Tencent Cloud

# Tencent Cloud EdgeOne

# Rule Engine

# Product Documentation

# Contents

# Rule Engine

## Overview

Last updated：2024-08-01 21:32:16

## Overview

The rule engine is designed to meet more flexible and fine-grained business requirements through a rich rule language. You can customize the match type as needed and apply it to the corresponding operations. Compared to the configuration of site acceleration, the priority of the rule engine is higher, meaning that the custom policies created by the rule engine will override the configurations of site acceleration.

## Use Cases

Provide custom configurations based on different conditions (subdomain name, path and file extension) when site-level configuration in **Site Acceleration** cannot meet your needs.
Provide basic features (caching and HTTPS) and acceleration features (custom cache key, URL rewrite and HTTP header modification).

## Key Terms

| Term | Description |
| --- | --- |
| Rule | It defines specific types of requests and the applicable operations. |
| Conditional Expression | It defines the logics that identify the requests. The followings are supported.<br>IF Note 1<br>ELSE IF<br>ELSE |
| Matching Condition | It defines the criteria that identifies the requests. The followings are included.<br>Matching type<br>Operator<br>Value |
| And/Or | Logical AND/OR, which can link multiple conditions. |
| Action | A wide range of feature configurations that can be applied to hit requests. |

**Note:**

Note 1:

An IF statement can be nested inside another IF statement, indicating that the nested one will be executed only after the other is met.

# Rule Priorities

| Range | Description |
|---|---|
| Site Acceleration vs Rule Engine | If the same operation is configured for both site acceleration and the rule engine, the rule engine has a higher priority and is the final effective configuration. |
| Single rule in the rule engine | If there exist nested IF conditions within an IF statement, the execution of the embedded IF statement necessitates the fulfillment of the outermost IF condition.<br>In the event of multiple coequal IF conditions, they are executed in relative order from top to bottom. That is, if multiple rules are matched simultaneously, the operations of the lower rules will supersede those of the upper rules.<br>In the event that IF, Else IF, and Else coexist, upon satisfying any one of the IF or Else IF conditions, the corresponding operation will be executed and concluded, precluding further matching of other rules under the current IF condition. If none are met, operations will be executed in accordance with the Else rule. |
| Multiple rules in the rule engine | The rules are executed in relative order, from top to bottom.<br>Note: You can place general or coarse-grained rules at the top as the default configuration and request-specific or finer-grained rules at the bottom. |

**Note:**

There are two scenarios with special execution:

Token authentication will be executed first no matter where it is placed. If a request hits two rules, token authentication will be executed first, as other operations will be performed only after authentication is passed.

For operations with redirect logic, such as URL redirection and forced HTTPS, their execution method is Break. This means that if the same request encounters both a redirect operation and other operations, the other operations below will not be executed after the redirect operation is executed.

# Example of Rule Priorities

**Example One: Nested IF Conditions within IF Matches**

The current user's node cache TTL rule configuration is as follows, with multiple nested IF conditions present.



The caching behavior of the user-requested URL is activated as follows:
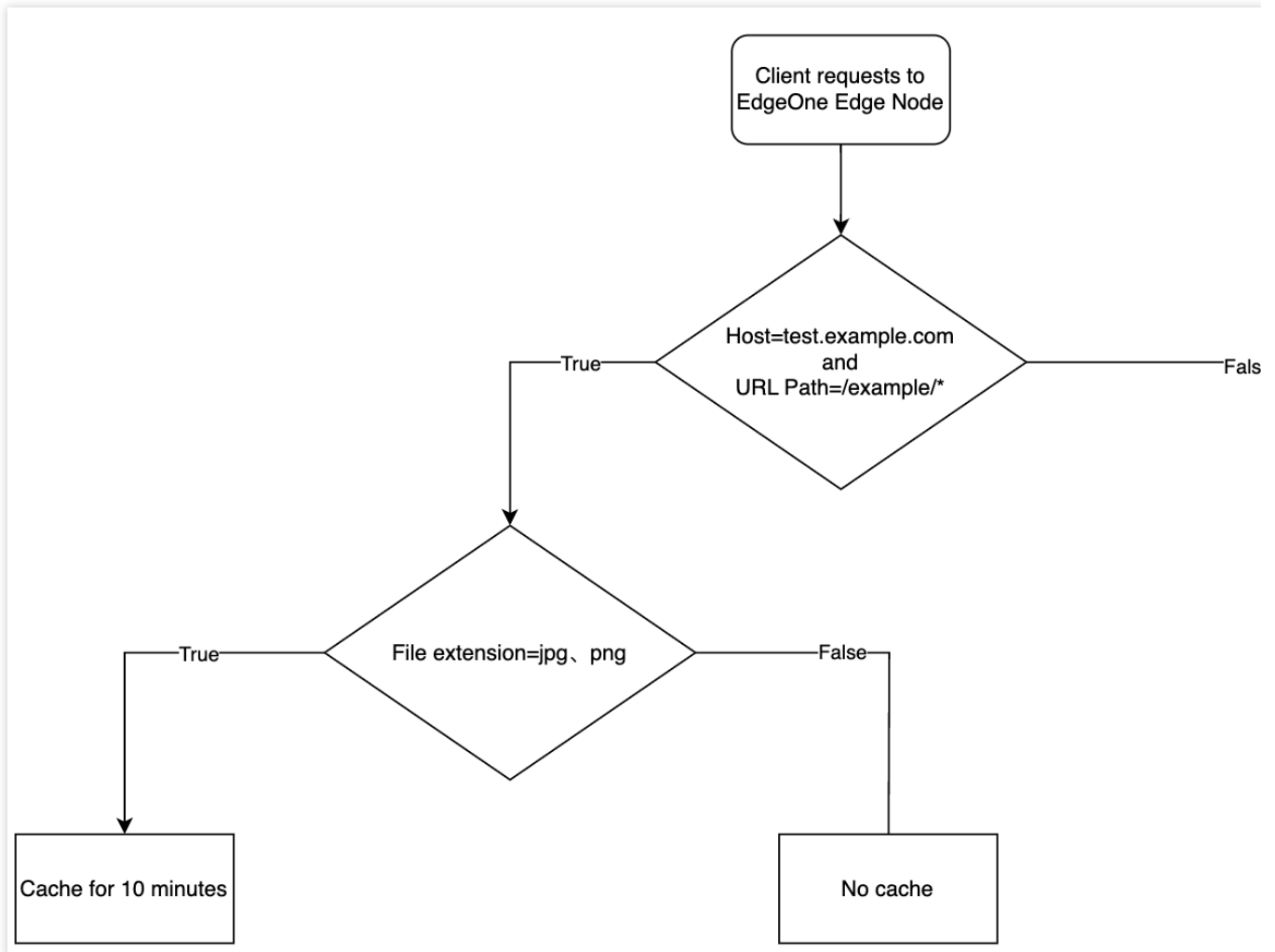
When the request URL is: `https://test.example.com/example/1.jpg` , the file is cached for a duration of 10 minutes.

When the request URL is: `https://test.example.com/example/1.mp4` , the file is not subjected to caching.

When the request URL is: `https://test.example.com/video/1.jpg` , it does not conform to the stipulated rule.

**Example Two: IF Condition Contains Multiple Parallel Else IF Matches**

The current user's node cache TTL rule configuration is depicted below, with multiple coequal Else IF conditions present.

The caching behavior of the user's requested URL will take effect as follows:

Host=test.example.com

True

False

File extension
=gif、png、bmp、jpeg、jpg

True → Cache for 7 days

False

File extension=aspx、jsp、php、asp

True → No cache

False

URL Path = /admin/*

True → No cache

False

Follow origin Cache-Control
When the headers do not exist,
follow EdgeOne Default Cache
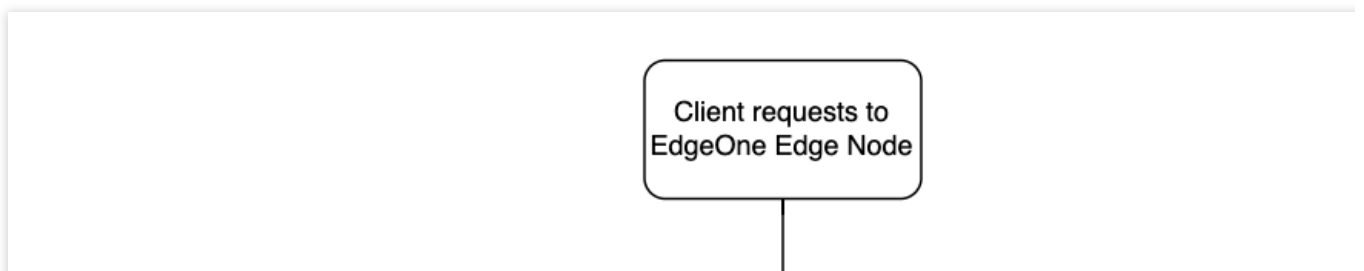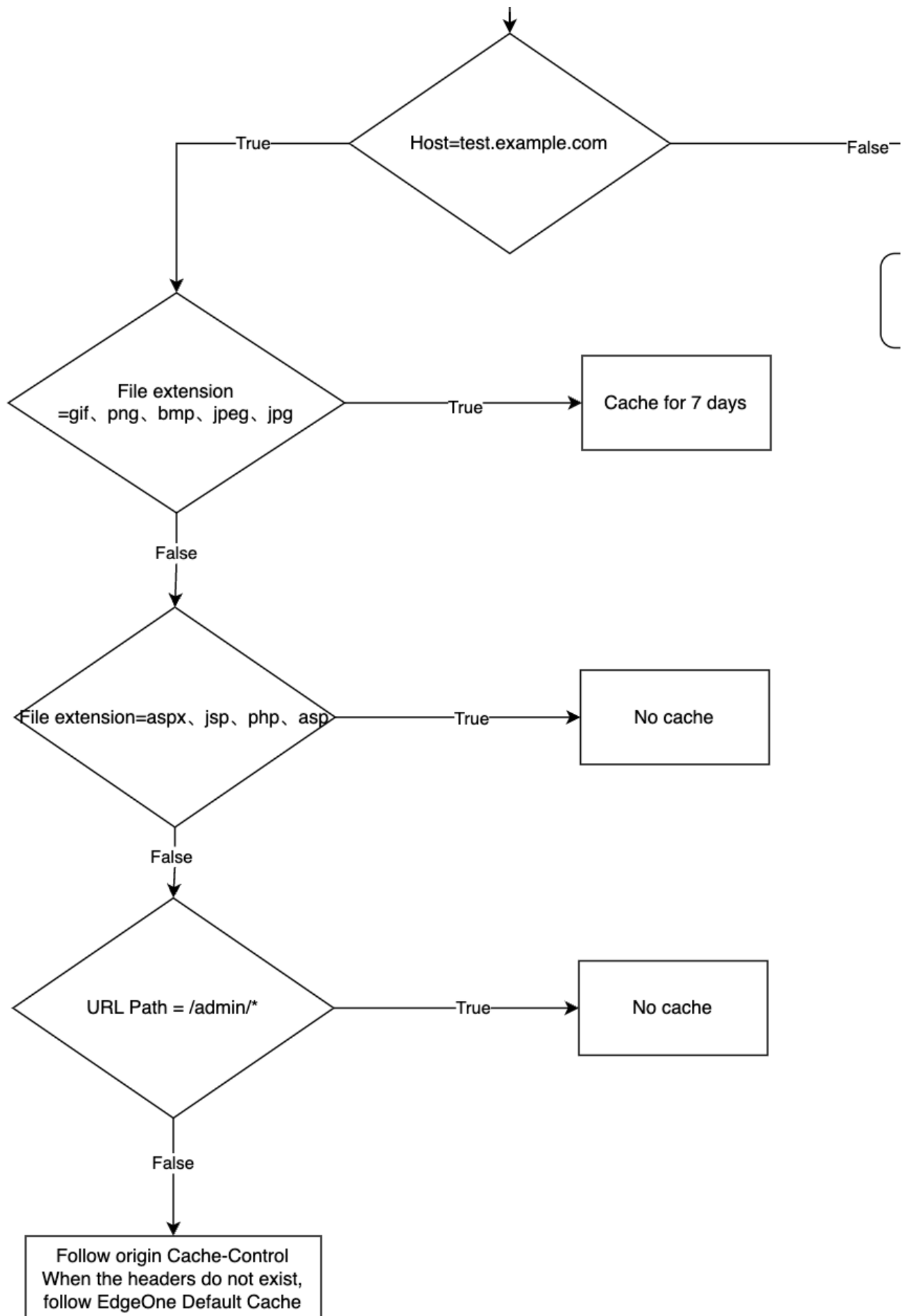
Rules

When the request URL is: `https://test.example.com/image/1.jpg` , the file is cached for a duration of 7 days.

When the request URL is: `https://test.example.com/index/1.jsp` , the file is not subjected to caching.

When the request URL is: `https://test.example.com/admin/1.php` , caching is not implemented.

## Example Three: Multiple Peer-Level IF Condition Matching

The current user's node cache TTL rule configuration is as follows. In the presence of multiple peer IF conditions, the effectiveness priority sequence of the subsequent conditions is the highest.



The caching behavior of the user's requested URL is activated as follows:

When the request URL is: `https://test.example.com/image/1.jpg` , the file is cached for a duration of 7 days.

When the request URL is: `https://test.example.com/admin/1.php` , the file is not subjected to caching.

When the request URL is: `https://test.exampel.com/admin/1.jpg` , the file is not subjected to caching.
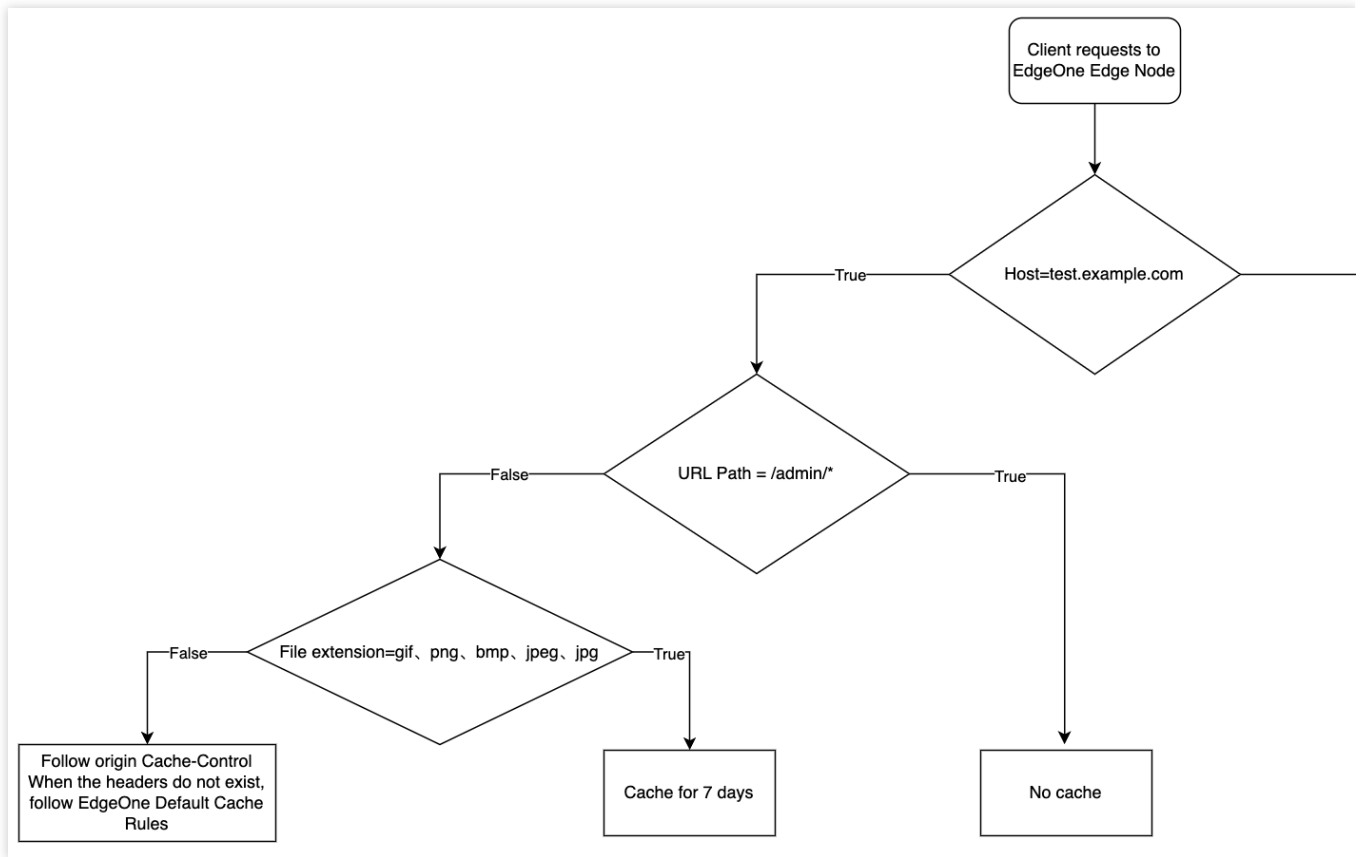
When the request URL is: `https://test.exampel.com/index/1.txt` , the file adheres to the source site's Cache-Control header settings. In the absence of such a header, it complies with the default caching policy of EdgeOne.

# Supported Matching Types and Actions

Last updated：2024-09-24 18:08:46

## Supported Matching Types

Supported matching types are listed in the following tables.

**Note**

1. URL Path and URL Full support wildcard match. If the `URL Path` is `/foo/*/bar` , both `/foo/example/bar` and `/foo/demo/bar` are valid values.

2. URL Path, URL Full, query string, file extension, file name and HTTP request header support enabling ignoring case (it is disabled by default).

| Type | Description | Sample values |
|------|-------------|---------------|
| HOST | Request Host | `www.example.com` |
| URL Path | Request URL path | If you need to match the `/example/foo/bar` path, then select the operator `Equal to` and enter the value `/example/foo/bar` .<br>If you need to match the `/example` directory and all files under the directory, then select the operator `Equal to` and enter the value `/example/*` . |
| URL Full | Full request URL, including the protocol, domain name, URL path, and query string. | If you need to fully match the `https://www.example.com/foo.jpg` path, then select the operator `Equal to` and enter the value `https://www.example.com/foo.jpg` .<br>If you need to match a path consisting of `https://www.example.com/foo.jpg` and a query string, then select the operator `Regular expression matching` and enter the value `https://www.example.com/foo\\.jpg($\|\\?)` .<br>If you need to match a path consisting of `https://www.example.com/foo.jpg` and the query string `test` , then select the operator `Regular expression matching` and enter the value `https://www.example.com/foo\\.jpg($\|\\?.*test=)` . |
| Query string | Query string in the request URL | Parameter name: key<br>Parameter value: value |

| File extension | File extension (file extension) of the request content | jpg, png, css |
|---|---|---|
| File name | File name of the request content | foo.txt |
| HTTP request header | HTTP request header | HTTP request header name: name<br>HTTP request header value: value |
| Client geo location | Country/region of the client IP | United States |
| Request protocol | Requested protocol type | HTTPS or HTTP |
| Client IP address | Client IP of the request | You can enter an IP address or IP address range.<br>IPv4: 192.168.0.0 or 192.168.0.0/24, where 0.0.0.0/0 matches all IPv4 addresses.<br>IPv6: 2001:db8:1234::1 or 2001:db8::/32, where ::/0 matches all IPv6 addresses. |
| All | Any site request | N/A |

## Operators

| Type | Description |
|---|---|
| Equal to | The request is equal to a specified value (value of the matching type). |
| Not equal to | The request is not equal to a specified value (value of the matching type). |
| Exist | A specified value exists in the request (HTTP header name or query parameter name). |
| Not exist | A specified value does not exist in the request (HTTP header name or query parameter name). |
| Regular expression matching | URL Path and URL Full support Google RE2 regular expression matching. |

## Supported Actions

Actions refer to a series of feature configurations performed after the requests hit the conditions. The supported actions and matching types are listed in the following tables.

## Cache configuration

| Action | Description | Supported Matching Types |
|---|---|---|
| Node Cache TTL | By configuring the cache TTL, you can optimize node cache to improve resource loading and update resources in a timely manner. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension |
| Browser Cache TTL | By adjusting the cache period of resources in browsers, you can optimize the browser cache and increase the loading speed of the requested resources. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>Client geo location<br>Client IP |
| Custom Cache Key | A cache key can be customized to suit your needs by setting the query string, HTTP header and URL case, so that requested resources can be loaded faster. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Status Code Cache TTL | You can specify a TTL period for origin response status codes, allowing the node to directly respond with non-2XX codes. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string |
| Cache Prefresh | Cached resources are validated via origin-pull before expiration, so that your site can respond to requests more rapidly. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension |
| Offline Cache | After offline caching is enabled, when | HOST |

| | your origin fails, and resources cannot be pulled through origin-pull normally, resources cached on nodes (even expired resources) can be used until the origin recovers. | URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| --- | --- | --- |

## Network optimization

| Action | Description | Supported Matching Types |
| --- | --- | --- |
| HTTP/2 | HTTP/2 (HTTP 2.0) requests are supported to accelerate sites and improve the web performance. | HOST |
| HTTP/3 (QUIC) | HTTP/3 (QUIC) requests are supported. HTTP/3 (QUIC) is used to accelerate site requests and improve data transfer efficiency and security. | HOST |
| WebSocket | EdgeOne supports the WebSocket protocol that allows the server to proactively send data to the client. | HOST |
| Maximum Upload Size | The maximum upload size is the maximum data volume that can be uploaded in a single client request. You can restrict it to improve the data transfer efficiency and optimize the network transfer. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header |
| Smart Compression | Smart Compression can automatically compress the resources to Gzip/Brotli files to reduce the files size and shorten the resource loading time. | HOST |
| Smart Acceleration | Smart acceleration refers to smart dynamic routing acceleration. After this feature is enabled, it will detect the node network latency in real time and use the smart algorithm to select the optimal transfer path, so as to handle both static and dynamic client requests more quickly, | HOST |

| | | |
|---|---|---|
| | stably, and securely. Smart dynamic routing minimizes problems such as high network latency, connection errors, and request failures. | |
| HTTP/2 Origin-Pull | Request origin-pull over the HTTP/2 protocol is supported. | HOST |
| Upstream Timeout | You can reasonably set the origin-pull request timeout based on the network link conditions and the data processing capability of the origin server to ensure normal origin-pull as requested. The origin-pull timeout is defined as follows. If there is no data response from the origin server after a node initiates an origin-pull request, no matter how long the duration is, the node considers it a timeout and actively disconnects from the origin server. | HOST |

## HTTPS optimization

| Action | Description | Supported Matching Types |
|---|---|---|
| Forced HTTPS | You can use 301 or 302 redirect to redirect HTTP client requests to HTTPS requests and send them to EdgeOne. | HOST |
| HSTS Configuration | Force clients such as browsers to establish connections to edge nodes over HTTPS for global website encryption. | HOST |
| SSL/TLS Security Configuration | Configure the protocol version and Cipher suite that are allowed to use when the client shakes hands with the edge server TLS as needed. | HOST |
| OCSP Stapling | Pre-cached OCSP responses are sent at the time of TLS handshake to improve the efficiency. | HOST |
| Origin-Pull HTTPS | You can specify the protocol that EdgeOne uses in the origin-pull request. | HOST |

## Modifying HTTP header

| Action | Description | Supported Matching Types |
|---|---|---|
| Modifying HTTP Response Headers | You can customize, add, and delete headers in HTTP responses from nodes to clients. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>Client geo location<br>Client IP |
| Client IP Header | The custom header can carry the real client IP to the origin. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Client IP Geographical Location | The custom header can carry the geographical location information of the client IP to the origin. | HOST<br>Client geo location<br>Client IP |
| Modifying HTTP Origin-Pull Request Headers | You can customize, add, and delete headers in HTTP origin-pull requests from nodes to the origin. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Host Header Rewriting | Host header rewriting enables you to rewrite the host header to the actual origin domain when the origin domain is different from the acceleration domain in the load balancing task. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP<br>Request protocol |

## Advanced configuration

| Action | Description | Supported Matching Types |
|---|---|---|
| Access URL Redirection | A node redirects the URL requested by the client to the destination URL based on the response status code. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Token Authentication | As an access control policy, token authentication supports creating rules to validate access and filter out unauthorized access requests. This effectively prevents your site resources from being maliciously hotlinked and thus protects your business. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header |
| Modify Origin | Configuration of primary and secondary sources, separate Path, and separate region Rules for complex origin-pull strategies. | HOST + Any matching type below：<br>URL Path<br>Client country/region<br>HTTP Request Header<br>Query string<br>File extension |
| Origin-Pull URL Rewrite | Based on specified rules, this feature rewrites the user request URL received by the node to the destination URL when the node sends the request to the origin server, which doesn't affect the node cache key. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Controlling Origin-pull Requests | You can specify which part of the query string and Cookie to be included in the request when it's forwarded to the origin. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header |

| | | Client geo location<br>Client IP |
|---|---|---|
| Redirect Following During Origin-Pull | When origin-pull is requested, the redirect will be based on the 302/301 status code of the origin server, and you can specify the maximum number of redirects (which is 3 by default and can be 1-5). | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Custom Error Page | You can redirect requests to a custom error page for specific error status codes returned by the origin. The redirection is performed when a 302 is returned. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Client IP |
| Range GETs | Range GETs can be enabled to reduce the consumption of large file origin-pulls and response time. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header |
| HTTP response | The HTTP response feature is supported by the rule engine. When the corresponding conditions are met, the EdgeOne node directly responds with the specified status codes and page content. | HOST<br>URL FULL<br>URL Path<br>File name<br>File extension<br>Query string<br>HTTP Request Header<br>Client geo location<br>Request protocol<br>Client IP<br>All (any site request) |

# Rule Management

Last updated：2024-01-25 11:28:53

The console supports a series of icons and buttons to manage rules, for example, sorting, copying, enabling, and disabling rules, as follows.

| Icon/Button | Description |
| --- | --- |
| | Drags a rule up or down. |
| | Pins a rule to the top. |
| | Pins a rule to the bottom. |
| | Edits a rule. |
| | Creates the same rule as the copied rule. |
| | Deletes a rule. |
| | Searches for a rule by rule name or keyword. |

| | |
|---|---|
|  | |
|  | Rule status<br>Enable: Publishes a rule to the production environment.<br>Disable: Saves a rule but does not publish it to the production environment. |
|  | Saves a rule but does not publish it to the production environment. |
|  | Saves and publishes a rule to the production environment. |
|  | If a single rule is complex and has multiple IF statements, you can add comments to them. Then, the rule navigation will be automatically generated on the right of the rule content to simplify viewing and locating. |

# variables

Last updated：2024-08-01 21:32:16

## Introduction

The variables of the rule engine allow you to dynamically extract and process data within request. These variables can not only store static values but also use for specific fields or information in the request, the value of which may change when processing each request. For example: the `http.request.host` variable, which can extract the `hostname` in each HTTP request. This capability enables the rule engine to handle more complex business logic.

## Content

| Name | Type | Description | Example |
| --- | --- | --- | --- |
| http.request.scheme | String | Client request protocol | http<br>https |
| http.request.zone | String | Site name | example.com |
| http.request.zoneid | String | Site ID | zone-2c2r77pc3796 |
| http.request.host | String | Hostname in the client request URI | www.example.com |
| http.request.full_uri | String | Full URI of the client request (not including #fragment) | htt-ps://www.example.org/articles/index?section=539061&expand=comments |
| http.request.method | String | Client request HTTP method | GET |
| http.request.uri | String | Client request URI path and query string | /articles/index?section=539061&expand=comments |
| http.request.uri.path | String | Client request URI path | /articles/index |
| http.request.file_extension | String | File extension of the client request file | jpg |
| http.request.filename | String | Filename of the client request file | bot.txt |

| http.request.uri.query | String | The whole query string of the client request, not including the `?` separator | section=539061&expand=comments |
| --- | --- | --- | --- |
| http.request.headers["key"] | String | The header value of the specified header name "key" in the client request, "key" can be replaced with your specified name | https://developer.mozilla.org |
| http.request.uri.args["key"] | String | The parameter value of the specified parameter name "key" in the client query string, "key" can be replaced with your specified name | value |
| http.request.version | String | The version of the HTTP protocol used in the client request | HTTP/1.0<br>HTTP/1.1<br>HTTP/2<br>HTTP/3 |
| http.request.ip | String | Client TCP IP address, for example: `1.1.1.1` | 93.184.216.34 |
| http.request.ip.port | String | Client Port | 1028 |
| http.request.ip.city | String | City associated with the client IP address | San Francisco |
| http.request.ip.continent | String | Continent code associated with the client IP address | AF: Africa<br>AS: Asia<br>EU: Europe<br>NA: North America<br>SA: South America<br>OC: Oceania<br>AN: Antarctica |
| http.request.ip.country | String | 2-letter country code in ISO 3166-1 Alpha 2 format associated with the client IP address | GB, see more in ISO 3166-1 Alpha 2 |

# Use Case

1. The custom origin-pull request header carries the information of the country where the client IP address is located back to the origin.



2. Custom origin-pull request headers allow the origin server to collect and analyze which domains have been accelerated by Tencent's EdgeOne.



3. Custom Cross-Origin Request Policy: Allows cross-origin requests from domains specified in the Origin header of the request.