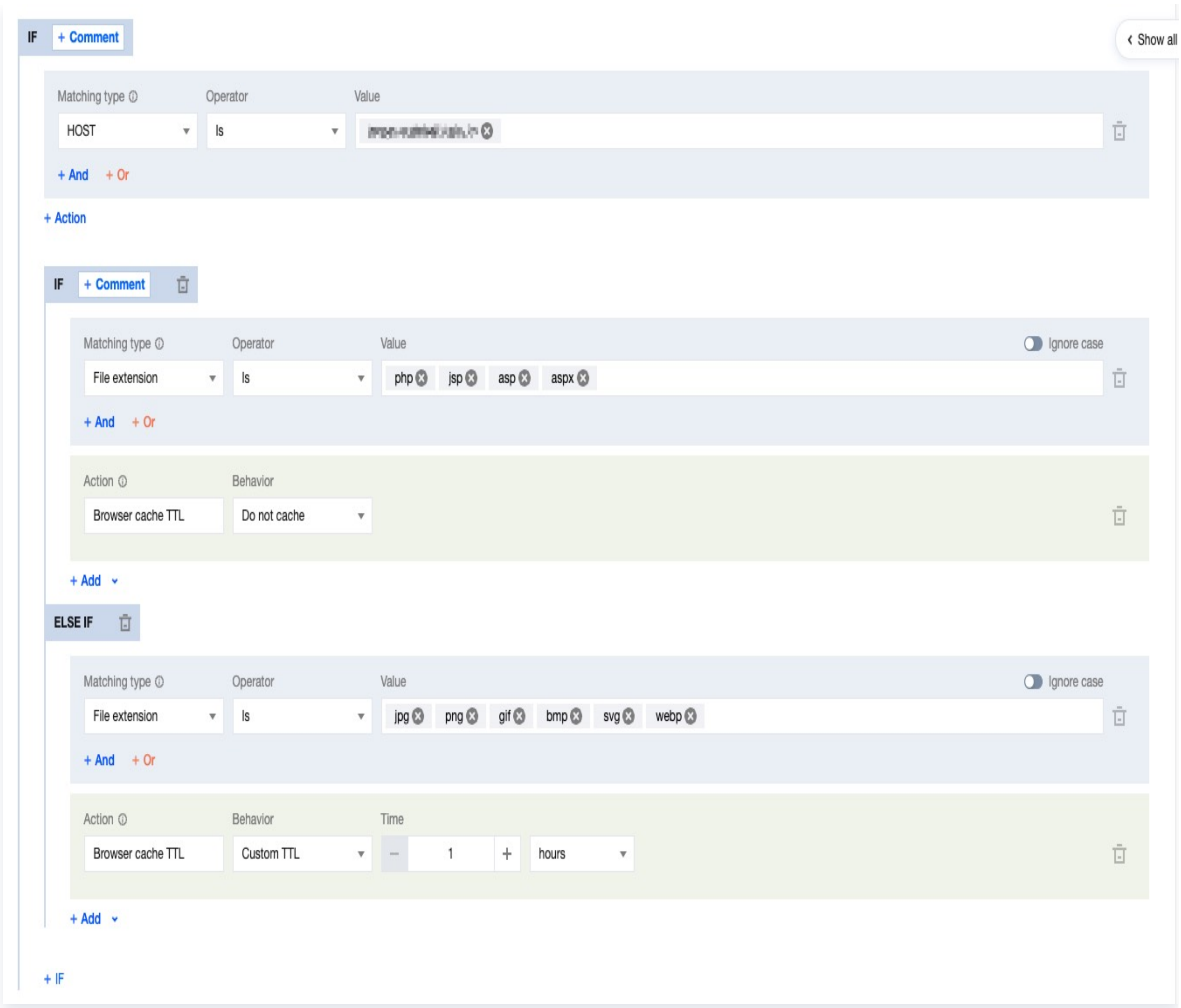
Scenario 2: Configuring browser cache TTL for specific domain names, paths, or file extensions request granularity.

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

1. Log in to the Edge Security Acceleration Platform Console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page.
4. On the rule editing page, first select the matching type as HOST, and the value as `www.example.com` as the outermost matching condition, then click **Add IF**.



5. Within the newly added IF condition, select the match type as file extension, add `php/jsp/asp/aspx` extensions, click **Action**, in the pop-up action list, select the action as **Browser Cache TTL,** and configure it as no-cache**.**
6. Repeat the above steps, add another IF condition, append `jpg/png/gif/bmp/svg/webp` suffixes, and configure it to cache for 1 hour.
7. After configuring the rules as shown below, click **Save and Publish** to complete the rule configuration.



IF + Comment

Matching type: HOST
Operator: Is
Value: [redacted]

+ 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 Cache

Last updated：2023-09-07 15:04:55

Overview

By default, if EdgeOne cannot establish a connection with the origin when fetching resources, it will respond with an error code. However, with offline caching enabled, EdgeOne can utilize the resources cached within its system (even if they have expired) when a connection with the origin cannot be established, until the origin connection is restored. This effectively ensures the availability and continuity of the service, thereby enhancing the user experience.

Note:
If no cache is available in EdgeOne, an error code will be returned.

Use Cases

- **Unstable Origin Server:** If your origin server is prone to failures or instability, enabling the offline caching feature can provide a better user experience during periods of origin server downtime. Even if the cached resources have expired, they can still continue to serve users, preventing situations where users cannot access the service during an origin server failure.
- **Key Business Assurance:** For certain critical operations, you may wish to ensure that users can still access key content on your website or application even when issues arise with the origin. Enabling offline caching allows users to access crucial resources even when the origin is experiencing difficulties, thereby ensuring business continuity.
- **Preventing Sudden Traffic Surges:** In certain scenarios, the origin server may be subjected to sudden traffic surges, leading to server overload or crashes. Enabling offline caching allows the service to continue providing for users during origin server failures, alleviating pressure on the origin server and aiding in its return to normal operation.

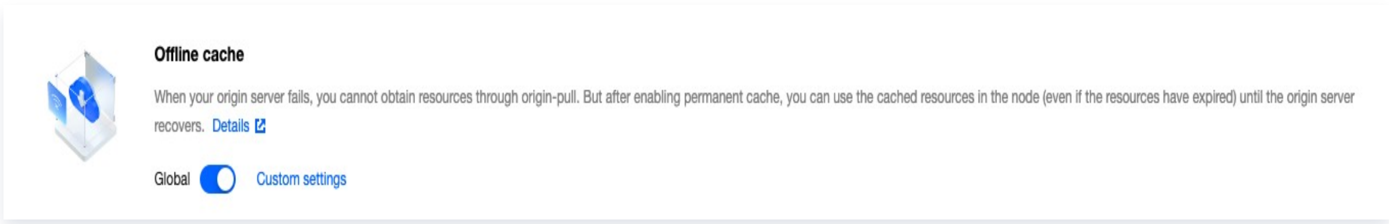
Instructions

Scenario 1: Configuring Offline Caching for All Domain Names of a Site

If you need to configure the enabling or disabling of offline caching for all domain names used to access a site, or as a fallback configuration at the site level, please refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List.** Within the site list, click on the **site** you wish to configure.
2. Within the site details page, click on **Site Acceleration** > **Cache Configuration**, locate the

offline cache card, and toggle the **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

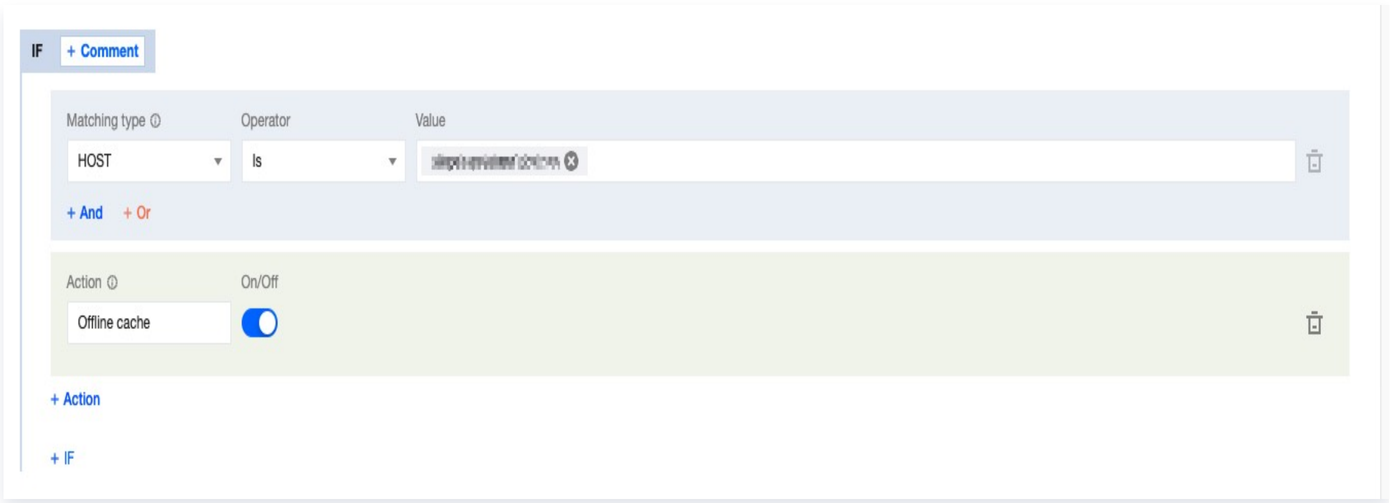
Global · Custom settings

- Default state: Enabled. If disabled, when the origin server fails and resources cannot be fetched normally, the node will transparently pass the origin server's response to the client request.

Scenario 2: Configuring Offline Caching for Specific Domain Names, Paths, or File Extensions at the Request Level

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

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page. On the rule editing page, select Host as the matching type and configure it as `www.example.com` .
4. Click on **Operation,** in the pop-up operation list, select **Offline Caching** and toggle it on.


IF + Comment

Matching type	Operator	Value
HOST	Is	[redacted]

+ And + Or

Action	On/Off
Offline cache	On

+ Action

+ IF

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

Cache prefresh

Last updated：2023-09-07 15:05:01

Overview

Upon expiration of cached resources within an EdgeOne node, EdgeOne will fetch the latest resource files from the origin server upon receiving corresponding client requests, which may lead to a significant increase in origin server traffic during peak times. The cache pre-refresh capability allows for verification of cache resource validity prior to cache expiration, eliminating the need to wait until after expiration for verification. This aids in maintaining resource timeliness and faster response to requests. The cache pre-refresh time can be configured as a percentage of the file cache TTL.

Use Cases

As the cache pre-refresh feature allows for early verification of resource validity from the origin server, it is recommended for use in scenarios where frequent content updates are required or where high user experience standards are demanded:

- **High Real-Time Requirements:** For rapidly updating content such as news or event pages, clients wish for users to access the most recent resources upon request. By enabling the cache pre-refresh feature, nodes verify and update the cache from the origin server prior to resource expiration, ensuring users access more recent resources, thereby eliminating additional wait time during user requests and enhancing user experience.
- **Reducing Origin Server Load:** For some hot resources, expiration may trigger a large number of requests to the origin server. By enabling the cache pre-refresh feature, these requests can be made in advance, reducing the concentration of a large number of requests to the origin server upon resource expiration, thereby alleviating the load on the origin server.

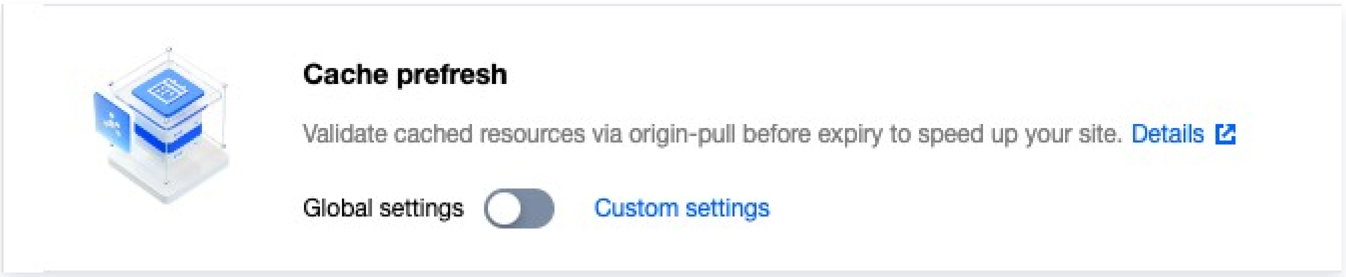
Instructions

Scenario 1: Configuring Cache Pre-Refresh for All Domain Names of a Site

To configure the same cache pre-refresh for all domain names used to access a site, or as a fallback configuration at the site level, refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the site that needs to be configured.
2. On the site details page, select **Site Acceleration** > **Cache Configuration**.
3. Within the cache pre-refresh card, click to enable **Global Site Settings**, and in the pop-up

confirmation box, enter the percentage value for the pre-refresh time.


Cache prefresh

Validate cached resources via origin-pull before expiry to speed up your site. Details

Global settings: Off

Custom settings

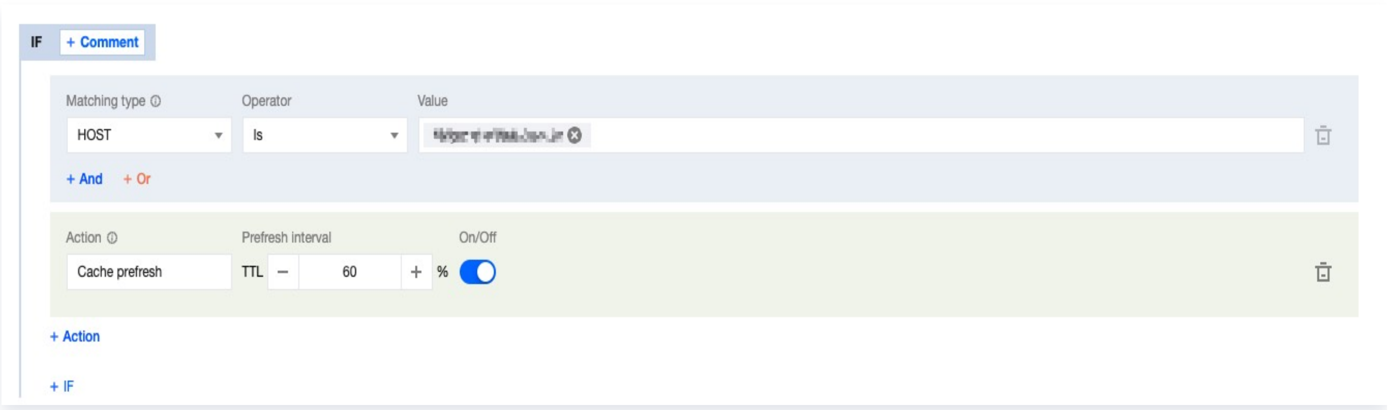
- **Configuration Status:** By default, it is disabled. You can enable it by clicking on the slider.
- **Prefresh Interval:** A percentage of the node cache TTL. Enter an integer from 1 to 99. Default value: 90%.

4. Click **Save** to deliver the configuration.

Scenario 2: Configuring Cache Pre-Refresh for Specified Domain Names, Paths, or File Extensions

If you need to configure different cache pre-refresh settings for different domain names, paths, or file extensions, for example, setting an earlier pre-refresh time - 60% for the domain name `www.example.com` under the `example.com` site, you can refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page. On the rule editing page, select Host as the matching type and configure it as `www.example.com`.
4. Click **Action**, in the pop-up operation list, select **Cache Prefresh** and set it to 60% of TTL.
5. The complete configuration is as shown. Click **Save and Publish** to finalize the rule configuration.


IF + Comment

Matching type	Operator	Value
HOST	Is	[illegible]

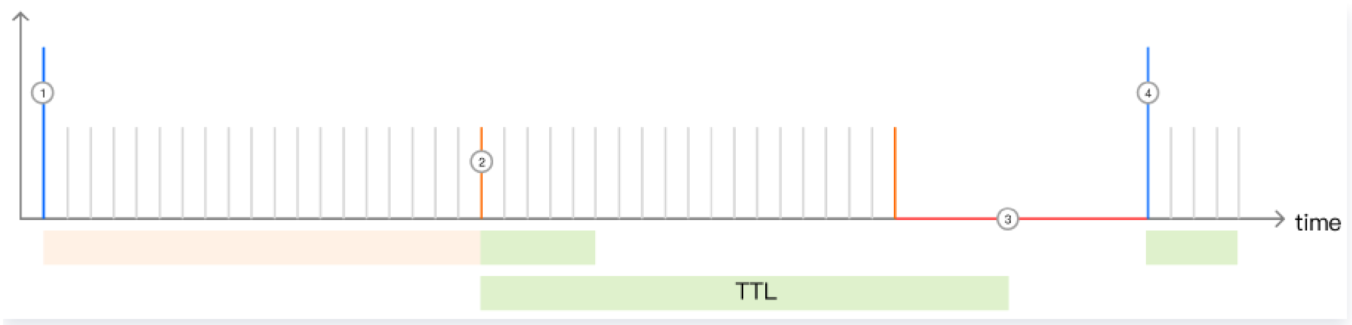
+ And + Or

Action	Prefresh interval	On/Off
Cache prefresh	TTL 60 %	On

+ Action

+ IF

Appendix: Functional Principle



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

1. Upon initial receipt of a client request, if the current node has not cached the file, it will fetch the resource from the origin server and cache it within the node, with a cache TTL of 10 seconds. If another client request is received within 0-7 seconds, the node will directly provide the resource from the cache, responding normally to the client request;
2. When the cached `test.jpg` within the node reaches the pre-refresh time, between the 8th and 10th second, if a client request is received, the node will continue to respond to the client request as usual, but will also asynchronously verify the validity of the cached resource from the origin server.
    - If the resource is valid, the cache TTL of the resource on the node is updated and reset to 10 seconds;
    - Should the resource become invalid, the latest valid resource will be retrieved from the origin server to the node, and the node cache TTL will be reset to 10 seconds;
3. If there are no client requests beyond the file's node cache TTL, the resource will expire on the node after the cache TTL time has elapsed.
4. Upon receiving the next client request, the node will initiate a request to the origin server to verify the validity of the resource. If a file update is detected, the latest file will be fetched; otherwise, the file will be re-cached and the cache refresh time will be reset to 10 seconds.

Clear and Preheat Cach

Cache Purge

Last updated：2023-09-07 15:05:08

Overview

Once your resource content is cached on EdgeOne edge nodes, during the cache validity period, users accessing these resources will be directly responded to by the EdgeOne edge nodes, without triggering a fetch from the origin. If your origin server updates the resource content during this time, to prevent users from still accessing the old resource files, you can manually clear all cached resources within all edge nodes by clearing the cache. After the cache is cleared, when users access the resources, EdgeOne will fetch the latest resources from the origin for response.

Quota Specifications

Different billing packages have different quotas. For more details, see Package Selection Comparison.

Scenarios

You might find this feature useful in the following scenarios:

- **Content Update**: When you have updated some resources on the origin server, but the cache on EdgeOne has not yet expired, and you want the client users to see the latest content immediately.
- **Error Correction:** If some erroneous content from your origin server has been cached on EdgeOne, to avoid business risks, you need to immediately clear these erroneous caches to ensure that EdgeOne no longer provides incorrect content.
- **Testing and Debugging:** During the development and debugging of a website, you may need to frequently modify and test the site content. To ensure that you are viewing the latest updated content rather than a cached old version, you can utilize the cache clearing feature.
- **Emergency Response:** In certain urgent situations, such as being under attack or having sensitive information published, you need to immediately remove the related content from EdgeOne.
- **Cache Policy Adjustment:** When you adjust or optimize the EdgeOne cache policy, you may need to clear the existing cache to ensure the new policy takes effect immediately.

Supported Types and Methods

EdgeOne's cache purging supports clearing various types of resources, as detailed below:

Supported Type	Details
URL Note 1	Matching URL node cache resources, such as `https://www.example.com/path/foo.jpg` .
Directory Note 1	Matching node cache resources in a directory, such as `https://www.example.com/path/` .
Hostname Note 1	Matching node cache resources with the Hostname, for example, `www.example.com` . Submission of URLs in the format of `*.test.com` is not supported, meaning the domain name cannot contain wildcards and the corresponding subdomain must be specified.
Cache-Tag Note 1	Clear the cache by matching the tag values (tags) in the `Cache-Tag` response header of the HTTP response packet, for example `Cache-Tag: tag1,tag2,tag3` . • Applicable exclusively to the Enterprise plan. • EdgeOne supports the recognition of the origin server response header `Cache-Tag` . Please add tag(s) to this header: ○ The maximum header size is 6 KB. ○ Multiple tags are separated by commas, with each tag not exceeding 128 characters. The maximum number of tags is 1,000. ○ Tags are case-insensitive, meaning that Tag1 and tag1 will be recognized as the same tag.
All Cache Note 1	Caching all resources of a site on nodes. Note 2

Note

Note 1: Cache clearing is divided into two methods: **Direct Deletion** and **Marking as Expired**:

- The default URL type is "Direct Deletion", which means the cache content is directly deleted. When a user requests a resource, EdgeOne will immediately fetch the latest resource from the origin, which increases the number of origin fetch requests in a short time and weakens the acceleration effect. If a large amount of content is submitted, it can put significant pressure on the origin server.
- For all purge types other than URL, the default action is "Mark as Expired", which means the cache is not directly deleted, but marked as expired. The next

time a request is made, it will carry `If-None-Match` and `If-Modified-Since` Note 3 to first check with the origin server for resource updates:

i. In case of no updates - If the `ETag` (if any) remains unchanged and the origin server returns 304 (Not Modified), the node cache is retained, effectively saving back-to-origin bandwidth;
ii. If updated - If the `ETag` (if any) changes and the origin server returns 200 (OK), the latest resource is fetched from the origin. The cached resource and the new resource are compared based on the `Last-Modified` and file size. If either parameter differs, the new resource replaces the expired cache on the node.

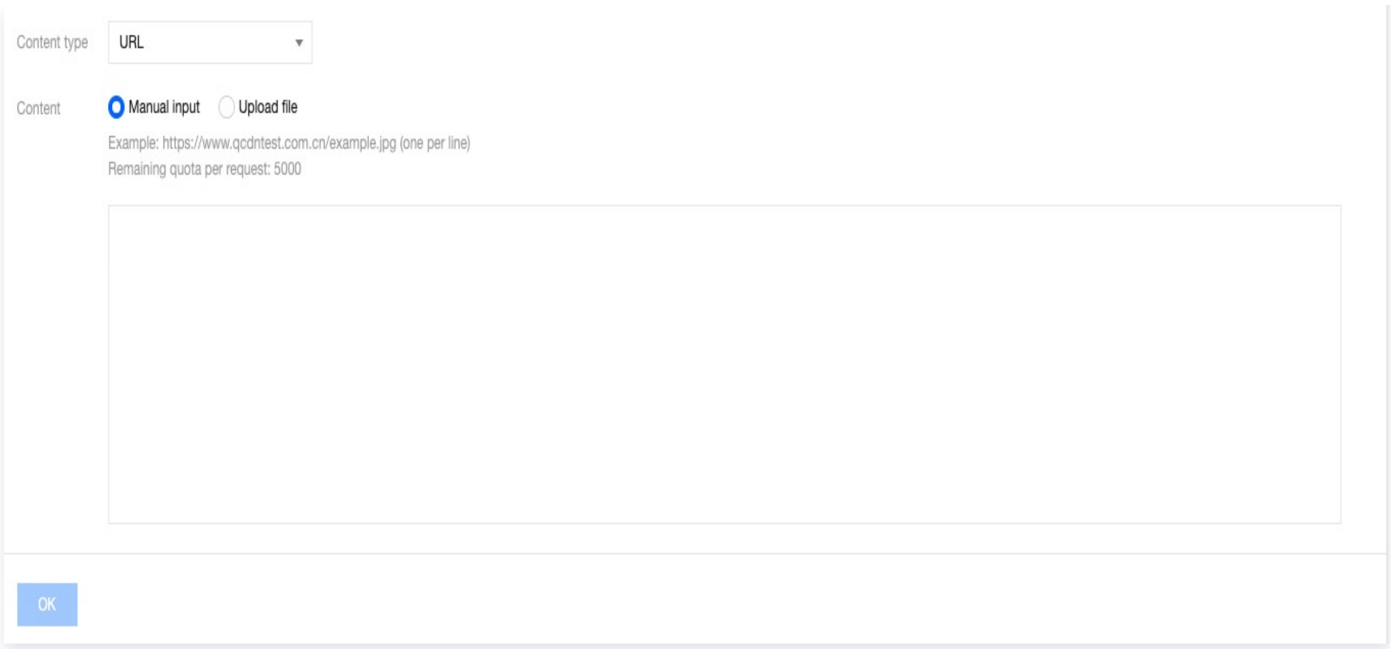
Note 2: If a wildcard domain, such as `*.foo.example.com` , is added under the current site (`example.com`), the cache cannot be cleared for all sub-domains under this wildcard domain. You need to submit separate cache clearing tasks for each specific sub-domain.
Note 3: When `If-None-Match` and `If-Modified-Since` are used together, `If-None-Match` has a higher priority, that is, it first checks whether the `ETag` has changed.

Instructions

Scenario 1: Clearing the Cache by Inputting Content

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

1. Log in to the Edge Security Acceleration Platform Console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, select **Site Acceleration** > **Cache Purging**.
3. On the Clear Cache page, select the type of resource to be cleared, enter the corresponding resource content, and click **Confirm Clear**.


Content type URL

Content [✓ Manual input] Upload file

Example: https://www.qcdntest.com.cn/example.jpg (one per line)
Remaining quota per request: 5000

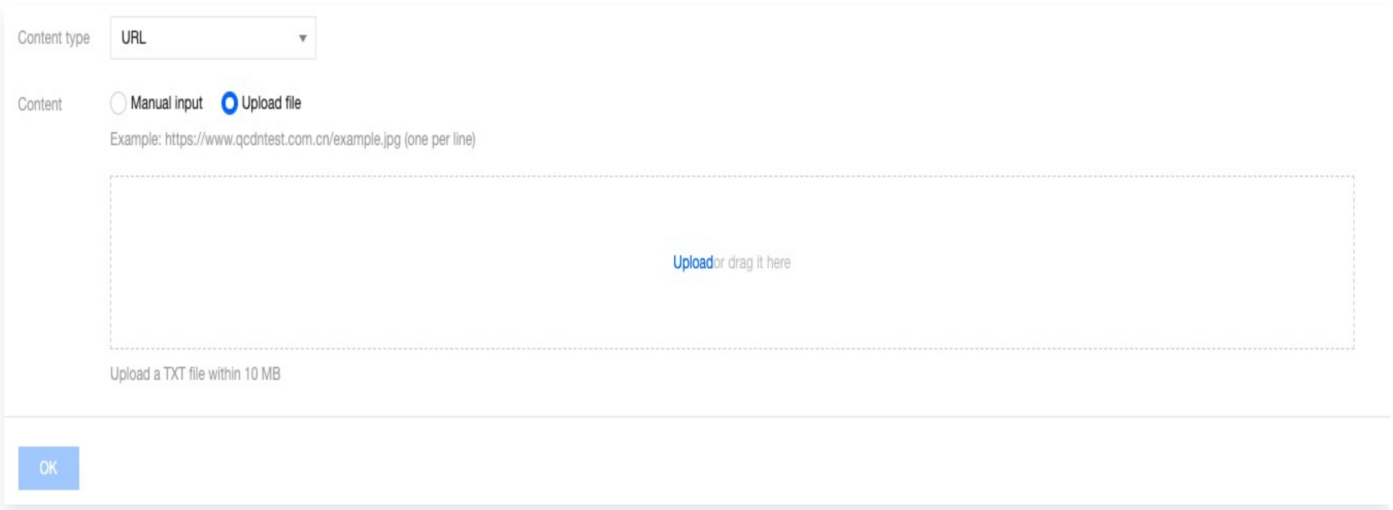
OK

4. Switch to the History tab to view the history records for a specified period (within the last month) and by purge type.

Scenario 2: Bulk Importing Content for Cache Purging via File Upload

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

1. Log in to the Edge Security Acceleration Platform Console . In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, select **Site Acceleration** > **Cache Purging**.
3. On the Clear Cache page, select the type of resource to be cleared, choose the "Upload File" method, and after uploading, click **Confirm Clear**.


Content type URL

Content Manual input [✓ Upload file]

Example: https://www.qcdntest.com.cn/example.jpg (one per line)

Uploador drag it here

Upload a TXT file within 10 MB

OK

4. Switch to the History tab to view the history records for a specified period (within the last month) and by purge type.

References

- How long does it take for cache purge and cache prefetch to take effect after each submission?

URL Pre-Warming

Last updated：2023-09-07 15:05:14

Overview

When new resources are released, the client's initial request for these resources may encounter a situation where there is no cache on EdgeOne, resulting in an inability to respond immediately and necessitating a return to the origin for retrieval. The cache pre-warming feature allows resources to be cached on EdgeOne in advance. Thus, even if the client requests for the first time, it can respond directly from the cache on EdgeOne without having to return to 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 enhancing the acceleration effect and alleviating the pressure on the origin.

Quota Specifications

Different billing packages have different quotas. For more details, see Package Selection Comparison.

Use Cases

You might find this feature useful 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 at the earliest possible time, reducing the delay when first accessed. For example, before a game business officially releases a new version of the installation package or upgrade package, the installation package resources can be pre-warmed to EdgeOne. After the official release, when users request to download these installation packages, they can directly obtain the installation package resources from the node, thereby improving the download speed.
- **Large-scale Event Operations:** Prior to a large-scale event, you would want to ensure that the key resources for the event are already cached on EdgeOne. This helps to ensure that when the event begins, client users can quickly access the required content, reducing delays and congestion caused by high traffic.
- **Anticipated Traffic Peaks:** If you anticipate a significant increase in website traffic during a specific period (for example, holiday promotions, news releases, etc.), you can use the cache pre-warming feature to ensure that key resources are already cached on the edge nodes. This helps to disperse the pressure of back-to-origin requests during peak periods and improve the access speed of client users.

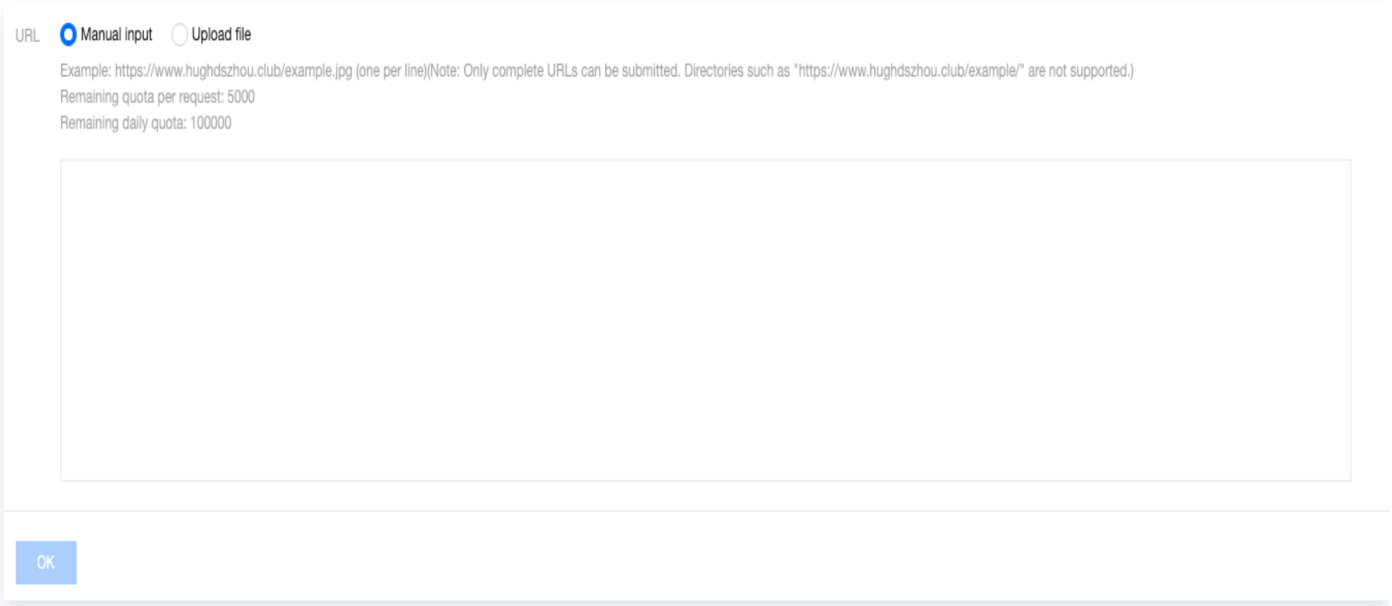
Note
- During resource pre-warming, a request will be simulated to pull the corresponding resource from the origin server. If there are many submitted pre-warming tasks, a large number of origin-pull requests will be generated, increasing the origin server bandwidth usage.
- If the resources to be pre-warmed conflict with the node cache, i.e., if EdgeOne has already cached a resource with the same name and it has not yet expired, it will still be valid and will not be overwritten by the pre-warmed resource. If the resource with the same name has changed, you can clear the corresponding node cache before pre-warming.

Instructions

Scenario 1: Pre-warming the cache by inputting content

If you have a small amount of content to pre-warm and it's convenient to enter it directly into the input box, you can follow these steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, select **Site Acceleration** > **Preheat Cache**.
3. On the Cache Prefetching page, input the corresponding resource content and click **Confirm Prefetch**.



URL [✓ Manual input] Upload file

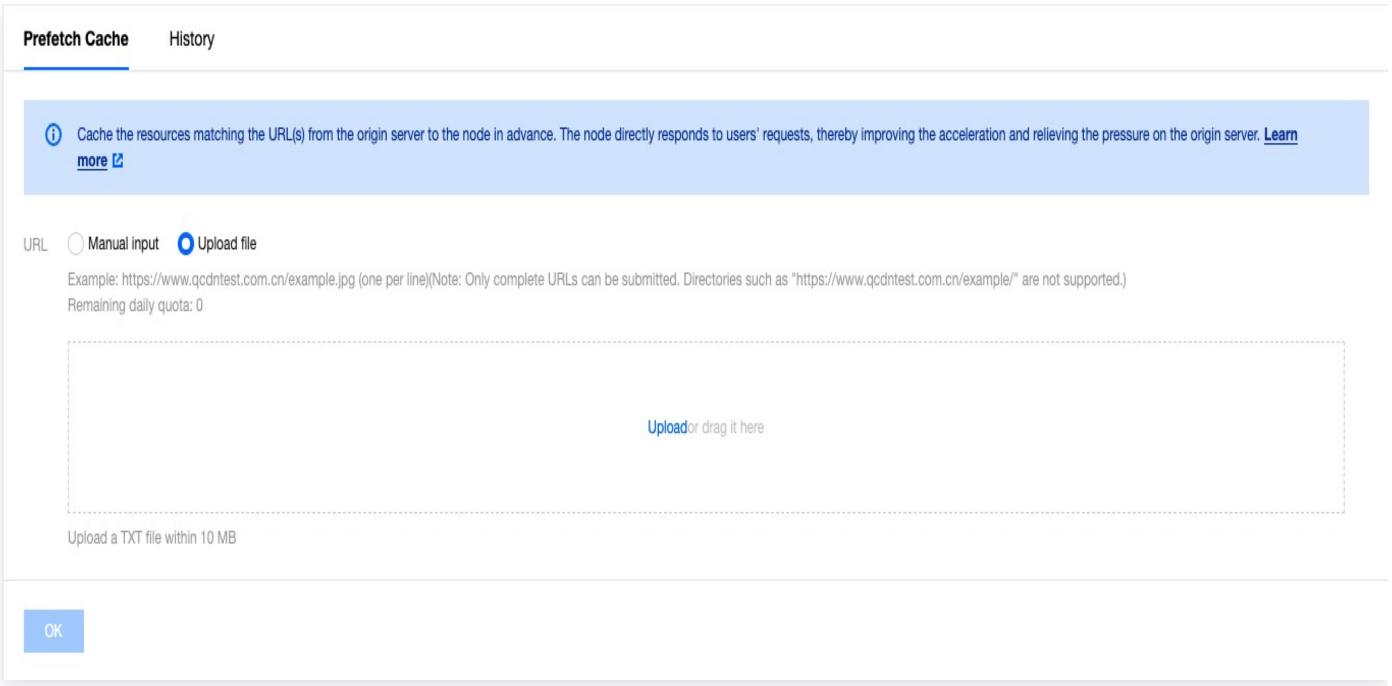
Example: https://www.hughdszhou.club/example.jpg (one per line)(Note: Only complete URLs can be submitted. Directories such as "https://www.hughdszhou.club/example/" are not supported.)
Remaining quota per request: 5000
Remaining daily quota: 100000

4. Switch to the **History** tab to view the historical records for a specified period (within the past month).

Scenario 2: Batch Importing Pre-warming Cache Content via File Upload

If you have a large amount of content to pre-warm or have placed the content in a file, you can choose to upload the file:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, select **Site Acceleration** > **Preheat Cache**.
3. On the Cache Prefetching page, select the "Upload File" method, and then click **Confirm Prefetch**.



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

URL Manual input [✓ Upload file]

Example: https://www.qcdntest.com.cn/example.jpg (one per line)(Note: Only complete URLs can be submitted. Directories such as "https://www.qcdntest.com.cn/example/" are not supported.)
Remaining daily quota: 0

Upload or drag it here

Upload a TXT file within 10 MB

4. Switch to the History tab to view the historical records for a specified period (within the past month).

References

- How long does it take for cache purge and cache prefetch to take effect after each submission?

How to improve the Cache Hit Rate of EdgeOne

Last updated：2023-09-07 15:05:23

This document outlines how to effectively utilize the various configurations on Tencent Cloud EdgeOne, in conjunction with your actual business scenarios, to optimize performance and enhance the cache hit rate of files within your site.

Background

Once your site is connected to Tencent Cloud EdgeOne and acceleration is enabled, static resources such as images and videos accessed by users will be cached in the edge nodes. When other users initiate the same requests, these requests will be directly responded to by the edge nodes, thereby avoiding the need for back-to-source requests.

If the cache hit rate is too low, it can result in a large number of back-to-source requests from users, placing significant processing pressure on the origin server and diminishing the user experience and site acceleration effect. You can optimize and improve the cache hit rate through the following configurations.

Note:
If you need to view the current cache hit analysis, you can do so on the console by navigating to **Data Analysis** > **Cache Analysis**. For more details, please refer to Cache Analysis.

Optimization Methods

Adjusting Node Cache TTL Configuration

The configuration of the node cache TTL directly affects whether EdgeOne caches the specified file resources and the corresponding cache time. If the file cache policy is set to no-cache or the cache time on the node is short, it can result in users accessing non-cached resources, leading to frequent back-to-source requests and a diminished user experience. By default, EdgeOne enables the default cache rules for the entire site. You can understand how EdgeOne's cache rules take effect by viewing the EdgeOne Content Cache Rules. To improve the cache hit rate, it is recommended that you configure cache rules individually based on file extensions within the rule engine.

Recommended Configuration

It is recommended that you configure personalized cache rules according to different file types in different scenarios:

1. For infrequently updated files, such as download resources and video files, it is recommended to configure a custom cache time on EdgeOne, set to more than 30 days, and enforce caching through EdgeOne nodes. Common download resource and video file formats are as follows:

File Type	File format
Audio/Video	mp4;mp3;ts;m4a;avi;m4s;ogg;mkv;mov;flv;rm;rmvb;swf;wav;wmv;rmi;aac
Packages	rar;7z;zip;gzip;dmg;gz;ios;tar;jar;br;bz2
Documents	doc;docx;xls;xlsx;pdf;ppt;pptx
Applications	apk;exe;bin
Others	vsv;iso;jar;swf;chunk;atlas

2. For frequently updated file content, such as image content, if the cache time is too long, users may access outdated content due to cache hits. Therefore, it is recommended to configure a custom cache time separately on EdgeOne. The cache time can generally be configured according to business needs, ranging from 1 to 7 days. Common image content formats are as follows:

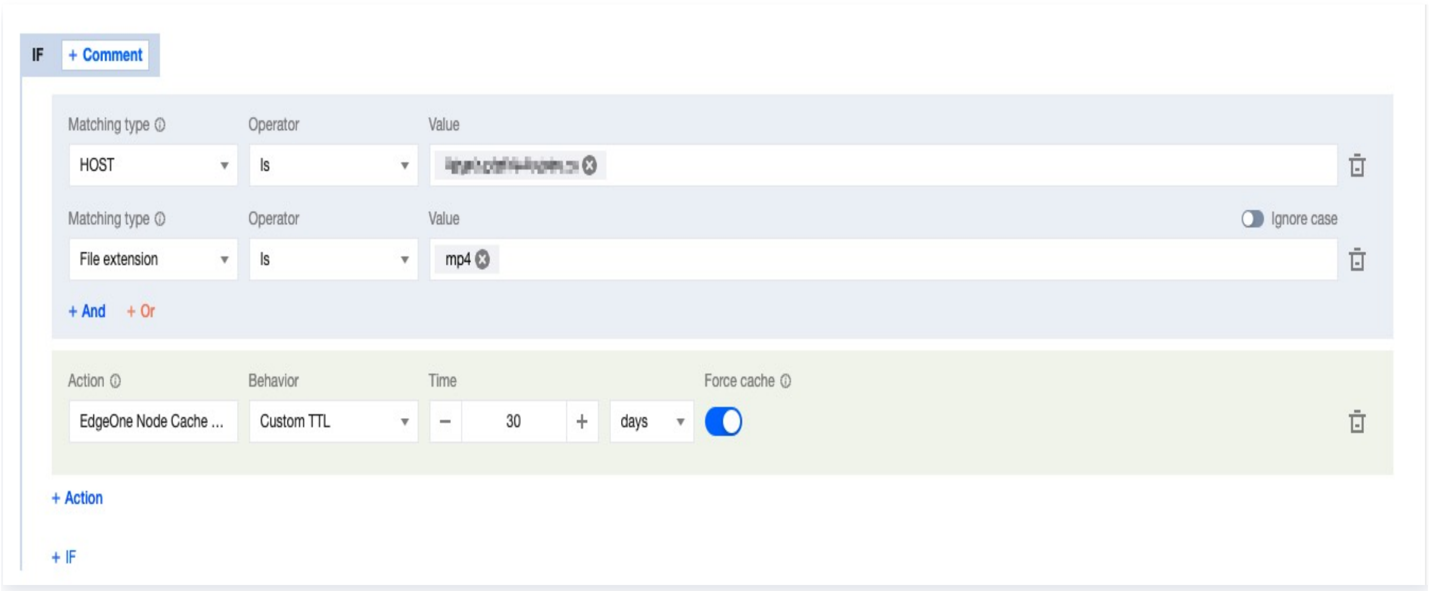
File Type	File format
Image	jpg;png;jpeg;webp;gif;heif;heic;kpg;ico
Webpages	html;htm;shtml;hml;js

3. Dynamic files, such as php, json files, etc., if cached, can result in incorrect content responses when accessed by users. Therefore, it is recommended to configure them separately on EdgeOne to prevent caching. Common dynamic file formats include:

File Type	File format
Dynamic Resources	php;aspx;asp;jsp;do;dwr;cgi;fcgi;action;ashx;axd;json

Optimization Examples

When you observe a low hit rate for resources 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 instance, in the current cache distribution, there are many misses for `.mp4` format files.
If you strictly adhere to the default caching rules of Tencent Cloud EdgeOne, the issue that arises is that the Cache-Control header is not responded to when responding to files. According to the default caching rules, the cache time for `.mp4` files on the node is 2 hours. Due to the short cache time, this file will frequently go back to the source. If you need to cache this file, you can go to the rule engine, add a new rule, and set the node cache TTL to cache for 30 days when the file extension is mp4, and enable force cache, i.e., ignore the CC header responded by the source station, and the node will forcibly cache this file. For detailed operations, please refer to Node Cache TTL.



IF + Comment

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

Ignore case

+ And + Or

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

+ Action

+ IF

A custom Cache Key directs the same type of request to a single cached file

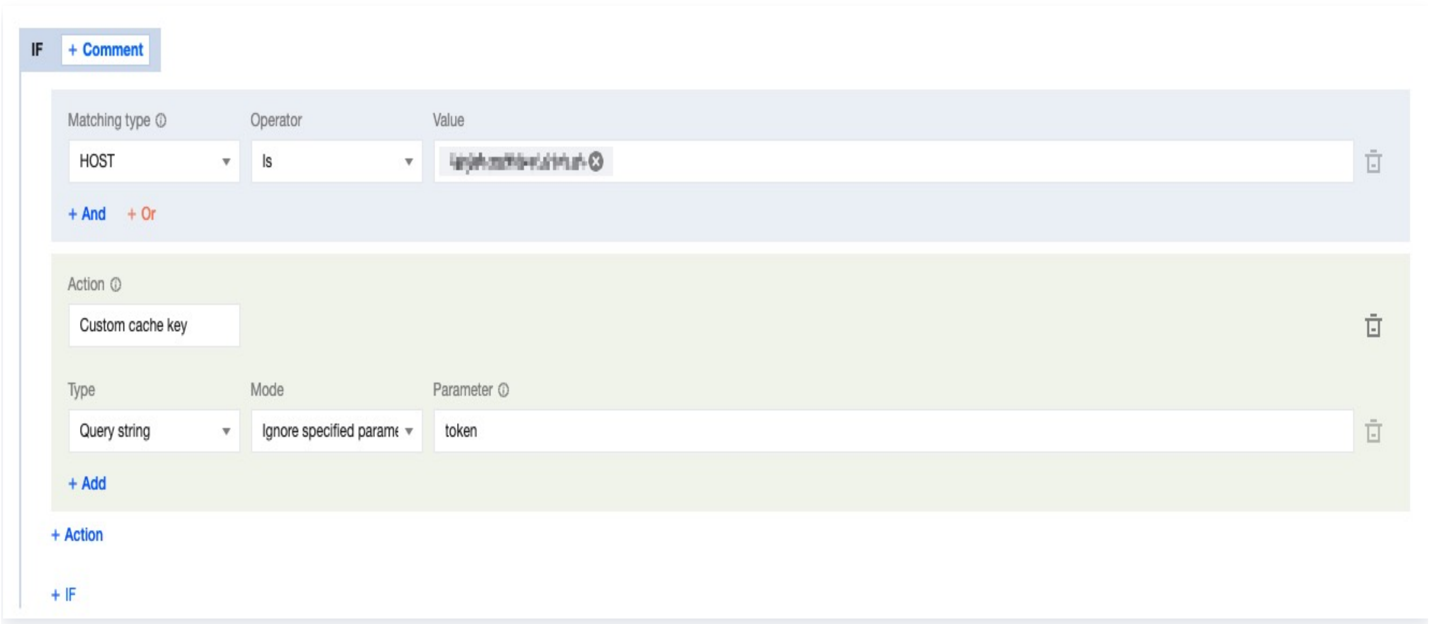
By default, EdgeOne generates a unique identifier for the cache key based on the user's accessed URL and query string, serving as the Cache Key for the file. When a similar request is made, the edge node determines whether the cache is hit by comparing whether the request matches the Cache Key in the cache. If the URL carries dynamic parameter content that does not affect the file version, such as a user ID, multiple caches will be established based on the different parameters, leading to a decrease in cache hit rate. This can be optimized by customizing the Cache Key.

Recommended Configuration

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

Optimization Examples

The current request URL is: `https://image.example.com/test.jpg?version=1.1&token=1234567890`. The parameter `version=1.1` affects the content of the image, while `token=1234567890` does not. To enhance the cache hit rate, you can ignore the token parameter in the custom Cache Key. For detailed operations, please refer to Customizing Cache Key.



Matching type: HOST
Operator: Is
Value: [redacted]

Action: Custom cache key

Type: Query string
Mode: Ignore specified parame
Parameter: token

Cache Prefetching

Cache prefetching allows EdgeOne to cache files in edge nodes in advance. When a user accesses these files, they can directly hit the cache, reducing the number of initial concurrent back-to-source requests and improving the cache hit rate. When you add a new site to EdgeOne or release new popular resources, it is recommended to prefetch the cache in advance. For detailed operations, please refer to Cache Prefetching.

Enable Cache Prefresh

When your files are primarily hot files and you need to ensure that these files are continuously cached within the node, you can enable cache pre-refresh. This means that before the node file cache expires, when a user requests a file within the node, the system will verify with the origin server whether the file has been updated. If it has not been updated, the cache time of the file within the node will be refreshed. For detailed operations, please refer to Cache Pre-Refresh.

Effective Utilization of the Vary Mechanism

When the origin server responds with a Vary header, the CDN will differentiate the cache based on the content specified by the Vary header. For a detailed explanation of the Vary feature, please refer to Vary Characteristics. If the current file does not require the Vary header to control its cache version, it is recommended to avoid responding with this header when the origin server responds, in order to reduce the number of cache versions created and improve the cache hit rate.

Learn more

- How to Determine if User Requests Have Hit the EdgeOne Node Cache

File Optimization
Smart Compression

Last updated：2023-09-07 15:05:33

Overview

EdgeOne enables Gzip or Brotli compression by default globally. When the client request header carries `Accept-Encoding: br, gzip` , `Accept-Encoding: br` , or `Accept-Encoding: gzip` , the node will intelligently compress based on the file's `Content-Type` . This compression effectively reduces the size of the resources and accelerates the transmission speed of the content.
Enabling intelligent compression can assist you in:

1. **Enhance User Experience**: By reducing resource size, the loading speed of web pages can be significantly improved, thereby providing a superior user experience. Particularly for websites with a wealth of CSS, JavaScript, and other resources, enabling compression can drastically cut down on loading times.
2. **Conserve Bandwidth**: Compressed resources will consume less network traffic, which can help reduce operational costs.

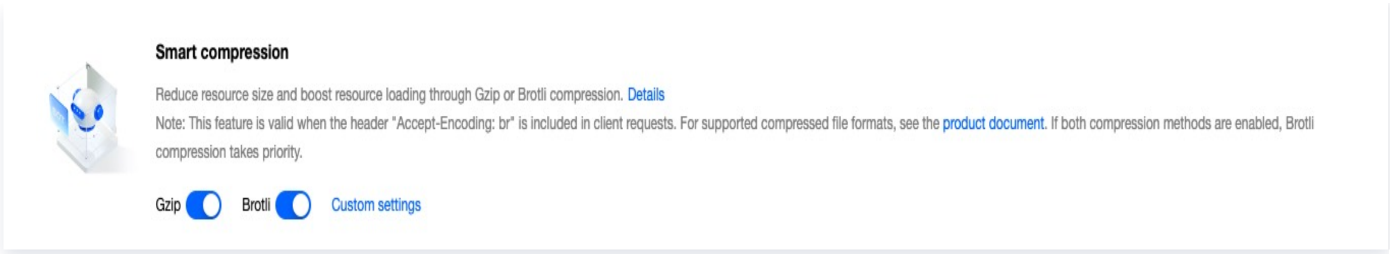
Instructions

Note:
By default, there is no need to modify this configuration. However, in certain scenarios, such as when the current client performs an MD5 check on the file or the current client cannot parse the specified compressed file, and you wish for the current site to use only Brotli compression, Gzip compression, or no compression at all, you can follow the steps below.

Scenario 1: Enabling/Disabling Smart Compression for All Domain Names of the Site

To enable or disable smart compression for all domain names used to access the current site, refer to the following steps:

1. Log in to the EdgeOne console . In the left sidebar, click **Site List**. Within the site list, click the **site** that needs to be configured to enter the site details page.
2. On the site details page, choose **Site Acceleration** > **File Optimization** to go to the Network Optimization details page.
3. Locate the Smart Compression configuration card. By default, all options are enabled. Click the **toggle** to configure whether to enable or disable this feature.



Smart compression

Reduce resource size and boost resource loading through Gzip or Brotli compression. Details

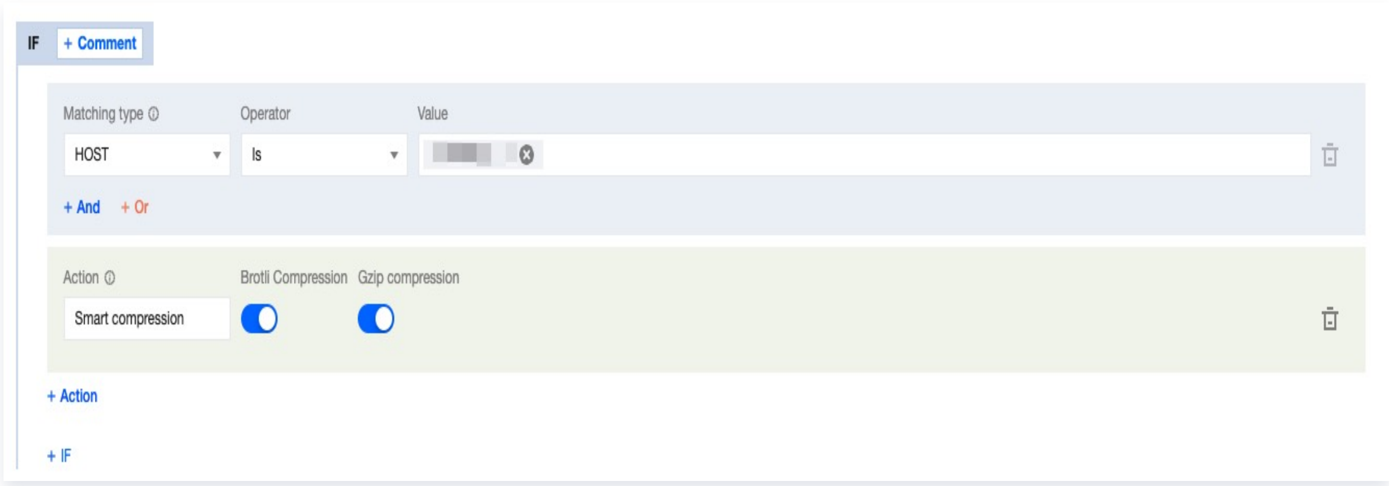
Note: This feature is valid when the header "Accept-Encoding: br" is included in client requests. For supported compressed file formats, see the product document. If both compression methods are enabled, Brotli compression takes priority.

Gzip ● Brotli ● Custom settings

Scenario 2: Enabling/Disabling Smart Compression for Specified Domain Names

To enable or disable smart compression for specified domain names, refer to the following steps:

1. Log in to the Edge Security Acceleration Platform Console . In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page.
4. On the page that appears, select **HOST** from **Matching type** and specify an operator and a value to match the requests of specified domain names.
5. Click on **Operation** > **Selection Box**. In the pop-up operation list, select **Smart Compression**. Click on the **Switch** to enable or disable Gzip or Brotli compression.



IF + Comment

Matching type	Operator	Value
HOST	Is	[redacted]

+ And + Or

Action	Brotli Compression	Gzip compression
Smart compression	●	●

+ Action

+ IF

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

References

Smart Compression Activation Rules

1. File size: 256 B - 30 MB
2. Smart Compression is a synchronous compression process that compresses files while retrieving them from the origin. When a compressed file is requested for the first time, the

node can directly respond with the compressed file.

3. By default, Smart Compression compresses based on `Content-Type` and supports the following types:

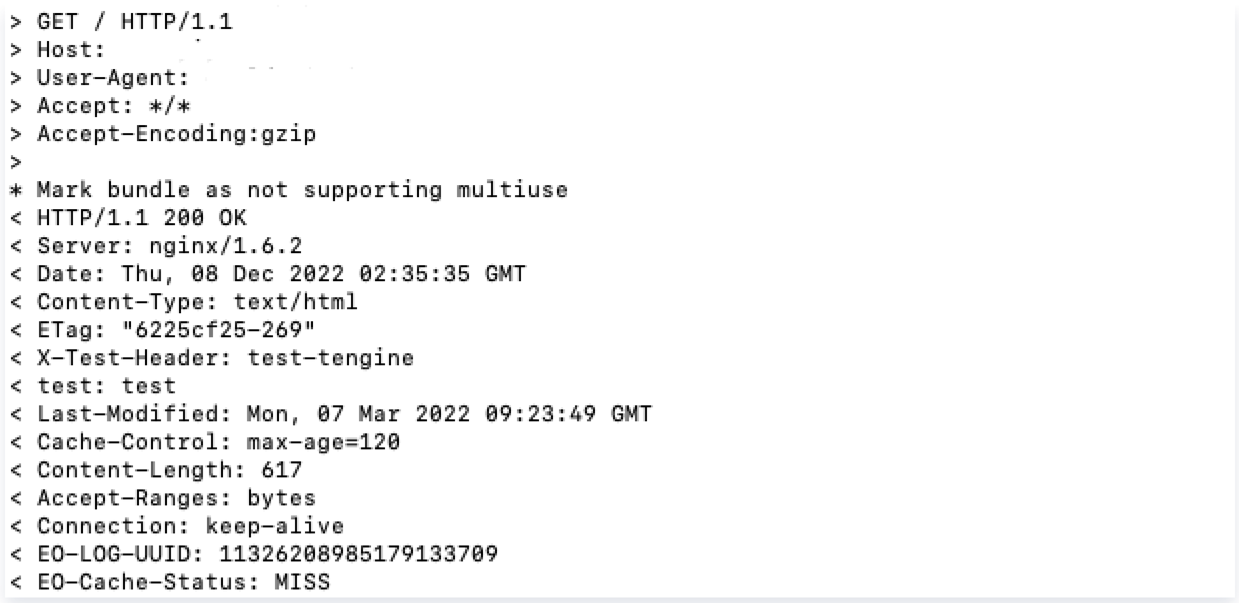
```
text/html
text/xml
text/plain
text/css
text/javascript
application/json
application/javascript
application/x-javascript
application/rss+xml
application/xmltext
image/svg+xml
image/tiff
text/richtext
text/x-script
text/x-component
text/x-java-source
text/x-markdown
text/js
image/x-icon
image/vnd.microsoft.icon
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, and the client request header `Accept-Encoding` carries both br and gzip:
    - If there is cached content within the node, it will respond according to the following rules:
        - If a node has cached content that is both Brotli and gzip compressed, it will prioritize responding with Brotli compression.
        - If a node only has cached content that is Brotli compressed, it will prioritize responding with Brotli compression.
        - If the node has cached resources compressed only in Gzip, Gzip compressed resources are returned first.
    - If there is no cached content within the node, it will prioritize responding with Brotli compression.
5. If only Brotli compression or Gzip compression is enabled and the request header carries `gzip` or `br` , the compression will not take effect, and the original resource will be returned.
6. If the origin server has enabled the compression feature and the server carries the response header: `Content-Encoding` , the intelligent compression feature will no longer be effective.

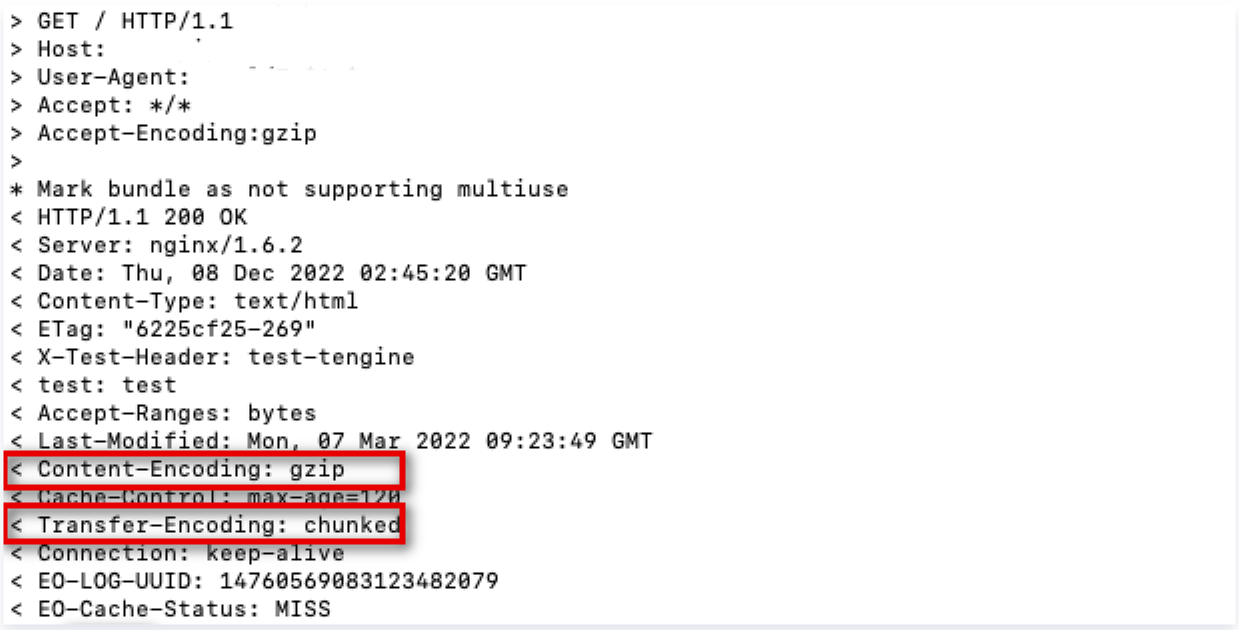
Sample Request

- Without Smart Compression enabled:
  On the first request for a gzip compressed file, if the node cache is not hit, the original file is fetched from the source and cached on the node. EdgeOne responds with the original file.



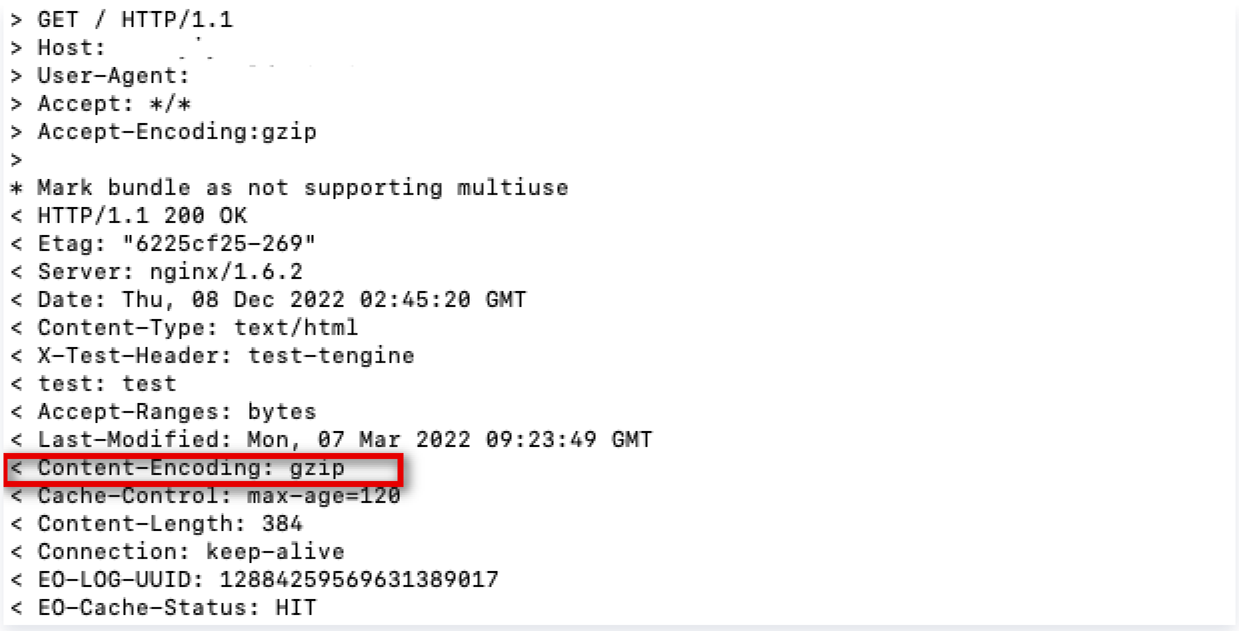
```
> GET / HTTP/1.1
> Host:
> User-Agent:
> Accept: */*
> Accept-Encoding:gzip
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.6.2
< Date: Thu, 08 Dec 2022 02:35:35 GMT
< Content-Type: text/html
< ETag: "6225cf25-269"
< X-Test-Header: test-tengine
< test: test
< Last-Modified: Mon, 07 Mar 2022 09:23:49 GMT
< Cache-Control: max-age=120
< Content-Length: 617
< Accept-Ranges: bytes
< Connection: keep-alive
< EO-LOG-UUID: 11326208985179133709
< EO-Cache-Status: MISS
```

- Enable Smart Compression
  - Upon the first request for a gzip compressed file, if the node cache is not hit, the file is retrieved from the source, compressed and cached by the node, and the compressed file is responded by EdgeOne:
    Smart Compression supports chunk streaming compression. If the request does not hit the node cache, the file will be responded in chunks after being retrieved from the source.



```
> GET / HTTP/1.1
> Host:
> User-Agent:
> Accept: */*
> Accept-Encoding:gzip
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.6.2
< Date: Thu, 08 Dec 2022 02:45:20 GMT
< Content-Type: text/html
< ETag: "6225cf25-269"
< X-Test-Header: test-tengine
< test: test
< Accept-Ranges: bytes
< Last-Modified: Mon, 07 Mar 2022 09:23:49 GMT
< Content-Encoding: gzip
< Cache-Control: max-age=120
< Transfer-Encoding: chunked
< Connection: keep-alive
< EO-LOG-UUID: 14760569083123482079
< EO-Cache-Status: MISS
```

  - Upon subsequent requests, if the Gzip compressed file is cached on the node, the node will directly respond with the compressed file.



```
> GET / HTTP/1.1
> Host:
> User-Agent:
> Accept: */*
> Accept-Encoding:gzip
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Etag: "6225cf25-269"
< Server: nginx/1.6.2
< Date: Thu, 08 Dec 2022 02:45:20 GMT
< Content-Type: text/html
< X-Test-Header: test-tengine
< test: test
< Accept-Ranges: bytes
< Last-Modified: Mon, 07 Mar 2022 09:23:49 GMT
< Content-Encoding: gzip
< Cache-Control: max-age=120
< Content-Length: 384
< Connection: keep-alive
< EO-LOG-UUID: 12884259569631389017
< EO-Cache-Status: HIT
```

Media Processing
Image Processing

Last updated：2023-09-07 16:15:21

Overview

The image processing feature allows you to adjust the size and format of images as needed. By directly processing, caching, and responding to images through EdgeOne edge servers, your business origin only needs to store the original images, thereby reducing image management costs. EdgeOne edge servers compress images without affecting visual perception to improve page loading speed and optimize image acceleration performance, enhancing user experience while maintaining image quality.

Supported Image Processing Capabilities

Resize

Feature	Parameter	Value (type/pixel)	Note
Resize	eo-img.resize	`w/<Width>` . For example: `w/100`	Changes the width to the specified value. The height is automatically adjusted based on the aspect ratio.
		`h/<Height>` . For example: `h/100`	Changes the height to the specified value. The width is automatically adjusted based on the aspect ratio.
		`w/<Width>/h/<Height>` . For example: `w/100/h/100`	Specify both the width and height
		`l/<Long>` . For example: `l/100`	Changes the long

			side of an image to the specified value. The short side is automatically adjusted based on the aspect ratio.
		`s/<Short>` . For example: `s/100`	Changes the short side of an image to the specified value. The long side is automatically adjusted based on the aspect ratio.

Format Conversion

Supports converting the original image into a specified format by carrying designated parameters.

Feature	Parameter	Supported Input Formats	Supported Output Formats
Format Conversion	eo-img.format	Static images: JPG, PNG, BMP, JP2, JXR, GIF, WebP, AVIF, HEIF	Supported static formats: JPG, PNG, BMP, JP2, JXR, GIF, HEIF, WebP, AVIF
		Animated Images: GIF, WebP, AVIF, HEIF	Static: JPG, PNG, BMP, JP2, JXR (using the first frame of a GIF animation as a single static image) Dynamic: GIF, WebP, AVIF, HEIF

Description

- The original image to be processed should not exceed 32 MB.
- The width and height of the input original image should not exceed 30,000 pixels, and the total pixels should not exceed 250 million pixels. For animated images, the product of the original image's width, height, and frame number should not exceed 250 million pixels.
- The input GIF format animation does not exceed 300 frames.

- The width and height of the output image must not exceed 9999 pixels.

Note:
In any of the following situations, image processing may fail, resulting in the original image being returned:
1. If any parameter of the original image or the resulting image exceeds the above limits, we will be unable to process the image and can only respond with the original image.
2. When incorrect request parameters are entered, the image will not be processed and the original image will be returned directly, as in the following scenarios:
    - Repeated input parameters: `eo-img.resize=w/100&eo-img.resize=w/200` will be considered as invalid arguments;
    - Spelling Error: Any parameter with a format or spelling error, such as `eo-img.resize=w=100` , will be considered an invalid argument;
    - Resize Parameter Error: The parameters `w/` (width) and `h/` (height) should not be mixed with `s/` (short side) and `l/` (long side). For instance, `w/300/s/200` is an invalid argument, and the image will remain unchanged.
3. If the Image Resize feature is disabled in the console, all eo-img related parameters are treated as standard query strings, and the image processing feature will not be triggered.
4. If an abnormal situation occurs that prevents the image from being processed normally, we will prioritize providing the original image. In subsequent requests, we will automatically attempt to reprocess the image.

Billing description

This feature is billed based on the number of image resizing requests. For more information, see Billing Overview .

Note
The image resize feature is currently free for a limited time. An announcement will be released when the free trial ends.

Directions

- You can directly implement image scaling by concatenating image processing related parameters in the client request URL. Refer to the following steps to understand how to use it.

- If you prefer not to alter the original client request URL, you can utilize the capabilities of EdgeOne's edge functions to automatically adapt the scale or convert the image format based on the client request information. For usage, please refer to: Implementing Adaptive Image Format Conversion through Edge Functions .

Adding Image Processing Parameters to the Client Request URL

If you need to resize images by adding relevant parameters to the client request URL, you can refer to the following steps:

1. Log in to the EdgeOne console and choose **Site Acceleration** > **Media Processing** on the left sidebar.
2. On the Media Processing page, activate the **Image Scaling** feature by toggling the switch to the 'On' position to start using it.

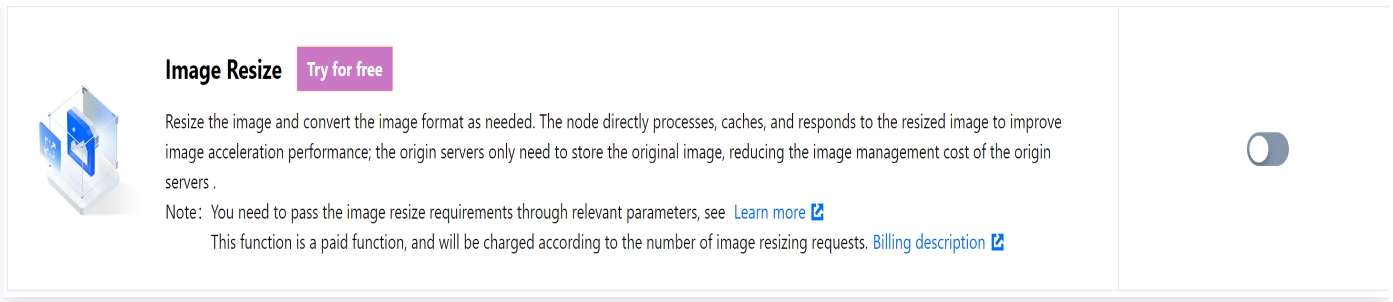


Image Resize Try for free

Resize the image and convert the image format as needed. The node directly processes, caches, and responds to the resized image to improve image acceleration performance; the origin servers only need to store the original image, reducing the image management cost of the origin servers .

Note: You need to pass the image resize requirements through relevant parameters, see Learn more

This function is a paid function, and will be charged according to the number of image resizing requests. Billing description

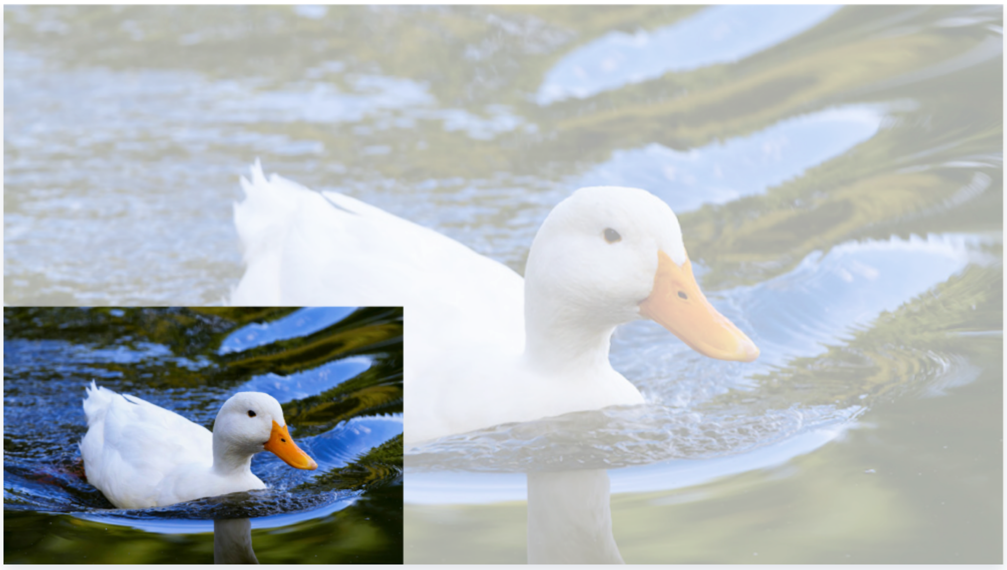
3. Once enabled, you simply need to append the `eo-img` related parameters to the client request URL to convey the image scaling requirements. EdgeOne will automatically complete the image processing based on the image processing parameters in the client request URL. For example, `https://www.example.com/foo.png?eo-img.resize=w/100` .

Image Processing Example

In the following examples, the original image is 500 × 280 pixels in resolution and 500 KB in size.

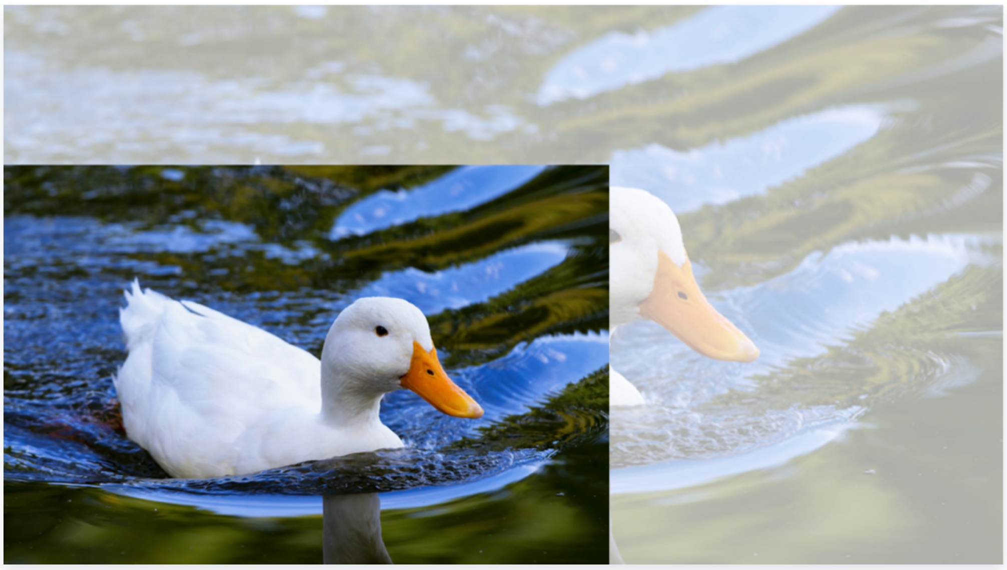
1. Change the width to 200 pixels, and the height is automatically adjusted based on the aspect ratio.

   Request URL: `http://www.example.com/foo.png?eo-img.resize=w/200` .



2. Change the height to 200 pixels, and the width is automatically adjusted based on the aspect ratio.

   Request URL: `http://www.example.com/foo.png?eo-img.resize=h/200` .



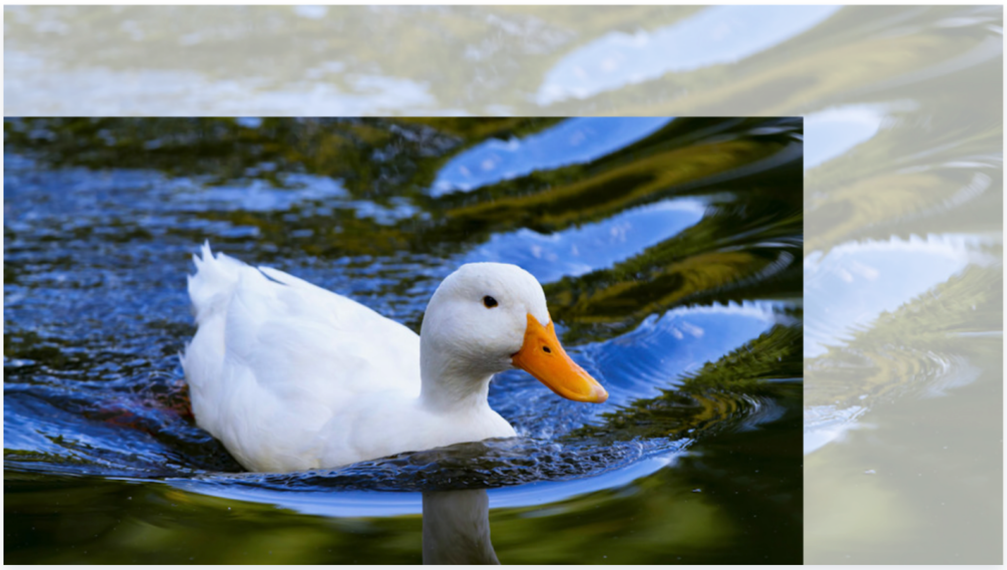
3. Change the width to 300 pixels and the height to 200 pixels.

   Request URL: `http://www.example.com/foo.png?eo-img.resize=w/300/h/200` .

> **Note**
> If both the width and height are specified, the aspect ratio of the original image may not be retained.

4. Change the long side to 400 pixels, and the short side is automatically adjusted based on the aspect ratio.

   Request URL: `http://www.example.com/foo.png?eo-img.resize=l/400` .



5. Change the short side to 200 pixels, and the long side is automatically adjusted based on the aspect ratio.

   Request URL: `http://www.example.com/foo.png?eo-img.resize=s/200` .

6. Convert the image to WebP format.
   Request URL: `http://www.example.com/foo.png?eo-img.format=webp` .
   Output image format: WebP
7. Change the width to 200 pixels, with the height automatically adjusted based on the aspect ratio, and convert the format to WebP.
   Request URL: `http://www.example.com/foo.png?eo-img.resize=w/200&eo-img.format=webp` .

Network Optimization
HTTP/2

Last updated：2023-09-07 15:05:47

What is HTTP/2?

By default, EdgeOne enables HTTP/2 access, supporting client-initiated requests via the HTTP/2 protocol. Hypertext Transfer Protocol Version 2 (HTTP/2 or HTTP 2.0) is the second major version of the HTTP protocol. It can effectively reduce network latency and accelerate site page loading.

Note:
1. If the client request does not utilize HTTP/2, EdgeOne is compatible with access via the HTTP 1.x protocol.
2. For configuring access requests, please refer to this document. If you need to configure HTTP/2 origin-pull, please see HTTP/2 Origin-pull.

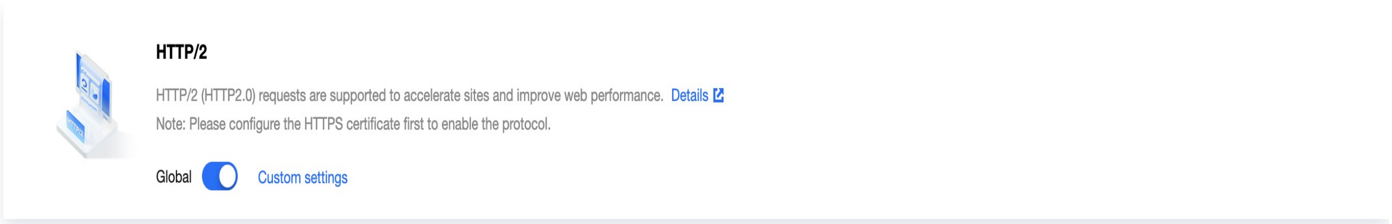
Preparations

You have configured SSL certificates for all domain names used to access the current site as instructed in Certificate Configuration.

Instructions

Scenario 1: Modifying HTTP/2 Support for All Domain Names of the Site

To enable or disable HTTP/2 for all domain names used to access the current site, refer to the following information:
1. Log in to the EdgeOne console. In the left sidebar, click **Site List**. Within the site list, click the **site** that needs to be configured to enter the site details page.
2. On the site details page, select **Site Acceleration** > **Network Optimization** to navigate to the Network Optimization details page.
3. Locate the HTTP/2 configuration card. By default, it is enabled, meaning that it supports site acceleration using HTTP/2 by default. You can toggle the **switch** to configure it on or off.


HTTP/2

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

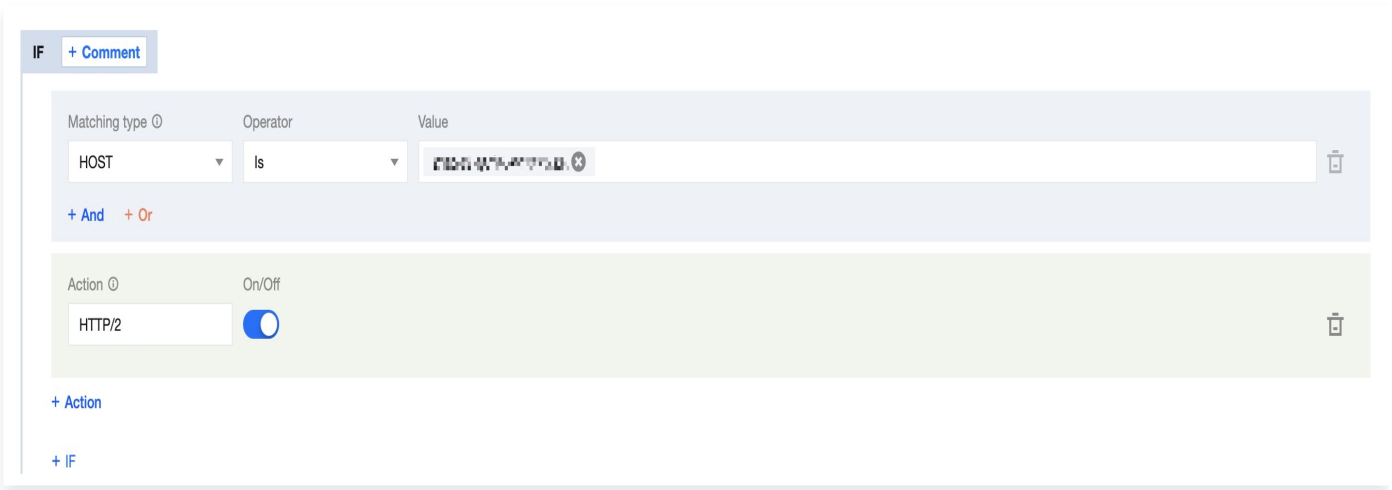
Note: Please configure the HTTPS certificate first to enable the protocol.

Global: On | Custom settings

Scenario 2: Enabling HTTP/2 for Specified Domain Names

To enable or disable HTTP/2 for specified domain names, refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to access the new rule editing page.
4. On the page that appears, select **HOST** from **Matching type** and specify an operator and a value to match the requests of specified domain names.
5. Click on **Action** > **Select Box**. In the pop-up action list, select **HTTP/2** as the action. Click on the **Switch** to configure it to be on or off.


IF + Comment

Matching type: HOST | Operator: Is | Value: [illegible]

+ And + Or

Action: HTTP/2 | On/Off: On

+ Action

+ IF

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

HTTP/3(QUIC)
Overview

Last updated：2023-09-08 10:21:58

What is HTTP/3?

HTTP/3 (HTTP over QUIC) is the next-generation internet transport protocol. It was developed to address the head-of-line blocking issue in HTTP/2. Unlike its predecessors, HTTP/3 is not based on TCP, but instead utilizes the QUIC protocol, which is based on UDP. Compared to HTTP/1.1 and HTTP/2, HTTP/3 offers faster connection establishment and data transfer speeds. It supports multiplexing, allowing for the simultaneous transmission of multiple requests and responses. Enhancements in congestion control and header compression significantly reduce latency and wait times, thereby improving website performance and user experience.

EdgeOne's support for HTTP/3

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

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 support in your App

- If your users need to access via an App, the App must integrate support for HTTP/3 or QUIC protocols. EdgeOne provides a QUIC SDK to help you complete the integration quickly. For more details, please refer to QUIC SDK Download and Integration Examples.
- If your site is primarily a website and the user's current browser supports and has enabled QUIC protocol access, all you need to do is enable HTTP/3 (QUIC) on EdgeOne. You can check whether the current browser supports it by using the HTTP/3 support detection tool."

Enable HTTP/3

Last updated：2023-09-08 10:51:51

This document outlines how to enable HTTP/3 protocol support within the EdgeOne platform.

Note:

1. In order to activate HTTP/3 support, it is necessary that the current access domain has already configured an HTTPS certificate for it to take effect.
2. If both HTTP/2 and HTTP/3 are activated, either HTTP/2 or HTTP/3 will be utilized, depending on the actual client request conditions.

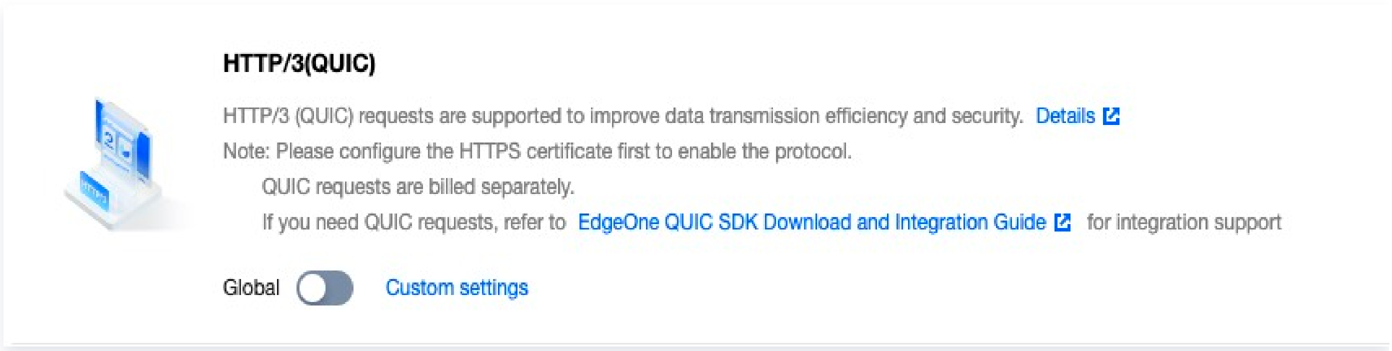
Billing description

HTTP/3 is a premium feature. For specific cost standards, please refer to: Value-added Service Fee - QUIC Requests.

Scenario 1: Enabling HTTP/3 Protocol Support for All Domain Names on the Site

To enable HTTP/3 protocol configuration for all domain names, please refer to the following steps:

1. Log in to the EdgeOne console. In the left sidebar, click **Site List**. Within the site list, click the **site** that needs to be configured to enter the site details page.
2. On the site details page, select **Site Acceleration** > **Network Optimization** to navigate to the Network Optimization details page.
3. Toggle the HTTP/3 module switch to enable or disable the HTTP/3 feature.



HTTP/3(QUIC)

HTTP/3 (QUIC) requests are supported to improve data transmission efficiency and security. Details

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

QUIC requests are billed separately.

If you need QUIC requests, refer to EdgeOne QUIC SDK Download and Integration Guide for integration support

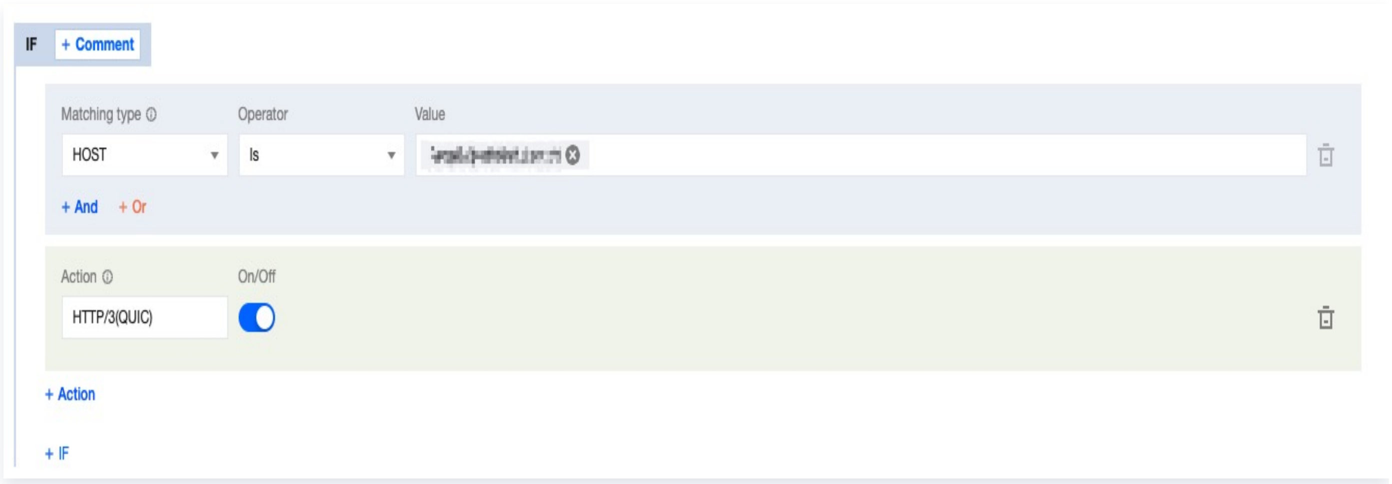
Global

- Off (default): HTTP/3 requests are not supported.
- Enabled: HTTP/3 requests are supported, utilizing HTTP/3 to expedite site requests.

Scenario 2: Enabling HTTP/3 Protocol Support for Specified Domain Names

To configure HTTP/3 protocol activation for specific domain names, please refer to the following steps:

1. Log in to the Edge Security Acceleration Platform Console . In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs to be configured.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to enter the new rule editing page.
4. On the page that appears, select **HOST** from **Matching type** and specify an operator and a value to match the requests of specified domain names.
5. From the **Operation** drop-down list, select **HTTPS/3**. Then, toggle the switch to enable this feature.



Matching type: HOST
Operator: Is
Value: [redacted]
Action: HTTP/3(QUIC)
On/Off: On

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

IPv6 access

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

Overview

EdgeOne provides a one-click feature to enable IPv6 access, allowing IPv6 clients to interact with nodes via the IPv6 protocol.

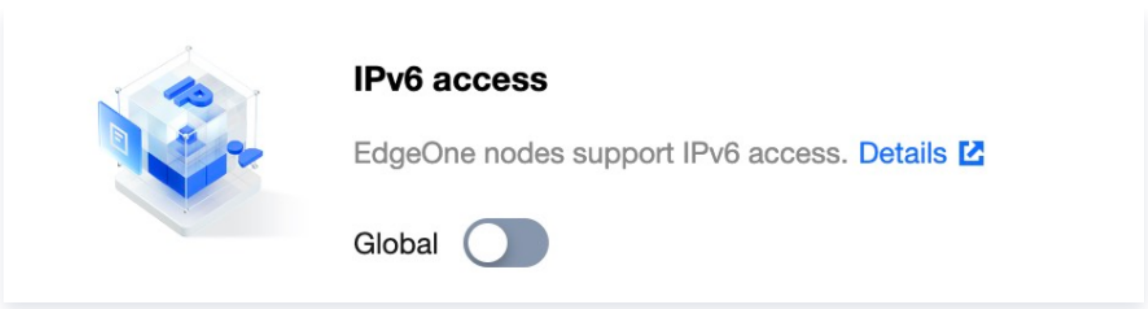
> **Note:**
> At present, most EdgeOne nodes support IPv6 access. You may contact us to confirm the specific resource coverage.

Use Cases

- **IPv6-only Network Environment:** Certain regions and organizations may already be utilizing IPv6-only network environments. Devices within these networks may not be able to directly access services based on IPv4. By enabling the IPv6 access feature, you can ensure these clients can access your accelerated resources seamlessly.
- **Dual Stack Network Environment:** In a dual stack network environment that supports both IPv4 and IPv6, clients can automatically choose to use either the IPv4 or IPv6 protocol to access accelerated resources based on network conditions. In some instances, IPv6 connections may be faster than IPv4, thus enabling IPv6 access can help enhance the performance for these clients.
- **Future Network Compatibility:** As IPv4 address resources gradually deplete, an increasing number of networks and devices will adopt IPv6. By enabling IPv6 access, you can ensure that your acceleration services remain compatible with these emerging networks and devices in the future.
- **Policy and Compliance Requirements:** Certain regions or industries may require enabling IPv6 support in services to meet policy or compliance requirements. In such cases, activating the IPv6 access feature can assist you in fulfilling these demands.

Instructions

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs configuration to enter the site details page.
2. On the site details page, select **Site Acceleration** > **Network Optimization** to navigate to the Network Optimization details page.
3. On the IPv6 access configuration card, click on the **Enable Globally** button to activate IPv6 access for all domain names on the site.



IPv6 access

EdgeOne nodes support IPv6 access. Details

Global

Maximum upload size

Last updated: 2023-09-07 15:06:12

Overview

The maximum upload size refers to the largest amount of data that can be uploaded in a single client request. If the uploaded data exceeds the set upper limit, EdgeOne will respond to the client with a 413 (Request Entity Too Large) status.

Note:

1. Only the EdgeOne Enterprise and Standard plans support the disabling of upload size limits.
2. By default, the maximum upload size limit is enabled, with an upper limit of 800MB.

Use Cases

1. **Large File Upload:** For businesses involving large file uploads, such as online video platforms, large game distribution, big data analysis, etc., you can increase the upper limit of the upload size or directly disable the size limit, allowing your large files to be uploaded without hindrance.
2. **Defending Against Malicious Uploads:** For businesses with frequent user interactions, such as social media, forums, or blogs, there may be instances of malicious users uploading excessively large files. You can enable size restrictions and lower the upper limit to prevent oversized files from being uploaded to the source station, thereby reducing the pressure on the source station, preventing potential security risks, and enhancing the user experience.
3. **Conserving Data Transfer:** For businesses sensitive to data usage, such as online education, virtual meetings, or API services, you may wish to conserve data by limiting the size of uploaded files. You can enable size restrictions and adjust the upper limit to a smaller value, reducing unnecessary data transfer and lowering data costs.

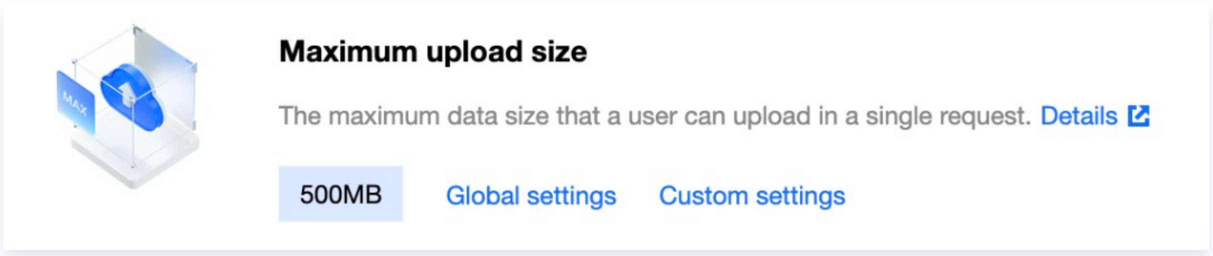
You can flexibly configure the "Maximum Upload Size" to meet the needs of various business scenarios.

Instructions

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

To configure the same maximum upload size for all domain names used to access a site, or as a fallback configuration at the site level, refer to the following steps:

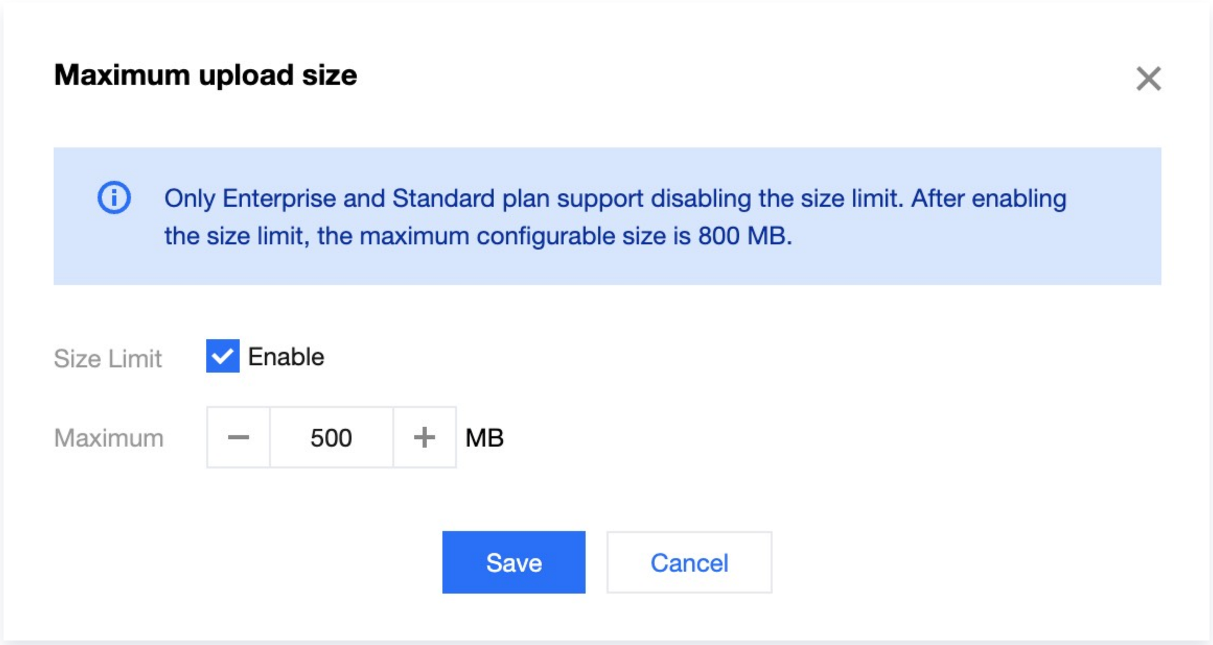
1. Log in to the EdgeOne console. In the left sidebar, click **Site List**. Within the site list, click the **site** that needs to be configured to enter the site details page.
2. On the site details page, select **Site Acceleration** > **Network Optimization** to navigate to the Network Optimization details page.
3. Locate the **Maximum Upload Size** configuration card and click on **Global Site Settings.**



Maximum upload size

The maximum data size that a user can upload in a single request. Details

4. Within the pop-up window, check the box to enable the size limit and modify the upper limit value. The changes will take effect immediately after clicking **Save**.



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

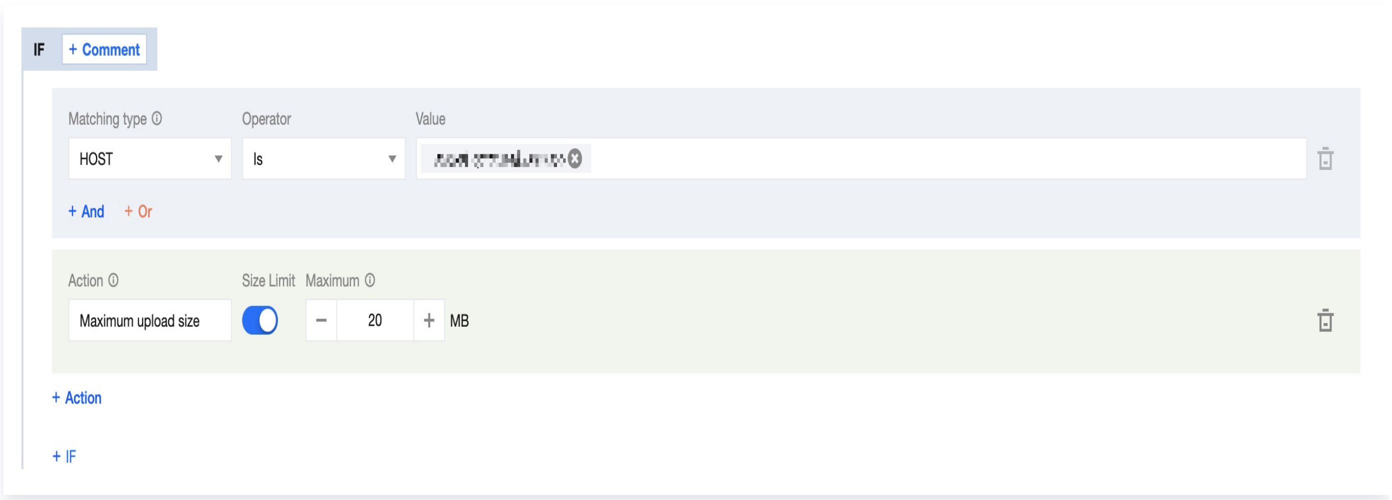
Maximum: 500 MB

Scenario 2: Configuring the maximum upload size for specific request granularities such as domain names, paths, or file extensions.

If you need to configure different maximum upload sizes for different domains, paths, or file extensions, for example, setting a maximum upload size of 20 MB for the domain `www.example.com` under the `example.com` site, you can follow the steps below:

1. Log in to the EdgeOne console. In the left sidebar, click **Site List**. Within the site list, click the **site** that needs to be configured to enter the site details page.
2. On the site details page, click **Rule Engine**.

3. On the rule engine management page, click **Create rule** to access the new rule editing page.
4. On the rule editing page, enter the rule name, select Host as the matching type, and configure it as `www.example.com`.
5. Click on **Action,** in the pop-up action list, select **Maximum Upload Size** as the operation, click on to enable the size limit, and configure the upper limit to 20 MB.



Matching type: HOST
Operator: Is
Value: [redacted]

Action: Maximum upload size
Size Limit: on
Maximum: 20 MB

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

WebSocket

Last updated：2023-09-07 15:06:26

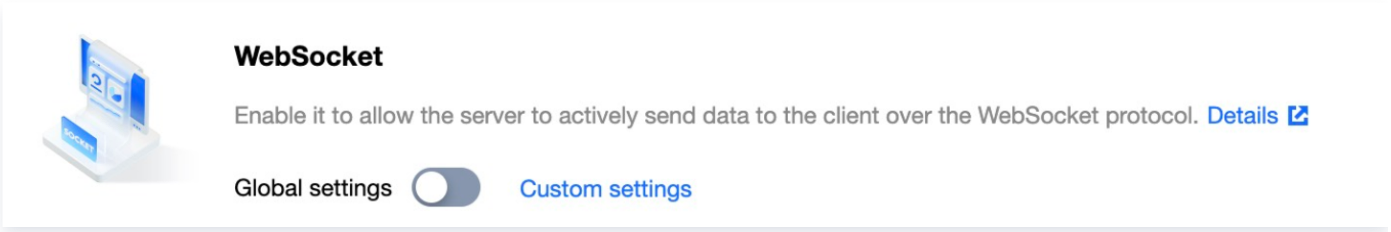
Overview

EdgeOne supports the activation of the WebSocket protocol for access, enabling the server to proactively push data to the client. The WebSocket protocol is a persistent protocol based on TCP, which facilitates full-duplex communication between the client and the server, allowing the server to actively send information to the client. Prior to the WebSocket protocol, Web Apps that implemented duplex communication between the client and the server had to continuously send HTTP request calls for inquiries, leading to increased service costs and inefficiency. Owing to the advantage of full-duplex communication, WebSocket is extensively employed in scenarios such as social networking subscription, collaborative work, market quotation broadcasting, interactive live streaming, online education, and the Internet of Things. It significantly conserves server resources and bandwidth, and enables more real-time communication.

Note:
1. Currently, only WebSocket based on HTTP/1.1 is supported, WebSocket based on HTTP/2 is not supported.
2. The maximum connection timeout supported is 300 seconds.

Scenario 1: Configuring WebSocket for All Domain Names on the Site

To enable or disable WebSocket for all domain names used to access a site, or to use it as a fallback configuration at the site level, refer to the following steps:
1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs configuration to enter the site details page.
2. On the site details page, select **Site Acceleration** > **Network Optimization** to navigate to the Network Optimization details page.
3. Locate the **WebSocket** configuration card and toggle the switch to enable or disable the WebSocket feature:



WebSocket

Enable it to allow the server to actively send data to the client over the WebSocket protocol. Details

Global settings Custom settings

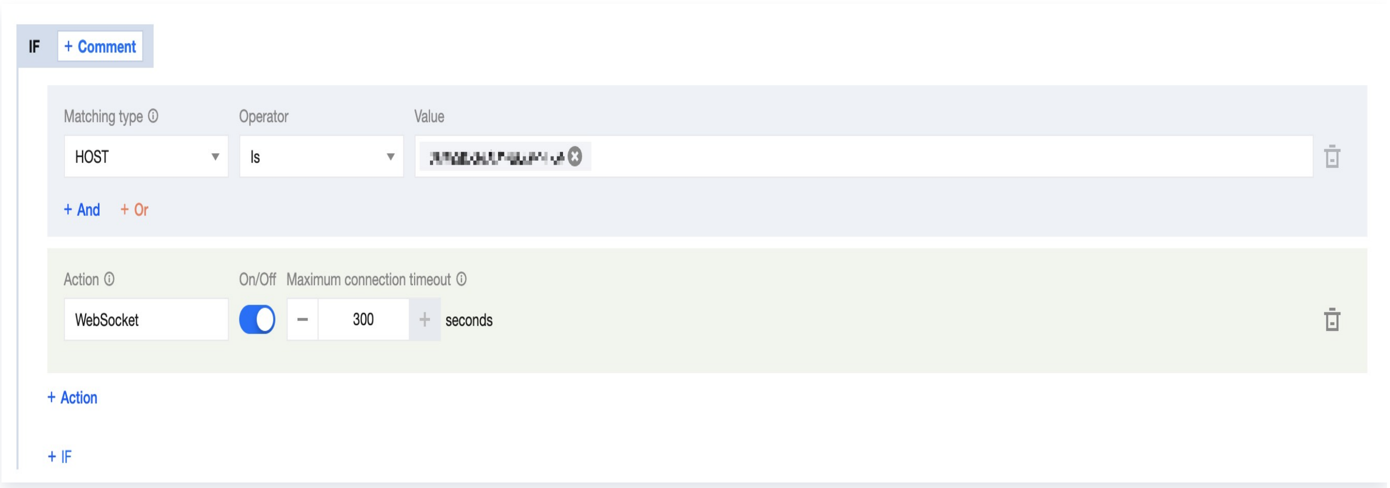
  - Off (default): WebSocket protocol is not supported. When enabled, WebSocket protocol is supported.
  - Maximum connection duration: If there is no data transmission within the timeout period, the connection will be terminated.

Scenario 2: Configuring WebSocket for Specified Domain Names

To configure WebSocket for specific domain names, such as setting up WebSocket for the domain name `www.example.com` under the site `example.com`, refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to navigate to the new rule editing page. On this page, select Host as the matching type and configure it as `www.example.com`.
4. Click on **Operation**, from the pop-up operation list, select **WebSocket**. Toggle the **switch** to enable it and configure the maximum connection timeout duration.

> **Note:**
> Maximum connection duration: Configurable between 1-300 seconds.



Matching type: HOST
Operator: Is
Value: [redacted]

Action: WebSocket
On/Off: On
Maximum connection timeout: 300 seconds

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

Client IP Geolocation Header

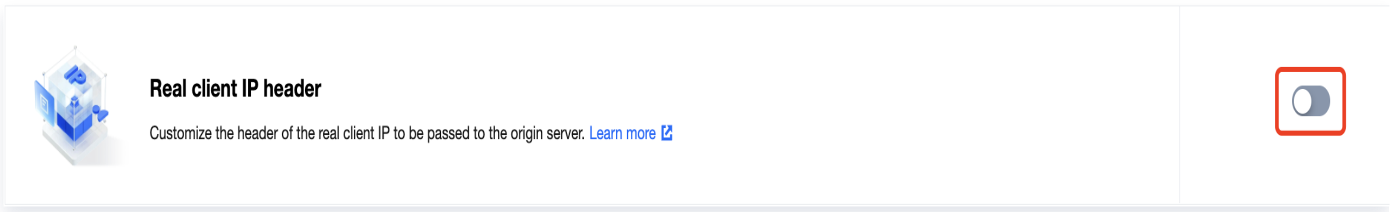
Last updated：2023-09-07 15:06:33

Overview

Real client IP header allows you to create a custom origin-pull HTTP request header that carries the real client IP information.

Instructions

1. Log in to the EdgeOne console and select **Site Acceleration** > **Network Optimization** from the left sidebar.
2. On the Network Optimization page, select the desired site and click to enable the "Real Client IP Header" feature.



Real client IP header

Customize the header of the real client IP to be passed to the origin server. Learn more

3. In the pop-up window, customize the name of this header, for example, Tencent-Client-IP, and click **Save**.

Client IP Geographical Location

Last updated：2023-09-07 15:06:39

Overview

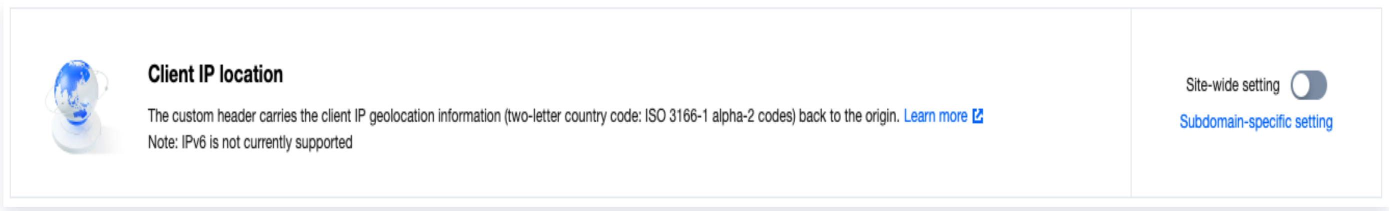
The custom header can carry the geographical location information of the client IP to the origin.

Note

- The country/region value is represented by an ISO 3166-1 alpha-2 code (a two-letter country/region code).
- Currently, IPv6 is not supported.

Instructions

1. Log in to the EdgeOne console and select **Site Acceleration** > **Network Optimization** from the left sidebar.
2. On the Network Optimization page, select the desired site and click ◯ to enable the "Client IP Geolocation" feature.



Client IP location

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

Note: IPv6 is not currently supported

3. In the pop-up window, you can customize the header name or use the default name "EO-Client-IPCountry". Click **Save** to apply the changes.

gRPC

Last updated：2023-09-08 10:52:10

Support for gRPC in EdgeOne

EdgeOne allows for the activation of gRPC protocol support within the console (disabled by default). Once enabled, it can simultaneously support HTTP/HTTPS/gRPC protocols, automatically adapting to the user's request protocol. That is, if an HTTP request is made, the HTTP protocol is used; if a gRPC request is made, the gRPC protocol is used.

Note:

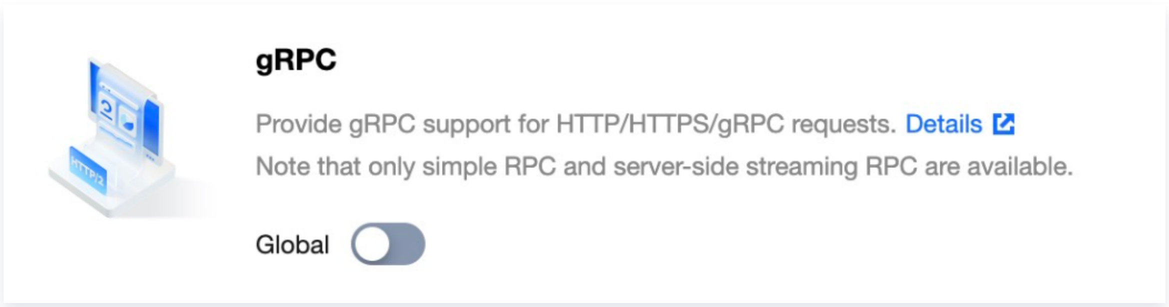
1. Currently, only Simple RPC and Server-side streaming RPC modes are supported;
2. gRPC is implemented based on full-link HTTP/2. Therefore, when enabling gRPC, please ensure that HTTP/2 access and HTTP/2 back-to-source are also enabled.

What Is gRPC?

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

Instructions

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** that needs configuration to enter the site details page.
2. On the site details page, select **Site Acceleration** > **Network Optimization** and locate the gRPC module.



gRPC

Provide gRPC support for HTTP/HTTPS/gRPC requests. Details
Note that only simple RPC and server-side streaming RPC are available.

Global

3. Toggle the switch of the gRPC module to enable or disable support for the gRPC protocol.

URL Rewrite
Access URL Redirection

Last updated：2023-09-07 15:06:52

Overview

A node redirects the URL requested by the client to the destination URL based on the response status code. This feature allows the URL redirection, which was originally generated and returned by the origin in your business scenario, to be constructed and returned directly by the EdgeOne edge node. This reduces the network latency of back-to-source and the load of the origin in generating URL redirection, thereby enhancing the client's access performance.

Scenarios

The following are common scenarios where URL rewriting is applicable:

- **Website Migration or Restructuring:** When a website undergoes migration or restructuring, changes to the URL structure may occur. To maintain the validity of old links, URL redirection can be used to redirect old URLs to new ones, ensuring that users and search engines can smoothly access the new resources.
- **Geolocation or Device Type Targeting:** Based on the user's geographical location or device type, URL redirection can be used to guide users to different resources or pages. For instance, providing specially optimized mobile pages for mobile device users, or offering pages in different languages based on the user's location.
- **Temporary Maintenance or Event Pages:** During temporary website maintenance or specific events, URL redirection can be used to guide users to maintenance notification pages or event pages, thereby enhancing the user experience.

Configure access URL redirection based on the granularity of requests, such as specific domain names, paths, or client regions

If you need to configure different access URL redirections for different domain names, paths, or client regions, etc. For instance, to configure access URL redirection for the `www.example.com` domain under the `example.com` site, and you currently wish to replace the path `/old-path/1234` with `/new-path/1234`, refer to the following steps:

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **Rule Engine**.
3. On the rule engine management page, click **Create rule** to enter the new rule editing page.

Using the current scenario as an example, you can follow the steps below:

3.1 On the rule editing page, select HOST equals www.example.com for the matching type.

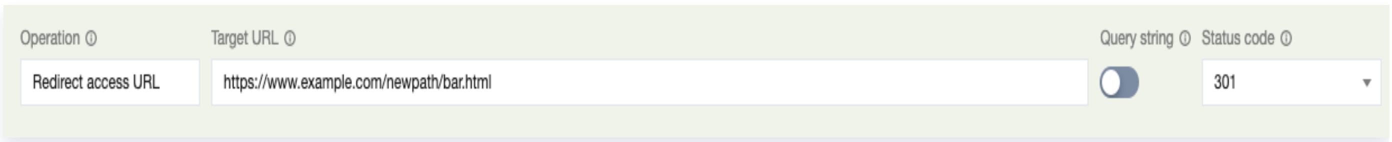
3.2 Click **Operation**, and in the pop-up operation list, select **Access URL Redirection**.

3.3 To configure the URL redirection rule for the current scenario, you can keep the target request protocol and target hostname to follow the request, configure the target path for regular expression replacement, and input the regular expression `^/old-path/(\d+)$` to match the target path, replacing it with `/new-path/$1` . The explanations for the related configuration items are as follows:

Configuration items	Note
Target protocol	The request protocol of the target redirect address defaults to follow the request, and it can support specifying redirection to HTTP/HTTPS protocol.
Target hostname	The Hostname part of the target redirection address defaults to follow the request and supports modification to a custom domain name.
Target path	The path part of the target redirection address offers three selection modes: • Follow Request: The default configuration follows the path of the request. • Customization: Define a complete path, replacing the original request path with the target path. For example, `/download` . • Regular Replacement: Supports path matching and replacement through Google RE2 regular expressions. It also supports referencing regular capture groups with `$num` , where `num` represents the group number, supporting up to `$9` . For instance, if you wish to replace the path `/old-path/1234` with `/new-path/1234` , you can configure the regular expression as `^/old-path/(\d+)$` . In the replacement path, you can configure it as `/new-path/$1` , where `$1` refers to the first captured group in the regular expression, i.e., the numeric part of the path.
Carry query parameters	Whether to carry the original query parameters to the target URL is enabled by default, meaning the original query parameters are included after redirection.
Status code	Select the response status code for redirection: 302 (default), 301,

	303, and 307.

4. The complete rule configuration is as follows. Click **Save and publish** to complete the rule configuration.



Operation	Target URL	Query string	Status code
Redirect access URL	https://www.example.com/newpath/bar.html		301

Origin-pull URL Rewrite

Last updated: 2023-09-07 15:06:58

Overview

Based on specified rules, this feature rewrites the user request URL received by the node to the destination URL when the node sends the request to the origin server, which doesn't affect the node cache key.

Scenarios

If the URL accessed by the client has already been published externally and is not suitable for modification, and the business origin server has changed the URL path on the origin server for some reason; or for search engine optimization (SEO), the URL accessed by the client and the path of the origin server URL are not consistent. In these cases, by setting the origin-pull URL rewrite rule, the node can rewrite the origin-pull URL to the actual URL corresponding to the resource on the origin server without changing the client access URL.

Instructions

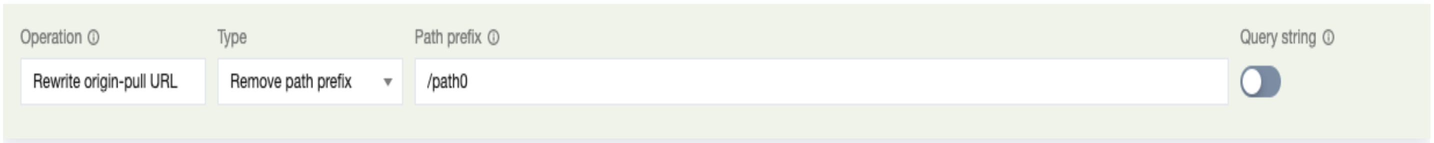
1. Log in to the EdgeOne console and select **Rule Engine** from the left sidebar.
2. On the Rule Engine page, select the desired site and configure the access URL redirection rules as needed. For information on how to use the Rule Engine, please refer to Rule Engine.
   Description of configuration items:

Local Disk Types	Note
Add path prefix	Add a specified path prefix to the request URL path. For instance, if the request URL is `http://www.example.com/path0/index.html` and the added path prefix is /prefix, the rewritten URL will be `http://www.example.com/prefix/path0/index.html`.
Remove path prefix	Remove the specified path prefix from the request URL. For instance, if the request URL is `http://www.example.com/path0/path1/index.html` and the specified path prefix to be removed is /path0, the rewritten URL will be `http://www.example.com/path1/index.html`.
Replace full path	Replace the entire request URL Path. For instance, if the request URL is `http://www.example.com/path0/index.html` and the entire path is replaced

	with /new/page.html, the rewritten URL will be `http://www.example.com/new/page.html`.

Parameter Configuration Sample Code

If the access URL redirection configuration for the request URL `https://www.example.com/path0/path1/foo.html` is as follows:



Operation	Type	Path prefix	Query string
Rewrite origin-pull URL	Remove path prefix	/path0	Off

Thus, the client request `https://www.example.com/path0/path1/foo.html?key1=value1` will be rewritten to `https://www.example.com/path1/foo.html?key1=value1` to fetch the requested resource during origin-pull. If the query string key1=value1 needs to be ignored during origin-pull, please use the origin-pull request parameter settings feature.

Modifying Header

Modifying HTTP Response Headers

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

Overview

You can customize, add, and delete headers in HTTP responses from nodes to clients, which will not affect the node cache.

Instructions

1. Log in to the EdgeOne console and select **Rule Engine** from the left sidebar.
2. On the Rule Engine page, select the desired site and configure the rules for modifying HTTP node response headers as needed. For information on how to use the Rule Engine, please refer to Rule Engine. The configuration items are explained as follows:

Local Disk Types	Note
Settings	Change the value of the specified header parameter to the set value, ensuring the header is unique. Note: If the specified header does not exist, it will be added.
Add	Add the specified header. See Note 1 for details.
DELETE	Deletes the specified header.

> **Note:**
> Note 1: If the header already exists, a new header will be added without overwriting the existing one.

Supports and Limits

- A custom header parameter must be in the following format:
  - Parameter Name: Composed of 1 to 100 characters, including digits 0-9, letters a-z, A-Z, and the special character `-`.
  - Parameter value: It can contain 1-1000 characters.
- During one HTTP request header modification operation, you can add up to 30 headers of different types, which will be executed in sequence from top to bottom.

- The following standard headers cannot be modified:

```
Accept-Ranges
Age
Allow
Authentication-Info
Cache-Control
Connection
Content-Encoding
Content-Length
Content-Location
Content-MD5
Content-Range
Content-Type
Date
Error
ETag
Expires
If-Modified-Since
Last-Modified
Meter
Proxy-Authenticate
Retry-After
Set-Cookie
Transfer-Encoding
Vary
WWW-Authenticate
```

Parameter Configuration Sample Code

Access-Control-Allow-Origin

This header requests permission to access cross-origin resources, so as to implement CORS.

- Header name: `Access-Control-Allow-Origin` .
- Header Value: Enter domain names and IPs, separated by commas (up to 1000 characters). Each must include `http://` or `https://` , such as `http://test.com,http://1.1.1.1` . "*" is supported.
- Value description:

Header value	Note
*	Exact Match: Add response header: `Access-Control-Allow-Origin:*` , allowing requests from all

	domains.
`http://cloud.tencent.com, https://cloud.tencent.com,http://www.b.com`	Fixed Match: • If the source `https://cloud.tencent.com` matches the list, the response will add the header: `Access-Control-Allow-Origin: https://cloud.tencent.com` . • If the source is `https://www.qq.com` and it does not hit the list, the cross-domain header will not be responded to.
`https://*.tencent.com`	Matching secondary wildcard domains: • If the source `https://cloud.tencent.com` matches the list, the response will add the header: `Access-Control-Allow-Origin: https://cloud.tencent.com` . • If the source is https://cloud.qq.com and it does not hit the list, the cross-domain header will not be responded to.
`https://cloud.tencent.com:8080`	Port Matching: • If the source is `https://cloud.tencent.com:8080` and it hits the list, the response will add the header: `Access-Control-Allow-Origin:https://cloud.tencent.com:8080` . • If the source is `https://cloud.tencent.com` and it does not hit the list, the cross-origin header will not be responded to. Note: Arbitrary port matching is not supported. If there is a specific port, it must be specified in the header value.

Access-Control-Allow-Methods

This header specifies the HTTP request methods allowed for cross-origin access.

- Header name: `Access-Control-Allow-Methods` .
- Header value: Multiple values can be set, such as `POST` , `GET` , and `POTIONS` .

Access-Control-Max-Age

This header specifies the validity period of the result of a preflight request in seconds.

Note
A non-simple CORS request requires a preflight request (that uses HTTP request methods) before being initiated. It is made to check whether the CORS request is secure and acceptable. Typical cases requiring preflight requests:
A CORS request uses methods other than `GET` , `HEAD` , and `POST` or is initiated via

> `POST` with the request data type other than `application / x-www-form-urlencoded` , `multipart / form-data` , and `text / plain` (such as `application / xml` or `text / xml`).

- Header name: `Access-Control-Max-Age` .
- Header value: Enter the number of seconds, for example, `1728000` .

Content-Language

This header specifies the language to be used by the accessed page.

- Header name: `Content-Language` .
- Header value: `zh-CN` or `en-US` .

Content-Disposition

This header activates download in the browser and sets the default name of the downloaded file.
When the server sends a file supported by the client browser (such as a TXT or JPG file), the browser opens the file by default. If you want to ask the user to save the file, configure the `Content-Disposition` field to override the browser's default behavior.

- Header name: `Content-Disposition` .
- Header Value: A common configuration example is: `attachment;filename=FileName.txt` .

Modifying HTTP Request Headers

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

Overview

You can customize, add, and delete headers in HTTP origin-pull requests from nodes to the origin.

Note
EdgeOne forwards `X-Forwarded-For` and `X-Forwarded-Proto` to the origin by default, so you don't need to configure them.

Operations Guide

1. Log in to the EdgeOne console and select **Rule Engine** from the left sidebar.
2. On the Rule Engine page, select the required site and configure the HTTP request header rules as needed. For information on how to use the Rule Engine, please refer to Rule Engine.

   Description of configuration items:

Local Disk Types	Note
Settings	Change the value of the specified header parameter to the set value, ensuring the header is unique. Note: If the specified header does not exist, it will be added.
Add	Add the specified headers. Please note: if the header already exists, it will overwrite the original header and remain unique.
DELETE	It deletes the specified header.

   Description of header types:

Header Type	Note
Custom	Customize the header. • Name: Composed of 1-100 characters, including digits 0-9, letters a-z, A-Z, and the special character '-'. • Value: 1 to 1000 characters (Chinese is not supported).

Preset header	Aggregated client header information based on the client's `User-Agent` information: • Client device type: `EO-Client-Device`. Values: `Mobile`, `Desktop`, `SmartTV`, `Tablet`, or `Others`. • Client Operating System: `EO-Client-OS`. Values: `Android`, `iOS`, `Windows`, `MacOS`, `Linux`, or `Others`. • Client browser type: `EO-Client-Browser`. Values: `Chrome`, `Safari`, `Firefox`, `Internet Explorer`, or `Others`.

Supports and Limits

- During one HTTP request header modification operation, you can add up to 30 headers of different types, which will be executed in sequence from top to bottom.
- The following standard headers cannot be modified:

Accept	Accept-Charset	Accept-Encoding	Accept-Language
Accept-Ranges	Age	Authorization	Cache-Control
chunked	close	Connection	Content-Encoding
Content-Length	Content-Range	Content-Type	Cookie
Date	Etag	Expect	Expires
From-Tencent-Lego-Cluster	From-Tencent-Lego-Cluster-Client-Info	From-Tencent-Lego-Cluster-Edge-Server-Info	From-Tencent-Lego-Dsa
From-Tencent-Lego-Dsa-Client-Info	From-Tencent-Lego-Dsa-Edge-Server-Info	From-Tencent-Lego-Overload	Host
identity	If-Match	If-Modified-Since	If-None-Match
If-Range	keep-alive	Last-Modified	Location
multirange	normal	Pragma	Proxy-Authorization

Proxy-Connection	Range	Referer	Server
Server-Timing	Set-Cookie	Transfer-Encoding	upgrade
Upgrade	Upgrade-Insecure-Requests	User-Agent	Via
X-Cache-Lookup	X-Forwarded-For	X-Last-Update-Info	x-redirect-to-self
X-Via	-	-	-

Custom Error Page

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

Overview

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.

Note
Only status codes returned for origin-pull requests are supported.

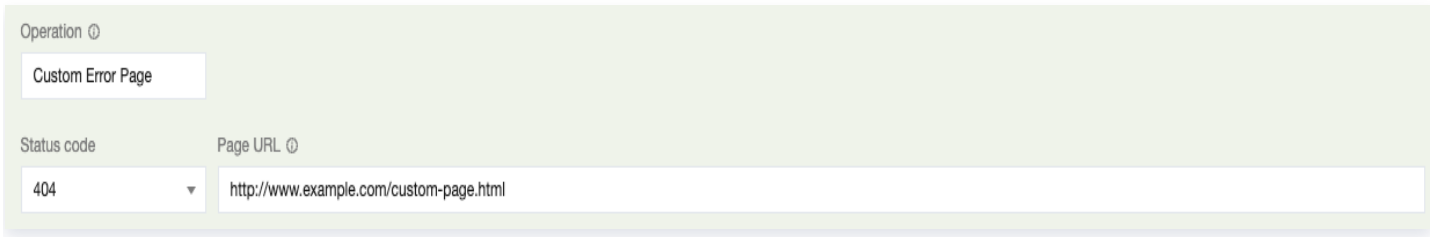
Instructions

1. Log in to the EdgeOne console and select **Rule Engine** from the left sidebar.
2. On the Rule Engine page, select the desired site and configure the custom error page rules as needed. For information on how to use the Rule Engine, refer to Rule Engine.
   Description of configuration items:

Configuration items	Note
Status code	Specify the error status code returned by the origin: • 4XX: 400, 403, 404, 405, 414, 416, 451 • 5XX: 500, 501, 502, 503, 504
Page URL	Specify the error page address, for instance: `http://www.example.com/custom-page.html`

Parameter Configuration Sample Code

The following configuration sample shows how to redirect requests to a custom error page for the 404 Not Found error:

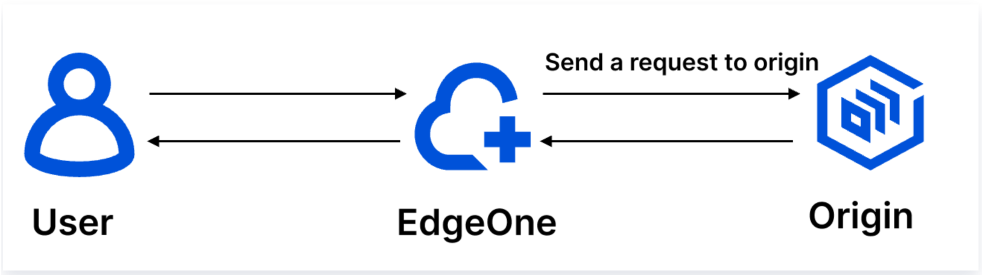

Operation
Custom Error Page
Status code
404
Page URL
http://www.example.com/custom-page.html

Request and Response Actions
Default HTTP Headers of Origin-Pull Requests

Last updated: 2023-09-07 16:19:08

Overview

By default, EdgeOne forwards all client request headers during origin-pull requests, also carrying the default request headers customized by EdgeOne. If further modifications to the HTTP headers of the origin-pull requests are required, please refer to 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-Client-IP

`EO-Client-IP` records the IP address of the client request that establishes a connection with EdgeOne. If the request has not passed through any proxy servers, this header IP is the real client IP address. If the request has passed through a proxy server, this header IP represents the IP address of the proxy server.

X-Forwarded-For

`X-Forwarded-For` is used to record the IP addresses of the proxy server and the real client. When a user's request reaches the EdgeOne edge node through multiple hops, this header can be used to view the real client IP address and the preceding proxy server address that reached the EdgeOne edge node. The value of this header is as follows:

- If the request sent to EdgeOne carries the `X-Forwarded-For` header, which has recorded the original client IP address, EdgeOne will append the IP address of the preceding proxy server that reached the EdgeOne edge node to the header value. Suppose the IP address of the preceding proxy server connected to the EdgeOne edge node is `10.1.1.1`, and the request carries `X-Forwarded-For: 192.168.1.1 (original client IP)` , then the origin-pull request header value is `X-Forwarded-For: 192.168.1.1.10,10.1.1.1` .

- If the request sent to the EdgeOne edge node does not contain the `X-Forwarded-For` header, EdgeOne will add the `X-Forwarded-For` header during the origin-pull request. The value of this header is the IP address of the preceding proxy server that established a connection with the EdgeOne edge node, and its value is the same as the `EO-Client-IP` header.

For more information, see X-Forwarded-For.

X-Forwarded-Proto

`X-Forwarded-Proto` is used to record the client's request protocol. It takes the value of the HTTP protocol used by the current client to initiate the request. The header values are:

- `X-Forwarded-Proto： http`
- `X-Forwarded-Proto： https`
- `X-Forwarded-Proto： quic`

For more information, see X-Forwarded-Proto.

CDN-Loop

`CDN-Loop` records the number of times a request has passed through the EdgeOne edge acceleration node, primarily to prevent request loops. Each time a client request passes through an EdgeOne edge node, the count of `CDN-Loop` increases by one and is marked in the request header. If the Loops value in the request header reaches ⩾ 16, the node will reject the request and respond with a 423 status code.

Example: `CDN-Loop: TencentEdgeOne; loops=3` .

EO-LOG-UUID

The `EO-LOG-UUID` header carries the unique identifier of the request. When an access exception occurs, the header is used for troubleshooting based on the full-linkage logs of user requests.

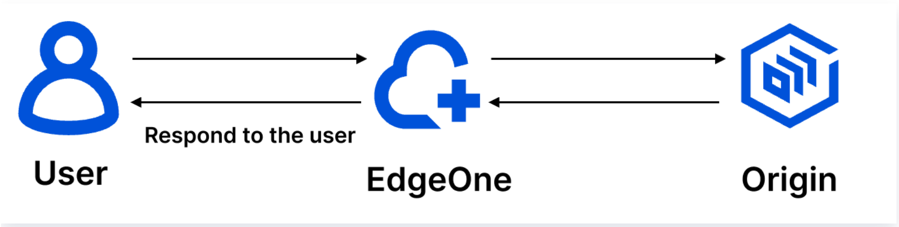
Example: `EO-LOG-UUID: 4105283880544427145` .

Default HTTP Response Headers

Last updated：2023-09-07 16:19:12

Overview

By default, EdgeOne transparently transmits the origin's response headers to the client, unless the client has custom configurations for adding, deleting, or modifying HTTP headers. The following sections will introduce the response headers defined by EdgeOne, which are sent to the client by default.



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 request has hit the cache. This header has two response statuses:

- `EO-Cache-Status: HIT` : The request hit the cached resource on the EdgeOne node and the node directly responds to the request.
- `EO-Cache-Status: MISS` : The request does not hit the cached resource on the EdgeOne node and the resource must be fetched from the origin.

Server

This header identifies the server name. The header value depends on the service upon which the Web Server is built. By default, if the origin's HTTP response headers include this header, it is passed through to the client. If the origin does not respond with this header, then the EdgeOne node will add it, with the value `Server: TencentEdgeOne` . For more details, please refer to Server.

Values of `Server` header for different origin types:

- When the origin is Tencent Cloud Object Storage (COS): `Server: tencent-cos` .
- When the origin is a Cloud Virtual Machine (CVM): `Server: nginx` , `Server: Apache` , `Server: tomcat` , or `Server: Microsoft-IIS` .
- Cloud Load Balancer (CLB) origin: `Server: openresty` .

Date

`Date` is used to indicate the most recent time the EdgeOne node updated the file from the origin. By default, if the origin's HTTP response header includes this header, it is passed through to the client. If the origin does not respond with this header, the EdgeOne node will add it, with the value set to the current time of the node server. For more details, please refer to Date.
For instance, if EdgeOne requests the origin and the origin responds with `Date: Sat, 07 Jan 2023 14:15:52 GMT`, and if the resource is set to cache for 1 hour on the CDN, then the client accessing this resource within 1 hour will receive the Date header value as: `Date: Sat, 07 Jan 2023 14:15:52 GMT`.

Connection

This header is used to indicate how persistent connections are managed during communication between the client and the server. By default, if the origin's HTTP response headers include this header, it is passed through to the client. If the origin does not respond with this header, EdgeOne will add it based on the following conditions:

- **HTTP/2 and QUIC requests**: This header is not added.
- If the current request uses HTTP1.0 and keepalive is not enabled, the header is set to: `Connection:close`.
- If the origin's response headers do not include `content-length` and are associated with the `transfer-encoding` header, then the header is set to: `Connection:close`.
- In other scenarios, the header is set to `Connection:keepalive`.

For more information, see Connection.

Alt-Svc

`Alt-Svc`, short for "Alternative-Service", lists alternative access methods for the current site. It is generally used for backward compatibility while supporting new access protocols such as QUIC. If the domain enables HTTP/3 (QUIC) access, EdgeOne will add this header to the HTTP response by default. For more information, please refer to Alt-Svc.

EO-LOG-UUID

The `EO-LOG-UUID` header carries the unique identifier of the request. When an access exception occurs, the header is used for troubleshooting based on the full-linkage logs of user requests.

Example: `EO-LOG-UUID: 4105283880544427145`.