边缘安全加速平台 EO

站点加速

产品文档

腾讯云

文档目录

站点加速
概述

最近更新时间：2023-08-30 15:23:36

站点加速服务是专门针对 HTTP/HTTPS 等应用层协议的互联网内容分发加速服务，尤其适合用于网站、在线应用、流媒体等内容的分发。站点加速可通过丰富的功能配置，如缓存优化，文件优化，网络优化等，帮助您实现更高效、更稳定的内容分发，提升业务用户的体验满意度，从而增强您的网站、应用或其他在线服务的竞争力。
您在接入 EdgeOne 安全加速服务后，您可以自定义配置以下站点加速服务：

分类	功能	使用场景	默认配置
访问控制	Token 鉴权	由 EdgeOne 节点对客户端访问进行鉴权，防止盗链行为。	关闭
智能加速		对动态资源访问请求进行智能加速，提升访问速度。	关闭
缓存配置	Vary 特性	根据源站响应的 Vary 头部来进行区分节点缓存内容，根据指定头部内容响应对应资源。	平台默认全局开启
	自定义 Cache Key	自定义文件在节点内的缓存键（Cache Key），决定文件的缓存规则以及是否命中缓存。	默认不忽略字符串缓存，详情请参见 缓存键（Cache Key）介绍
	节点缓存 TTL	自定义文件在节点内的缓存时间，决定文件是否要缓存在 EdgeOne 节点内以及相应缓存时间。	请参考 EdgeOne 节点默认缓存规则
	状态码缓存 TTL	自定义错误状态码在 EdgeOne 节点内的缓存时间，降低在源站异常时的回源请求压力。	404 缓存10s，其它异常状态码不缓存
	浏览器缓存 TTL	控制浏览器是否缓存文件，减少访问请求流量。	默认为遵循源站 `Cache-Control`
	离线缓存	控制 EdgeOne 是否在源站异常时使用已缓存（即使文件已过期）的文件响应客户端请求，保障在源站异常时提供正常服务响应。	开启
	缓存预刷新	提前让 EdgeOne 节点在缓存过期前回源验证文件有效性，保证节点内已缓存最新资源文件。	关闭

文件优化	智能压缩	由 EdgeOne 节点进行 Gzip 压缩或 Brotil 压缩以减少文件传输大小。	开启 Gzip 压缩和 Brotil 压缩
媒体处理	图片处理	由 EdgeOne 节点自动对图片进行缩放和格式转换操作，无需源站存储多种大小和格式图片副本，减少图片存储成本。	关闭
网络优化	HTTP /2	要求 EdgeOne 节点支持 HTTP/2 访问。	开启
	HTTP/3	要求 EdgeOne 节点支持 HTTP/3 访问。	关闭
	IPv6 访问	要求 EdgeOne 节点支持 IPv6 访问。	关闭
	最大上传大小	EdgeOne 节点限制用户可上传的最大文件大小，防止用户恶意上传大型文件，减少传输流量。	开启，最大限制800MB
	WebSocket	要求 EdgeOne 节点支持 WebSocket 传输并配置超时连接时长。	关闭
	客户端 IP 头部	指定 EdgeOne 节点回源时使用指定请求头部携带客户端 IP。	关闭
	客户端 IP 地理位置头部	指定 EdgeOne 节点回源时使用指定请求头部携带客户端 IP 地理位置。	关闭
	开启 gRPC	要求 EdgeOne 节点支持 gRPC 访问。	关闭
URL 重写	访问 URL 重写	自定义修改重定向用户访问的 URL 地址。	N/A
	回源 URL 重写	自定义修改重定向用户需回源请求的 URL 地址。	N/A
修改头部	修改 HTTP 节点响应头	自定义修改 EdgeOne 节点响应用户请求时携带的 HTTP 头部。	N/A
	修改 HTTP 回源请求头	自定义修改 EdgeOne 节点回源时携带的 HTTP 头部。	N/A
自定义错误页面		自定义修改源站在响应4xx、5xx状态码时客户端显示的错误页面。	N/A

访问控制
Token 鉴权

最近更新时间：2024-02-26 10:00:26

功能简介

Token 鉴权是一种实现原理简单、可靠性高的访问控制策略，通过配置鉴权规则进行 URL 访问校验，可有效防止站点资源被恶意盗刷。该功能的使用需要客户端和 EdgeOne 配合，客户端负责发起加密的 URL 请求，EdgeOne 负责根据预先设定的规则对 URL 进行合法性验证。

功能原理

Token 鉴权的实现主要由以下两部分配合：
客户端：根据鉴权规则（包括鉴权算法、密钥）发起鉴权 URL 的请求。
EdgeOne 节点：对鉴权 URL 中的鉴权信息（md5 字符串+时间戳）进行校验。当校验通过时，该访问请求才会被视为合法请求，节点正常响应。若校验失败，则节点拒绝该访问，直接返回 403。

Token 鉴权 URL 生成和校验工具

EdgeOne 提供了 Token 鉴权 URL 的生成工具和校验工具，开发者可以使用该工具快速准确地生成和校验符合要求的防盗链 URL。

操作步骤

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，设置触发该规则的匹配条件。
5. 单击**操作** > **选择框**，在弹出的操作列表内，选择操作为**Token 鉴权**，参数配置说明如下：

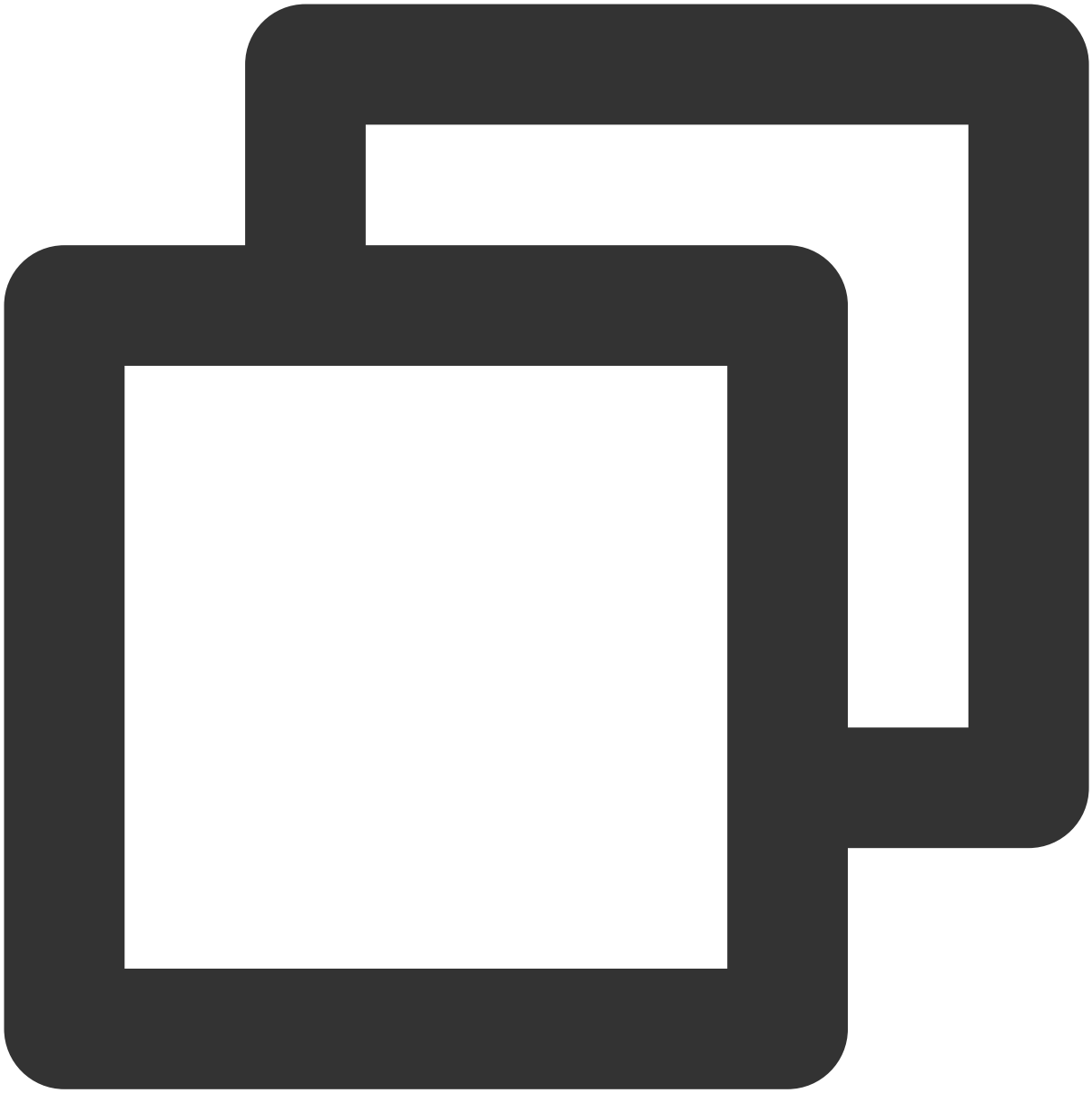
配置项	说明
鉴权方式	目前支持 4 种鉴权签名计算方式，请您根据访问 URL 格式选择合适的方式。详情请参见鉴权方式。

密钥（主）	主用密码，由 6-40 位大小写英文字母或数字组成。
密钥（备）	备用密码，由 6-40 位大小写英文字母或数字组成。
鉴权参数	鉴权参数名称，节点将校验此参数名对应的值。由 1 - 100位大小写字母、数字或下划线组成。
有效时长	鉴权 URL 的有效时长，单位：秒（1 - 630720000），用于判断客户端访问请求是否过期： 若当前时间超过“timestamp + 有效时长”时间，则为过期请求，直接返回 403。 若当前时间未超过“timestamp+ 有效时长”时间，则请求未过期，继续校验 md5 字符串。

鉴权方式

方式 A

鉴权 URL 格式



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

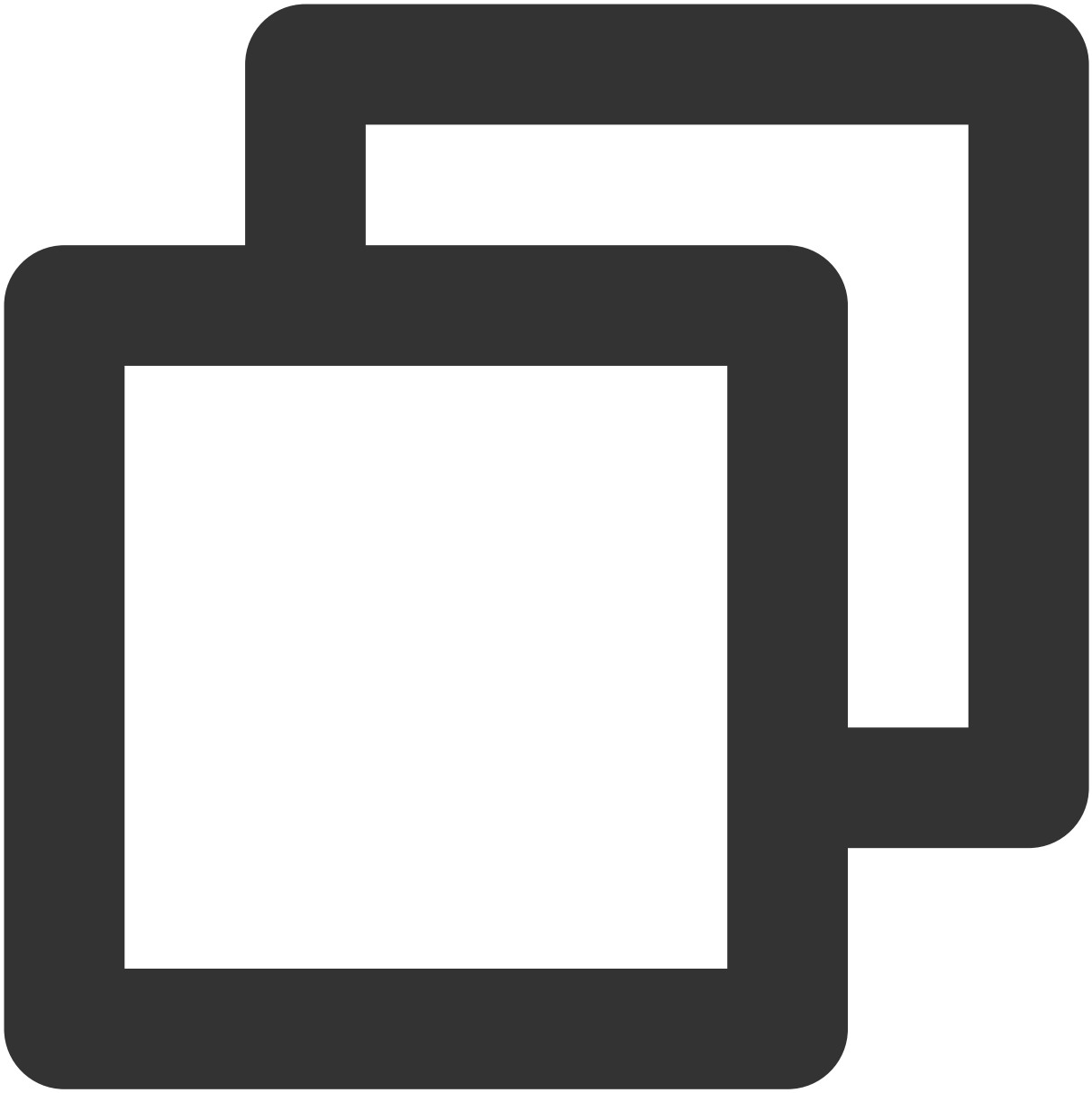
鉴权字段说明

字段	说明
Hostname	站点加速域名。
Path	资源访问路径，鉴权时需以 `/` 开头。

sign	自定义设置的鉴权参数名称。
timestamp	十进制整型正数的 Unix 时间戳，是从 UTC 时间1970年01月01日00时00分00秒到现在的总秒数，其定义与所在时区无关。
rand	0 - 100位随机字符串，由大小写字母与数字组成。
uid	用户 ID，暂未使用，默认为0。
md5hash	通过 MD5 算法计算出的固定长度为32位的字符串： 算法：MD5（Path-timestamp-rand-uid-密钥）。 鉴权逻辑：若请求未过期，则节点比较此字符串值与请求 URL 中携带的 `md5hash` 值：两值相同，鉴权通过，响应请求；两值不同，鉴权失败，返回403。

方式 B

鉴权 URL 格式



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

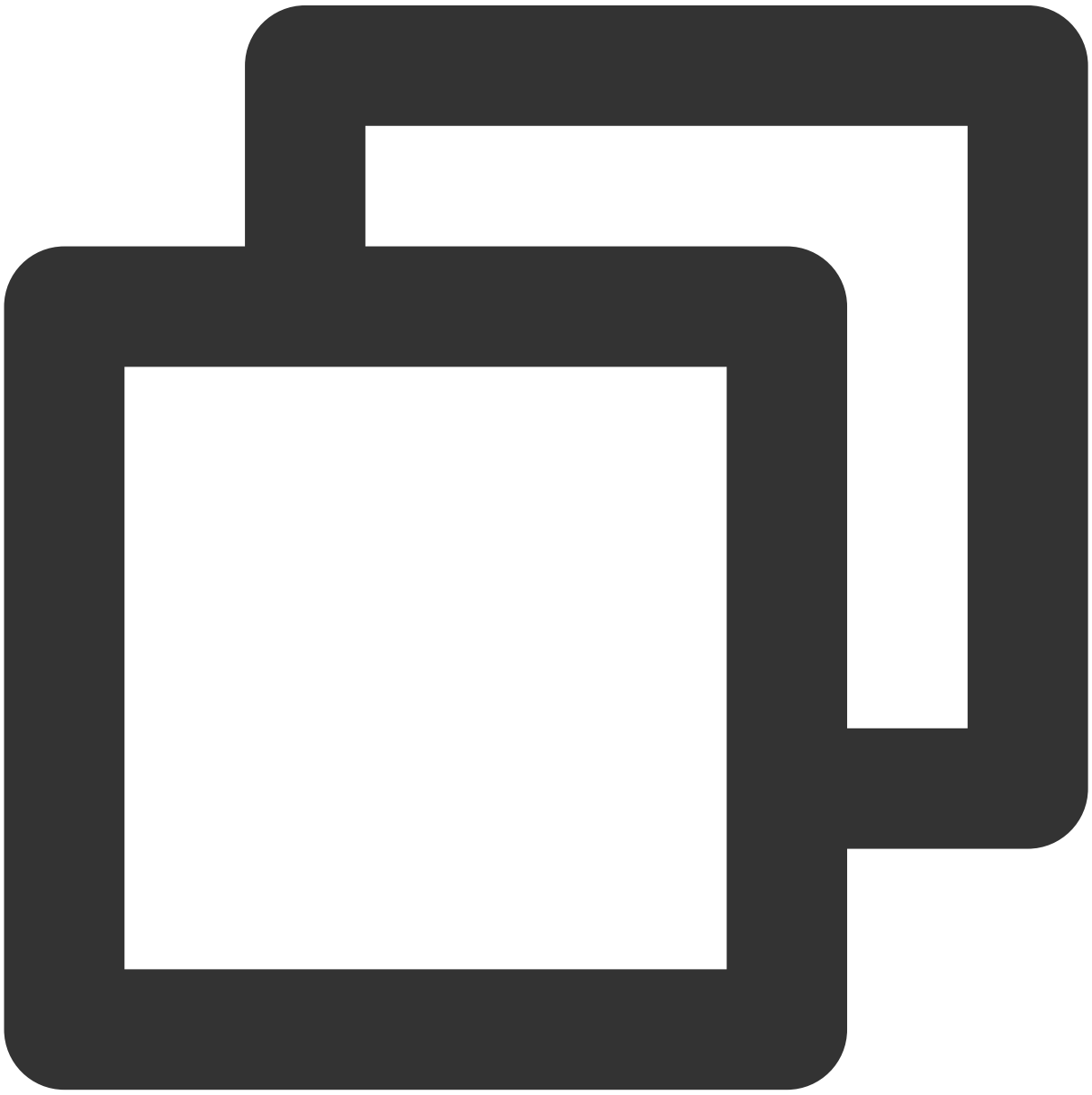
参数说明

字段	说明
Hostname	站点加速域名。
Path	资源访问路径，鉴权时需以 `/` 开头。

timestamp	时间戳参数。 格式：YYYYMMDDHHMM，UTC+8 时间，例如201807301000。
md5hash	通过 MD5 算法计算出的固定长度为32位的字符串： 算法：MD5（密钥 + timestamp + Path）。 鉴权逻辑：若请求未过期，则节点比较此字符串值与请求 URL 中携带的 `md5hash` 值：两值相同，鉴权通过，响应请求；两值不同，鉴权失败，返回403。

方式 C

鉴权 URL 格式



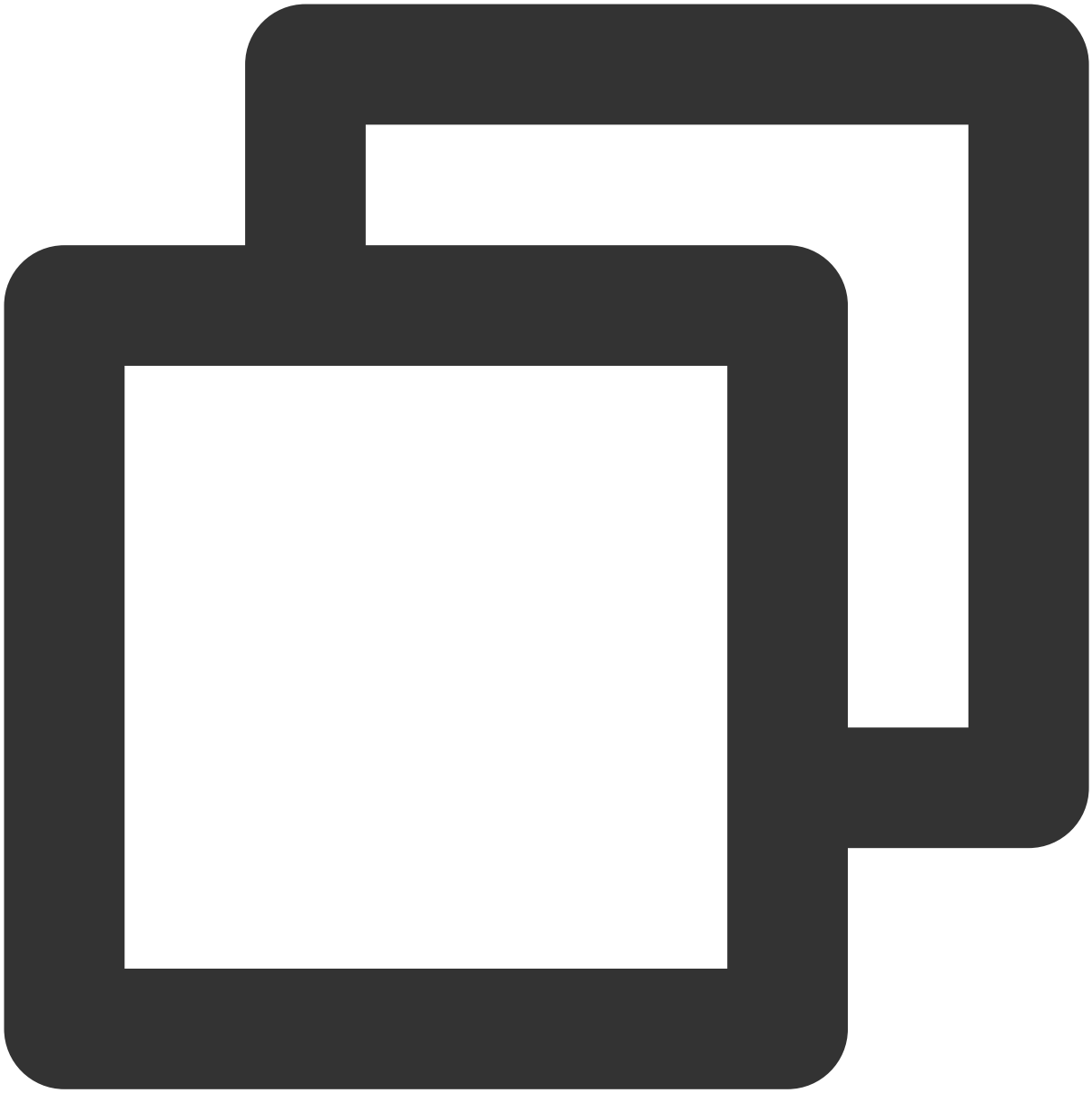
```
http://Hostname/md5hash/timestamp/Filename
```

参数说明

字段	说明
Hostname	站点加速域名。
Path	资源访问路径，鉴权时需以 `/` 开头。
timestamp	时间戳参数。 格式：十六进制整型正数的 Unix 时间戳，是从 UTC 时间1970年01月01日00时00分00秒到现在的总秒数，其定义与所在时区无关。
md5hash	通过 MD5 算法计算出的固定长度为32位的字符串： 算法：MD5（密钥+ Path + timestamp ）。注：计算时，十六进制的 timestamp 需过滤掉进制数标识0x。 鉴权逻辑：若请求未过期，则节点比较此字符串值与请求 URL 中携带的 `md5hash` 值：两值相同，鉴权通过，响应请求；两值不同，鉴权失败，返回403。

方式 D

鉴权 URL 格式



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

参数说明

字段	说明
Hostname	站点加速域名。
Path	资源访问路径，鉴权时 需以 `/` 开头。

sign	自定义设置的鉴权参数名称。
t	自定义设置的时间戳参数名称
timestamp	时间戳参数。 格式：十进制整型正数的 Unix 时间戳，是从 UTC 时间1970年01月01日00时00分00秒到现在的总秒数，其定义与所在时区无关；或十六进制整型正数的 Unix 时间戳，是从 UTC 时间1970年01月01日00时00分00秒到现在的总秒数，其定义与所在时区无关。
md5hash	通过 MD5 算法计算出的固定长度为32位的字符串： 算法：MD5（密钥 + Path+ timestamp）。注：计算时，十六进制的 timestamp 需过滤掉进制数标识0x。 鉴权逻辑：若请求未过期，则节点比较此字符串值与请求 URL 中携带的 `md5hash` 值：两值相同，鉴权通过，响应请求；两值不同，鉴权失败，返回403。

配置示例

假设请求 `http://www.example.com/test.jpg` 符合鉴权方式 A，则可配置如下：

获取鉴权参数

Path： `/foo.jpg` 。

timestamp：服务端生成鉴权URL的时间为2022年03月15日10:30:32（UTC+8），转换为十进制的整形数值为 `1647311432` 。

rand：生成随机数为 `J0ehJ1Gegyia2nD2HstLvw` 。

uid： `0` 。

密钥： `3C9mxSGzc8ZadmGNzE` 。

md5hash：MD5（Path-timestamp-rand-uid-密钥）= MD5（ `/foo.jpg` - `1647311432` - `J0ehJ1Gegyia2nD2HstLvw` - `0` - `3C9mxSGzc8ZadmGNzE` ）= ecce3150cbdaac83b116d937777ca77f。

鉴权 URL

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

节点鉴权

当节点服务器接受到客户端通过加密 URL 发出的请求时，解析出 URL 中的 timestamp 参数，加上配置的有效时长“1秒”，与当前时间比较：
若当前时间超过”timestamp + 有效时长“时间，则为过期请求，直接返回403。
若当前时间未超过”timestamp + 有效时长“时间，则请求未过期，继续第3步。
节点服务器通过获取的鉴权参数计算 `md5hash` 值，与请求 URL 中携带的 `md5hash` 值做比较：两值相同，鉴权通过，响应请求；两值不同，鉴权失败，返回403。

注意事项

1. 鉴权通过后，节点会自动忽略 URL 中鉴权相关的参数进行缓存，从而提高缓存命中率，减少回源量。
2. URL 中不能包含中文。

智能加速

最近更新时间：2023-10-11 10:11:02

功能简介

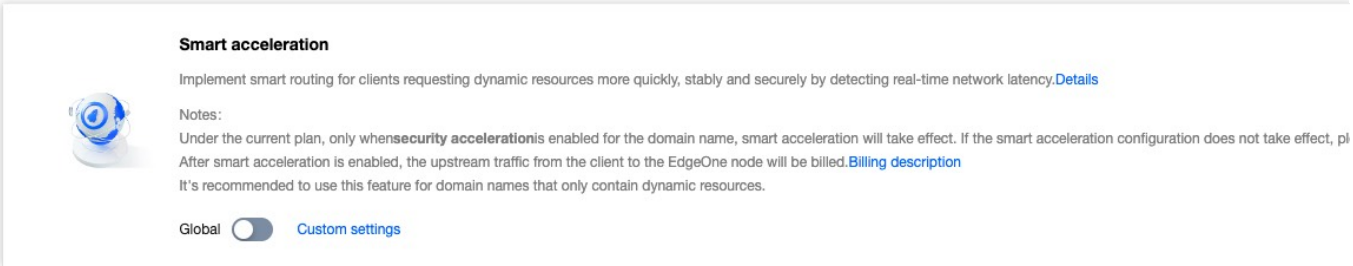
当您的站点提供的服务为**纯动态内容**服务或者**动静态内容混合**时，其中用户对动态内容请求需要回源请求来根据用户响应不同的资源内容，此时可能因为客户端所处地域、运营商的差异，网络环境错综复杂，在跨地区、跨运营商访问时，导致用户访问请求慢、丢包率高等情况。
智能加速能实时调整和优化网络路径，启用此功能后，EdgeOne 将实时检测节点网络延迟，通过智能算法选择最佳访问路径，并根据实时的网络状况，动态调整资源分配和利用，来帮助提升用户访问体验、保障业务连续性。

计费说明

此功能为收费服务，开启智能加速后，在原计费项的基础上，客户端用户与 EdgeOne 节点的上行流量将同时计入安全加速流量费用，同时会根据业务请求数收取增值服务费。详情请参见 计费概述。

场景一：站点所有域名开启智能加速

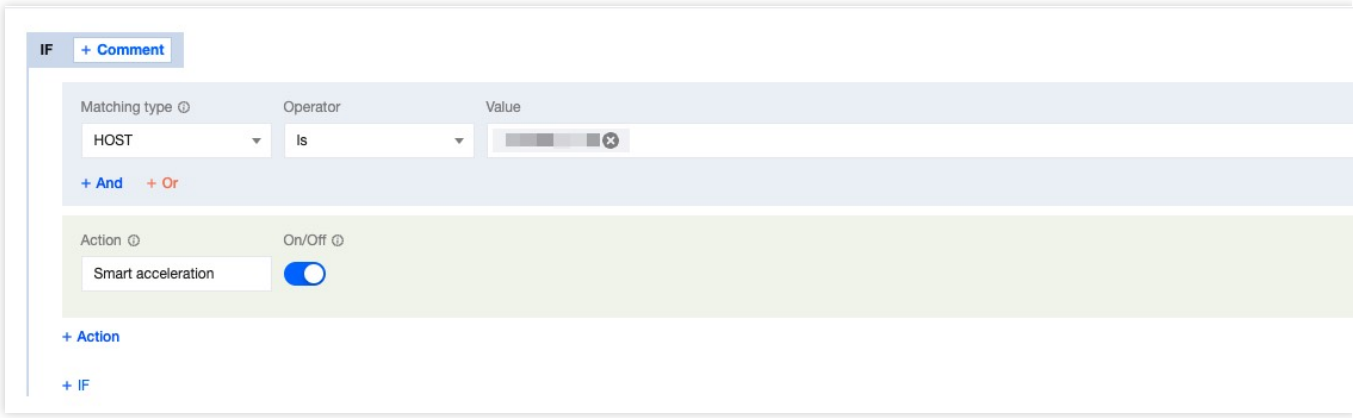
若您的站点均为动态资源或者为动静结合资源，需要对整个接入站点开启智能加速，可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速** > **智能加速**，进入智能加速详情页面。
3. 找到智能加速配置卡片，默认为关闭状态，单击**开关**可配置开启/关闭。



场景二：针对指定域名开启智能加速

若您的站点下只有某个域名为纯动态资源或者动静结合资源，需要单独针对指定域名开启智能加速，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，选择 Host 匹配类型以匹配指定域名的请求。
5. 单击**操作** > **选择框**，在弹出的操作列表内，选择操作为**智能加速**，单击**开关**开启/关闭。



6. 单击**保存并发布**，即可完成该规则配置。

相关参考

什么是静态资源，什么是动态资源？

静态资源：用户多次访问某一资源，返回**相同**内容。例如：图片、视频、软件安装包、压缩包文件、CSS、JavaScript 文件等不经常改变的内容。
动态资源：用户多次访问某一资源，返回**不同**内容。即需要根据用户请求实时更新、用户间交互等动态内容。例如：API 接口、`jsp`、`asp`、`php`、`perl` 和 `cgi` 格式文件等。

缓存配置

概述

最近更新时间：2023-09-14 10:45:56

当您的站点接入 EdgeOne 后，EdgeOne 边缘节点将根据缓存配置的规则来决定是否缓存客户端请求响应的资源文件，边缘节点缓存该文件后，当有其他用户发起相同的文件请求时，可由 EdgeOne 边缘节点直接响应，可有效避免长链路回源情况，以更快的速度为用户响应最新的文件请求。

您可以根据以下使用场景自定义您的站点缓存：

分类	使用场景	功能
自定义 EdgeOne 节点缓存规则	站点/域名下的文件资源类型多样，需要自定义各类资源在节点内的缓存时间，确保用户访问最新文件的同时，减少回源请求量； 站点/域名下包含动态资源，需要避免该资源内容被缓存。	节点缓存 TTL
	请求同一路径文件时，不同的 URL 携带的参数、请求头等内容将指向不同的文件； 请求同一路径文件时，URL 所携带参数、请求头等不影响文件版本，需要指向同一份缓存文件。	自定义 Cache Key
源站异常下缓存控制	在源站响应异常时，需要尽量保护源站不再受损，同时正常响应客户请求。	状态码缓存 TTL 离线缓存
控制浏览器缓存	希望进一步提升网页的加载速度，减少流量消耗，允许浏览器缓存一定时长的静态资源文件。	浏览器缓存 TTL
清除缓存	节点内缓存文件已过期或有违规资源被缓存时，清除节点内已缓存资源。	清除缓存
预热缓存	刚完成域名接入或文件更新时，需要将文件提前缓存到 EdgeOne 节点内，以提升加速效果，减少高峰期回源量。	预热缓存
缓存文件预刷新	针对有持续用户访问的文件，需要保障该文件在节点内持续有缓存，避免缓存过期后集中回源，可通过缓存预刷新来验证文件的有效性并刷新缓存时间。	缓存预刷新

如果您还需要了解更多的缓存规则，可以参考如下：

了解 EdgeOne 的内容缓存规则

缓存键（Cache Key）介绍

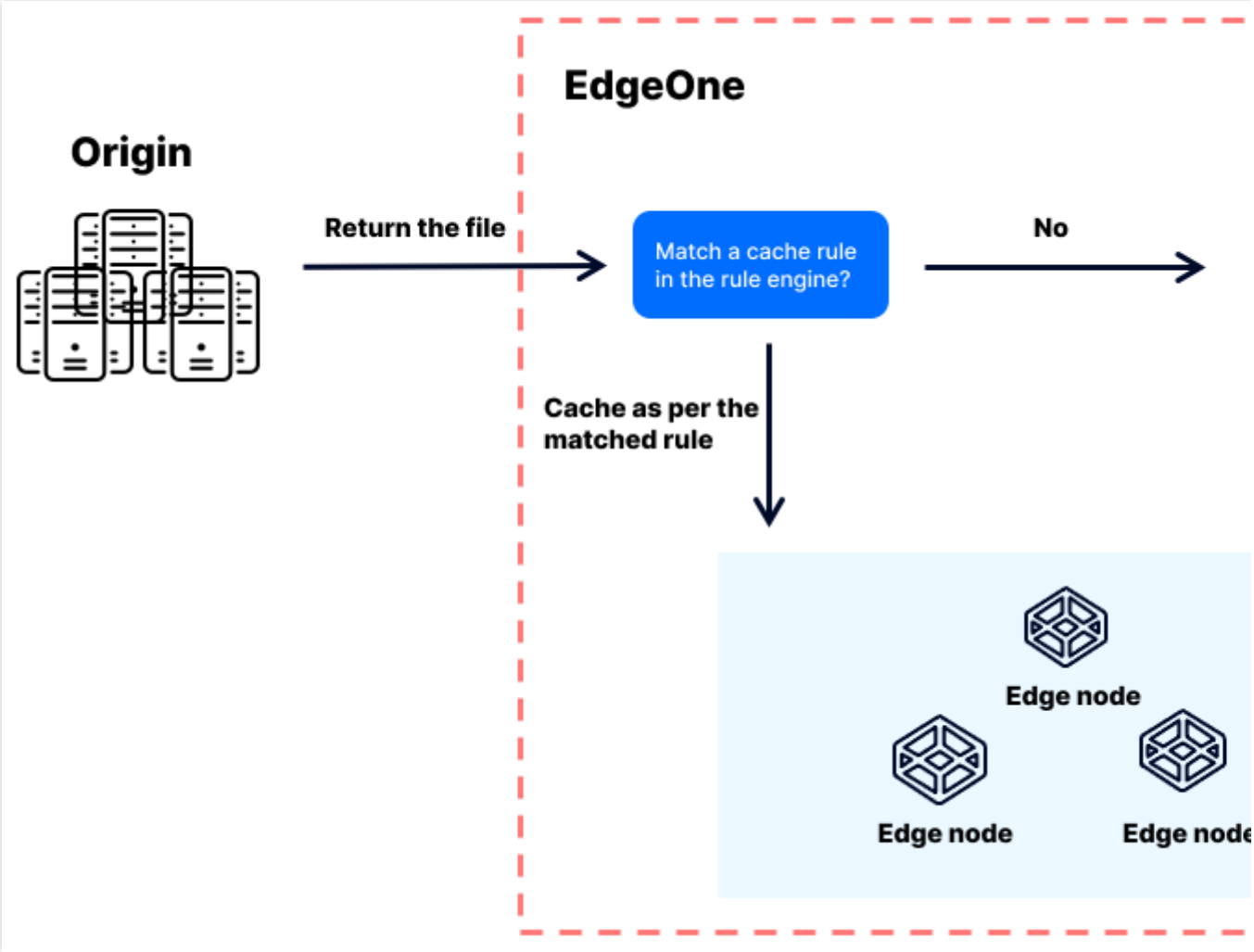
了解 Vary 特性的作用

EdgeOne 缓存规则介绍
EdgeOne 内容缓存规则

最近更新时间：2023-05-25 15:19:48

概述

当客户端向 EdgeOne 边缘节点发起 HTTP 请求后，节点将判断当前文件是否命中缓存，如未命中，则回源向源站发起请求获取最新文件，在源站正确响应文件后，EdgeOne 将根据用户设置的缓存规则结合平台默认缓存策略，对文件进行缓存。您可以通过查看 如何配置缓存规则 来了解如何自定义设置您的文件缓存规则。缓存规则配置后，将按照以下顺序匹配生效：



注意：

缓存规则仅在源站响应状态码为200、206的情况下生效，如果源站响应为404状态码，则节点将缓存该状态码10s，其他状态码均不缓存。

1. 缓存规则将优先匹配规则引擎内缓存规则，按照从上往下的优先级顺序进行匹配，最上方规则优先级最高，如该文件在规则引擎内匹配成功，则按照该缓存规则进行缓存。
2. 规则引擎内未匹配到相应的规则时，则按照站点加速内的全局节点缓存策略进行缓存，全局缓存策略默认为 EdgeOne 的默认缓存策略，您可以根据需求自定义修改。

缓存规则

EdgeOne 支持配置三种缓存策略配置，分别为：

默认缓存策略：遵循 EdgeOne 默认缓存策略，根据 HTTP 响应头内的 `Cache-Control` 及其他缓存头部来决定文件在节点内的缓存时间。

不缓存：通过规则引擎内指定文件不缓存，或全局文件不缓存，适用于动态文件或更新频繁的文件内容。

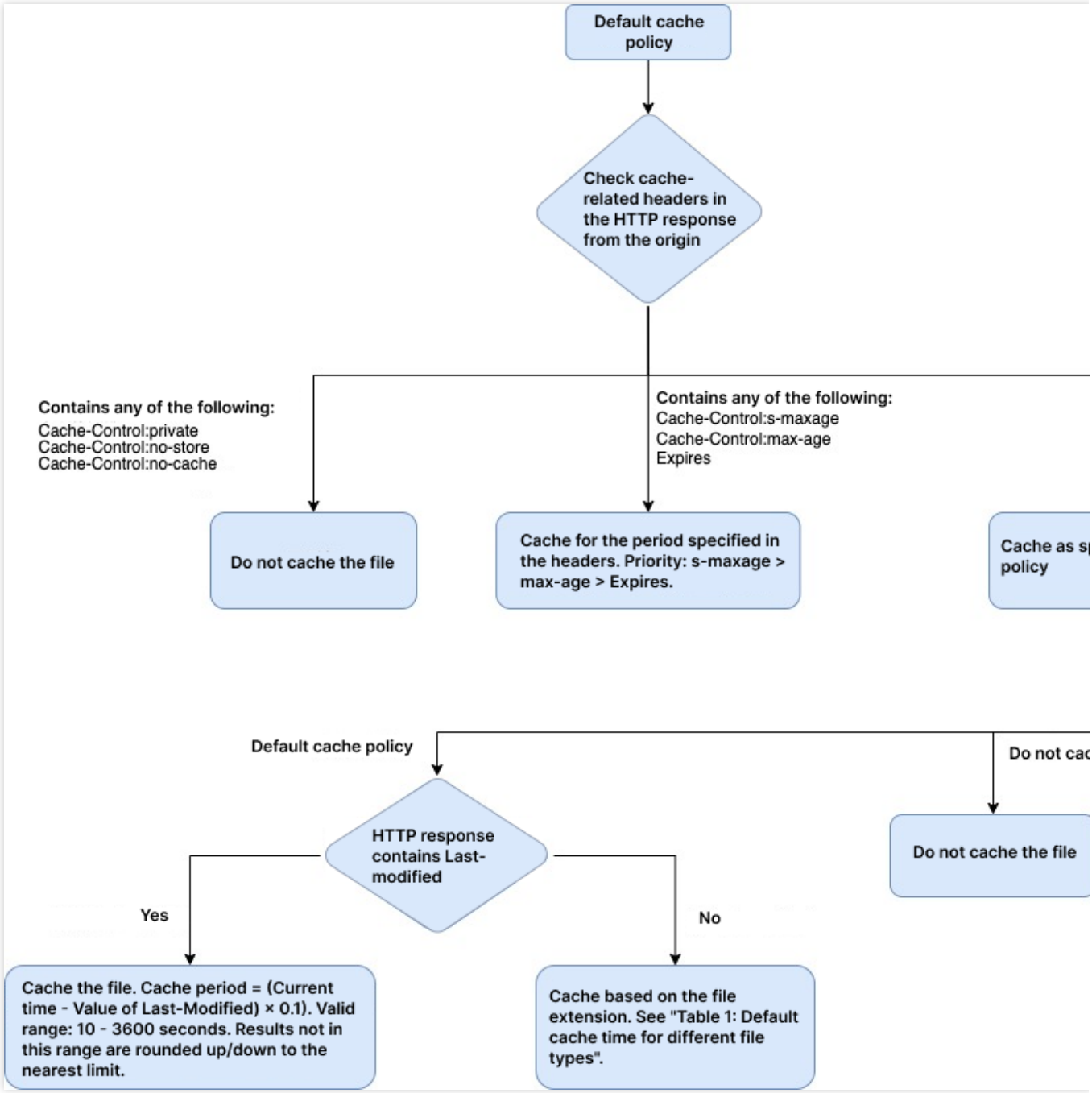
自定义缓存时间：按照自定义缓存时间缓存文件。

注意：

文件缓存在 EdgeOne 节点后，平台具有文件冷热淘汰机制，如果当前缓存文件长时间未有请求，则可能在未达到最大缓存时间时提前从节点缓存中删除。

默认缓存策略

EdgeOne 的默认缓存策略如下：



在默认缓存策略下，节点将根据源站是否携带缓存头部来控制该文件在节点的缓存动作及缓存时间，缓存规则如下：

1. 当 HTTP 响应头中，包含了以下任意的不缓存头部时，文件不缓存：

`Cache-Control:private`

`Cache-Control:no-store`

`Cache-Control:no-cache`

2. 当 HTTP 响应头中，包含以下任意一个缓存头部时，文件将按照缓存头部中设定的缓存时间进行缓存：

`Cache-Control:s-maxage`

`Cache-Control:max-age`

`Expires`

如果同时存在以上多个响应头，则缓存时间按照 s-maxage > max-age > Expires 的优先级顺序判断，按照优先级高的头部所设定时间缓存。

3. 当 HTTP 响应头不包含以上任意的缓存头部时，则会根据在规则中所配置的缓存行为执行：

默认缓存策略：

如果 HTTP 响应头内带有 `Last-Modified`，则缓存时间 = (当前时间 - `Last-Modified`) * 0.1，计算结果在10秒 ~ 3600秒及之间的，取计算结果时间；小于10秒的，按照10秒处理；大于3600秒的，按照3600秒处理。

如果 HTTP 响应头内无 `Last-Modified`，则依据文件后缀，按照平台默认缓存规则进行缓存，不同文件后缀的缓存时间如下：

表1：默认文件缓存时间

文件类型		后缀	缓存时间
动态文件		php、aspx、asp、jsp、do、dwr、cgi、fcgi、action、ashx、axd、json	不缓存
静态文件	图片	jpg、png、jpeg、webp、gif、heif、heic、kpg、ico	缓存2小时
	音视频	mp4、mp3、m3u8、ts、m4a、avi、m4s、ogg	
	网页	html、js、css	
	压缩包	zip、7z、tar、br、gz、rar、bz2	
	文档	doc、docx、xls、xlsx、pdf、ppt、pptx	
	应用程序	apk、exe、bin	
	其它	vsv、iso、jar、swf、chunk、atlas	
其他文件		N/A	不缓存

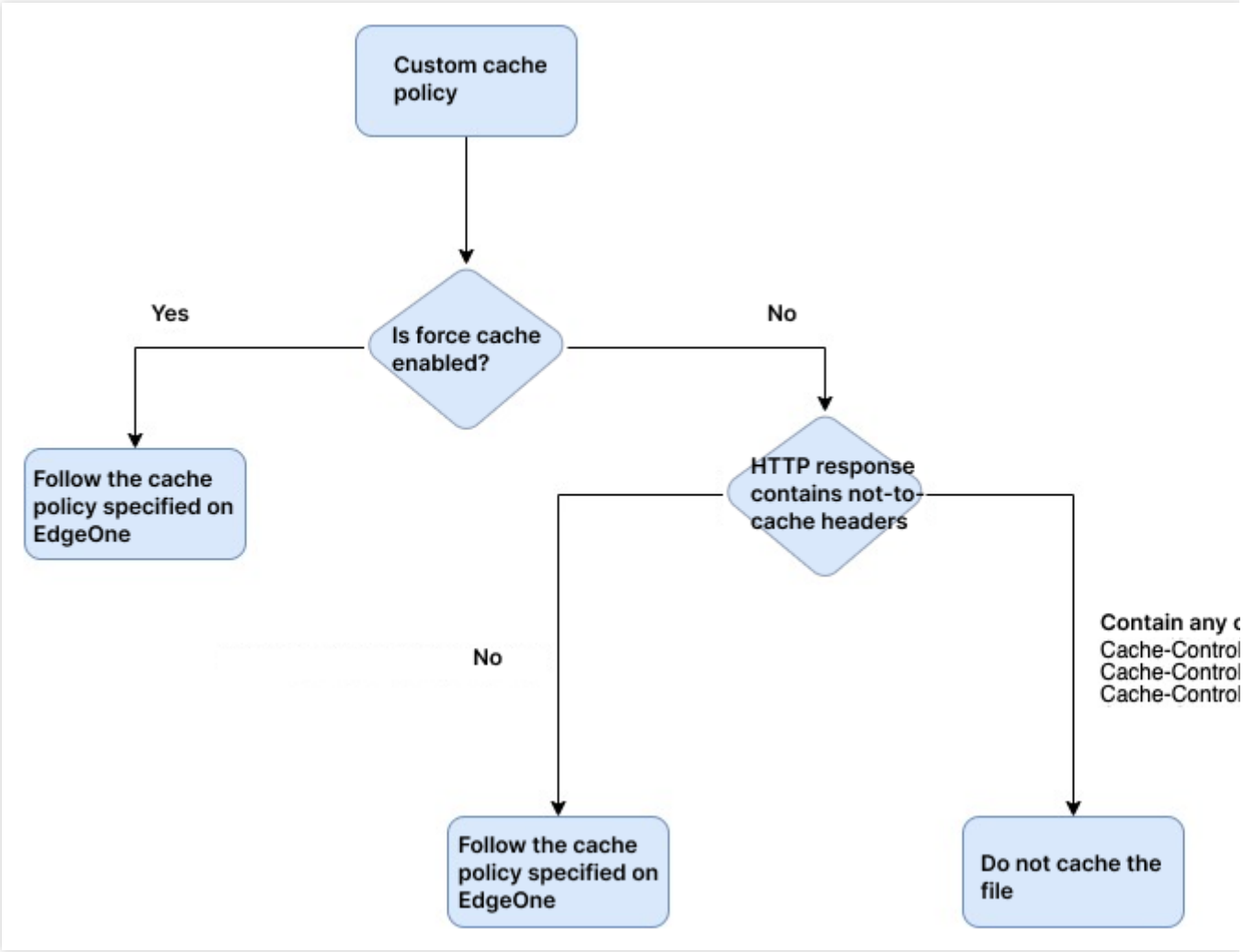
不缓存：HTTP 响应头中不包含以上任意缓存头部的情况下，不缓存

自定义时间：如果 HTTP 响应头中不包含以上任意缓存头部，则按照平台自定义配置的缓存时间缓存。

不缓存策略

如果在 EdgeOne 规则引擎或全局站点设置内，配置缓存规则为不缓存，则该文件无论源站是否带有 `Cache-Control` 及其他缓存头部，均不缓存文件。

自定义缓存时间



自定义缓存时间可以帮助您按照自定义配置的时间来缓存文件，支持开启/关闭强制缓存：

开启强制缓存：默认开启，无论源站是否携带有 `Cache-Control` 及其他缓存头部，均无视对应头部配置，按照 EdgeOne 平台内配置的自定义缓存时间缓存该文件。

关闭强制缓存：关闭强制缓存后，如果源站的 HTTP 响应头中携带有以下任意的不缓存响应头时，文件将不会缓存：

`Cache-Control:private`

`Cache-Control:no-store`

`Cache-Control:no-cache`

如果不包含以上的任意 HTTP 响应头，则文件将按照 EdgeOne 平台内配置的自定义缓存时间缓存该文件。

了解更多

如何配置节点缓存规则

如何清除节点内缓存文件

如何提前将热点文件缓存至节点中

缓存键（Cache Key）介绍

最近更新时间：2023-08-07 16:30:42

什么是缓存键（Cache Key）

缓存键（Cache Key）用于决定用户访问的文件资源是否命中 EdgeOne 边缘缓存内容，是节点内缓存资源的唯一标识。缓存命中可以帮助您的站点：
减少回源请求量，降低源站带宽消耗。
提高用户的访问请求速度。
缓存键的工作原理如下所示：
当文件被缓存到 EdgeOne 边缘节点中时，节点将根据缓存键规则来为文件生成一个对应的缓存键标识，默认情况下，缓存键通过客户端请求 URL 和查询字符串计算得出。当有其它客户端向该边缘节点发起 HTTP 请求时，节点将根据 Cache Key 的计算规则，来比对该 HTTP 请求与节点内所有缓存资源的缓存键（Cache Key）是否一致，如果一致时，则将对应的缓存资源直接响应给该客户端，即缓存命中。

缓存键（Cache Key）的作用场景

缓存键（Cache Key）用于 EdgeOne 节点为不同版本的文件建立多版本缓存内容，即使客户端通过同一路径请求，也可以通过缓存键（Cache Key）的计算规则来正确响应用户文件。

您可以通过以下场景示例来了解如何正确配置 Cache Key 来帮助您正确匹配请求的缓存文件，同时降低请求的回源率。
例如：用户 A 和用户 B 分别有以下请求：
请求 A：`https://www.example.com/Image/test.jpg?version=1&time=1651752743` 。
请求 B：`https://www.example.com/Image/test.jpg?version=2&time=1651758319` 。
场景一：用户访问的文件路径完全相同，但是根据查询字符串内携带的 `version` 参数不同，将会有版本的区分，以上请求分别为两张不同的图片，此时应在 Cache Key 中保留会影响文件版本的参数 `version` ，来保证节点能正确缓存并响应对应的文件内容。
场景二：用户访问的 URL 中查询字符串的内容完全不影响文件内容，以上请求所对应的文件一致，不影响文件版本，此时应该在 Cache Key 计算中忽略所有查询字符串，来提高文件在节点内的命中率，减少请求回源。

允许自定义的 Cache Key 内容及生效规则

EdgeOne 允许用户自定义 Cache Key 规则，支持配置查询字符串、HTTP 请求头或者 cookie 进行区分缓存。您可以通过 自定义 Cache Key 来了解如何配置自定义的 Cache Key 规则。

注意：

1. 缓存键计算以客户端向节点发起的 HTTP 请求内容为准，回源 URL 改写、回源跟随重定向、回源 HTTP 请求头、访问 URL 重写均不影响缓存键的计算。
2. 当在 EdgeOne 内配置了 Token 鉴权时，鉴权内容不会参与至缓存键计算中。
3. 当在 EdgeOne 内开启了图片处理参数，请求中携带的图片处理参数将默认参与 Cache Key 计算。

查询字符串

查询字符串是指请求 URL 中 `?` 之后的字符串参数（包含一个或多个参数，用 `&` 分隔），例如：`https://www.example.com/images/example.jpg?color=blue&size=large` 中的 `color=blue&size=large`。

EdgeOne 支持自定义保留指定查询字符串的内容来区分缓存。

注意：

当保留查询字符串作为缓存键时，如果参数的位置顺序发生变化，缓存键也会变化。

例如：当客户端分别发起以下请求：

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

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

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

配置	缓存行为
查询字符串配置为全部保留	请求 A 和请求 B 所携带参数内容不同，对应不同的缓存版本，请求 A 和请求 C 参数内容一致，但是顺序不同，也对应不同的缓存版本。
查询字符串配置为全部忽略	请求 A、B、C 均对应同一份缓存版本。
查询字符串配置为保留指定参数 `Type`	请求 A、B 的参数顺序及内容完全一致，对应同一份缓存版本；请求 B 和请求 C 的 参数内容相同，但是顺序不同，对应不同的缓存版本。
查询字符串配置为忽略指定参数 `Type`	请求 A 和请求 B 剩余参数的内容不一致，对应不同的缓存版本；请求 A 和请求 C 剩余参数的内容一致，但是顺序不一致，对应不同的缓存版本。

HTTP 请求头

EdgeOne 支持将指定的 HTTP 请求头加入缓存键 Cache Key 的计算中，EdgeOne 边缘节点将根据请求头内容来建立不同的缓存版本。请求头的顺序变化，不影响 Cache Key的计算。

例如：指定HTTP请求头 `User-Agent` 加入缓存键计算中，以下请求 A 与请求 B 的 URL 及参数内容一致，但是 `User-Agent` 头内容不一致，将对应不同的缓存版本。

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

Cookie

EdgeOne 支持将指定的 Cookie 内参数加入到缓存键的计算中，根据 Cookie 参数及内容区分缓存版本。Cookie 内多个参数参与 Cache Key 计算时，参数的顺序变化不影响 Cache Key 的计算。
例如：指定 Cookie 内参数 `User` 加入缓存键计算中，以下请求 A 和请求 B 参数内容相同，对应同一份缓存，请求 A 与请求 C 参数内容不同，对应不同版本的缓存。
请求 A：`https://www.example.com/Image/test.jpg?version=1&type=a`，携带 `Cookie：User=A;ID=1`。
请求 B：`https://www.example.com/Image/test.jpg?version=1&type=a`，携带 `Cookie：User=A;ID=2`。
请求 C：`https://www.example.com/Image/test.jpg?version=1&type=a`，携带 `Cookie：User=B;ID=1`。

了解更多

了解 EdgeOne 内容缓存规则
如何配置自定义Cache Key
如何配置节点缓存规则
如何清除节点缓存资源

Vary 特性

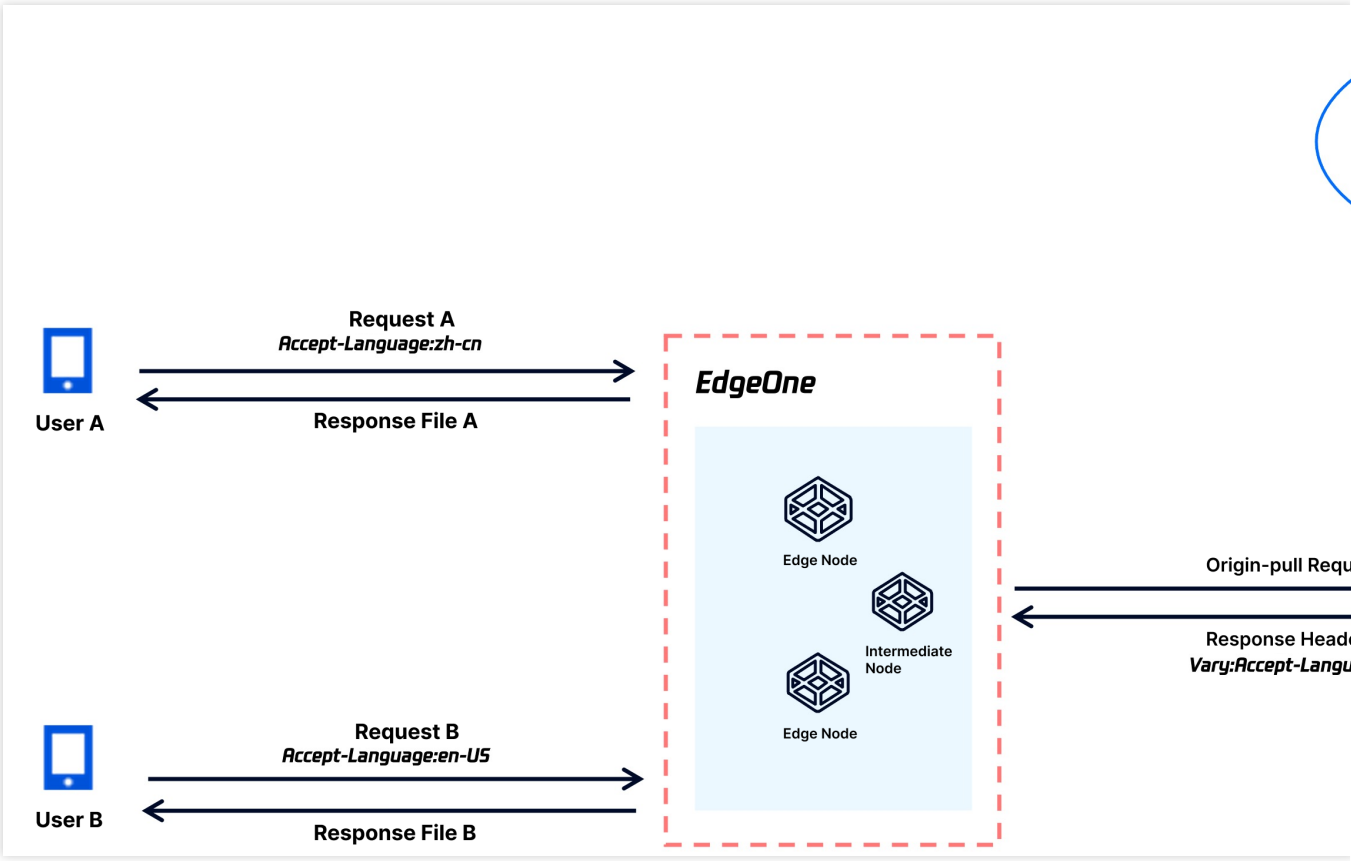
最近更新时间：2023-05-09 10:47:02

EdgeOne 对 Vary 的支持情况

EdgeOne 当前已全平台默认支持 Vary 特性，您无需做任何配置，直接在源站响应文件时，根据需要在响应头内增加 Vary 头部即可。Vary 头部标准请参见：Vary。

什么是 Vary 特性

Vary 是在 HTTP/1.1 后新增的一个 HTTP 响应头，当客户端使用同一 URL 向源站服务器发起请求时，如果源站服务器存在不同版本的文件内容时，可能被中间缓存系统（例如：浏览器缓存、内容分发网络 CDN ）缓存，无法区分场景响应文件。因此，源站服务器可以通过在 HTTP 响应中增加 Vary 头来告知中间缓存系统通过哪个请求头部来区分缓存版本内容。
例如：当客户端请求都为 `https://www.example.com/test.pdf` 时，源站中为了区分客户端语言响应不同版本的文件，在 HTTP 响应头中增加了 `Vary: Accept-Language` ，EdgeOne 将在建立缓存时，将根据客户端所请求的 `Accept-Language` 内容建立不同版本的缓存。
当用户 A 发起请求 URL 为 `https://www.example.com/test.pdf` ，携带的请求头 `Accept-Language:zh-cn` ，则 EdgeOne 将响应文件 A。
当用户 B 发起请求 URL 为 `https://www.example.com/test.pdf` ，携带的请求头 `Accept-Language:en-US` ，则 EdgeOne 将响应文件 B。

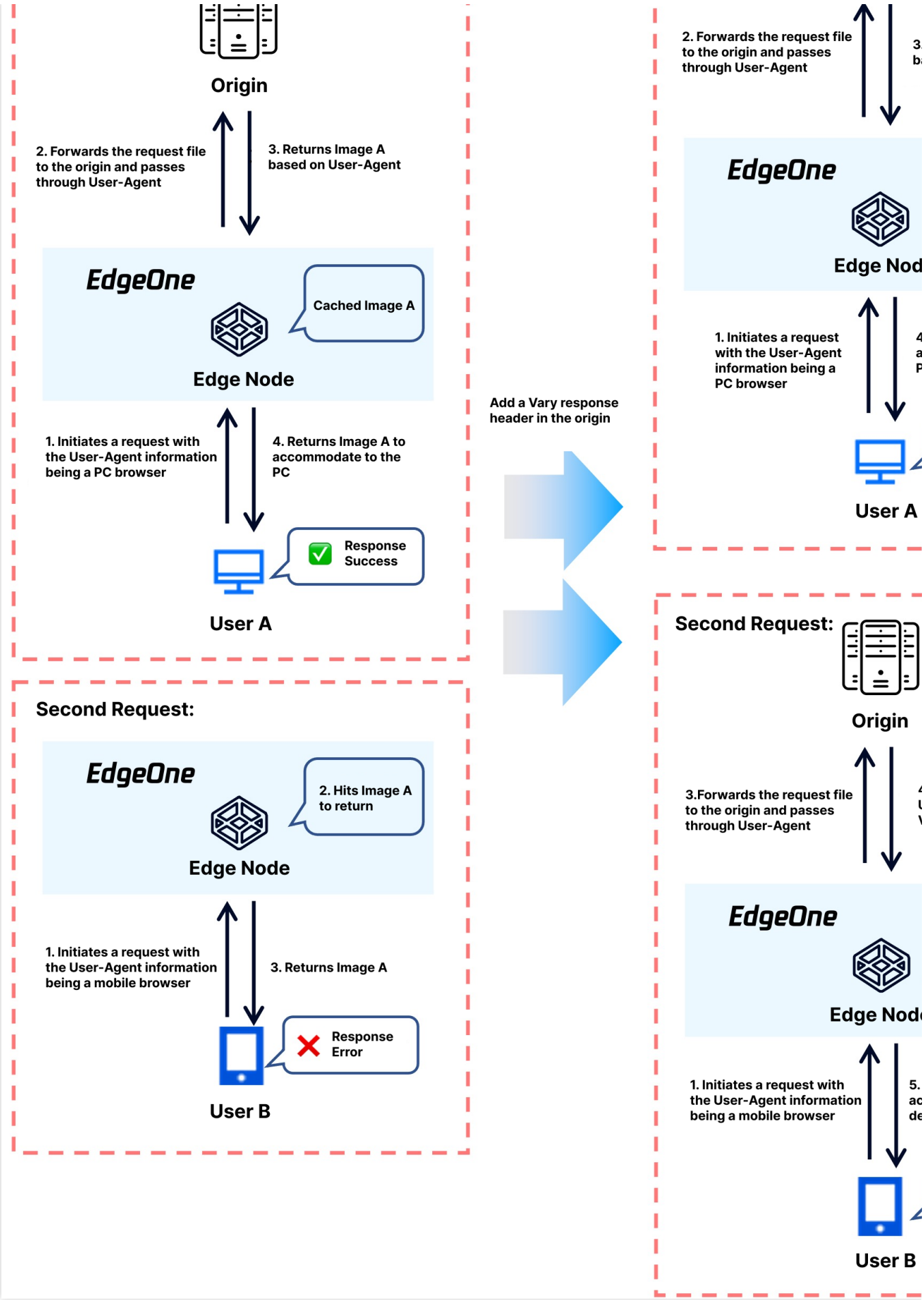


Vary 的应用场景

通过灵活地使用 Vary 头部，可由源站响应控制哪些文件是需要根据指定的 HTTP 请求头来区分缓存，可帮助您解决以下场景问题：

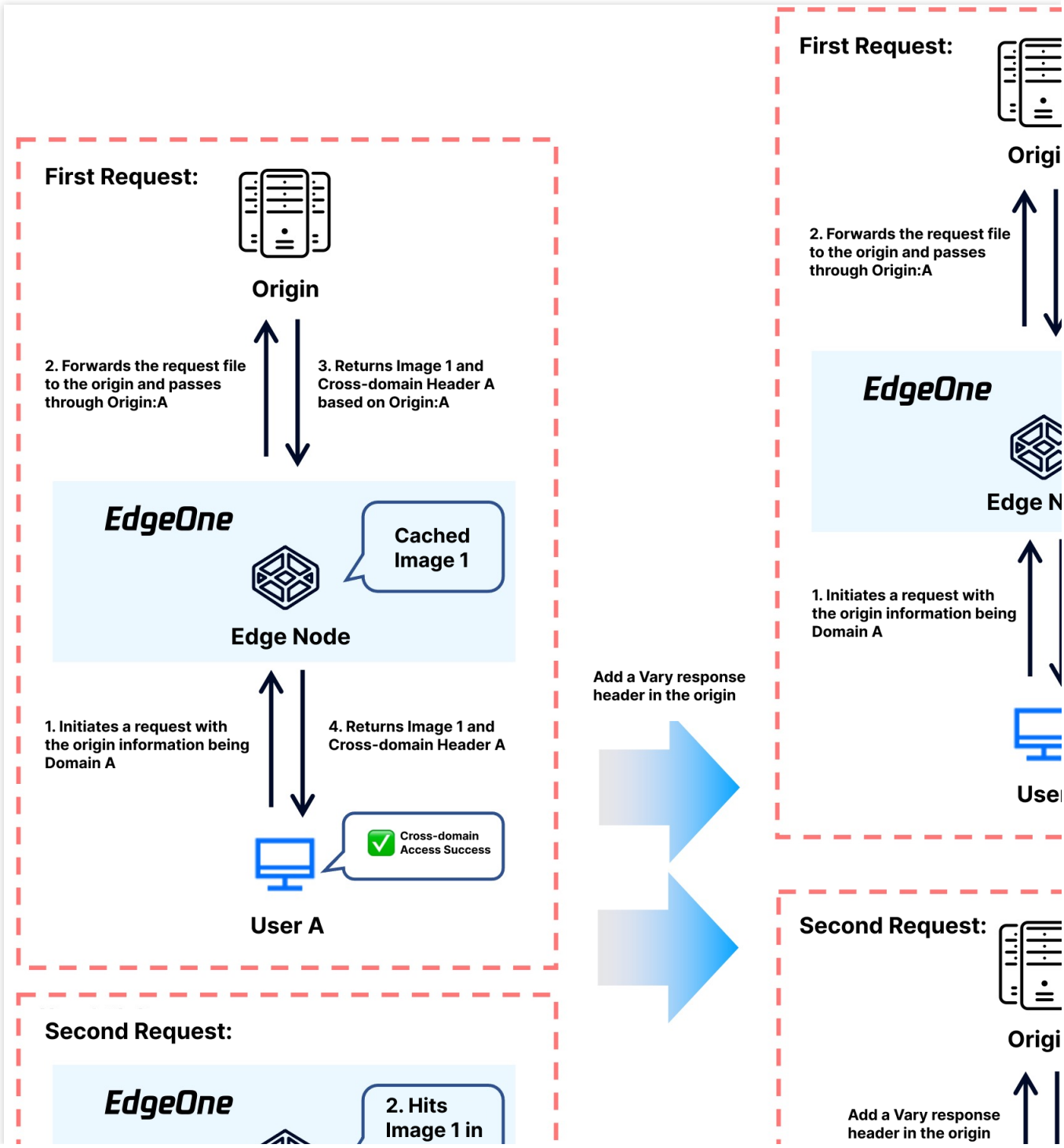
场景一：URL 相同，根据客户端类型区分响应文件

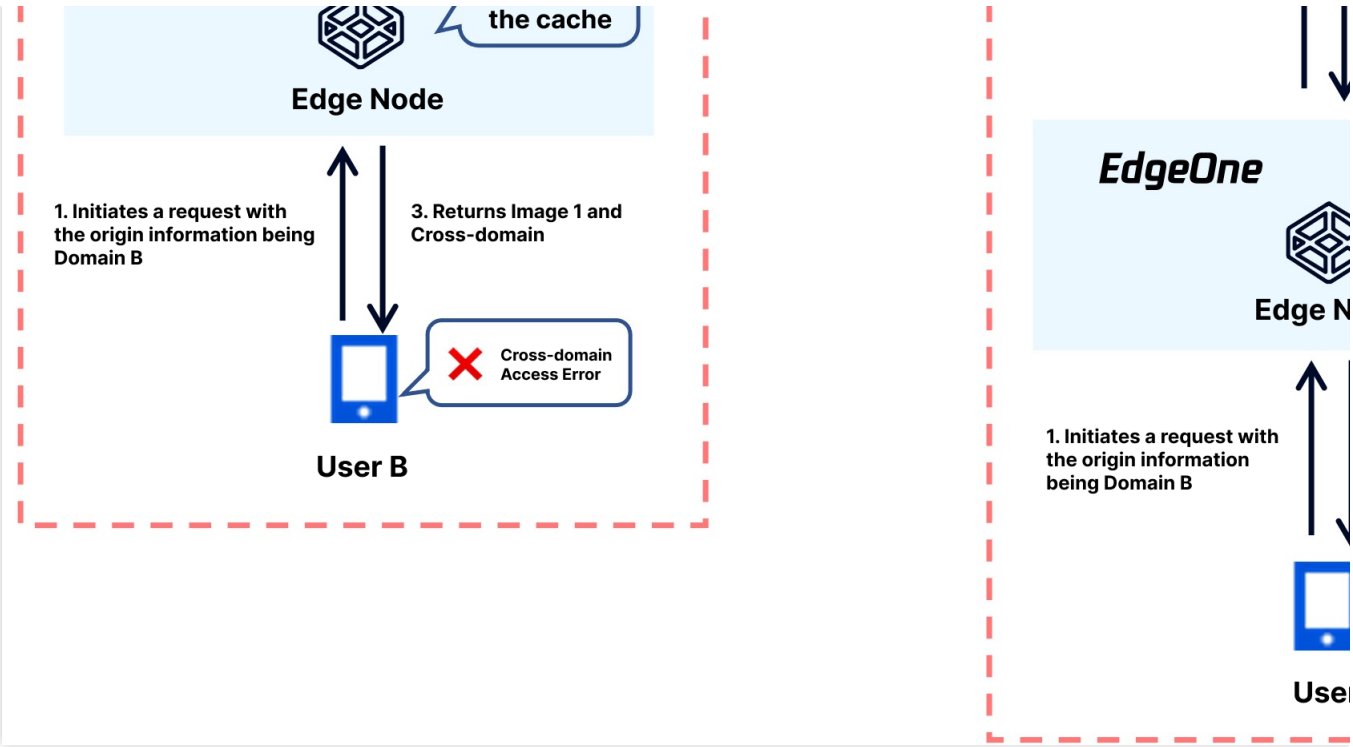
假设当前网站同时具有 PC 端和移动端页面时，为了让用户访问方便，网站的 PC 端和移动端的资源均使用相同 URL 访问，但是因为屏幕分辨率的差异，前端为了将内容适配到移动端，图片的分辨率大小是不一致的，此时需要能根据用户在请求时携带的 `User-Agent` 头部来区分响应的图片 A 适配 PC 端，图片 B 适配移动端。为了避免响应的图片内容被缓存导致移动端访问的内容与 PC 端内容一致，此时源站可在响应图片时增加 `Vary:User-Agent` 来指定节点根据用户请求的 `User-Agent` 头部区分缓存。


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
2. Forwards the request file to the origin and passes through User-Agent
EdgeOne
Edge Nod
1. Initiates a request with the User-Agent information being a PC browser
User A
Second Request:
Origin
3.Forwards the request file to the origin and passes through User-Agent
EdgeOne
Edge Nod
1. Initiates a request with the User-Agent information being a mobile browser
User B

场景二：URL 相同，根据访问来源响应跨域头

假设当前页面内存在同一个图片文件分别被域名 A 和域名 B 引用时，为了避免出现跨域错误，通常需要在源站响应文件时为该文件增加 `Access-Control-Allow-Origin` 来实现允许跨域访问，但是如果当前文件被缓存，则可能出现跨域头一起被缓存导致出现跨域访问错误的问题。此时源站可以在响应 `Access-Control-Allow-Origin` 跨域头的同时，增加 `Vary:origin` 头部，来指定节点根据用户请求的 `Origin` 头部区分缓存。




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
EdgeOne
Edge N
1. Initiates a request with the origin information being Domain B
User

缓存配置
自定义 Cache Key

最近更新时间：2024-01-02 10:01:01

功能简介

当您需要将同一路径的请求 URL 通过请求参数、cookie 或者 HTTP 请求头的区别指向不同文件时，或者将携带不同参数的请求 URL 均指向同一份文件时，自定义 Cache Key 支持自定义调整资源在节点内的缓存 Cache Key 标识，包括拼接查询字符串、拼接 HTTP 标头或 Cookie 信息等，以便请求 URL 能根据不同场景正确获取对应的缓存资源。您可以通过 缓存键（Cache Key）介绍 来了解什么是 Cache Key。

使用场景

场景一：用户访问的文件路径完全相同，但是根据携带的查询字符串、HTTP 请求头、Cookie 内容，将会有版本的区分，可通过自定义 Cache Key 来调整该类型文件的缓存键来区分文件缓存。
场景二：用户访问的 URL 中查询字符串的内容完全不影响文件内容，以上请求所对应的文件一致，不影响文件版本，可通过自定义 Cache Key 来调整该类型文件的缓存键来使请求命中同一份文件缓存。

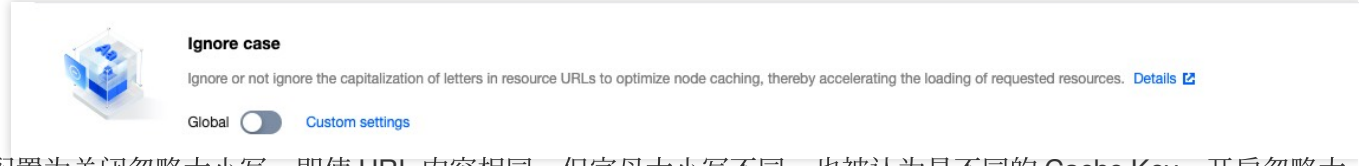
操作步骤

场景一：针对站点所有域名配置自定义 Cache Key

若您需要对整个接入站点配置自定义 Cache Key，或作为站点级兜底配置，可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**站点加速 > 缓存配置**，找到查询字符串/忽略大小写卡片，单击全局站点设置进行配置。
说明：
全局配置仅可配置**查询字符串**和**忽略大小写**，如需更丰富的自定义 Cache Key 配置项请参见 场景二规则引擎处的配置步骤。

默认配置为全部保留，即保留原请求 URL 的全部查询参数作为 Cache Key。支持其他选项：a. 全部忽略：忽略整个查询字符串；b. 保留指定参数：仅保留查询字符串中指定的参数；c. 忽略指定参数：仅忽略查询字符串中指定的参

数。



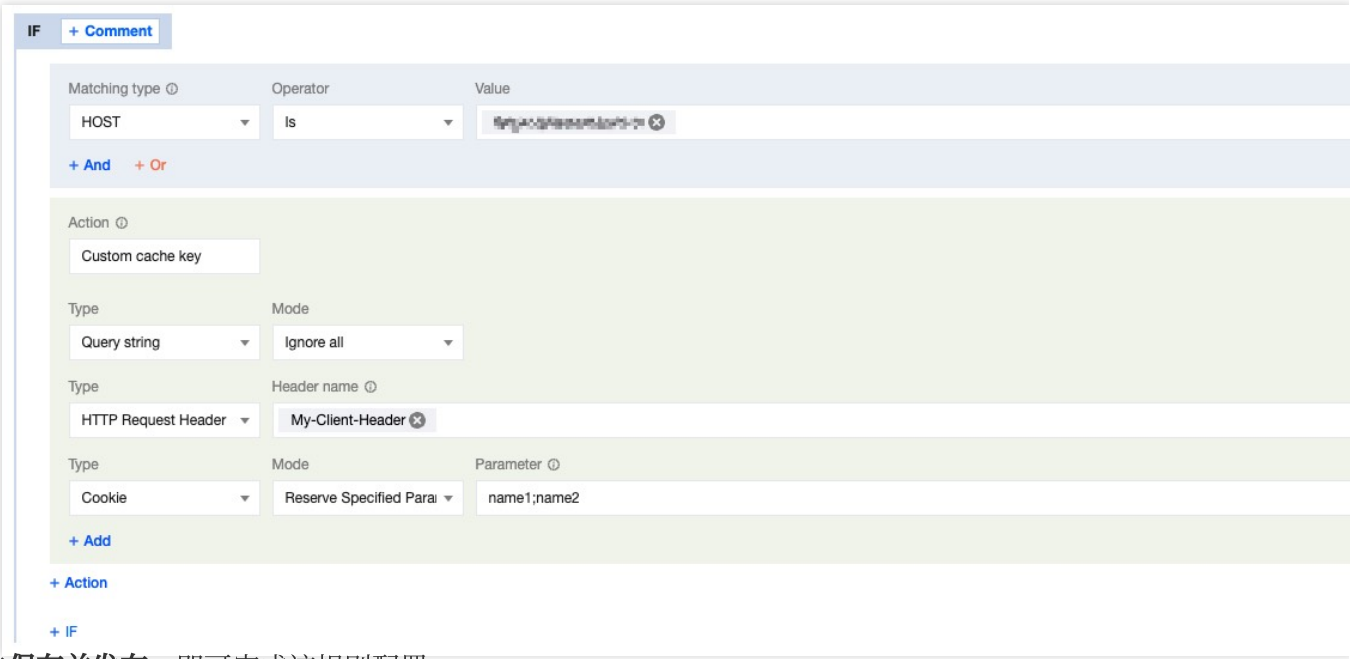
默认配置为关闭忽略大小写，即使 URL 内容相同，但字母大小写不同，也被认为是不同的 Cache Key，开启忽略大小写后，则字母大小写不同将被认为是相同的 Cache Key。

场景二：针对指定域名，路径或文件后缀等请求粒度配置自定义 Cache Key

若您需要针对站点 `example.com` 站点下的 `www.example.com` 域名配置自定义 Cache Key 规则为忽略所有查询字符串，将 HTTP 请求头 `My-Client-Header` 及 Cookie 内的参数 `name1` 、 `name2` 作为Cache Key、可以参考以下步骤配置：

操作步骤

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面，单击**规则引擎。**
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。在规则编辑页面，选择 Host 为匹配类型，配置为 `www.example.com` 。
4. 单击**操作，**在弹出的操作列表内，选择操作为**自定义 Cache Key；**
5. 单击类型下方的**添加**，可添加自定义 Cache Key 的类型，以本示例场景为例，添加查询字符串、HTTP 请求头、Cookie 进行配置并填写对应的内容，完整的规则配置如下所示：



6. 单击**保存并发布**，即可完成该规则配置。

生效示例

配置完成后，Cache Key 由 URL+My-Client-Header+Cookie 组成：忽略全部查询字符串，拼接 `My-Client-Header` 和保留指定参数后的 Cookie。
则客户端请求 A：
URL：`https://www.example.com/path/demo.jpg?key1=value1&key2=value2` 。
HTTP 请求头：含 `My-Client-Header:fruit` 。
Cookie：name1=yummy;name2=tasty;name3=strawberry。
与客户端请求 B：
URL：`http://www.example.com/path/demo.JPG?key1=value1&key2=value2&key3=value3` 。
HTTP 请求头：含 `My-Client-Header:fruit` 。
Cookie：name1=yummy;name2=tasty;name3=blueberry。
与客户端请求 C：
URL：`http://www.example.com/path/demo.JPG?key1=value1&key2=value2&key3=value3&key4=value4` 。
HTTP 请求头：含 `My-Client-Header:sea` 。
Cookie：name1=yummy;name2=tasty;name3=fish。
A 和 B 请求将会命中同一份缓存资源，C 命中另一份缓存资源。

相关参考

支持的头部名称说明：

头部类型	说明
自定义	自定义头部。 名称：1 - 100 个字符，由数字 0 - 9、字符 a - z、A - Z，及特殊符 - 组成。 值：1 - 1000 个字符，不支持中文。
预设头部	根据客户端 User-Agent 信息聚合的头部： 客户端设备类型：`EO-Client-Device` 取值：`Mobile` ，`Desktop` ，`SmartTV` ，`Tablet` 或 `Others` 客户端操作系统：`EO-Client-OS` 取值：`Android` ，`iOS` ，`Windows` ，`MacOS` ，`Linux` 或 `Others` 客户端浏览器类型：`EO-Client-Browser` 取值：`Chrome` ，`Safari` ，`Firefox` ，`IE` 或 `Others`

节点缓存 TTL

最近更新时间：2024-05-14 14:20:47

功能简介

节点缓存 TTL 用于决定资源是否在 EdgeOne 节点缓存以及在节点内的缓存时长，用户在请求缓存过期的文件或者未缓存的文件时，节点将不会直接响应用户请求，而是会回源站获取最新资源进行响应，并根据缓存规则决定是否缓存到 EdgeOne 。缓存文件并让用户请求命中，可以帮助您：

减少回源请求量，降低源站带宽消耗。

提高用户的访问请求速度。

您可以根据业务需求，自定义针对不同资源配置缓存时间，优化不同资源的缓存策略，提升请求资源的加载速度。

了解更多缓存说明请查看 EdgeOne 内容缓存规则 。

说明：

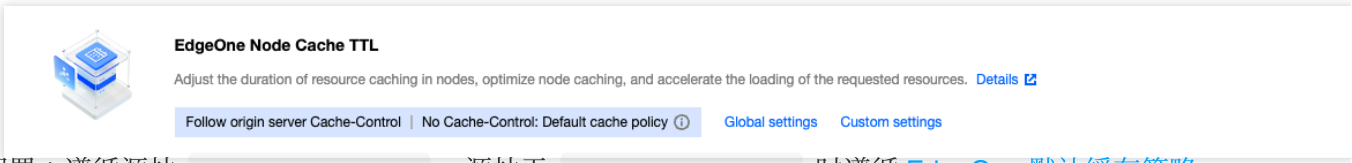
1. 若源站资源更新后，需要立刻更新节点的缓存，可使用 清除缓存 功能主动清除节点未过期的旧缓存，保证后续请求可以获取到源站最新的资源。
2. 请不要缓存动态资源到边缘节点内，以避免用户访问到错误的内容。
3. 文件缓存在 EdgeOne 节点后，平台具有文件冷热淘汰机制，如果当前缓存文件长时间未有请求，则可能在未达到最大缓存时间时提前从节点缓存中删除。

操作步骤

场景一：针对站点所有域名配置节点缓存 TTL

若您需要对整个接入站点配置相同的节点缓存 TTL，或作为站点级兜底配置，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击站点列表，在站点列表内单击需配置的站点。
2. 在站点详情页面，单击**站点加速 > 缓存配置**，找到节点缓存 TTL 卡片。
3. 单击**站点全局设置**，进行配置。详细配置说明可参见 EdgeOne 内容缓存规则。



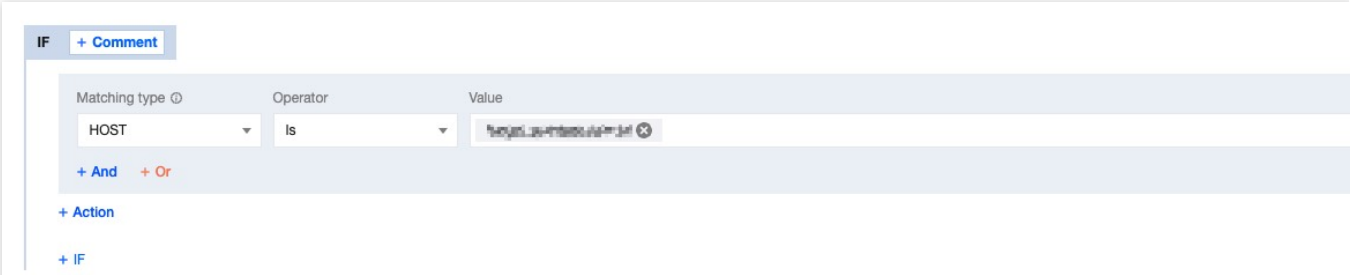
默认配置：遵循源站 `Cache-Control` ，源站无 `Cache-Control` 时遵循 EdgeOne 默认缓存策略。

场景二：针对指定域名，路径或文件后缀等请求粒度配置节点缓存 TTL

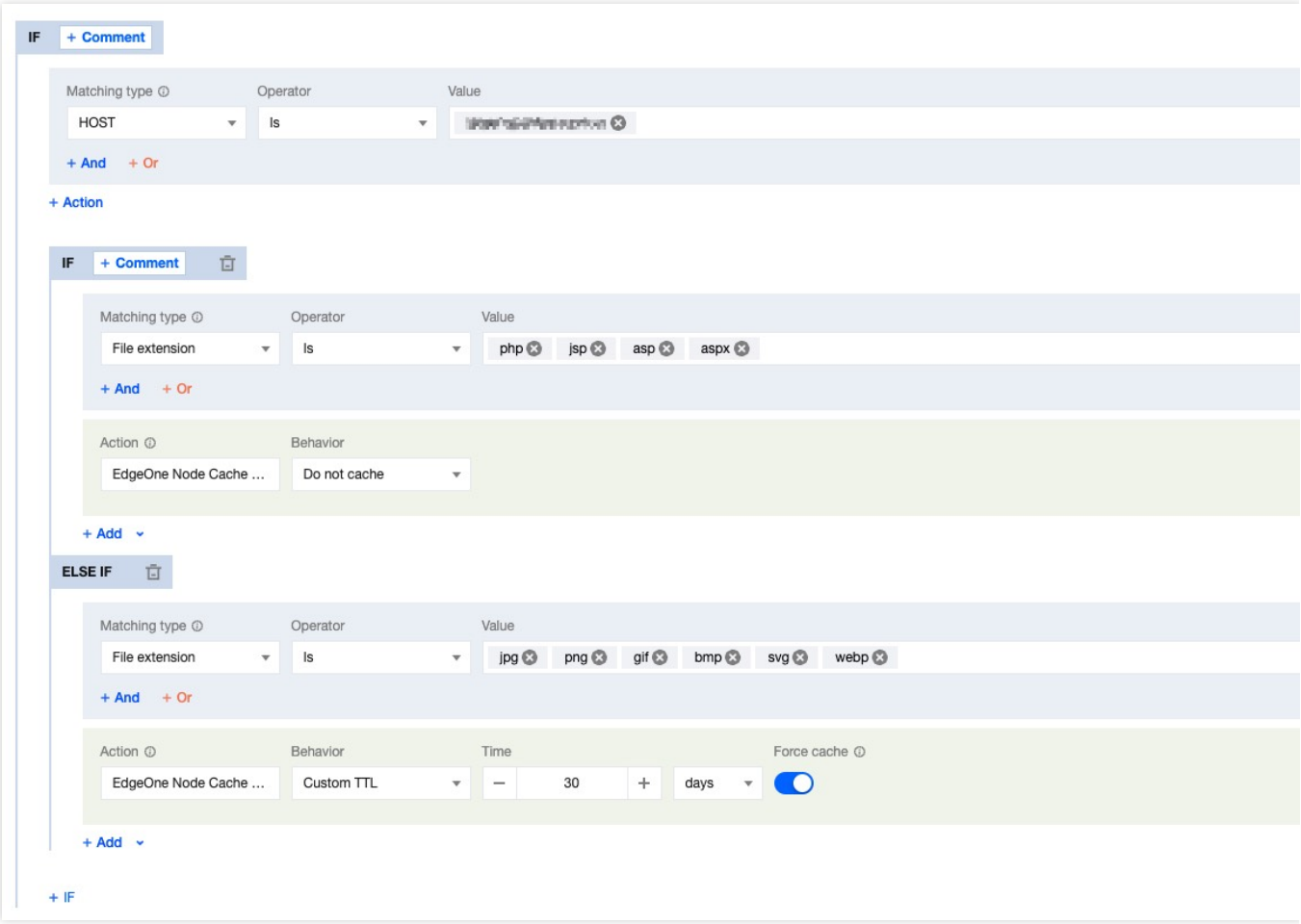
若您需要针对不同域名，路径或文件后缀等配置不同的节点缓存 TTL，例如：针对 `www.example.com` 域名下， `php/jsp/asp/aspx` 后缀的文件不缓存， `jpg/png/gif/bmp/svg/webp` 后缀的文件缓存 30 天。可参

考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，先选择匹配类型为 HOST，值为 `www.example.com` 作为最外层匹配条件，单击**添加 IF**。



5. 在新增的 IF 条件内，选择匹配类型为文件后缀，添加 `php/jsp/asp/aspx` 后缀，单击**操作**，在弹出的操作列表中，选择操作为**节点缓存 TTL**，配置为不缓存。
6. 重复以上步骤，再新增一条 IF 条件，添加 `jpg/png/gif/bmp/svg/webp` 后缀，配置为缓存 30 天。
7. 配置后规则如下，单击**保存并发布**，即可完成该规则配置。



相关参考

如何判断用户请求是否命中 EdgeOne 节点缓存？

状态码缓存 TTL

最近更新时间：2023-10-11 10:16:18

功能简介

EdgeOne 回源获取资源时，若源站成功响应资源，则 EdgeOne 会响应给客户端请求并缓存在 EdgeOne 中，以便下次直接响应。若源站响应为4xx或5xx等异常状态码， EdgeOne 拿不到资源，则下次请求仍会触发回源，源站可能会有较大压力。配置状态码缓存 TTL，则在缓存时间内， EdgeOne 直接响应异常状态码，而不是全部触发回源，可以减轻源站压力，提升响应速度。

目前支持配置以下状态码：

4xx：400、401、403、404、405、407、414。

5xx：500、501、502、503、504、509、514。

说明：

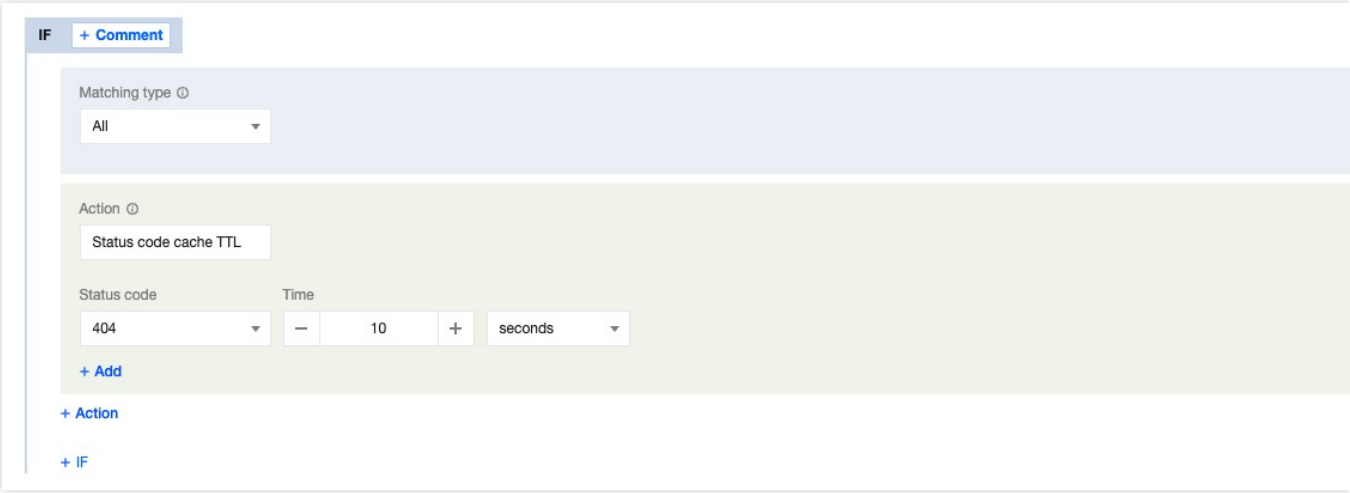
EdgeOne 默认对 404 状态码缓存10s。

操作步骤

场景一：针对站点所有域名配置状态码缓存 TTL

若您需要对整个接入站点配置自状态码缓存 TTL，或作为站点级兜底配置，可参考以下步骤：

1. 登录 边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。在规则编辑页面，选择匹配类型为全部（站点任意请求）
4. 单击**操作**，在弹出的操作列表内，选择操作为**状态码缓存**，配置对应的缓存状态码及缓存时间。

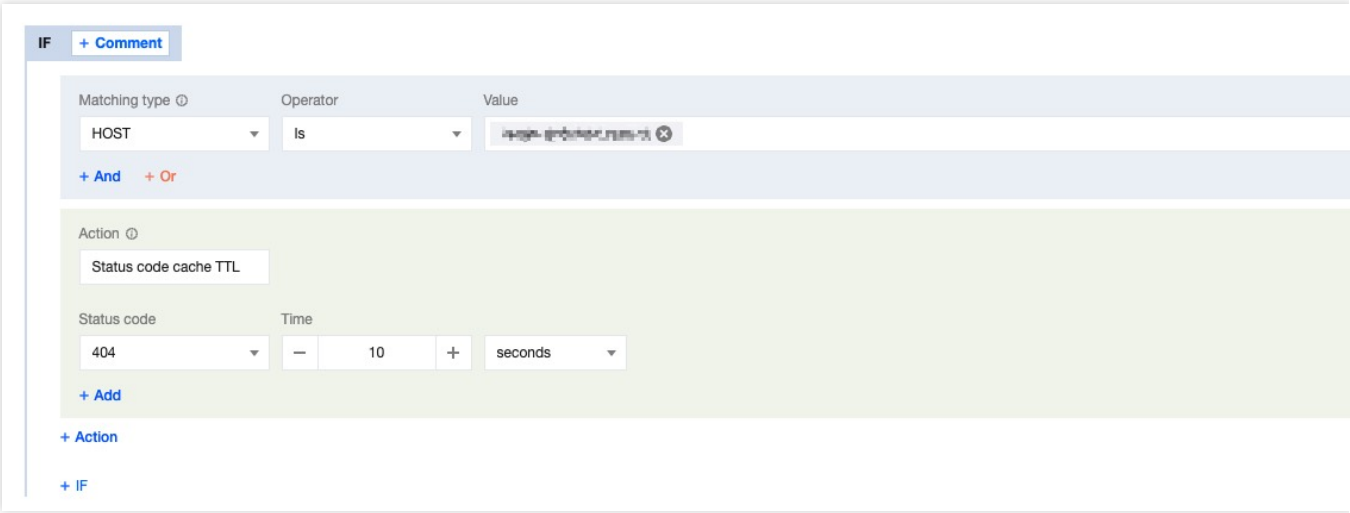


5. 单击保存并发布，即可完成该规则配置。

场景二：针对指定域名，路径或文件后缀等请求粒度配置状态码缓存 TTL

若您需要针对不同域名，路径或文件后缀等配置不同的状态码缓存 TTL，例如：针对 `example.com` 站点下的 `www.example.com` 域名配置状态码缓存 TTL。可参考以下步骤：

1. 登录 边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面，单击**规则引擎。**
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。在规则编辑页面，选择 Host 为匹配类型，配置为 `www.example.com` 。
4. 单击**操作，**在弹出的操作列表内，选择操作为**状态码缓存**，配置对应的缓存状态码及缓存时间。



5. 单击保存并发布，即可完成该规则配置。

浏览器缓存 TTL

最近更新时间：2024-05-08 20:40:57

功能简介

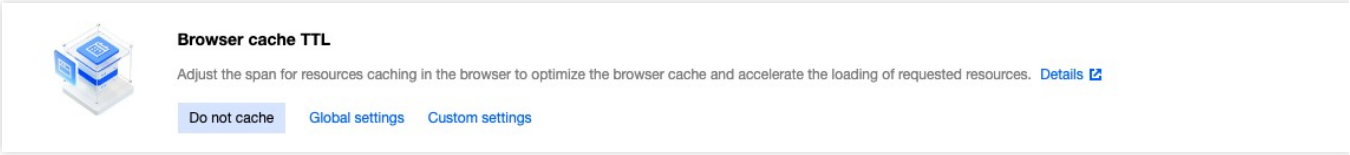
客户端浏览器缓存 TTL 即资源在浏览器中的缓存时长，默认遵循源站的 `Cache-Control` 头部，您可在不修改源站配置的情况下，通过配置 EdgeOne 的浏览器缓存 TTL 来控制资源在浏览器中的缓存时长。
EdgeOne 通过设置响应客户端时的 `Cache-Control` 响应头实现浏览器缓存 TTL。支持以下配置：
遵循源站：遵循源站的 `Cache-Control`，若源站无 `Cache-Control` 则不做任何更改；
不缓存：无论源站是否携带 `Cache-Control`，均控制浏览器为不缓存文件；
自定义时间：无论源站是否携带 `Cache-Control`，将修改 `max-age` 为指定的缓存时间。

您可以通过调整浏览器缓存 TTL 来控制资源在浏览器的缓存时间，优化不同资源的缓存策略，提升请求资源的加载速度。

操作步骤

场景一：针对站点所有域名配置浏览器缓存 TTL

若您需要对整个接入站点配置相同的浏览器缓存 TTL，或作为站点级兜底配置，可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面内，单击**站点加速 > 缓存配置**，找到浏览器缓存 TTL 卡片，单击**全局站点设置**进行修改。

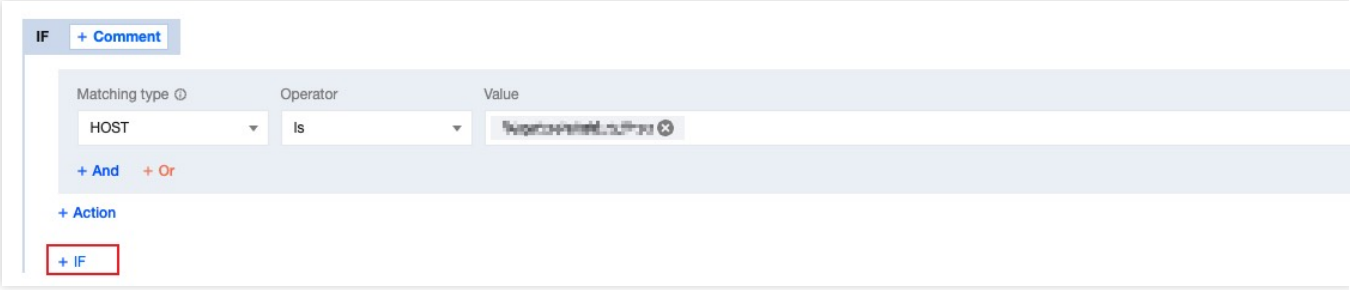


默认配置：支持遵循源站的 `Cache-Control`，若源站无 `Cache-Control` 则不缓存。

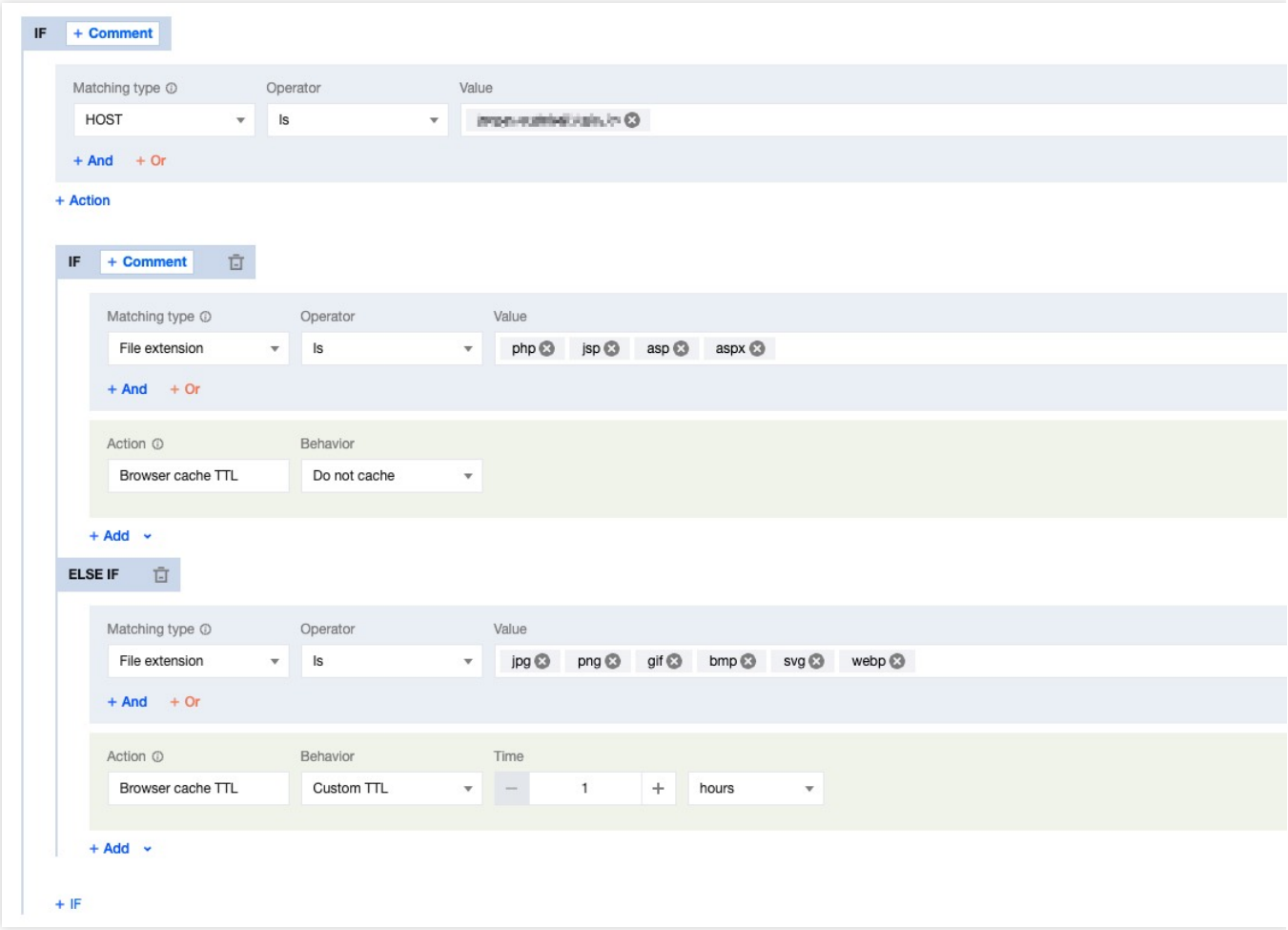
场景二：针对指定域名，路径或文件后缀等请求粒度配置浏览器缓存 TTL

若您需要针对不同域名，路径或文件后缀等配置不同的浏览器缓存 TTL，例如：针对 `www.example.com` 域名，`php/jsp/asp/aspx` 后缀的文件不缓存，`jpg/png/gif/bmp/svg/webp` 后缀的文件缓存 1 小时。可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。

3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，先选择匹配类型为 HOST，值为 `www.example.com` 作为最外层匹配条件，然后点击**添加 IF**；



5. 在新增的 IF 条件内，选择匹配类型为文件后缀，添加 `php/jsp/asp/aspx` 后缀，单击**操作**，在弹出的操作列表中，选择操作为**浏览器缓存 TTL**，配置为不缓存。
6. 重复以上步骤，再新增一条 IF 条件，添加 `jpg/png/gif/bmp/svg/webp` 后缀，配置为缓存 1 小时。
7. 配置后规则如下，单击**保存并发布**，即可完成该规则配置。


IF
+ Comment
Matching type
Operator
Value
HOST
Is
+ And
+ Or
+ Action
IF
+ Comment
Matching type
Operator
Value
File extension
Is
php
jsp
asp
aspx
+ And
+ Or
Action
Behavior
Browser cache TTL
Do not cache
+ Add
ELSE IF
Matching type
Operator
Value
File extension
Is
jpg
png
gif
bmp
svg
webp
+ And
+ Or
Action
Behavior
Time
Browser cache TTL
Custom TTL
1
hours
+ Add
+ IF

离线缓存

最近更新时间：2023-10-11 10:17:07

功能简介

默认情况下，若 EdgeOne 回源获取资源时无法与源站建立连接，则会响应错误码。启用离线缓存后，当 EdgeOne 无法与源站建立连接时，可使用 EdgeOne 中已缓存的资源（即使资源已过期），直到源站恢复连接。可有效保障业务的可用性和连续性，提升用户体验。

说明：

若 EdgeOne 中无缓存可用，则响应错误码。

使用场景

源站不稳定：如果您的源站服务器容易出现故障或不稳定，开启离线缓存功能可以在源站故障期间提供更好的用户体验。即使缓存的资源已过期，仍然可以继续为用户提供服务，避免用户在源站故障时遇到无法访问的情况。

关键业务保障：对于一些关键业务，您可能希望确保在源站出现问题时，用户仍能访问网站或应用的关键内容。启用离线缓存可以确保用户在源站出现故障时仍然能访问关键资源，保障业务的连续性。

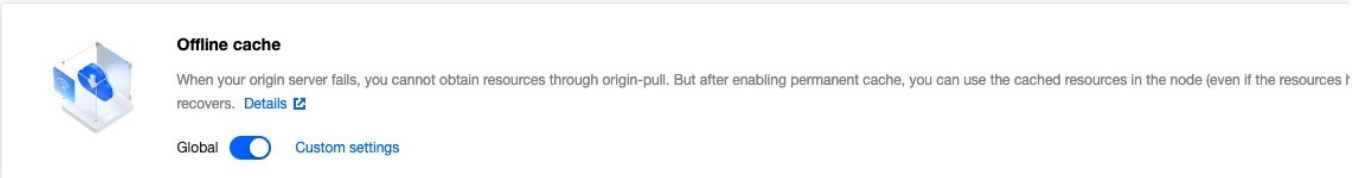
避免突发流量冲击：在某些情况下，源站可能会受到突发流量的冲击，导致服务器过载或崩溃。启用离线缓存功能可以在源站故障期间继续为用户提供服务，减轻源站的压力，有助于源站恢复正常运行。

操作步骤

场景一：针对站点所有域名配置离线缓存

若您需要对整个接入站点配置开启/关闭离线缓存，或作为站点级兜底配置，可参考以下步骤：

1. 登录 边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面内，单击**站点加速 > 缓存配置**，找到离线缓存卡片，单击开关开启。

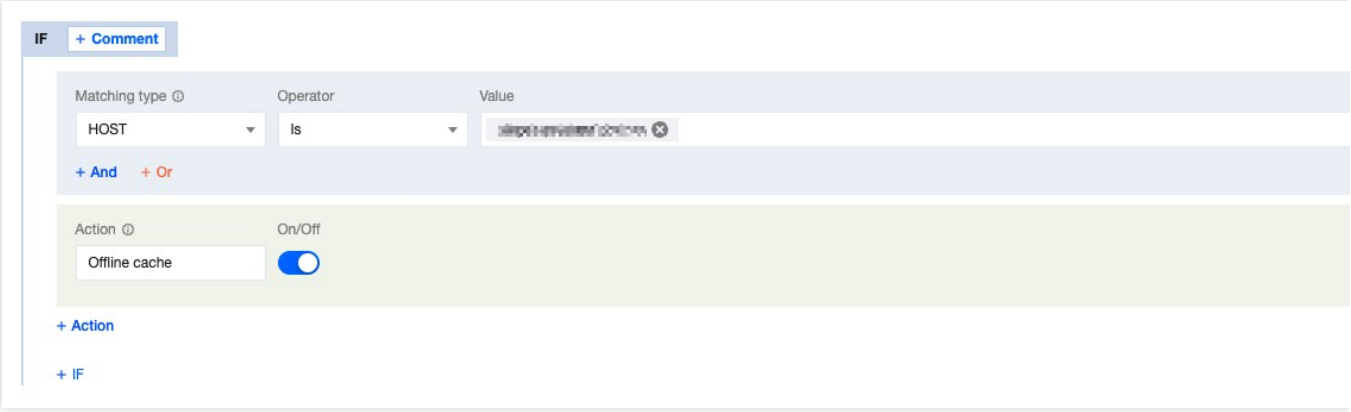


默认状态：开启。若关闭，则当源站故障，即无法正常回源拉取资源时，则节点会透传源站响应给客户端请求。

场景二：针对指定域名，路径或文件后缀等请求粒度配置离线缓存

若您需要针对不同域名，路径或文件后缀等配置不同的离线缓存，例如：针对 `example.com` 站点下的 `www.example.com` 域名开启离线缓存。可参考以下步骤：

1. 登录 边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面，单击**规则引擎。**
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。在规则编辑页面，选择 Host 为匹配类型，配置为 `www.example.com` 。
4. 单击**操作，**在弹出的操作列表内，选择操作为**离线缓存**，点击开关开启。



5. 单击**保存并发布**，即可完成该规则配置。

缓存预刷新

最近更新时间：2023-12-14 17:41:07

功能简介

缓存资源在 EdgeOne 节点内过期后，EdgeOne 在收到对应的客户端请求时，将回源获取最新资源文件，在高峰期可能导致回源量大幅上涨。缓存预刷新能力可以在缓存资源过期之前就回源验证缓存资源是否有效，不用等到过期后再验证，有助于保持资源的实时性，更快响应请求。缓存预刷新时间可按照文件缓存 TTL 的百分比进行配置。

使用场景

因缓存预刷新功能可以提前回源验证资源有效性，建议您在需要频繁更新内容或对用户体验要求较高的场景中使用：

高实时性要求：对于需要快速更新的内容，如新闻、活动页面等，客户希望用户在请求时能够获取到最新的资源。通过启用缓存预刷新功能，节点在资源过期前就回源验证并更新缓存，从而确保用户在访问时能够获得较新的资源，从而避免在用户请求时产生额外的等待时间，提高用户体验。

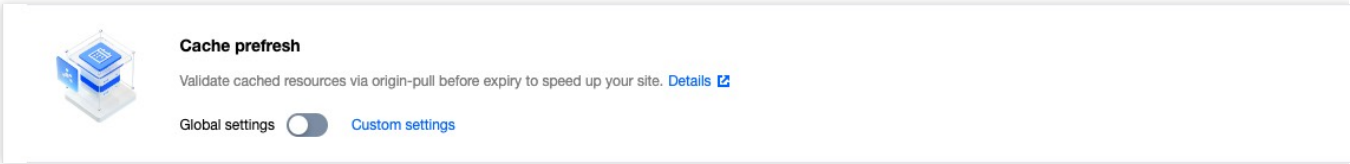
降低回源压力：对于一些热点资源，在过期后可能会引发大量的回源请求。启用缓存预刷新功能，可以将这些回源请求提前进行，减少在资源过期时集中产生大量回源请求，从而降低回源压力。

操作步骤

场景一：针对站点所有域名配置缓存预刷新

若您需要对整个接入站点配置相同的缓存预刷新，或作为站点级兜底配置，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的站点。
2. 在缓存配置页面，选择所需站点，单击**站点加速 > 缓存配置，**找到**缓存预刷新**配置卡片。



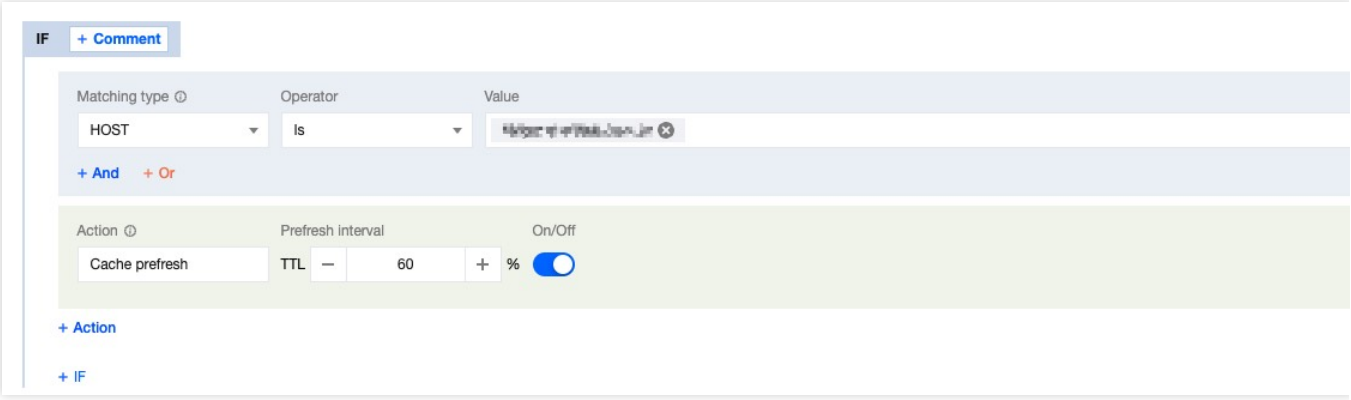
配置状态：默认为开启，可点击滑块关闭。

预刷新时间：占节点缓存 TTL 的百分比，可输入1-99整数。默认90%。

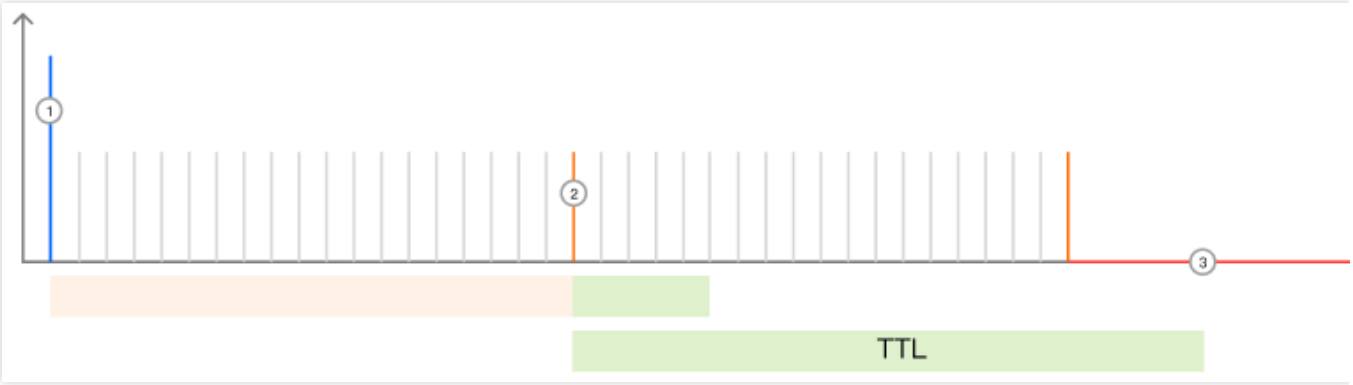
场景二：针对指定域名，路径或文件后缀等请求粒度配置缓存预刷新

若您需要针对不同域名，路径或文件后缀等配置不同的缓存预刷新，例如：针对 `example.com` 站点下的 `www.example.com` 域名配置更提前的预刷新时间 - 60%。可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面，单击**规则引擎。**
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。在规则编辑页面，选择 Host 为匹配类型，配置为 `www.example.com` 。
4. 单击**操作**，在弹出的操作列表内，选择操作为**缓存预刷新**，配置为 TTL 的60%。
5. 完整配置如下所示，单击**保存并发布**，即可完成该规则配置。



附：功能原理



假设指定图片 `test.jpg` 在节点缓存 TTL 为10秒，缓存预刷新时间为 TTL 的 80%（即8秒），则：

1. 节点首次收到客户端请求时，当前节点未缓存该文件，将回源拉取资源并缓存在节点，缓存 TTL 为10秒，在0-7秒内，如果再收到客户端请求，节点直接从缓存中提供资源，正常响应客户端请求；

2. 在该节点内缓存的 `test.jpg` 到达预刷新时间，在第8-10秒时，如果收到客户端请求，节点仍正常响应客户端请求，但同时会异步回源验证缓存资源是否有效；
若资源有效，则更新节点上的资源的缓存 TTL，重置为10秒；
若资源已失效，则从源站获取最新的有效资源至节点，并将其节点缓存 TTL 重置为10秒；
3. 如果超过文件在节点缓存 TTL 时，无客户端请求，则超过缓存 TTL 时间后资源将在节点上过期；
4. 节点下次收到客户端请求，节点会向源站发起回源请求验证资源是否有效，如果出现文件更新则回源拉取最新文件，否则重新缓存该文件并刷新缓存时间为 10秒。

清除和预热缓存
清除缓存

最近更新时间：2024-03-21 09:30:03

功能简介

当您的资源内容缓存至 EdgeOne 边缘节点后，在缓存有效期内，用户在访问该资源时，将直接由 EdgeOne 边缘节点响应，不会触发回源。如果此时您的源站更新了资源内容，为了避免用户仍然访问到旧的资源文件，可以通过清除缓存来手动清除所有边缘节点内已缓存的资源。缓存被清除后，用户在访问资源时， EdgeOne 将回源获取最新的资源以进行响应。

限额说明

不同计费套餐有不同限额，详见：套餐选型对比。

适用场景

以下场景中您可能需要用到此功能：
内容更新：当您在源站上更新了一些资源，但 EdgeOne 上的缓存尚未过期，此时您希望立即让客户端用户看到最新的内容。
错误修复：如果您源站上有一些错误的内容被缓存到了 EdgeOne 上，为避免业务风险，需要立即清除这些错误缓存，确保 EdgeOne 不再提供错误的内容。
测试和调试：在开发和调试网站时，您可能需要频繁地修改和测试网站内容。为了确保看到的是最新更改后的内容而不是缓存的旧版本，您可以使用清除缓存功能。
应急响应：在某些紧急情况下，例如遭受攻击或发布了敏感信息，您需要立即从 EdgeOne 上移除相关内容。
缓存策略调整：当您调整或优化 EdgeOne 缓存策略时，您可能需要清除现有缓存以确保新策略立即生效。

支持类型

EdgeOne 清除缓存支持根据多种类型清除，详情如下：

支持类型	详情
URL	匹配 Url 的节点缓存资源，例如 `https://www.example.com/path/foo.jpg` 。

目录	匹配目录的节点缓存资源，例如 `https://www.example.com/path/` 。
Hostname	匹配 Hostname 的节点缓存资源，例如 `www.example.com` 。不支持提交 `*.test.com` 格式的 URL，即域名中不能包含通配符，需要写明对应的子域名。
Cache-Tag	匹配 HTTP 应答包中 `Cache-Tag` 响应头的标签值（tags）清除缓存，例如 `Cache-Tag: tag1,tag2,tag3` 。 仅适用企业版套餐。 EdgeOne 支持识别源站响应头 `Cache-Tag` ，请添加 tag(s) 至此头部： 头部大小最大为 6KB。 多个 tag 用“,”分隔，单个 tag 不超过128字符，tag 个数上限为1,000。 tag 忽略大小写，即 Tag1 和 tag1 会识别为相同的 tag。
全部缓存	站点在节点的全部缓存资源。 若当前站点（ `example.com` ）下接入了泛域名，例如 `*.foo.example.com` ，则无法针对该泛域名下所有缓存生效，需单独提交其各个具体子域名的清除缓存任务。

EdgeOne 清除缓存分为**直接删除**和**标记过期**两种方式，详情如下：
URL 类型以及 Cache-Tag 类型默认为“直接删除”，即直接删除缓存内容，当用户请求资源时，EdgeOne 会立即回源获取最新资源，短时间内增加回源请求量并减弱加速效果。如果提交的内容较多，源站会有较大压力。
其它清除类型默认为“标记过期”，即不会直接删除节点缓存，而是将缓存标记为过期。若节点缓存了 `Last-Modified` 和 Etag 头部，那么下次用户请求时，节点会携带 `If-None-Match` 和 `If-Modified-Since` 回源校验资源是否有更新，响应 304 还是 200 是由源站决定的，一般来说：
若无更新 - 源站返回 304（Not Modified），则节点继续使用缓存响应，可有效节省回源带宽；
若有更新 - 源站返回 200（OK），则节点从源站获取最新资源，并对比缓存资源和新资源的 `Last-Modified` 和 `Content-Length` ，任一头部值不同则用新资源覆盖节点过期缓存。

注意

EdgeOne 节点默认针对编码前和编码后的 URL 区分缓存，因此，刷新时，也需要分别提交。若 URL 含有特殊字符，那么在控制台提交 URL 刷新：
https://example.com/d/default_avatar.png?x-oss-process=image/resize,w_600,1_800，则会刷新编码前 URL 对应的缓存；
https://example.com/d/default_avatar.png?x-oss-process=image%2Fresize%2Cw_600%2C1_800，则会刷新编码后 URL 对应的缓存。

操作步骤

场景一：通过输入内容清除缓存

若您需要清除的内容不多，方便直接于输入框中输入内容，可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。

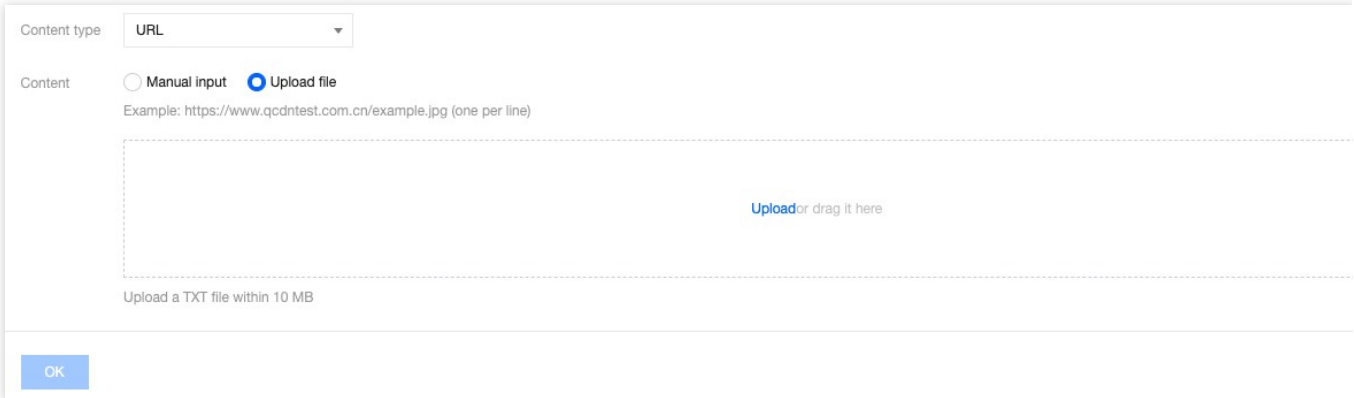
2. 在站点详情页面，单击**站点加速 > 清除缓存**。
3. 在清除缓存页面，选择需清除的资源类型，输入对应的资源内容，单击**确定清除**。

4. 切换至历史记录 Tab 页，可查看指定时间段（近一个月内）和清除类型的历史记录。

场景二：通过上传文件批量导入内容清除缓存

若您需要清除的内容较多或已将内容放置在一份文件中，可选择上传文件的方式：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面，单击**站点加速 > 清除缓存**。
3. 在清除缓存页面，选择需清除的资源类型，选择“上传文件”方式，上传后单击**确定清除**。



4. 切换至历史记录 Tab 页，可查看指定时间段（近一个月内）和清除类型的历史记录。

相关参考

清除缓存和预热缓存每次提交内容后需要多久才能生效？

预热缓存

最近更新时间：2023-12-18 15:25:58

功能简介

当业务发布新资源时，客户端首次请求这些资源可能会遇到 EdgeOne 上没有缓存的情况，导致无法立即响应，需要回源获取。预热缓存功能允许预先将资源缓存到 EdgeOne。这样，即使客户端首次请求，也可以直接从 EdgeOne 的缓存中响应，无需回源。预热缓存的实现方式是提交需要预热的 URL，然后将匹配这些 URL 的资源从源站提前缓存到 EdgeOne ，从而提升加速效果并缓解源站压力。

限额说明

不同计费套餐有不同限额，详见 套餐选型对比。

使用场景

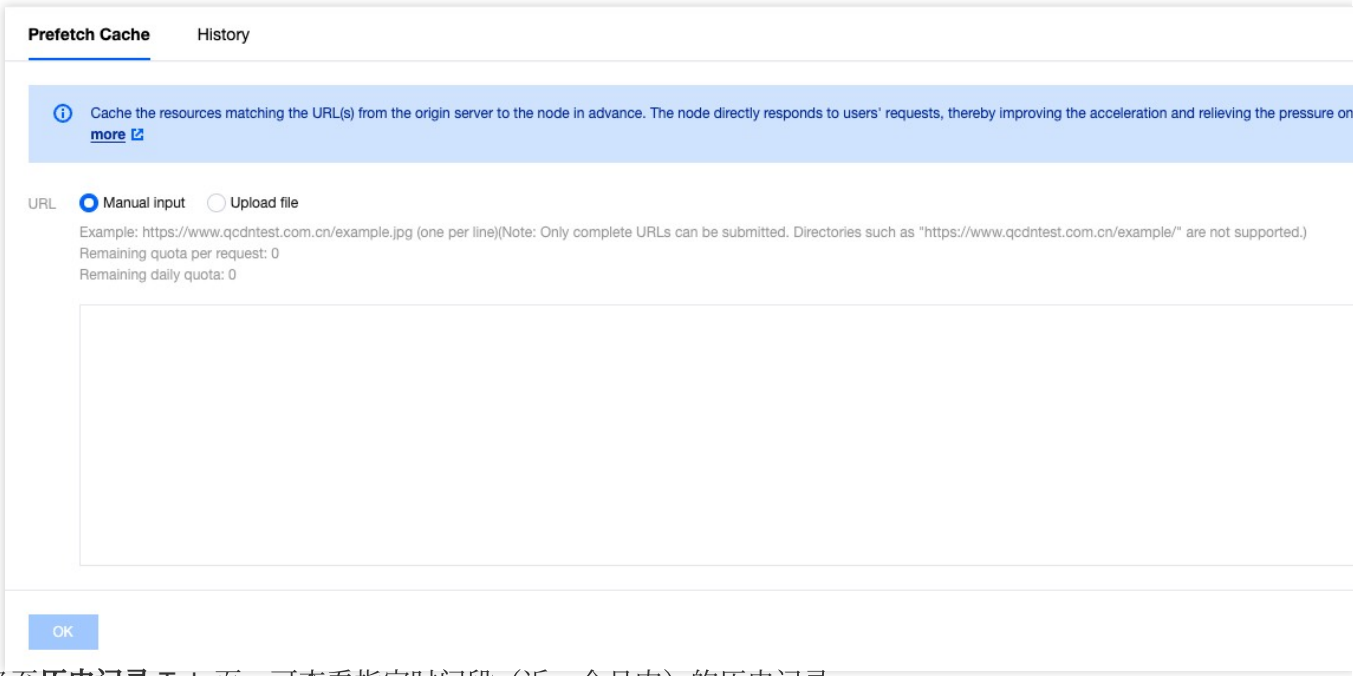
以下场景中您可能需要用到此功能：
新发布的内容：当您的业务发布了新的内容或更新了现有内容时，您希望确保这些内容 EdgeOne 上立即可用，以便客户端用户能够在第一时间访问到最新的内容，以减少首次访问时的延迟。例如游戏业务正式对外发布新版本安装包或升级包前，可将安装包资源预热至 EdgeOne 。正式发布后，用户请求下载这些安装包时，可直接从节点获取安装包资源，提升下载速度。
大型活动运营：在大型活动之前，您希望确保活动的关键资源已经在 EdgeOne 上缓存，这样有助于确保活动开始时，客户端用户能够快速访问到所需的内容，降低因高流量导致的延迟和拥塞。
预期的流量高峰：如果您预计在某个特定时间段内网站流量会有显著增长（例如，节假日促销、新闻发布等），可以使用预热缓存功能来确保边缘节点上已经缓存了关键资源。这有助于分散高峰时段的回源请求压力，提高客户端用户的访问速度。
注意：
预热资源时会模拟请求，回源拉取对应资源，若提交的预热任务较多，则会产生较多回源请求，源站带宽增大。
若预热的资源与节点缓存有冲突，即若 EdgeOne 已缓存同名资源，且尚未过期，则仍有效，不会被预热的资源覆盖。如同名资源有变动，您可在预热前先清除对应的节点缓存。

操作步骤

场景一：通过输入内容预热缓存

若您需要预热的内容不多，方便直接于输入框中输入内容，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**站点加速** > **预热缓存**。
3. 在预热缓存页面，输入对应的资源内容，单击**确定预热**。

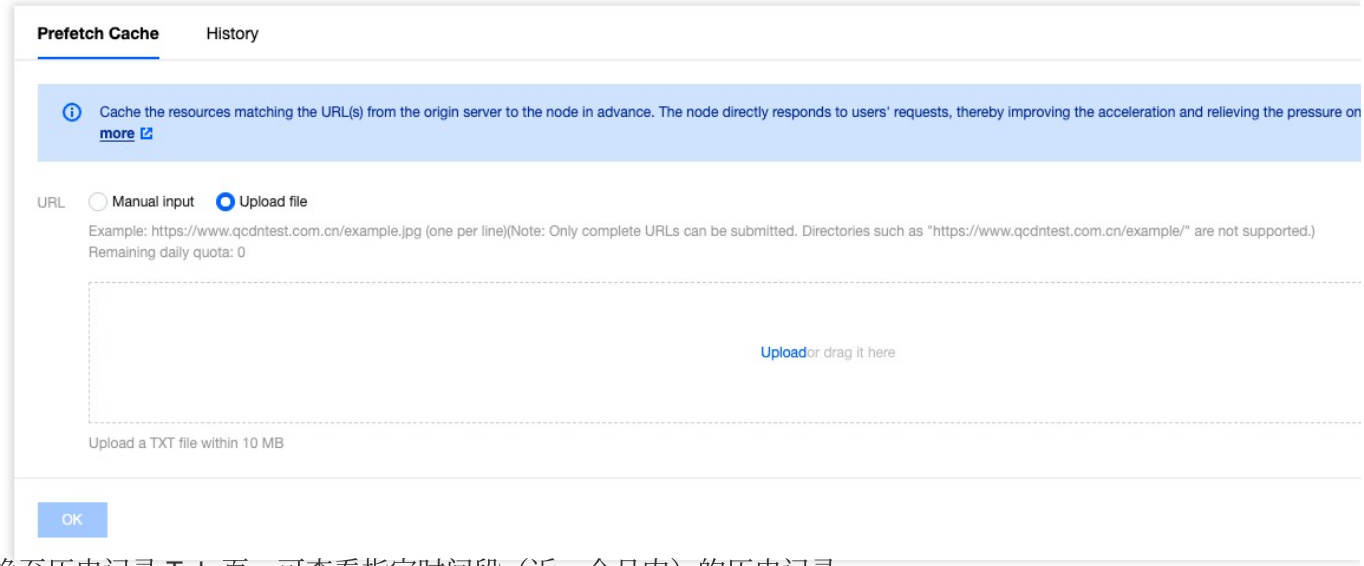


4. 切换至**历史记录** Tab 页，可查看指定时间段（近一个月内）的历史记录。

场景二：通过上传文件批量导入预热缓存内容

若您需要预热的内容较多或已将内容放置在一份文件中，可选择上传文件的方式：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**站点加速** > **预热缓存**。
3. 在预热缓存页面，选择“上传文件”方式，上传单击**确定预热**。



4. 切换至历史记录 Tab 页，可查看指定时间段（近一个月内）的历史记录。

相关参考

清除缓存和预热缓存每次提交内容后需要多久才能生效？

如何提高 EdgeOne 的缓存命中率

最近更新时间：2023-08-08 14:31:08

本文介绍了如何合理利用 EdgeOne 上的各项配置，结合您的实际业务场景进行调优，提高站点内文件的缓存命中率。

背景介绍

当您的站点接入 EdgeOne 并开启加速后，当用户访问静态资源如图片、视频时，EdgeOne 将会把响应的静态文件缓存在边缘节点中，当有其他用户发起重复请求时，将会由边缘节点直接响应请求，避免回源请求。

如果缓存的命中率过低，会造成用户大量请求回源，则会给源站带来大量的处理压力，降低了用户的访问体验和站点加速效果。您可以通过以下配置调优，来优化提升缓存命中率。

说明：

如果您需要查看当前缓存的命中分析，可以在控制台上，通过**数据分析 > 缓存分析**来查看，详情请参见 缓存分析。

优化方式

1. 调节节点缓存 TTL 配置

节点缓存 TTL 的配置将直接影响 EdgeOne 是否缓存指定的文件资源以及对应的缓存时间，如果文件缓存策略为不缓存或者在节点上的缓存时间较短，则会导致用户访问未命中缓存，频繁回源，降低访问体验。

EdgeOne 默认情况下针对全局站点开启了默认缓存规则，您可以通过查看 EdgeOne 内容缓存规则 来了解 EdgeOne 的缓存规则是如何生效的，为了提高缓存的命中率，建议您可以在规则引擎内根据文件后缀单独配置缓存规则。

建议配置

建议您在不同场景下根据不同的文件类型配置个性化的缓存规则：

1. 不常更新的文件，例如：下载资源、视频文件等内容，建议在 EdgeOne 上单独配置自定义的缓存时间，缓存时间配置在30天以上，且通过 EdgeOne 节点强制缓存；常见的下载资源及视频文件格式如下：

音视频	mp4;mp3;ts;m4a;avi;m4s;ogg;mkv;mov;flv;rm;rmvb;swf;wav;wmv;rmi;aac
压缩包	rar;7z;zip;gzip;dmg;gz;ios;tar;jar;br;bz2
文档	doc;docx;xls;xlsx;pdf;ppt;pptx
应用程序	apk;exe;bin
其他	vsv;iso;jar;swf;chunk;atlas

2.经常更新的文件内容，例如：图片内容，如果缓存时间过长，可能会因为命中缓存导致用户访问时获取到过期的内容。因此，建议在 EdgeOne 上单独配置自定义的缓存时间，一般缓存时间根据业务需求配置，可配置1-7天。常见的图片内容格式如下：

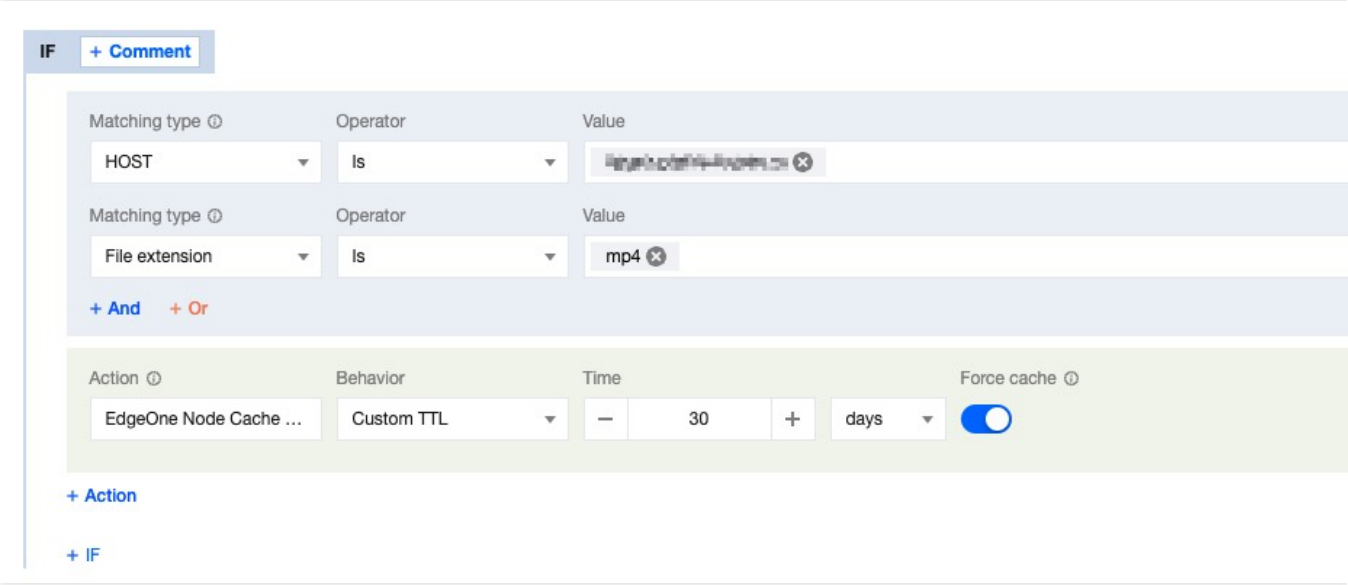
图片	jpg;png;jpeg;webp;gif;heif;heic;kpg;ico
网页	html;htm;shtml;hml;js

3.动态文件，例如：php、json文件等，如果被缓存将会导致用户访问无法正确响应内容。因此，建议在 EdgeOne 上单独配置为不缓存。常见的动态文件格式如下：

动态资源	php;aspx;asp;jsp;do;dwr;cgi;fcgi;action;ashx;axd;json

优化示例

当您在缓存分析中查看资源的命中率较低，可以在右侧查看具体的文件后缀，查看哪些类型的资源命中率出现了大量的 miss。例如：当前缓存分布中，`.mp4` 格式文件存在较多未命中缓存。
如果您完全按照 EdgeOne 的默认缓存规则，存在的问题是因为响应文件时，未响应 Cache-Control 头部，根据默认缓存规则，`.mp4` 文件在节点上的缓存时间为2小时，因为缓存时间短，导致该文件将频繁回源。如果您的需要缓存该文件，可以前往规则引擎，新增一条规则，设定当文件后缀等于mp4时，节点缓存 TTL 自定缓存30天，且开启强制缓存，即无视源站响应的CC头，节点强制缓存该文件。详细操作步骤可查看：节点缓存 TTL。



2. 自定义缓存键 Cache Key 将同一类型请求指向一份缓存文件

默认情况下，EdgeOne 将根据用户访问的 URL 和查询字符串生成该缓存键的唯一标识符，作为该文件的Cache Key，当有相同请求时，边缘节点将通过比对请求是否与缓存内的 Cache Key 一致来判断是否命中该缓存。如果

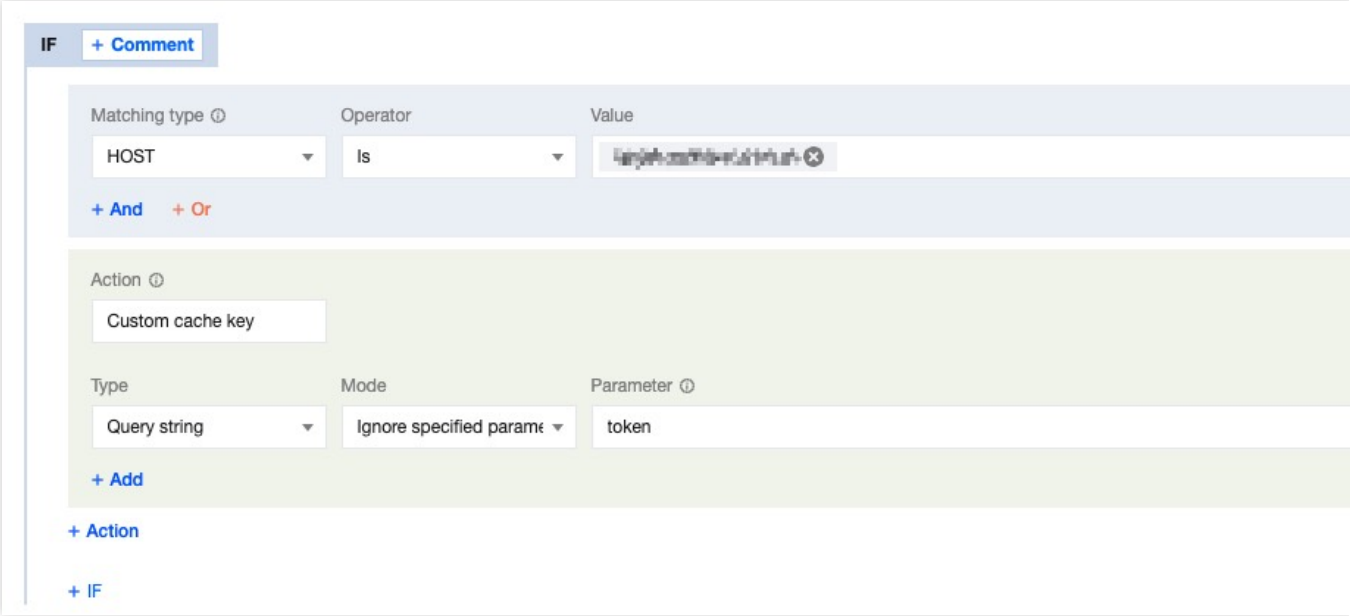
URL 内携带了不影响文件版本的动态参数内容，比如用户标识 ID，则将根据该参数的不同建立了多份缓存，导致缓存命中率下降，您可以通过自定义缓存键 Cahche Key 来优化。

建议配置

当请求 URL 中携带的部分参数不会影响文件的版本时，建议通过保留或者忽略指定的参数内容，来提高缓存命中率。

优化示例

当前请求URL为：`https://image.example.com/test.jpg?version=1.1&token=1234567890`，其中参数 `version=1.1` 将影响图片的内容，`token=1234567890` 则不影响，为了提高缓存的命中率，可以在自定义 Cache Key 中忽略 token 参数。详细操作可查看：自定义 Cache Key。



3. 预热缓存

预热缓存可以让 EdgeOne 提前将文件缓存至边缘节点中，当有用户访问时，可直接命中缓存，减少首次并发回源量，提高文件的缓存命中率。当您新增站点至 EdgeOne 时，或者有新增的热门资源发布时，建议提前预热缓存，详细操作可参考：预热缓存。

4. 开启缓存预刷新

当您的文件主要是热点文件时，如需保障该文件能在节点内持续有缓存，您可以通过开启缓存预刷新，在节点文件缓存过期之间，用户请求节点内文件时，即向源站校验该文件是否出现更新，未更新则刷新该文件在节点内的缓存时间。详细操作可参考：缓存预刷新。

5. 合理利用 Vary 机制

当源站响应了 Vary 头时，CDN将根据 Vary 头指定的内容来进行区分缓存，Vary 的原理说明详见 Vary 特性。如果当前文件不需要通过 Vary 头来控制文件的缓存版本，建议您在源站响应时，避免响应该头部，以减少建立的缓存版本，提高缓存命中率。

了解更多

如何判断用户请求是否命中 EdgeOne 节点缓存

文件优化
智能压缩

最近更新时间：2023-12-15 09:47:56

功能简介

EdgeOne 默认全局开启 Gzip 或 Brotli 压缩，当客户端请求头中携带有 `Accept-Encoding：br,gzip` 或者 `Accept-Encoding：br` 或者 `Accept-Encoding：gzip` 时，节点将根据文件的 `Content-Type` 来进行智能压缩，压缩后文件可以有效地减少资源的大小，加快内容的传输速度。开启智能压缩可以帮助您：

1. **提高用户体验**：通过减少资源大小，可以显著提高网页加载速度，从而提供更好的用户体验。特别是对于拥有大量 CSS、JavaScript 等资源的网站，启用压缩可以大幅度减少加载时间。
2. **节省流量**：压缩后的资源会占用更少的网络流量，这将有助于降低运营成本。

操作步骤

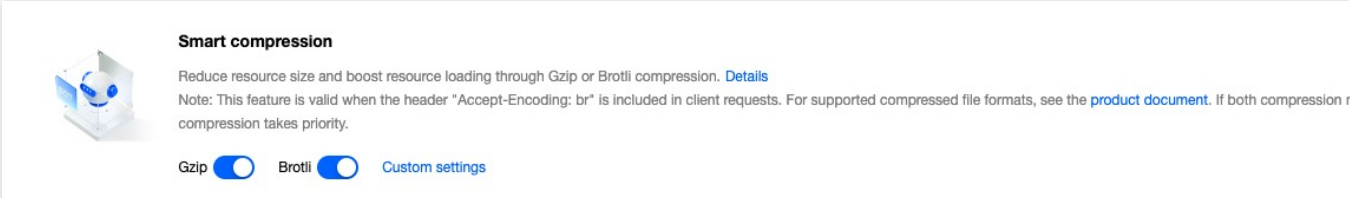
说明：

默认情况下，您无需修改该配置。如果在某些场景中，例如：当前客户端会对文件进行 MD5 校验或者当前客户端不支持解析指定压缩文件，您希望当前站点只使用 Brotli 压缩或者 Gzip 压缩，或者不进行压缩，可以参考以下步骤操作。

场景一：站点所有域名开启/关闭智能压缩

若您需要对整个接入站点开启/关闭智能压缩，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速 > 文件优化**，进入网络优化详情页面。
3. 找到智能压缩配置卡片，默认为全部开启状态，单击**开关**可配置开启/关闭。

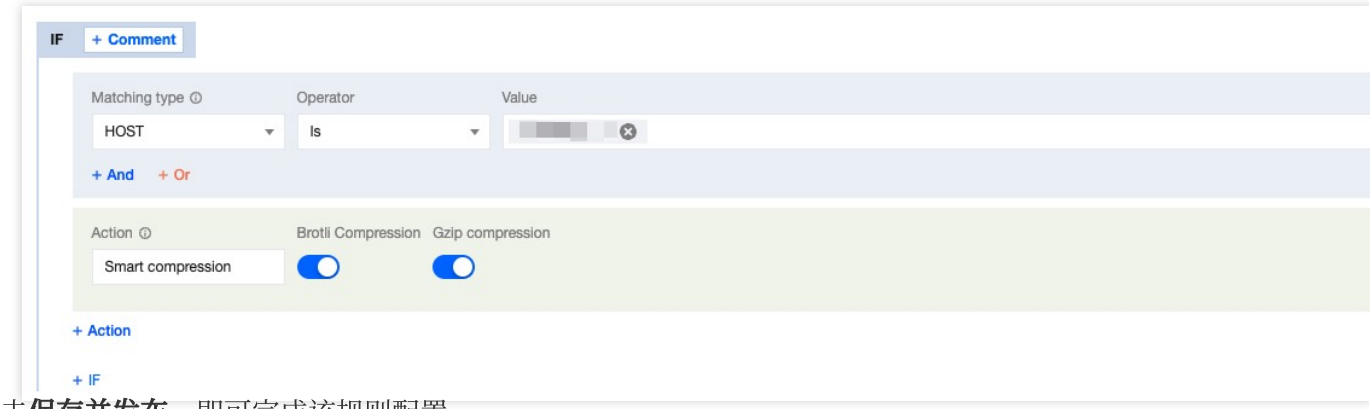


场景二：针对指定域名开启/关闭智能压缩

若您只需要针对指定域名开启/关闭智能压缩，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。

2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，选择 Host 匹配类型以匹配指定域名的请求。
5. 单击**操作** > **选择框**，在弹出的操作列表内，选择操作为 **智能压缩** ，单击**开关**开启/关闭 Gzip 或 Brotli 压缩 。

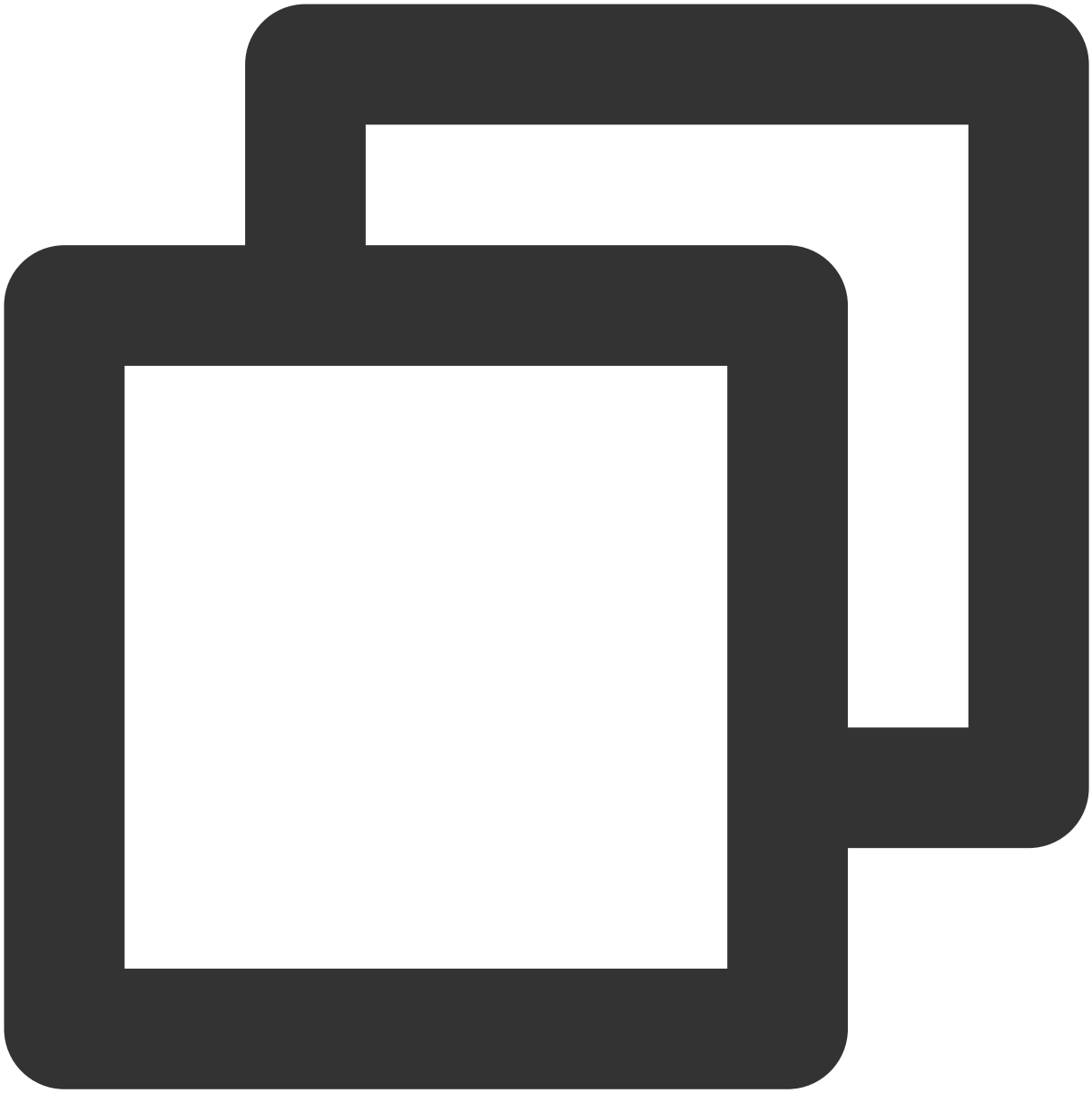


6. 单击**保存并发布**，即可完成该规则配置。

相关参考

智能压缩生效规则

1. 智能压缩支持的文件大小范围：256B - 30MB。
2. 智能压缩为同步压缩，在回源获取文件同时压缩文件，首次请求压缩文件时节点可直接响应压缩后的文件。
3. 智能压缩默认根据 `Content-Type` 压缩，支持以下类型：



```
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. 如果您当前同时开启 Gzip 压缩和 Brotli 压缩，且客户端请求头 `Accept-Encoding` 同时携带有 br 和 gzip 时：

如果节点内已有缓存内容，则按照如下规则响应：

若节点同时有 Brotli 和 gzip 压缩的缓存内容，则优先响应 Brotli 压缩。

若节点仅有 Brotli 压缩的缓存内容，则优先响应 Brotli 压缩。

若节点仅有 gzip 压缩的缓存内容，则优先响应 Gzip 压缩。

如果节点内未有缓存内容，则优先响应 Brotil 压缩。

5. 仅开启 Brotli 压缩时，若请求压缩头为 gzip，则压缩不会生效，将返回原始资源；仅开启 Gzip 压缩时，若请求压缩头为 br，则压缩不会生效，将返回原始资源。
6. 若源站开启了压缩功能，且服务端携带响应头：`Content-Encoding`，则智能压缩功能将不再生效。

请求示例

未开启智能压缩

首次请求 gzip 压缩文件，未命中节点缓存，回源获取原文件并缓存至节点，EdgeOne 响应原文件：

```
> 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
```

开启智能压缩

首次请求 gzip 压缩文件，未命中节点缓存，回源获取文件，节点同步压缩并缓存压缩后的文件，EdgeOne 响应压缩文件：

智能压缩支持 chunk 流式压缩，若请求未命中节点缓存，回源获取文件后会以 chunk 的方式响应。

```
> 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
```

再次请求，命中节点 gzip 压缩文件的缓存，节点直接响应压缩文件。

```
> 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
```

媒体处理
图片处理

最近更新时间：2023-12-18 09:51:43

功能简介

图片处理功能允许您根据需求调整图片尺寸和转换图片格式。通过 EdgeOne 边缘服务器直接处理、缓存和响应图片，您的业务源站只需存储原始图像，从而降低了图片管理成本。 EdgeOne 边缘服务器在不影响视觉感受的前提下压缩图片，以提高页面加载速度并优化图片加速性能，在保持图像质量的同时，增强用户体验。

支持的图片处理能力

重置大小

能力	参数名	参数值（type/pixel）	说明
重置大小	eo-img.resize	`w/<Width>`，例如：`w/100`	指定宽度，高度自适应
		`h/<Height>`，例如：`h/100`	指定高度，宽度自适应
		`w/<Width>/h/<Height>`，例如：`w/100/h/100`	指定宽高
		`l/<Long>`，例如：`l/100`	指定长边，短边自适应
		`s/<Short>`，例如：`s/100`	指定短边，长边自适应

格式转换

支持通过携带指定参数将原图转换为指定格式

能力	参数名	支持输入格式	支持输出格式
格式转换	eo-img.format	静态图片：jpg、png、bmp、jp2、jxr、gif、webp、avif、heif	均静态：jpg、png、bmp、jp2、jxr、gif、heif、webp、avif
		动态图片：gif、webp、avif、heif	静态：jpg、png、bmp、jp2、jxr（取 gif 动画首帧作为单一静态图像） 动态：gif、webp、avif、heif

限制说明

处理图片原图大小不超过32MB。
输入的原图宽、高不超过30000像素且总像素不超过2.5亿像素；针对动图，原图宽 x 高 x 帧数不得超过2.5亿像素。
输入的 gif 格式动画帧数不超过300帧。
输出的图片宽、高设置不得超过9999像素。
注意：
以下任意情况，可能导致图片处理失败，返回原图：
1. 图片处理原图和结果图像的任何一个参数超过以上限制，我们将无法进行图片处理，而只能响应原图。
2. 输入错误的请求参数时，图片将不会被处理，直接返回原图，例如以下几种情况：
重复输入参数：`eo-img.resize=w/100&eo-img.resize=w/200`，将视为非法传参；
拼写错误：任何格式错误或拼写错误的参数，例如 `eo-img.resize=w=100`，都会被视为非法传参；
重置大小参数错误：参数 `w/`（宽度）和 `h/`（高度）不应与 `s/`（短边）和 `l/`（长边）混用。例如：`w/300/s/200` 是非法传参，图片将保持原状。
3. 若在控制台内关闭了图片缩放功能，则所有 eo-img 相关参数被视为普通查询字符串，不会触发图片处理功能。
4. 若出现其他异常情况导致无法正常处理图片，我们会优先提供原始图片，在后续的请求中，我们会自动尝试重新进行图片处理。

计费说明

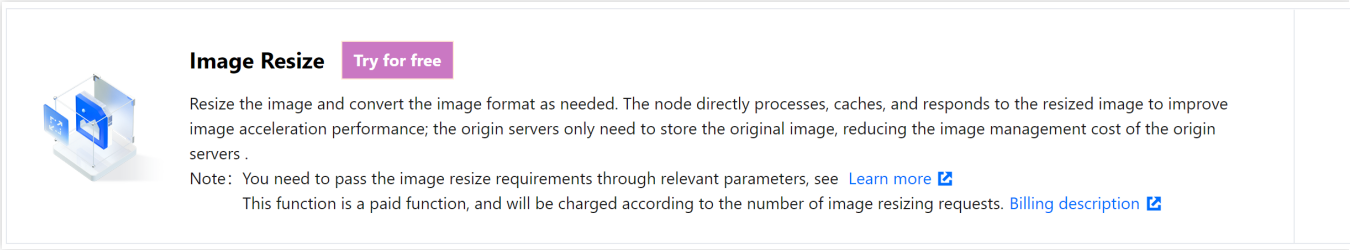
此功能为收费功能，将根据图片缩放请求数计费，详见 增值服务用量单元费用（后付费）。
说明：
此功能正在限时免费，请关注后续计费通知。

使用方式

您可以直接通过在客户端请求 URL 中拼接图片处理相关参数来实现图片缩放，可参考以下步骤了解如何使用。
如果您不希望变更原始客户端请求 URL，也可以通过 EdgeOne 边缘函数的能力，自动根据客户端请求信息，自适应缩放或转换图片格式。使用方式请参见：通过边缘函数实现自适应图片格式转换。

在客户端请求 URL 中增加图片处理参数

如果您需要通过在客户端请求 URL 中增加相关参数实现图片缩放，可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**站点加速 > 媒体处理**。
3. 在媒体处理页面，单击**图片缩放**的“开关”开启，即可开始使用。



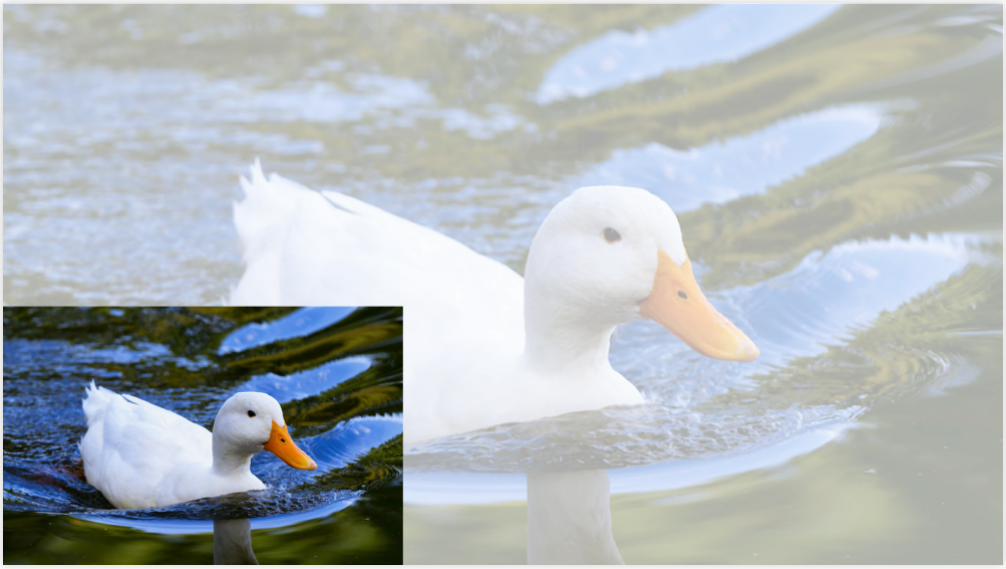
4. 开启后，您只需要通过在客户端请求 URL 后拼接 `eo-img` 相关参数传递图片缩放需求，EdgeOne 将自动根据客户端请求 URL 内的图片处理参数完成图片处理。例如: `https://www.example.com/foo.png?eo-img.resize=w/100` 。

图片处理示例

处理的原图为 500*280，500 KB，处理示例如下：

1. 指定宽度为200px，高度自适应。

请求 url： `http://www.example.com/foo.png?eo-img.resize=w/200` 。



2. 指定高度为200px，宽度自适应。

请求 url： `http://www.example.com/foo.png?eo-img.resize=h/200` 。

3. 指定宽度为300px，高度为200px。

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

注意：

同时指定宽高，会按照指定的值缩放，不再保持原图长宽比。

4. 指定长边为400px，短边自适应。

请求 url： `http://www.example.com/foo.png?eo-img.resize=l/400` 。

5. 指定短边为200px，长边自适应。

请求 url： `http://www.example.com/foo.png?eo-img.resize=s/200` 。

6. 指定图片转换格式为 webp。

请求 url： `http://www.example.com/foo.png?eo-img.format=webp` 。

输出图片格式：webp。

7. 指定宽度为200px，高度自适应，并转换格式为 webp。

请求 url： `http://www.example.com/foo.png?eo-img.resize=w/200&eo-img.format=webp` 。

网络优化

HTTP/2

最近更新时间：2024-01-02 10:04:48

什么是 HTTP/2？

EdgeOne 支持客户端以 HTTP/2 协议发起请求。HTTP/2（即 HTTP 2.0，超文本传输协议第2版），是 HTTP 协议的第二个主要版本，能有效减少网络延迟，提高站点页面加载速度。

注意：

1. 若客户端请求不使用 HTTP/2，EdgeOne 兼容 HTTP 1.x 协议访问。
2. 配置访问请求请参考此文档，若需配置 HTTP/2 回源，请参见 HTTP/2 回源。
3. 自 2023 年 11 月 23 日起，出于安全考虑（详情请参见关于针对 HTTP/2 协议漏洞的 DDoS 攻击防护），对于增量站点，HTTP/2 默认保持为关闭，用户可按需开启。

前提条件

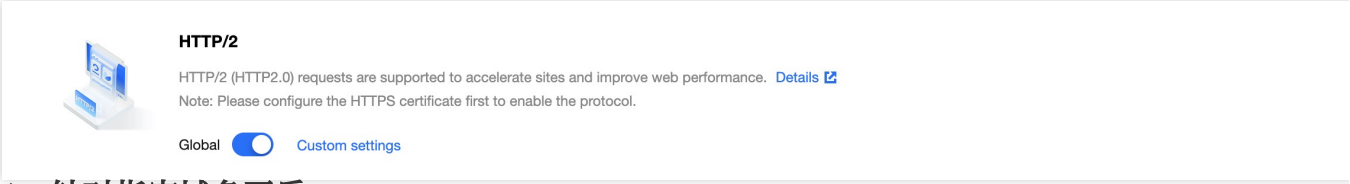
当前站点的访问域名，均已配置 SSL 证书。如何配置 SSL 证书请参考：证书配置。

操作步骤

场景一：针对站点所有域名修改 HTTP/2 支持

若您需要对整个接入站点开启或关闭 HTTP/2，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速 > 网络优化**，进入网络优化详情页面。
3. 找到 HTTP/2 配置卡片，默认为关闭状态，单击**开关**可配置开启/关闭。



场景二：针对指定域名开启 HTTP/2

若您只需要针对指定域名开启或关闭 HTTP/2，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。

3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，选择 Host 匹配类型以匹配指定域名的请求。
5. 单击**操作** > **选择框**，在弹出的操作列表内，选择操作为 **HTTP/2**，单击**开关**配置开启/关闭。

6. 单击**保存并发布**，即可完成该规则配置。

HTTP/3(QUIC)
概述

最近更新时间：2023-12-15 15:02:59

什么是 HTTP/3

HTTP/3 （HTTP over QUIC）是下一代的互联网传输协议，为了解决 HTTP/2 中存在的队头阻塞问题，HTTP/3 不再基于 TCP 建立，改为使用基于 UDP 协议的 QUIC 协议实现。与 HTTP/1.1 和 HTTP/2 相比，HTTP/3 提供了更快的连接建立和数据传输速度，同时支持多路复用，可以同时传输多个请求和响应，并在阻塞控制、头部压缩等方面做了提升，从而显著减少了延迟和等待时间，提高了网站的性能和用户体验。

EdgeOne 对 HTTP/3 的支持

EdgeOne 已同时支持 HTTP/3 和 QUIC 协议，其中 QUIC 协议支持以下版本：

标准	支持的版本
Google QUIC（gQUIC）	Q039、Q043、Q046、Q050
IETF QUIC（iQUIC）	draft-27、draft-29、RFC 9000

如何在您的 App 中使用支持 HTTP/3 或 QUIC 访问

如果您的用户需要通过 App 应用访问，需要 App 内集成支持 HTTP/3 或 QUIC 协议，EdgeOne 提供了 QUIC SDK 可帮助您快速完成集成。详情可参见 QUIC SDK 下载和集成示例。
如果您的站点是以网站为主，用户当前浏览器支持 QUIC 协议并启用了 QUIC 协议访问。您的站点只需要在 EdgeOne 上启用 HTTP/3即可支持。您可以通过 检测当前浏览器是否支持 HTTP/3 来查看当前浏览器是否支持。

启用 HTTP/3

最近更新时间：2023-12-18 10:57:34

本文介绍了 EdgeOne 平台内如何启用支持 HTTP/3 协议支持。

注意：

1. 如需开启 HTTP/3 支持，需当前访问域名已 配置 HTTPS 证书，才可生效。
2. 若同时开启 HTTP/2 和 HTTP/3 ，则根据实际客户端请求使用 HTTP/2 或 HTTP/3。

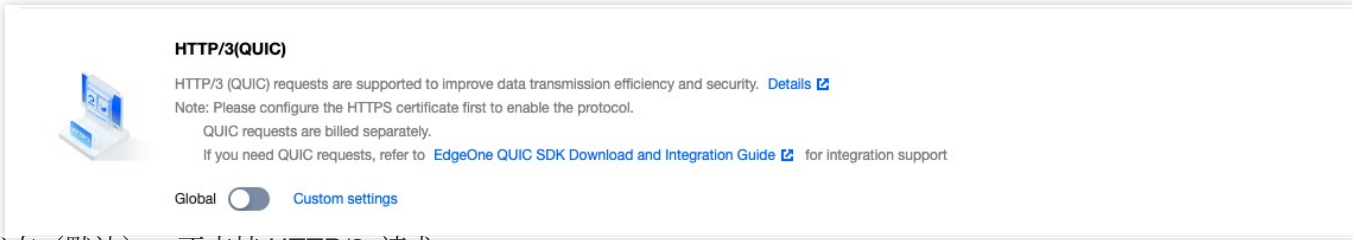
计费说明

HTTP/3 为收费功能，具体的费用标准可参考：增值服务费-QUIC 请求

场景一：针对站点所有域名开启支持 HTTP/3 协议

若您需要针对所有域名配置启用 HTTP/3 协议，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速** > **网络优化**，进入网络优化详情页面。
3. 单击 HTTP/3 模块的“开关”，开启或关闭 HTTP/3 功能。



关闭状态（默认）：不支持 HTTP/3 请求。

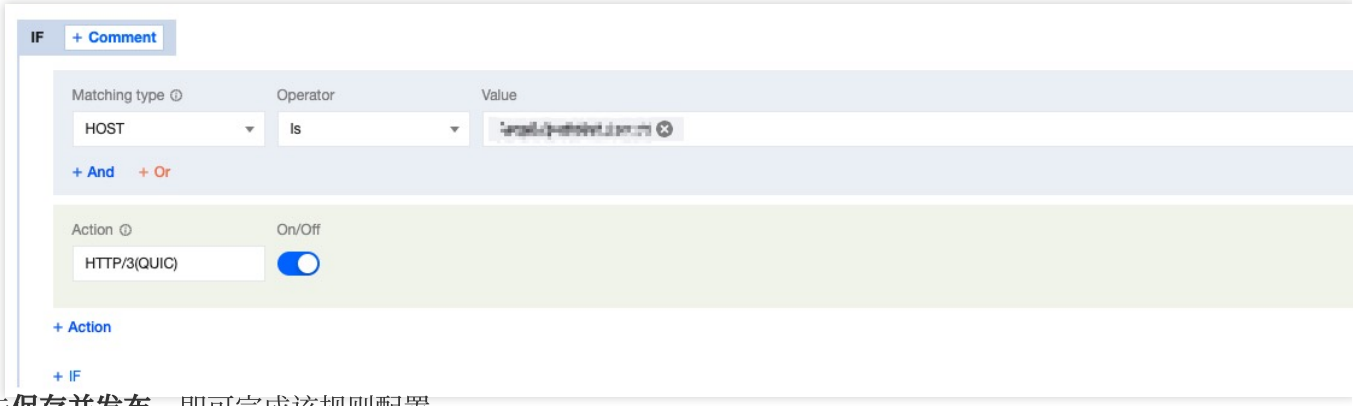
开启状态：支持 HTTP/3 请求，使用 HTTP/3 加速站点请求。

场景二：针对指定域名开启支持 HTTP/3 协议

若您需要针对不同域名配置启用 HTTP/3 协议，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点。**
2. 在站点详情页面，单击**规则引擎。**
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，选择 Host 匹配类型以匹配指定域名的请求。

5. 单击**操作** > **选择框**，在弹出的操作列表内，选择操作为 HTTP/3，将开关切换为开启。



6. 单击**保存并发布**，即可完成该规则配置。

QUIC SDK
SDK 概览

最近更新时间：2023-06-29 11:20:56

腾讯云 EdgeOne QUIC SDK 是一个基于 QUIC 协议的开发工具包，它提供了简单易用的 API 接口，能够帮助开发者更快速地集成 QUIC 协议到自己的应用中，通过 QUIC 协议发起请求，为应用提供稳定、高质量的网络传输。当前已支持 Android 和 iOS 平台。

什么是 QUIC

QUIC（Quick UDP Internet Connections）是一种通用、安全、多路复用的传输层新型网络协议。目前标准的 HTTP/3 协议正是基于 QUIC 实现的，QUIC 支持 0-RTT 建立连接和无队头阻塞的多路复用，可以比较容易的在端侧实现用户态拥塞控制，更大限度地利用网络带宽进行实际的数据传输，在丢包率和网络延迟较高的弱网环境也可提供高质量的数据通信。同时，QUIC 还支持连接迁移，在移动端频繁切换网络的场景中，也可平滑过渡，保证网络不中断。

计费说明

当前 EdgeOne QUIC SDK 处于公测期间，暂不收费。

协议版本支持

EdgeOne QUIC SDK 同时支持 IETF QUIC 和Google QUIC，各版本支持如下：

标准	支持的版本
Google QUIC（gQUIC）	QO43、Q046、Q050、Q051
IETF QUIC（iQUIC）	draft-29、RFC 9000

SDK 下载和集成指引

最近更新时间：2024-05-08 21:35:46

SDK下载

腾讯云 EdgeOne 的 QUIC SDK 支持 iOS 和 Android 操作系统。

Android ZIP下载

iOS ZIP下载

接入步骤

Android SDK 接入

iOS SDK 接入

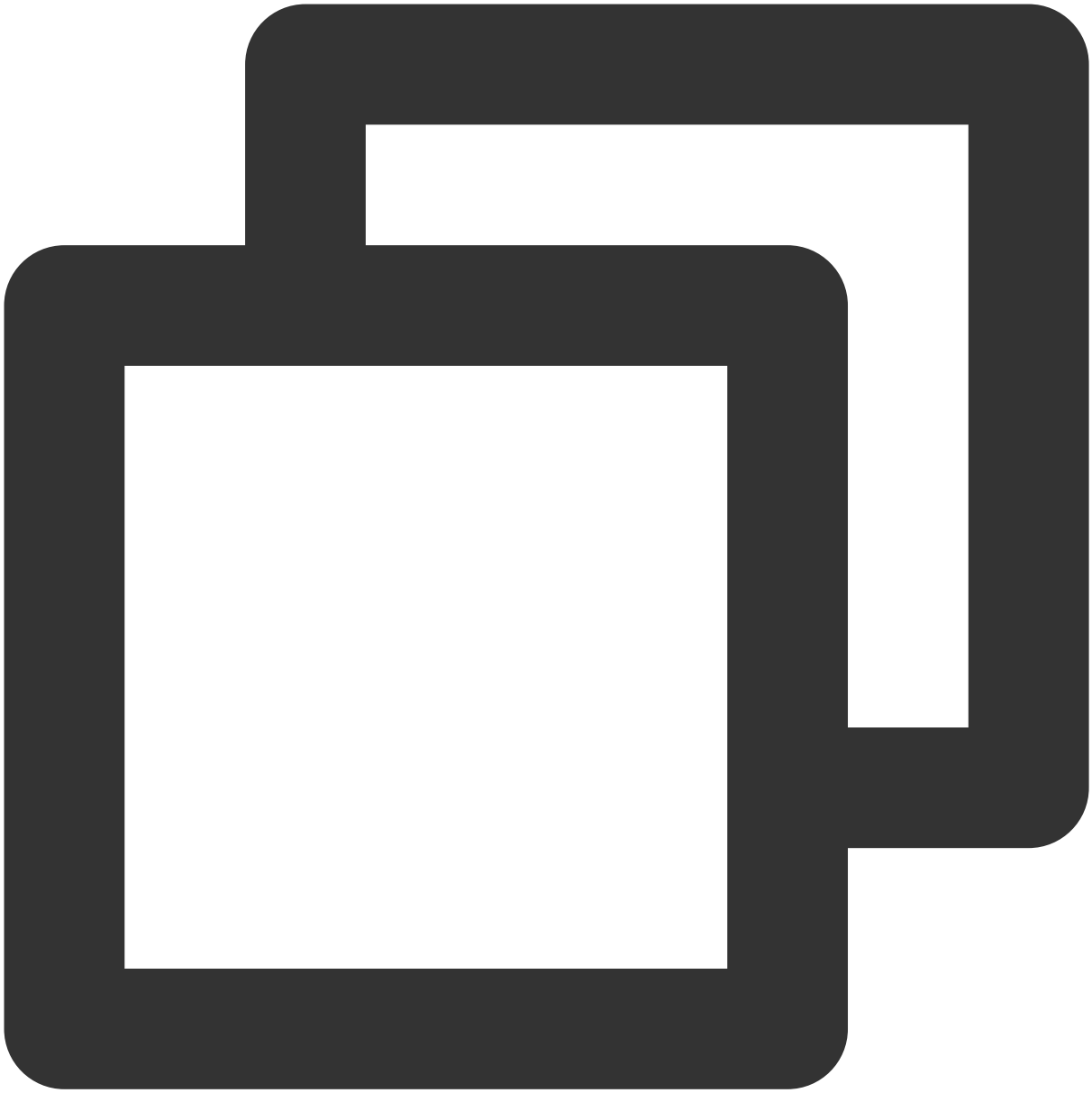
1. 开发环境要求。

Android Studio 2.0+

Android 5.0（SDK API 21）及以上系统

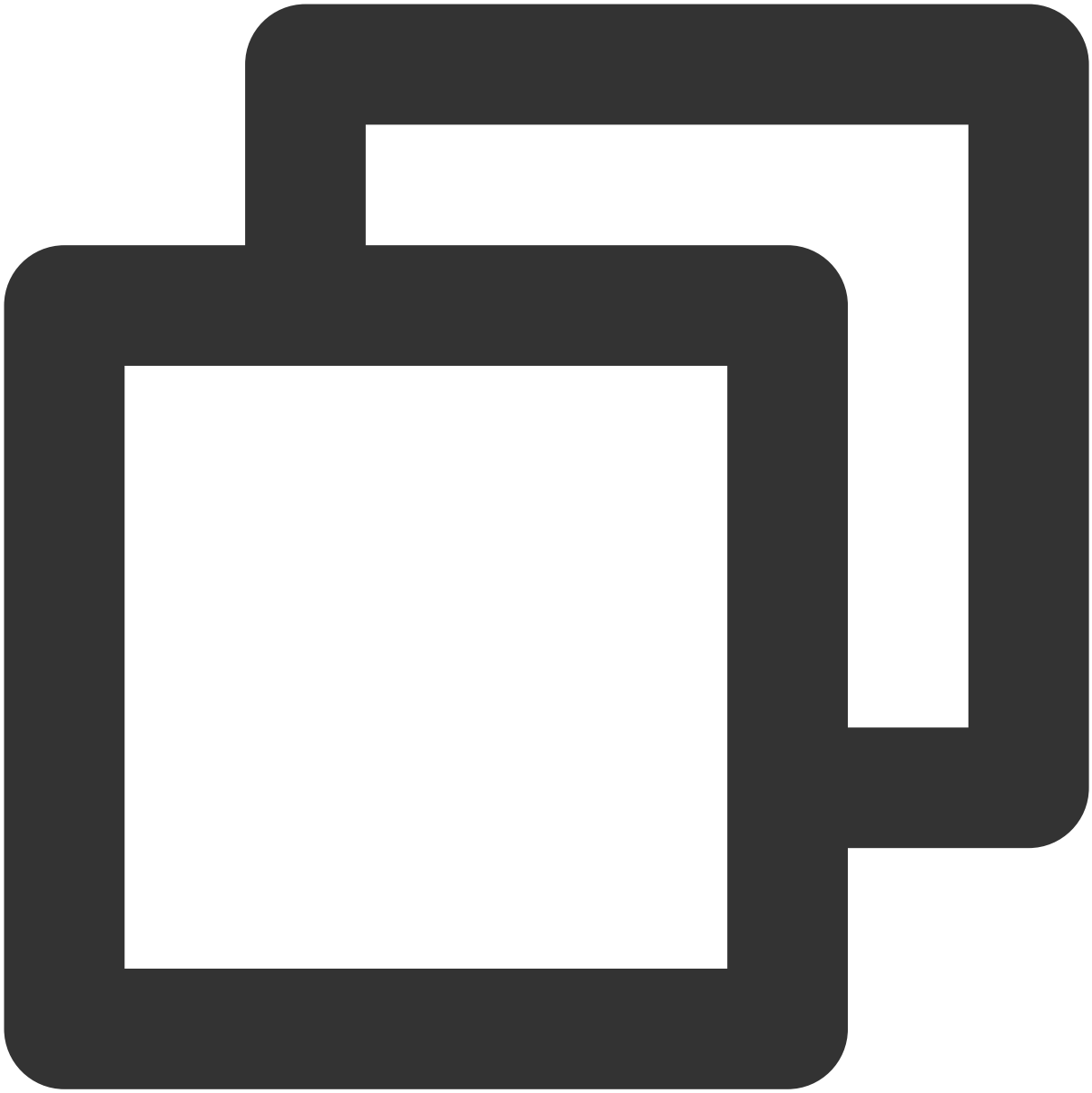
2. 下载 `TQUIC_Android_SDK.zip` 并解压，将目录中的 `AAR` 文件复制到工程目录下（app/libs）。

3. 在 build.gradle 中配置依赖项，引用 sdk 及 okhttp。



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

4. 配置权限，在 AndroidManifest.xml 中配置 App 的权限， QUIC SDK需要以下权限：



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

1. 开发环境要求。

Xcode 10.0+

iOS 10.0 以上的 iPhone 或者 iPad 真机

项目已配置有效的开发者签名

2. 下载 `TQUIC_iOS_SDK.zip` 并解压，将目录中的 `framework` 文件复制到工程目录下。

3. 在 XCode 中导入 TQUICiOS.framework 和 Tquic.framework。

说明：

Tquic.framework 为动态库，需要设置为 Embed & Sign。

4. 编译选项Other Linker Flags配置 `-ObjC、-l"c++"`。

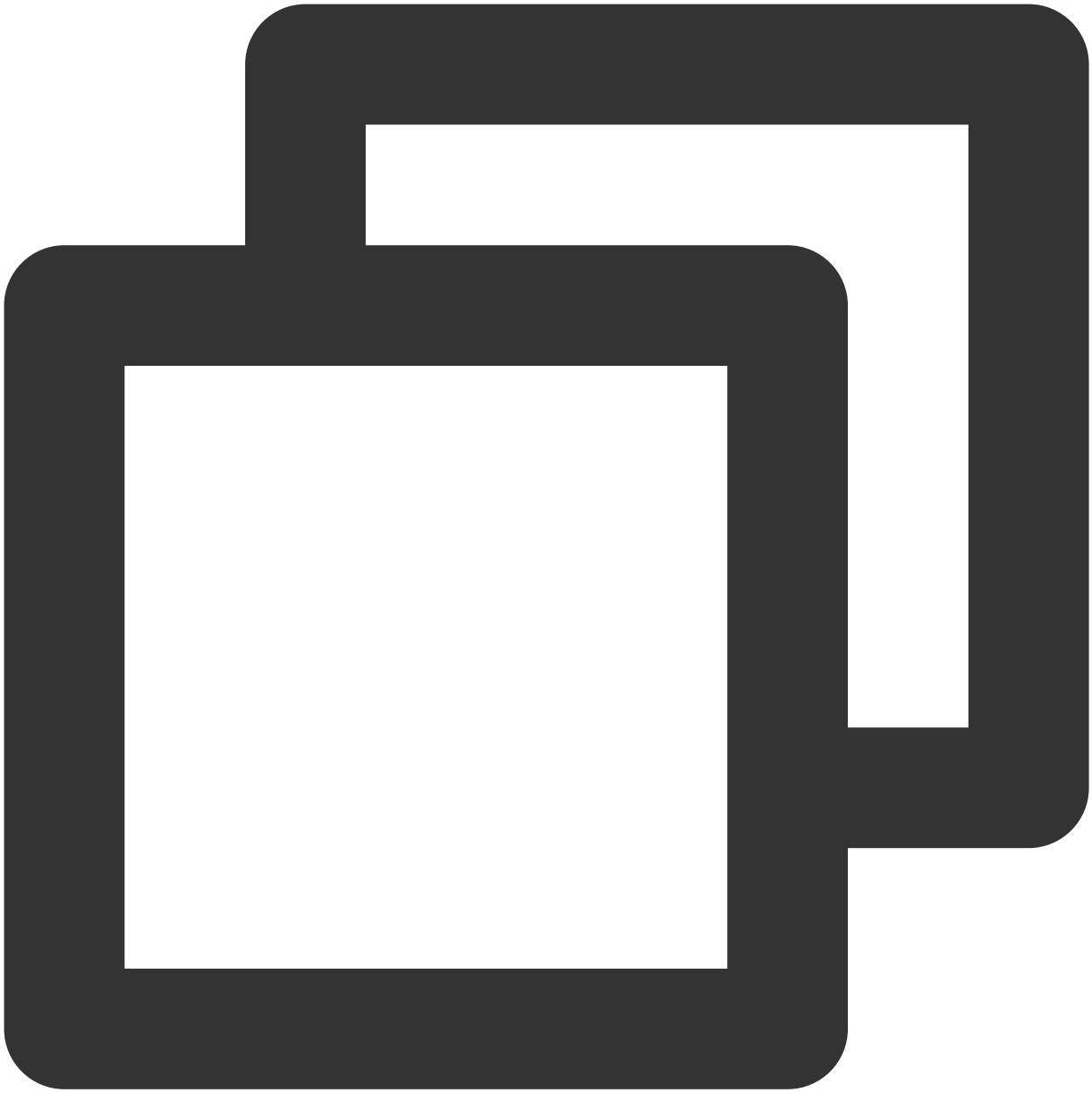
代码示例

Android

最近更新时间：2023-06-29 11:20:56

以下是 Android 客户端创建 QUIC 请求代码示例，详细 API 说明可参考：Android API 文档

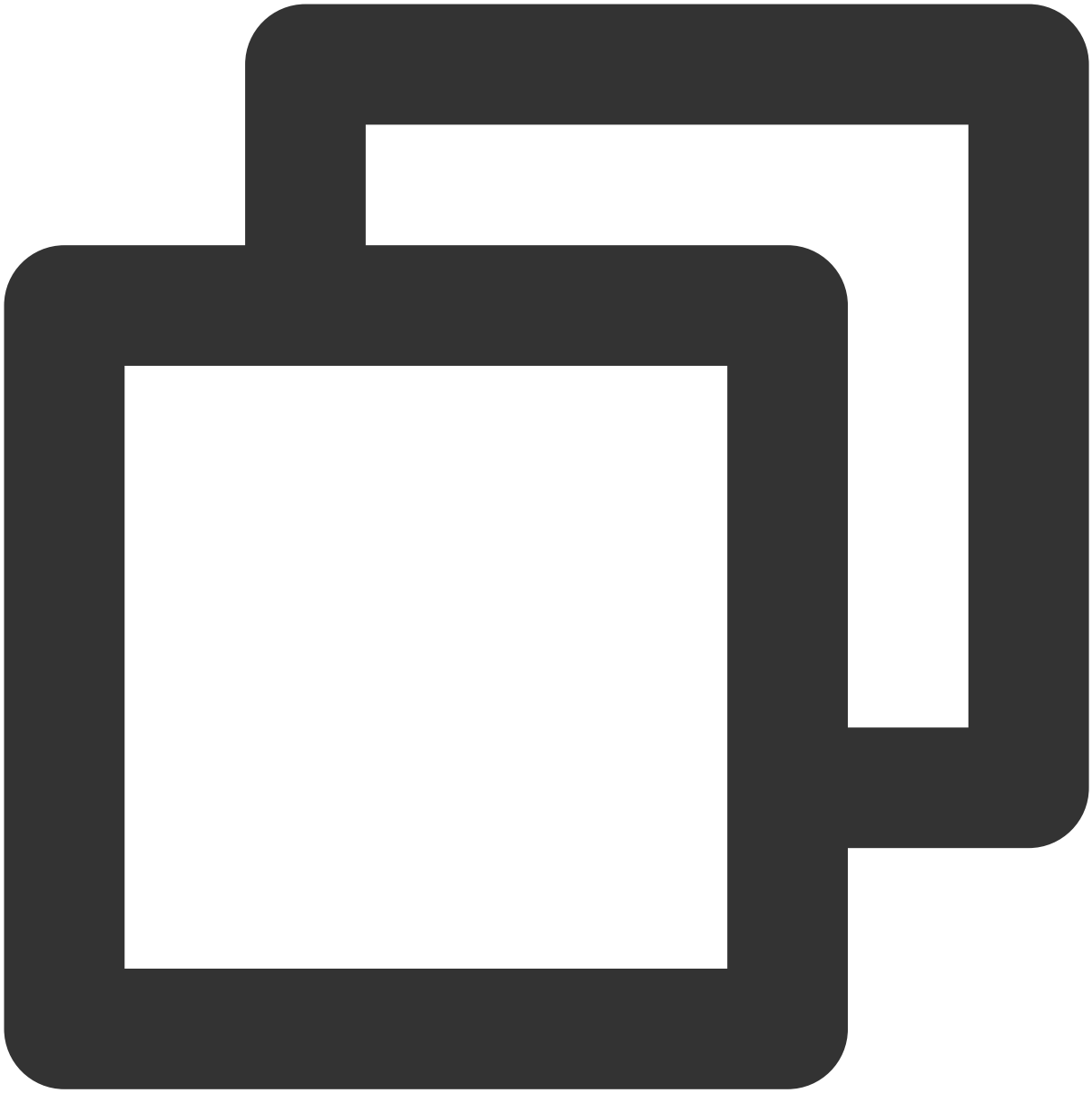
创建 Get 请求



```
//创建QuicClient，初始化QUIC配置，建议创建后作为全局变量使用，接口说明可参考API文档
QuicClient quicClient = new QuicClient.Builder()
                        .setCongestionType(QuicClient.CONGESTION_TYPE_BBR) //使用BBR
                        .setConnectTimeoutMillis(6 * 1000)   //配置连接超时时间
                        .build();

//创建QuicRequest，配置请求url
String url="";
QuicRequest request = new QuicRequest.Builder(url).get().build();

//异步执行请求，获取结果，QuicCall使用说明参考API文档
```

```
quicClient.newCall(request).enqueue(new QuicCallback() {
    @Override
    public void onResponse(QuicCall call, QuicResponse response) throws IOException
         //请求执行成功，获取响应数据
         ResponseBody body = response.body();
         if(body != null) {
             String res = body.string();
         }
    }

    @Override
    public void onFailed(QuicCall call, int errorCode, String error) {
        //请求执行失败，返回错误信息
    }
});
```

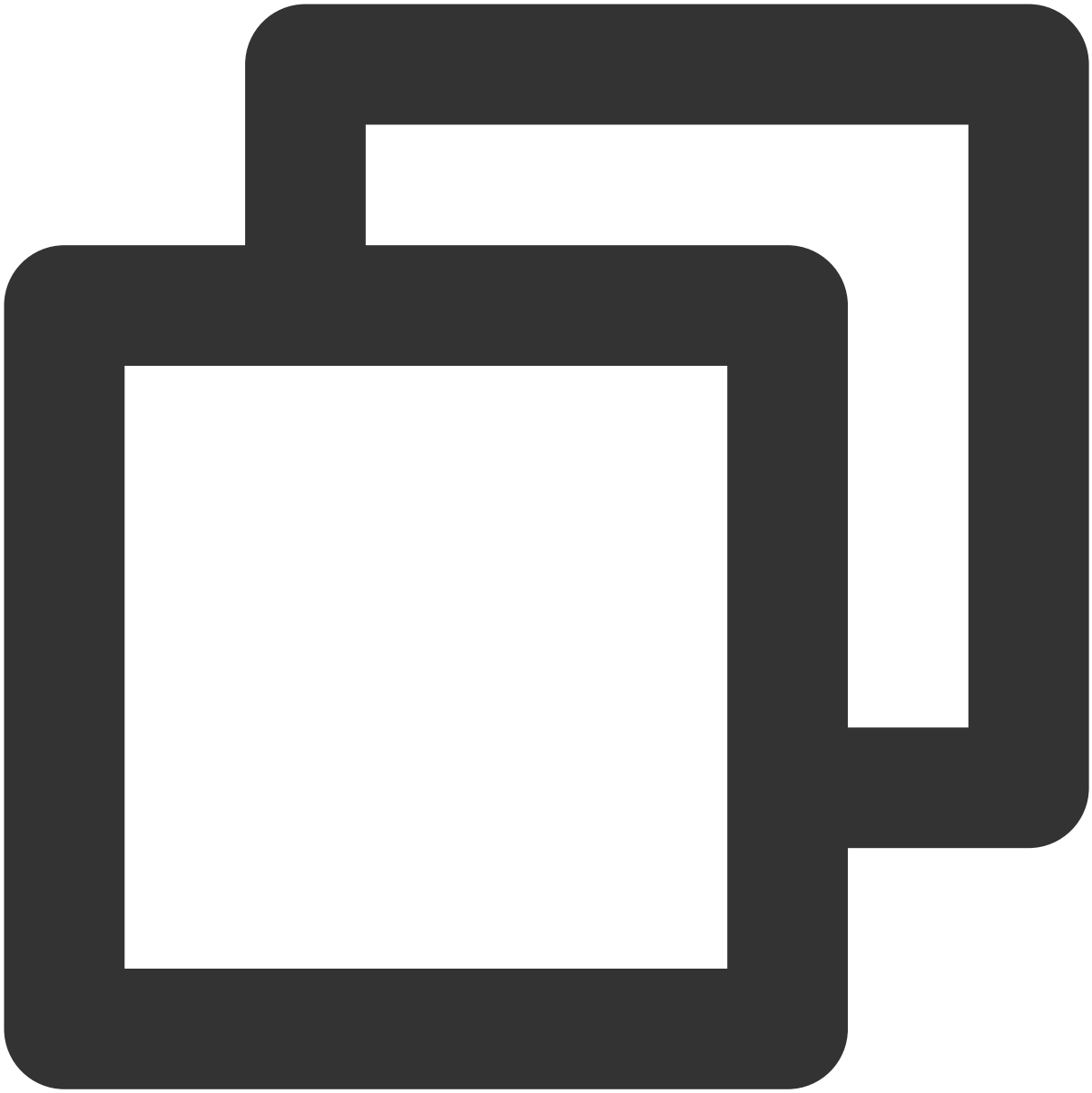
创建 Post 请求



```
//创建QuicClient，初始化QUIC配置，建议创建后作为全局变量使用，详细配置说明参考API文档
QuicClient quicClient = new QuicClient.Builder()
                        .setCongestionType(QuicClient.CONGESTION_TYPE_BBR) //使用BBF
                        .setConnectTimeoutMillis(3 * 1000)  //配置连接超时时间
                        .build();

//构造body数据
String body="your body string";
RequestBody requestBody = RequestBody.create(MediaType.parse("application/json"), b

//创建QuicRequest
```

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

//异步执行请求，获取结果，QuicCall使用说明参考API文档
quicClient.newCall(request).enqueue(new QuicCallback() {
    @Override
    public void onResponse(QuicCall call, QuicResponse response) throws IOException
         //请求执行成功，获取响应数据
         ResponseBody body = response.body();
         if(body != null) {
             String res = body.string();
         }
    }

    @Override
    public void onFailed(QuicCall call, int errorCode, String error) {
        //请求执行失败，返回错误信息
    }
});
```

取消请求



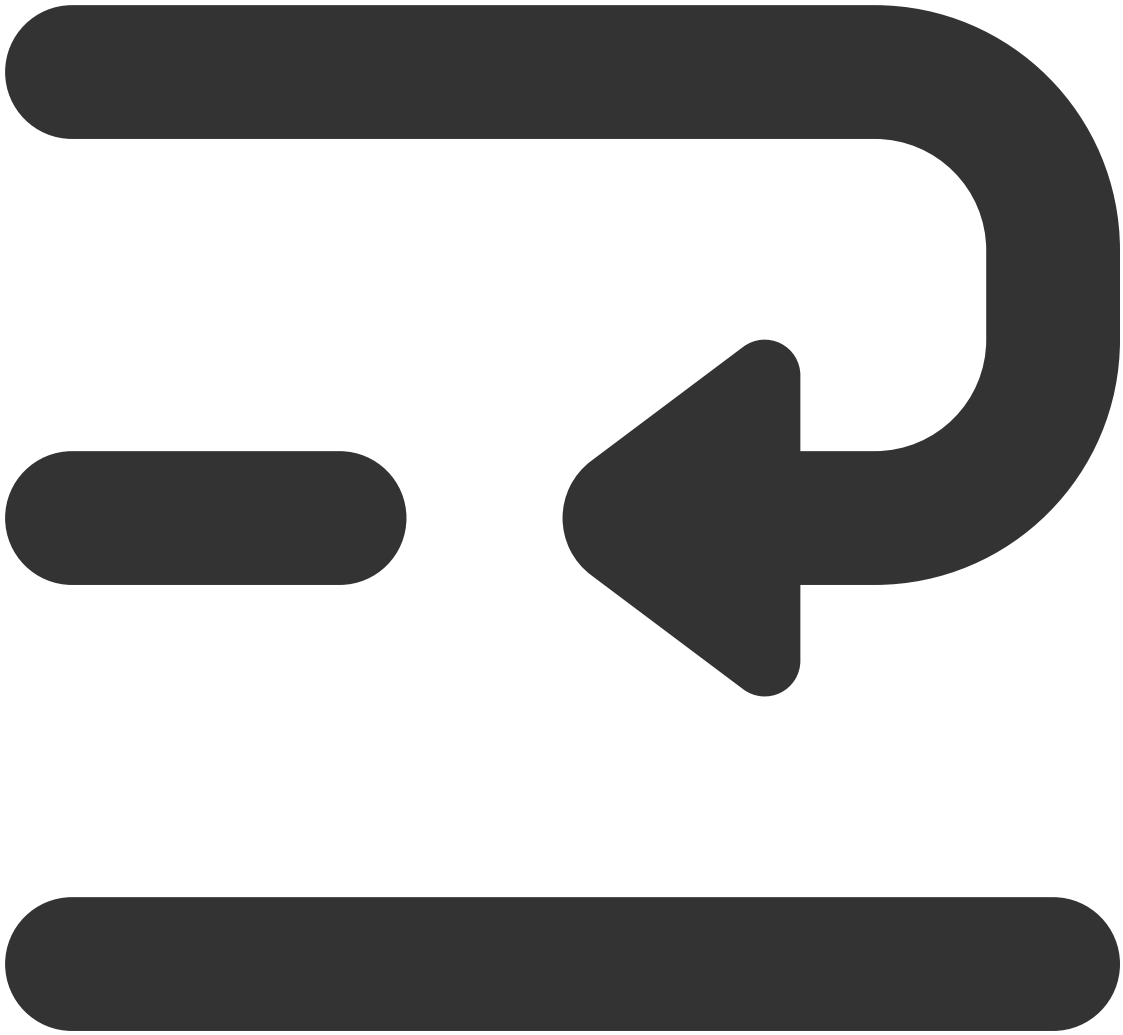
```
...
//创建QuicCall，使用说明参考API文档
QuicCall quicCall = quicClient.newCall(request);

//发起请求
...

//通过QuicCall中的cancel方法请取消请求
quicCall.cancel();
```

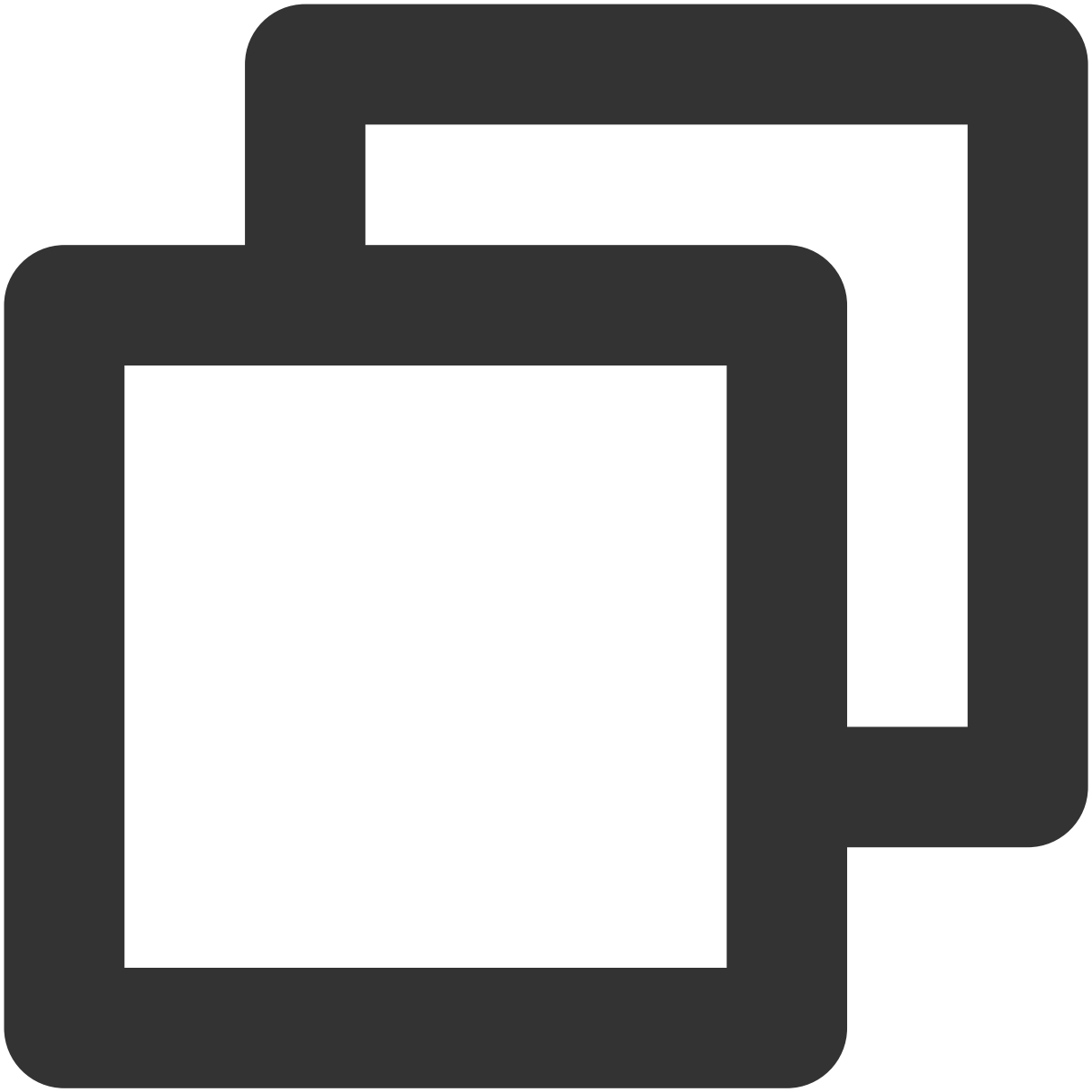
iOS

最近更新时间：2023-06-29 11:20:56

以下是 iOS 客户端创建 QUIC 请求代码示例，详细 API 说明可参考：iOS API 文档

创建 Get 请求

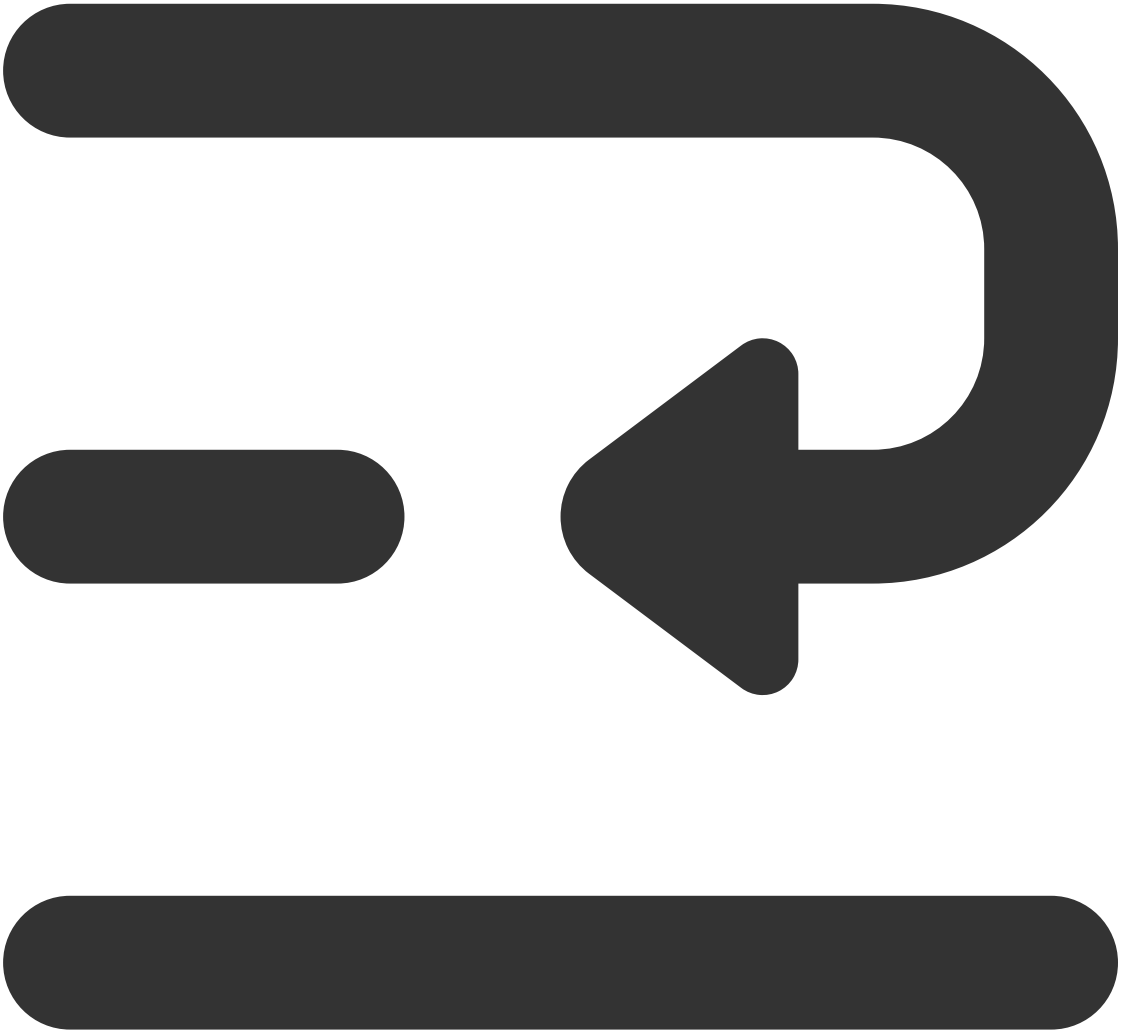


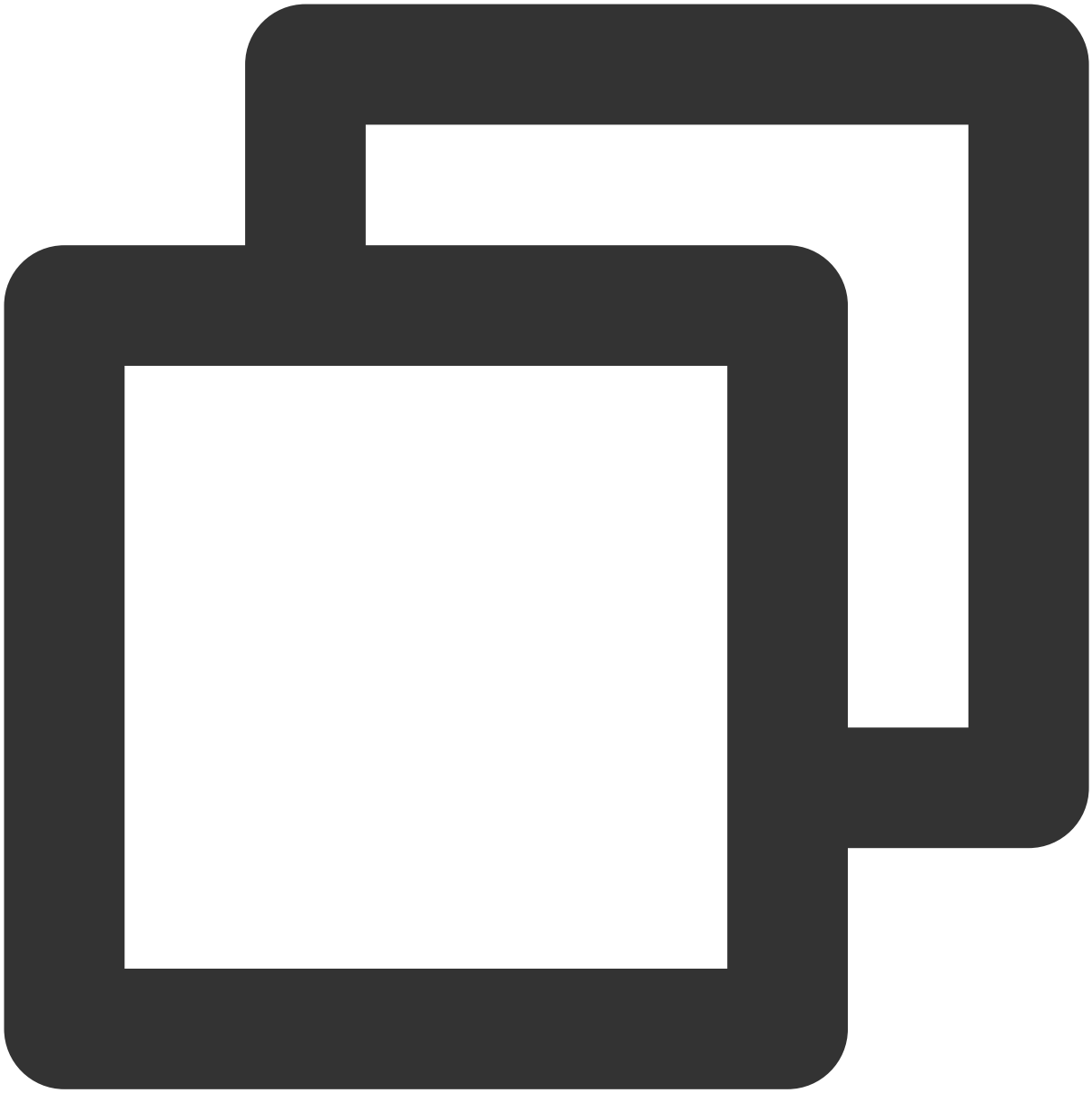


```
// Session配置
TQUICURLSessionConfiguration *quicSessionConfiguration = [TQUICURLSessionConfigurat
// 拥塞控制算法
quicSessionConfiguration.congestionType = TQUICCongestionTypeBBR;
// 连接超时时间
quicSessionConfiguration.connectTimeoutMillis = 6 * 1000;
// 创建SessionManager，使用说明参考API文档
TQUICHTTPSessionManager *quicSessionManager = [[TQUICHTTPSessionManager alloc] init

//（可选）通过TQUICHTTPRequestSerializer/TQUICHTTPResponseSerializer配置序列化，使用说明
```

```
// 发起GET请求
[quicSessionManager GET:@"url"
             parameters:nil
                headers:nil
        timeoutInterval:0
       downloadProgress:nil
                success:^(TQUICURLSessionTask * _Nonnull task, id  _Nullable respon
        // 请求执行成功，获取响应数据
        // 获取 Headers，由于是 HTTP 协议，因此这里 response 可以强转为 NSHTTPURLResponse
        NSDictionary *headers = [(NSHTTPURLResponse *)task.response allHeaderFields
        // 获取 Body
        id body = responseObject;

        }      failure:^(TQUICURLSessionTask * _Nullable task, NSError * _Nonnull e
        // 请求执行失败，获取错误信息
        NSInteger errorCode = error.code;
}];
```

创建 Post 请求

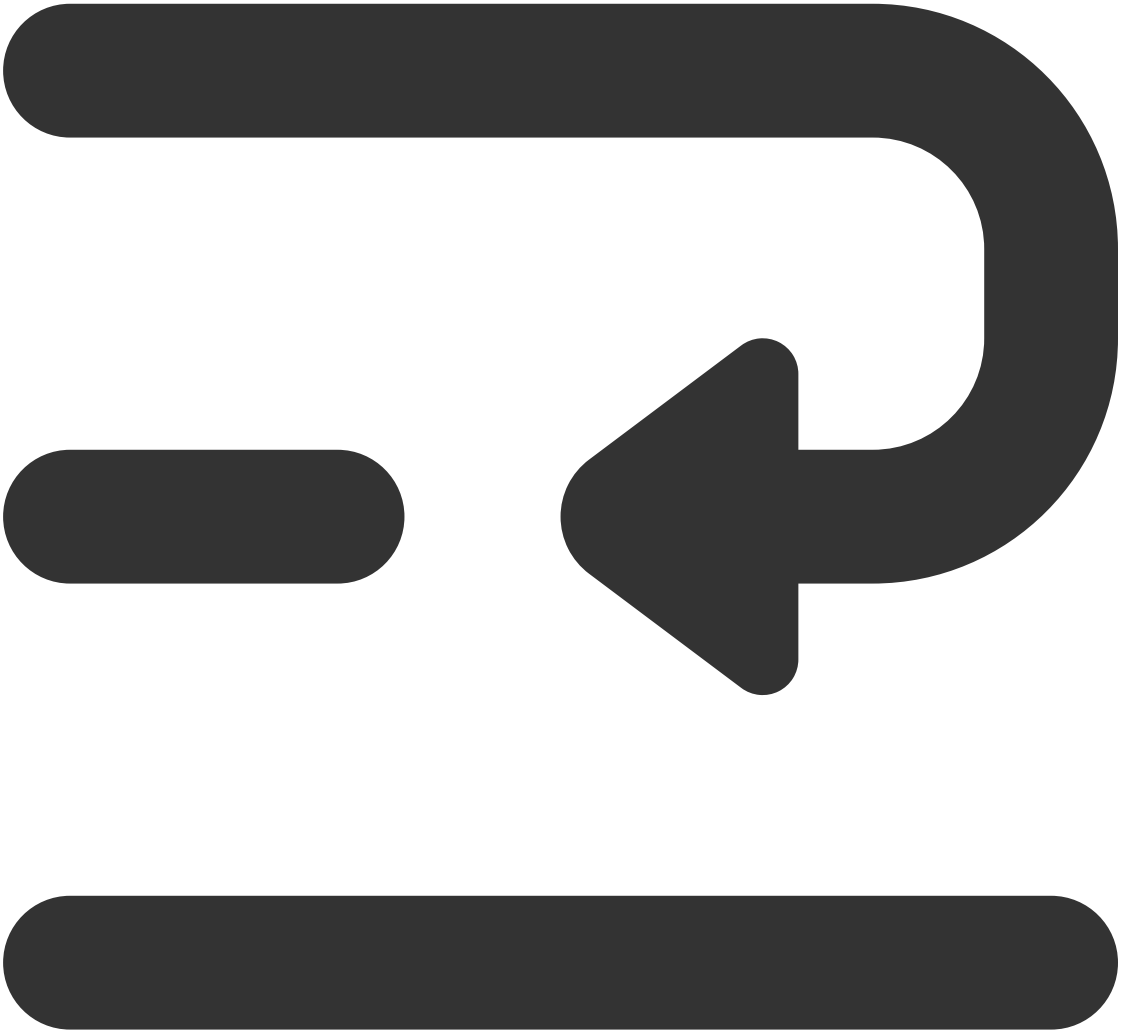


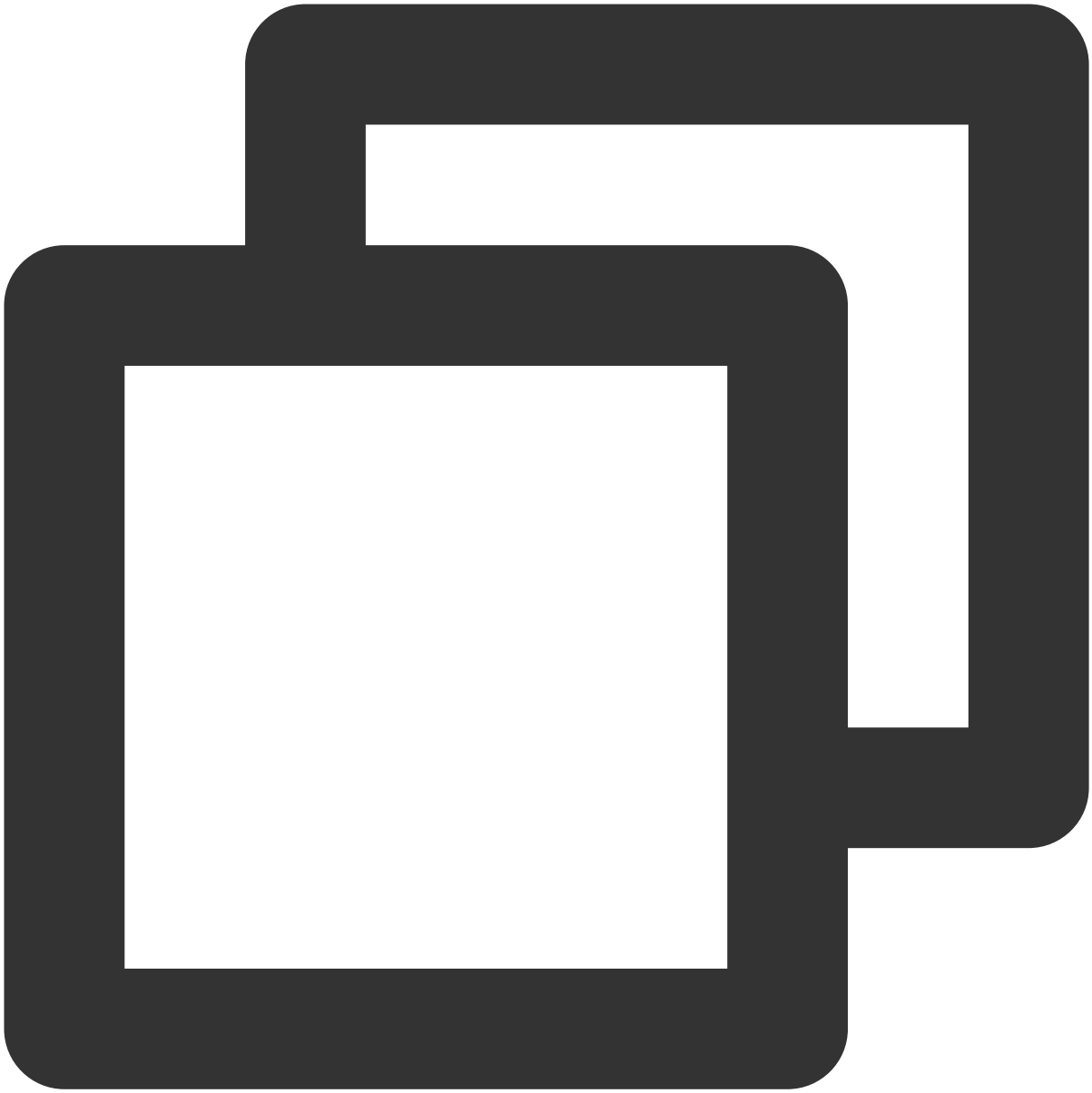


```
// Session配置
TQUICURLSessionConfiguration *quicSessionConfiguration = [TQUICURLSessionConfigurat
// 拥塞控制算法
quicSessionConfiguration.congestionType = TQUICCongestionTypeBBR;
// 连接超时时间
quicSessionConfiguration.connectTimeoutMillis = 6 * 1000;
// 创建SessionManager，使用说明参考API文档
TQUICHTTPSessionManager *quicSessionManager = [[TQUICHTTPSessionManager alloc] init

//（可选）通过TQUICHTTPRequestSerializer/TQUICHTTPResponseSerializer配置序列化，使用说明
```

```
//构造body数据
NSData *bodyData;
//发起Post请求
[quicSessionManager POST:@"url"
                    body:bodyData
                    headers:nil
                    timeoutInterval:timeInterval
                    uploadProgress:^(NSProgress * _Nonnull uploadProgress) {

                    } success:^(TQUICURLSessionTask * _Nonnull task, id  _Nullable
                    // 请求执行成功，获取响应数据
                    // 获取 Body
                    id body = responseObject;

                    } failure:^(TQUICURLSessionTask * _Nullable task, NSError * _No
                    // 请求执行失败，获取错误信息
                    NSInteger errorCode = error.code;
}];
```

取消请求





```
// 创建SessionManager

...
//使用SessionDataTask发起请求，使用说明见API文档
TQUICURLSessionDataTask *dataTask =
[quicSessionManager dataTaskWithRequest:request
                          uploadProgress:nil
                        downloadProgress:nil
                        completionHandler:^(NSURLResponse * _Nonnull response, id  _

        //请求完成回调
```

```
}];

//发起请求
[dataTask resume];

//取消请求
[dataTask cancel];
```

API 文档

Android

最近更新时间：2023-06-29 11:20:56

API概览

API	描述
QuicClient	QUIC 主功能入口，包括创建 QuicCall 实例、QUIC 配置、获取版本号等。
QuicCall	用来管理 QUIC 请求（发起请求、取消请求、获取状态）
QuicRequest	请求信息封装
QuicResponse	请求响应结果封装
QuicCallback	请求回调接口
QuicNetStats	QUIC网络状态信息

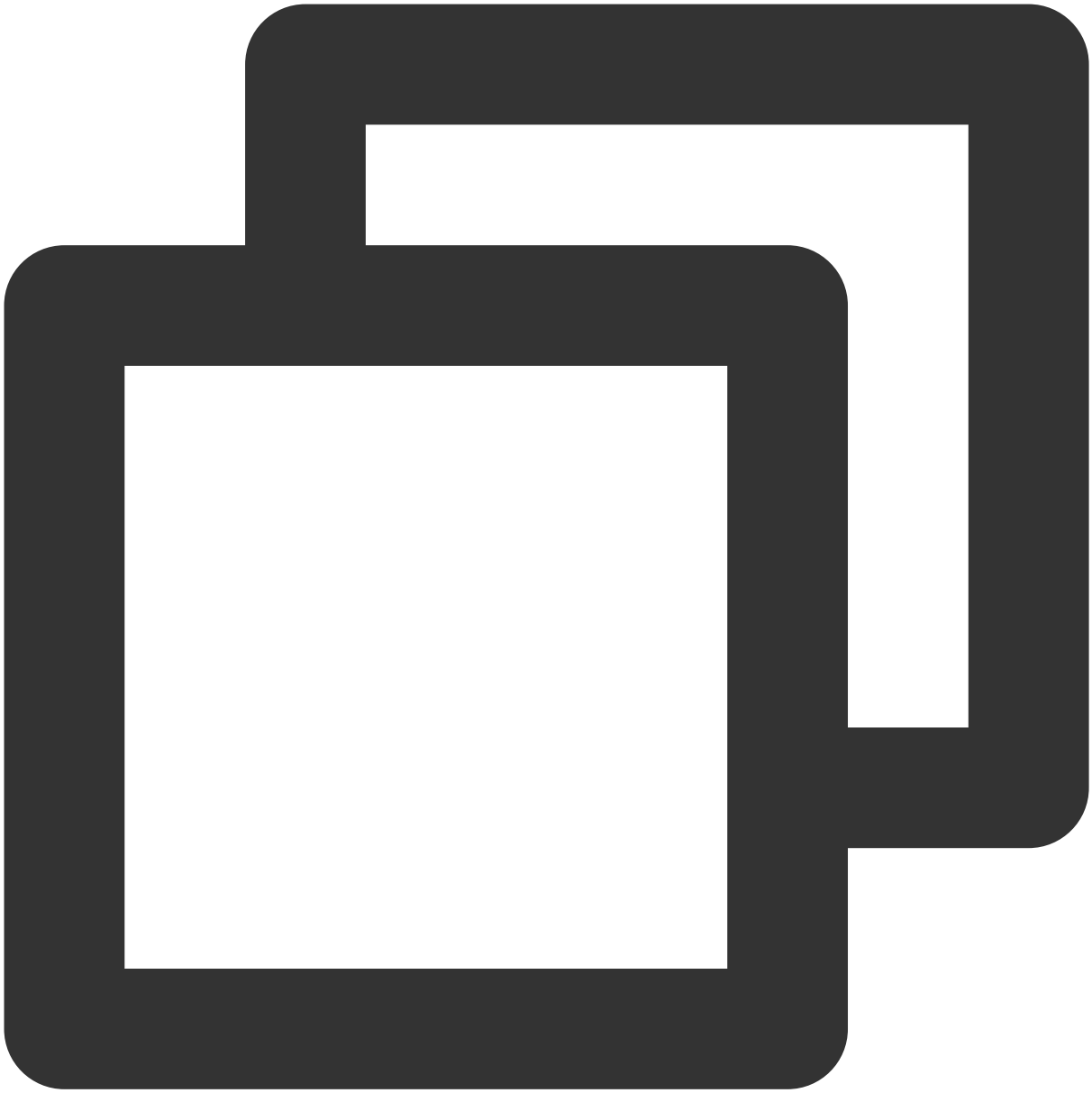
QuicClient

QUIC 主功能入口，包括创建 QuicCall 实例、QUIC 配置、获取版本号等。

API	描述
newCall	创建 QuicCall 实例
Builder	QUIC 配置信息构建
getVersion	获取 SDK 版本号

newCall

创建 QUIC 请求，每次请求都需要调用此方法创建新的 QuicCall

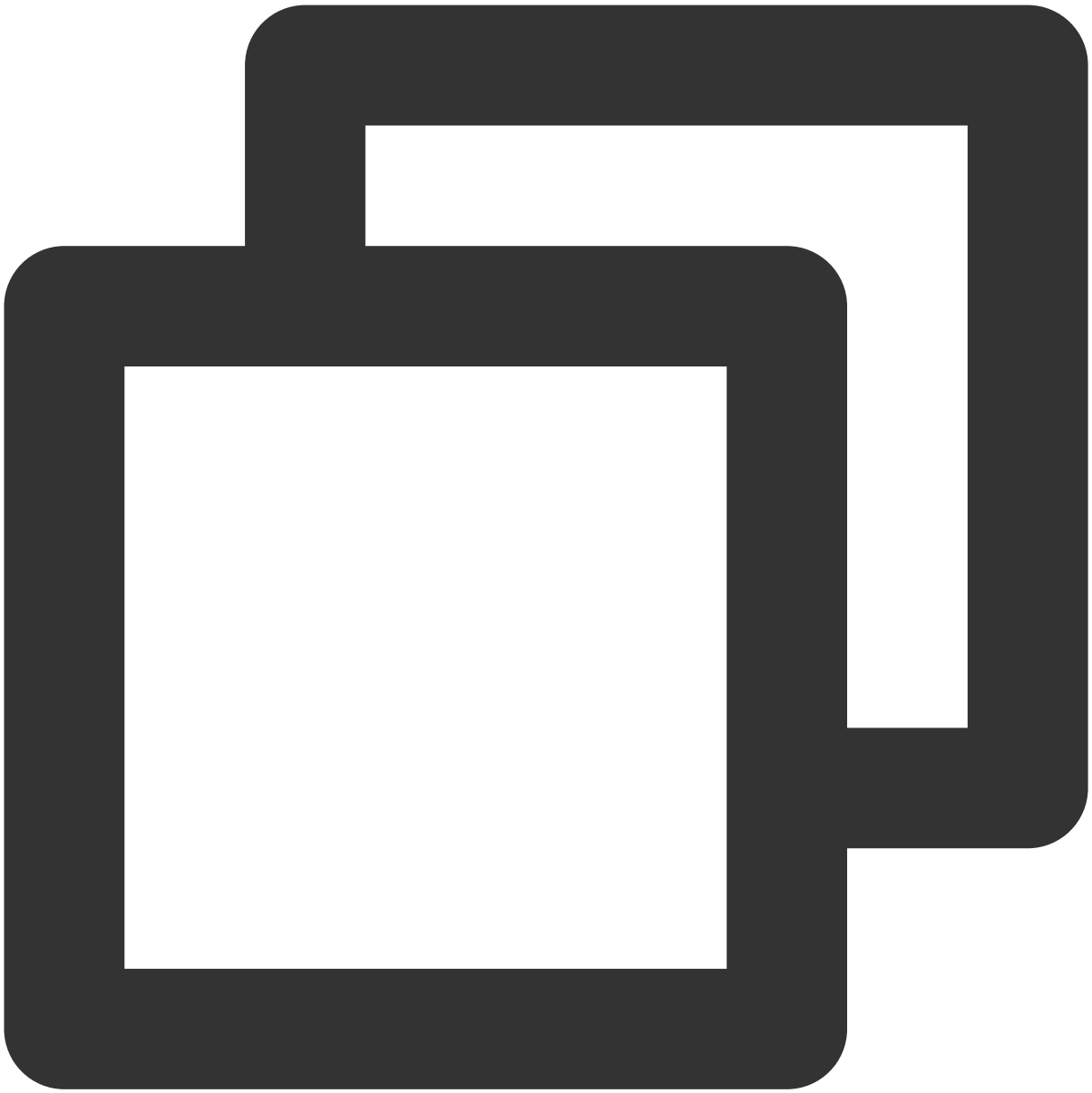


```
QuicCall newCall(QuicRequest request)
```

参数	描述
request	请求信息封装，可参考：QuicRequest

getVersion

获取 SDK 版本号，静态方法



```
String getVersion()
```

Builder

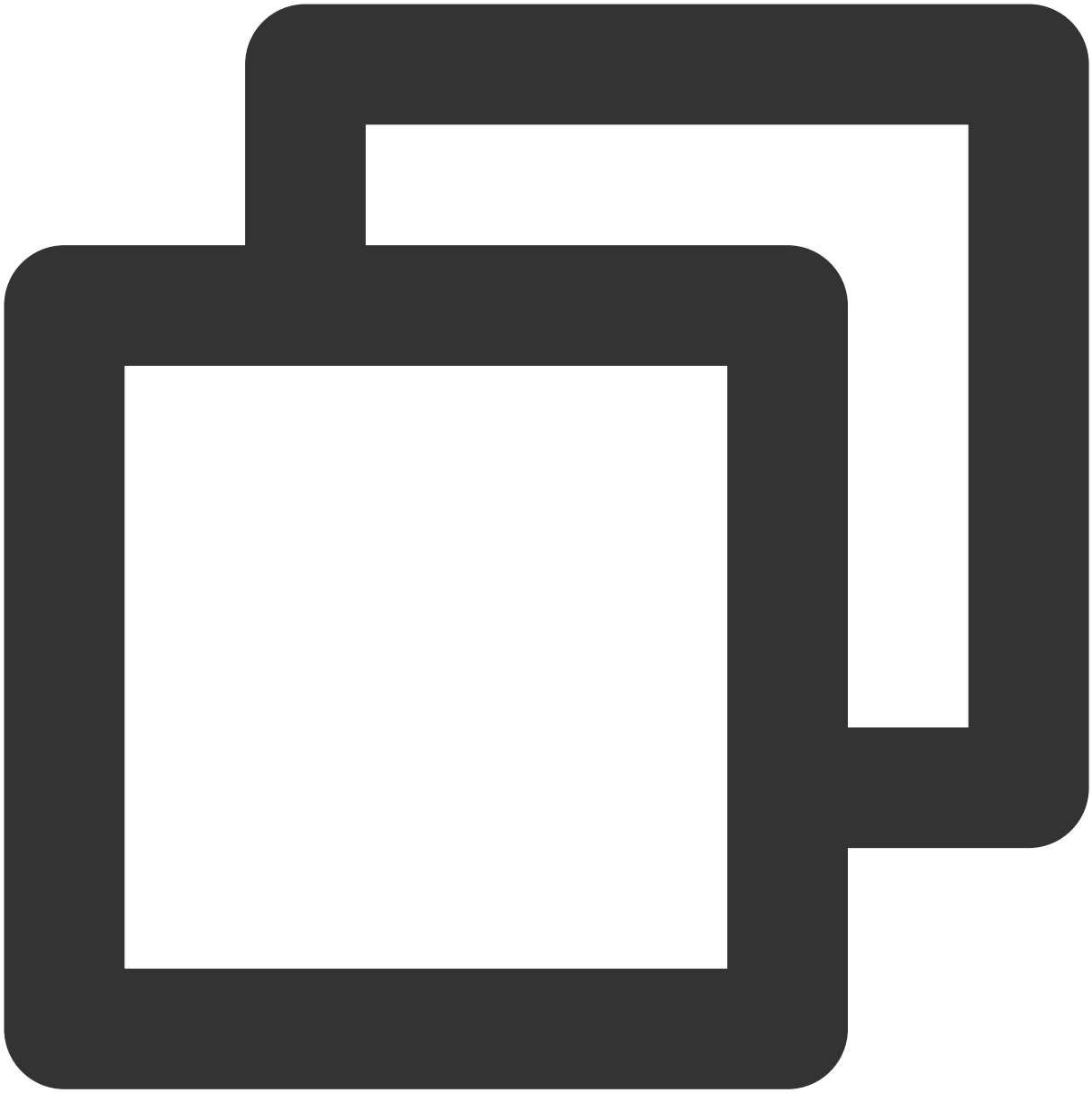
QUIC 配置接口

API	描述
setQuicVersion	设置 QUIC 协议版本号

setCongestionType	设置拥塞算法
setConnectTimeoutMillis	设置连接超时时间
setTotalTimeoutMillis	设置请求总超时时间
setIdleTimeoutMillis	设置空闲连接超时时间
setSupportIpV6	是否支持 IPv6
build	创建 QuicClient

setQuicVersion

设置使用的 QUIC 协议版本号



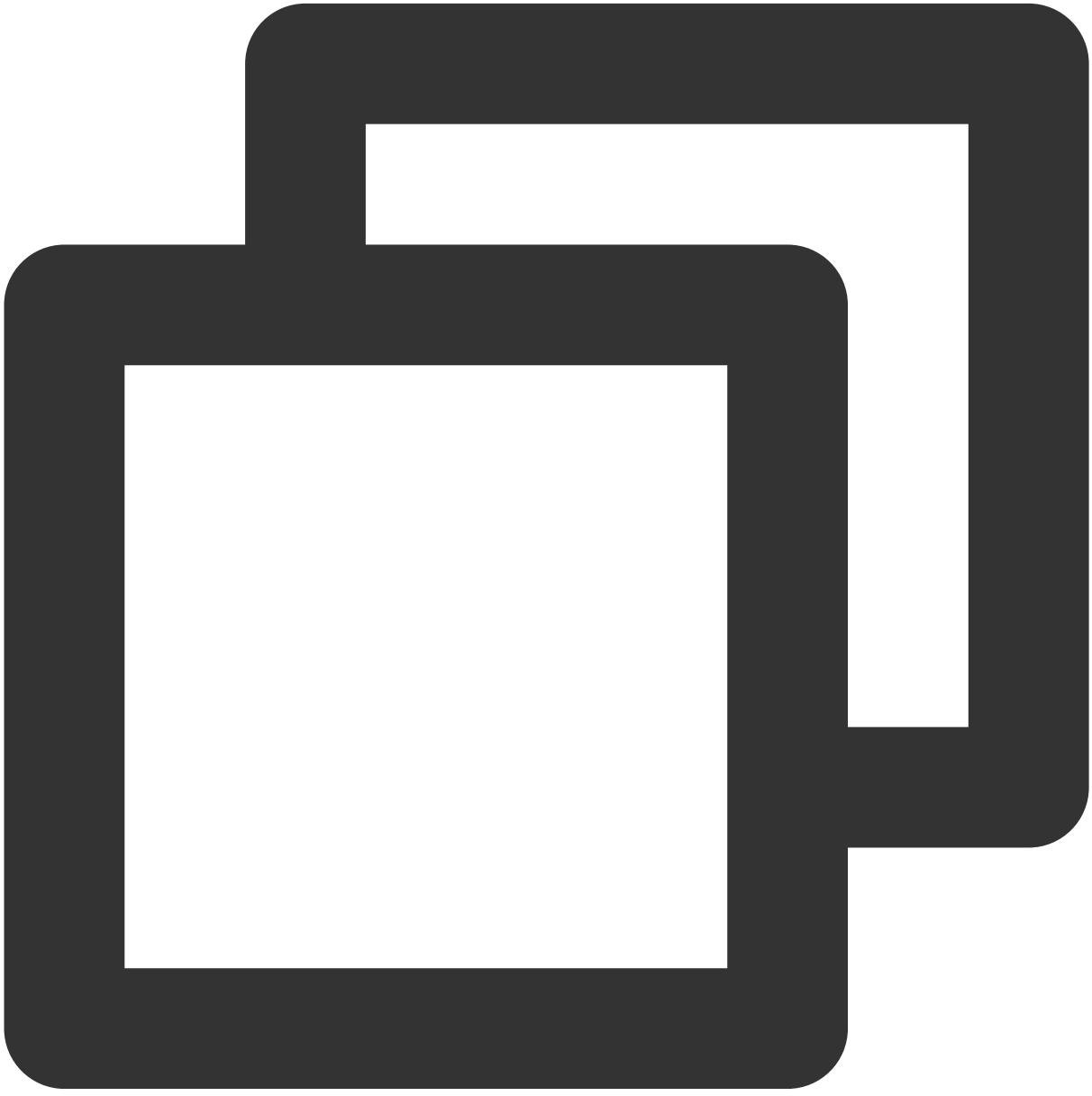
```
Builder setQuicVersion(int quicVersion)
```

参数	描述
quicVersion	设置 QUIC 协议版本号 支持的版本：Q043、Q046、Q050、Q051、draft-29、RFC-V1（RFC 9000） 取值： QuicClient.QUIC_VERSION_Q43 (默认值) QuicClient.QUIC_VERSION_Q46 QuicClient.QUIC_VERSION_Q50

	QuicClient.QUIC_VERSION_Q51 QuicClient.QUIC_VERSION_IETF_DRAFT_29 QuicClient.QUIC_VERSION_IETF_RFC_V1

setCongestionType

设置拥塞算法



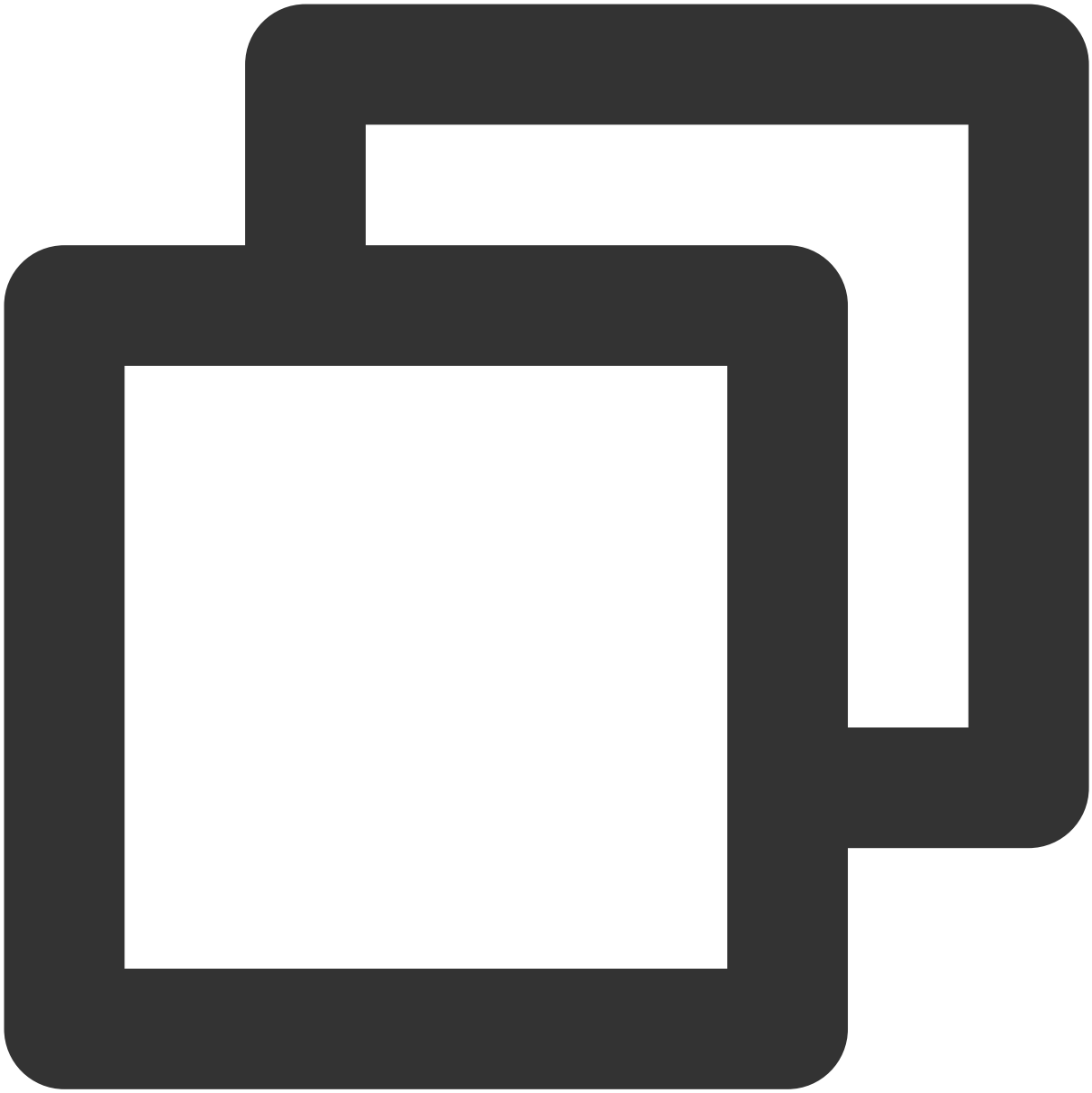
```
Builder setCongestionType(int congestionType)
```

参数	描述

congestionType	支持的拥塞算法（CubicBytes、RenoBytes、BBR、PCC、GCC） 取值： QuicClient.CONGESTION_TYPE_BBR （默认值） QuicClient.CONGESTION_TYPE_RENO_BYTES QuicClient.CONGESTION_TYPE_BBR QuicClient.CONGESTION_TYPE_PCC QuicClient.CONGESTION_TYPE_GCC

setConnectTimeoutMillis

设置连接超时时间

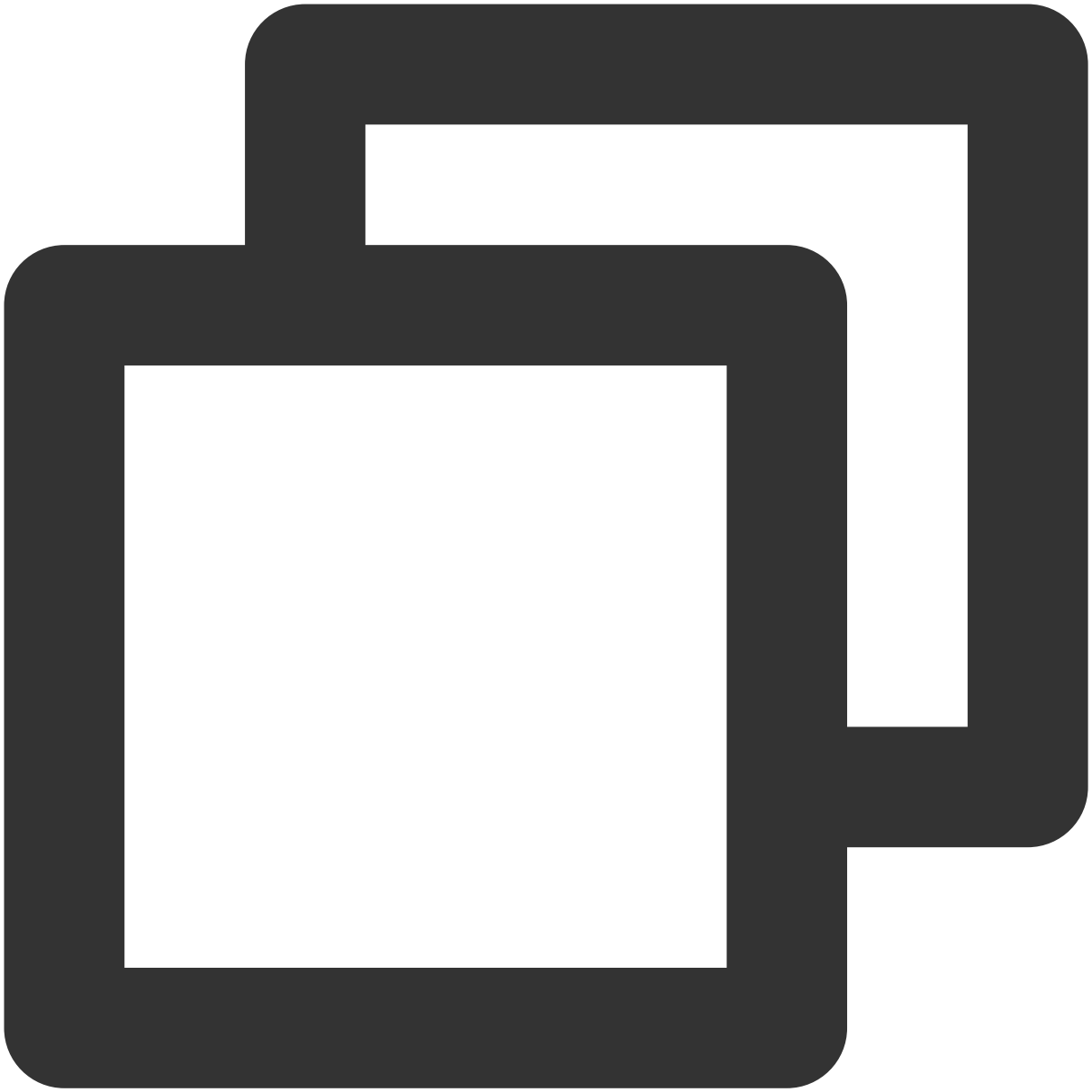


```
Builder setConnectTimeoutMillis(int connectTimeoutMillis)
```

参数	描述
connectTimeoutMillis	连接超时时间，单位：毫秒 默认值：60000

setTotalTimeoutMillis

设置请求总超时时间，包括数据流读写

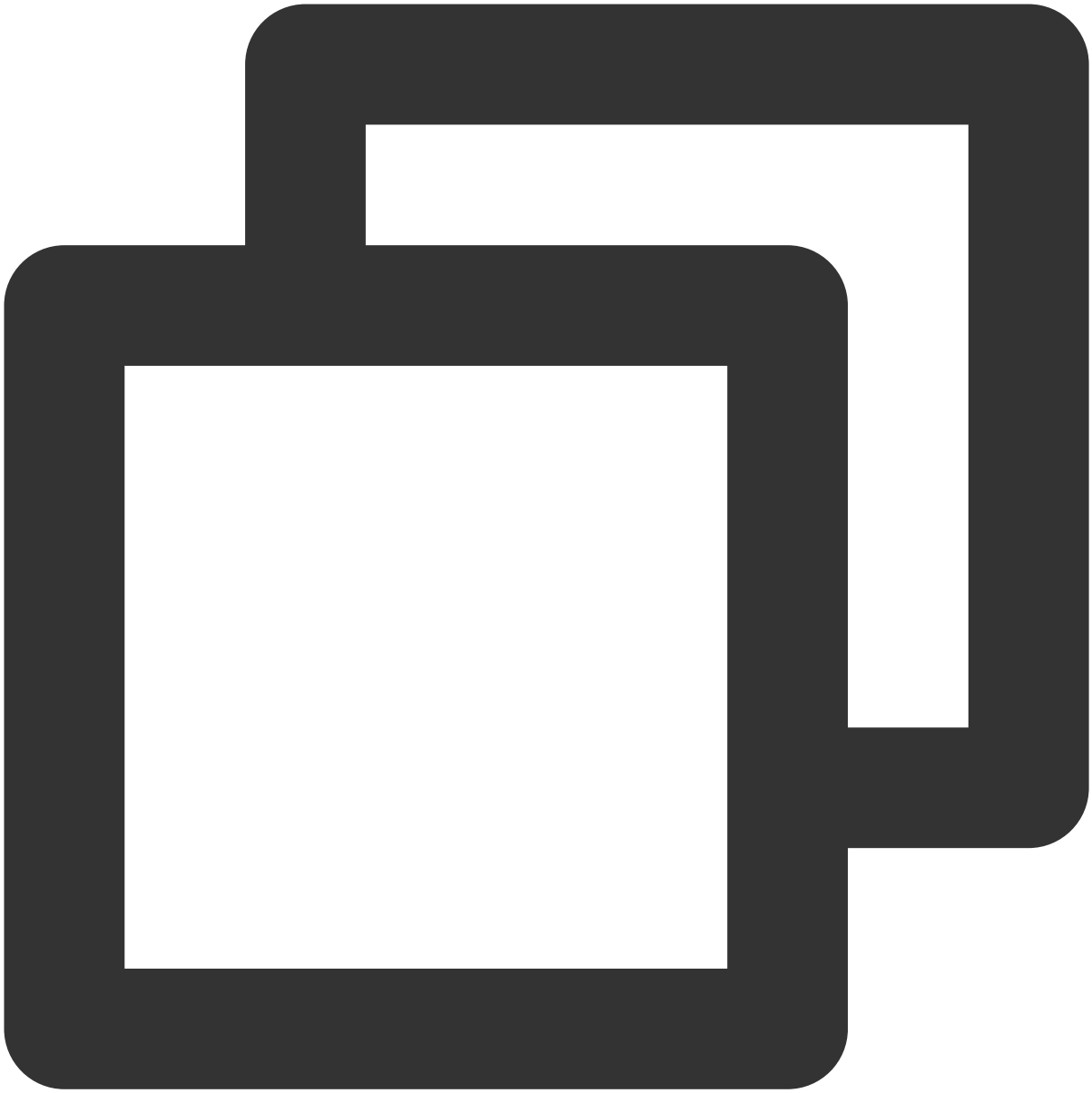


```
Builder setTotalTimeoutMillis(int totalTimeoutMillis)
```

参数	描述
totalTimeoutMillis	请求总超时时间，单位：毫秒 默认值：0（不超时）

setIdleTimeoutMillis

设置空闲连接超时时间，主要用于连接复用参数调整，超时后连接关闭不能再复用

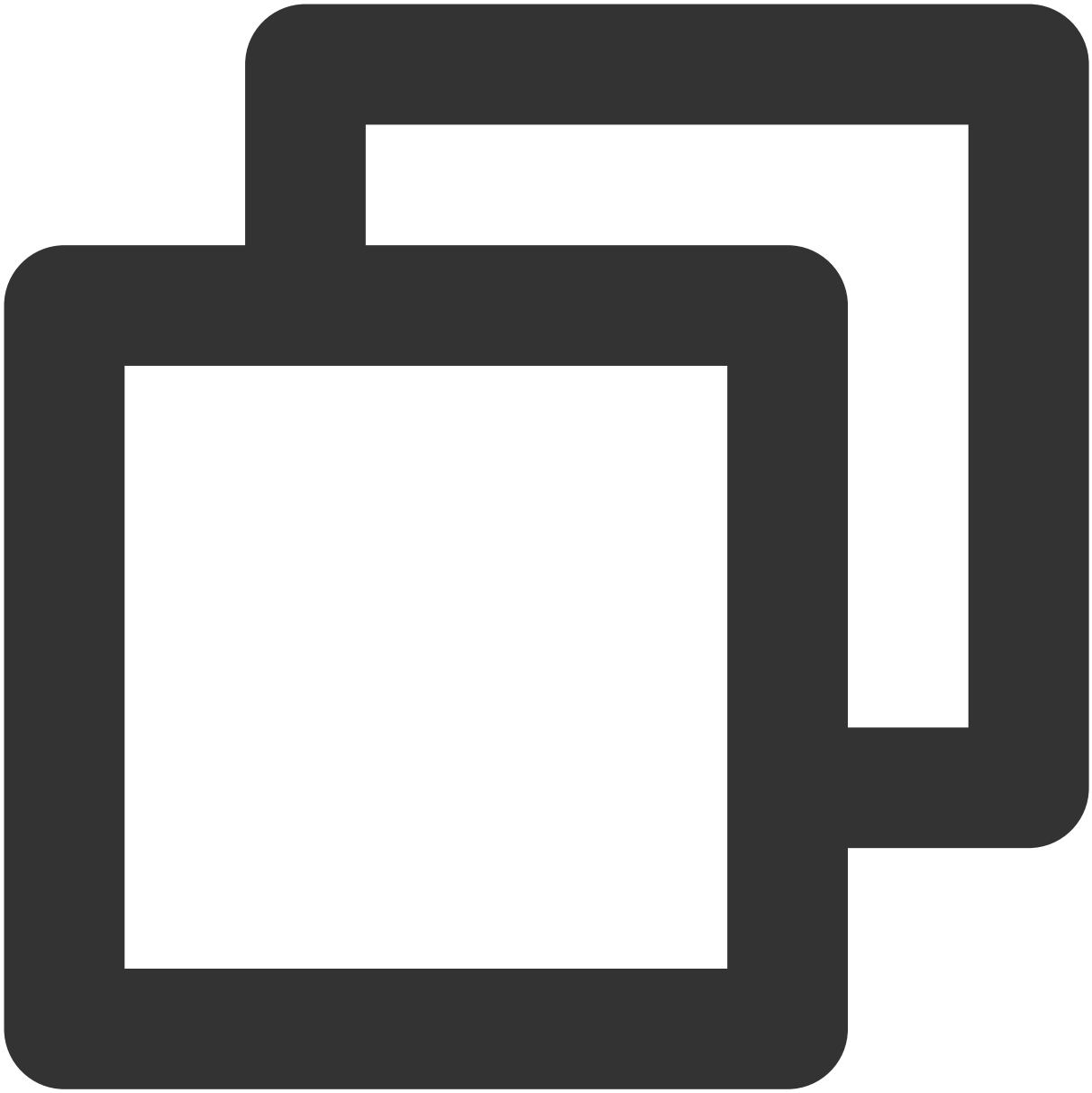


```
Builder setIdleTimeoutMillis(int idleTimeoutMillis)
```

参数	描述
idleTimeoutMillis	空闲连接超时时间，单位：毫秒 默认值：90000

setSupportIpV6

是否支持 IPv6

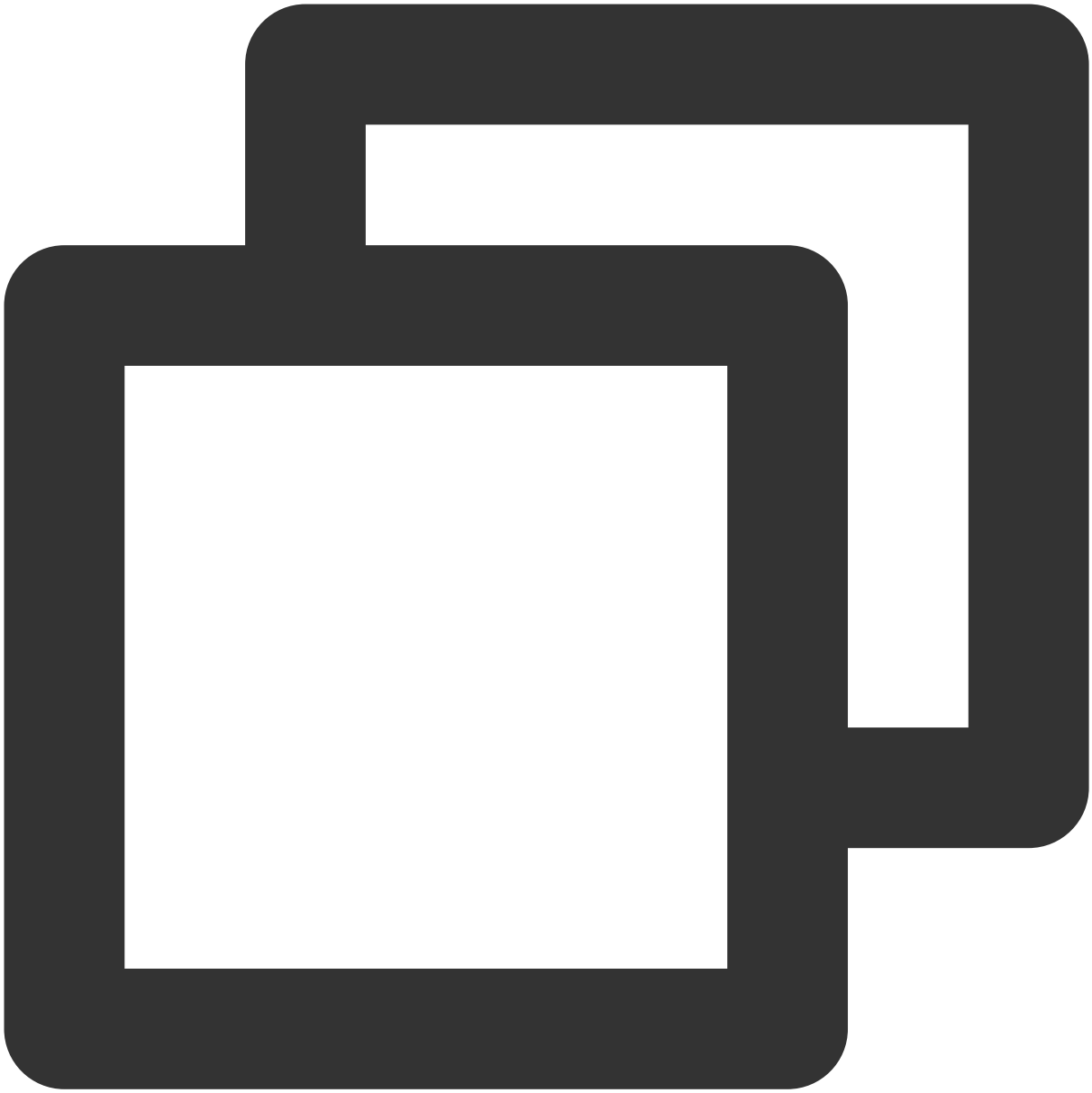


```
Builder setSupportIpV6(boolean supportIpV6)
```

参数	描述
supportIpV6	是否支持 IPv6，可选 true 或 false 默认值：false

build

创建 QuicClient



```
QuicClient build()
```

QuicCall

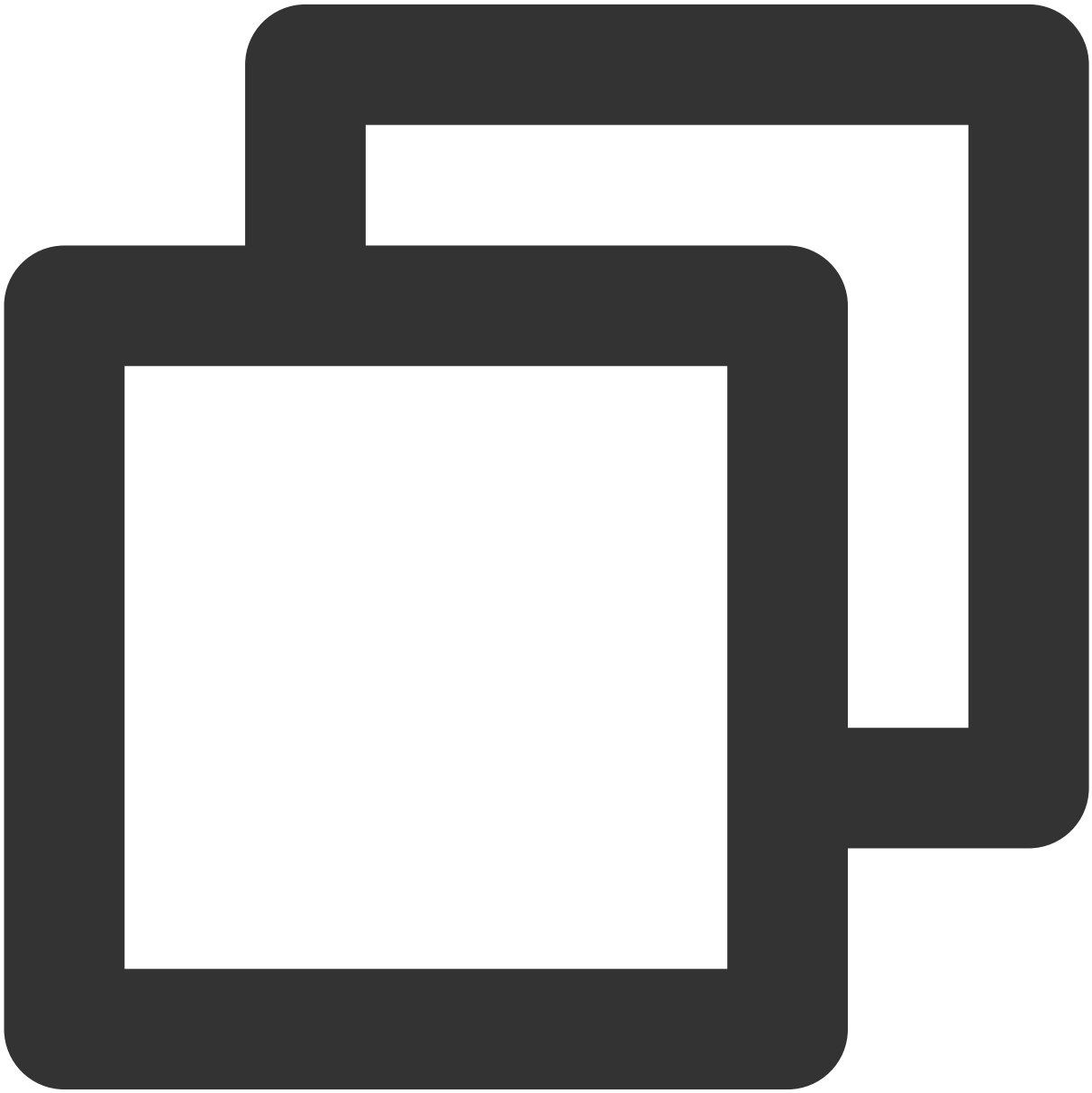
用来管理 QUIC 请求（发起请求、取消请求、获取状态信息）

API	描述
enqueue	异步发起 QUIC 网络请求

cancel	取消请求
getQuicNetStats	获取 QUIC 网络状态信息，详见：QuicNetStats

enqueue

异步发起 QUIC 请求，请求加入到队列中，请求响应通过回调函数通知

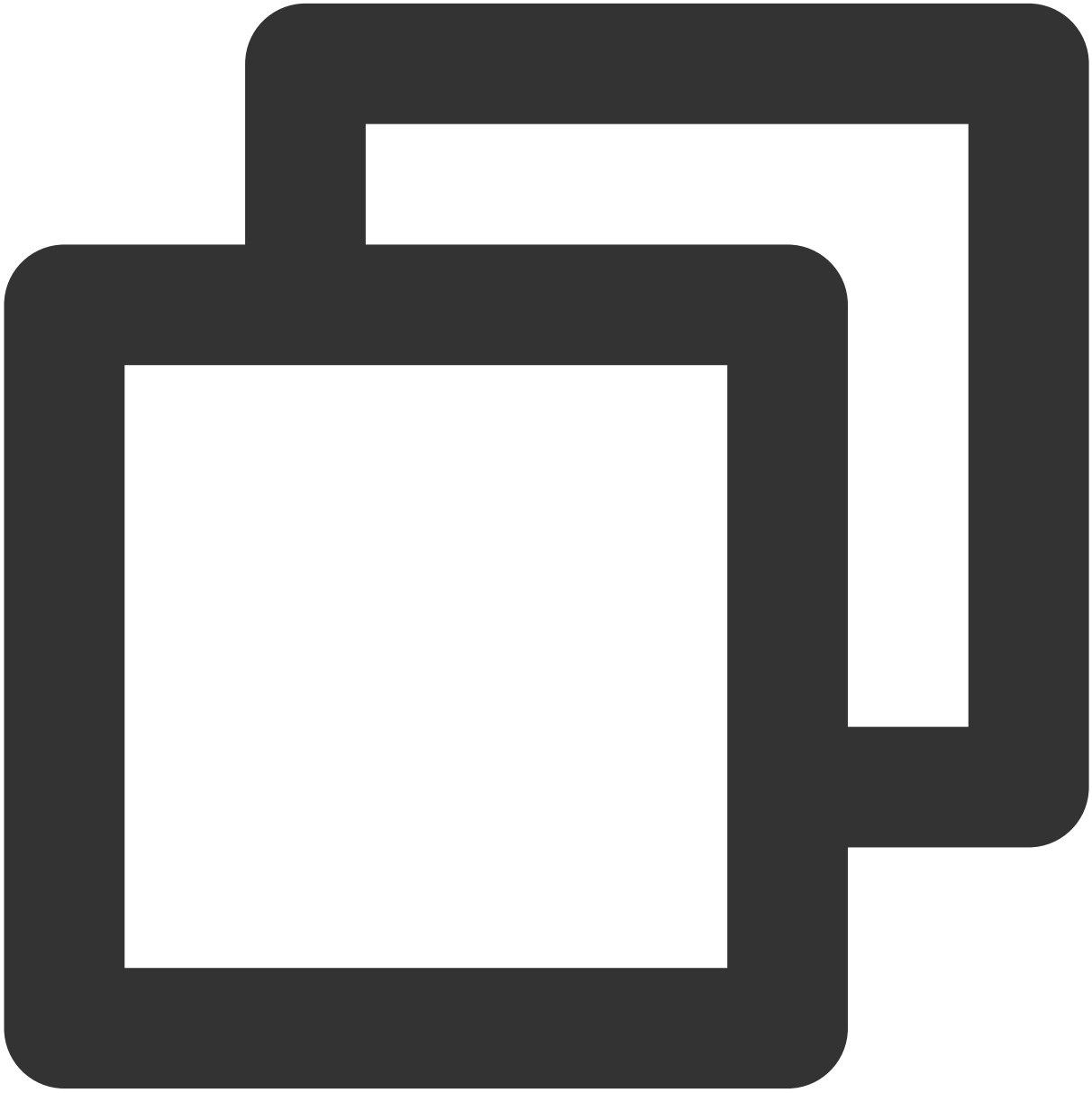


```
void enqueue(QuicCallback callback)
```


参数	描述
callback	请求响应结果回调函数，详见：QuicCallback

cancel

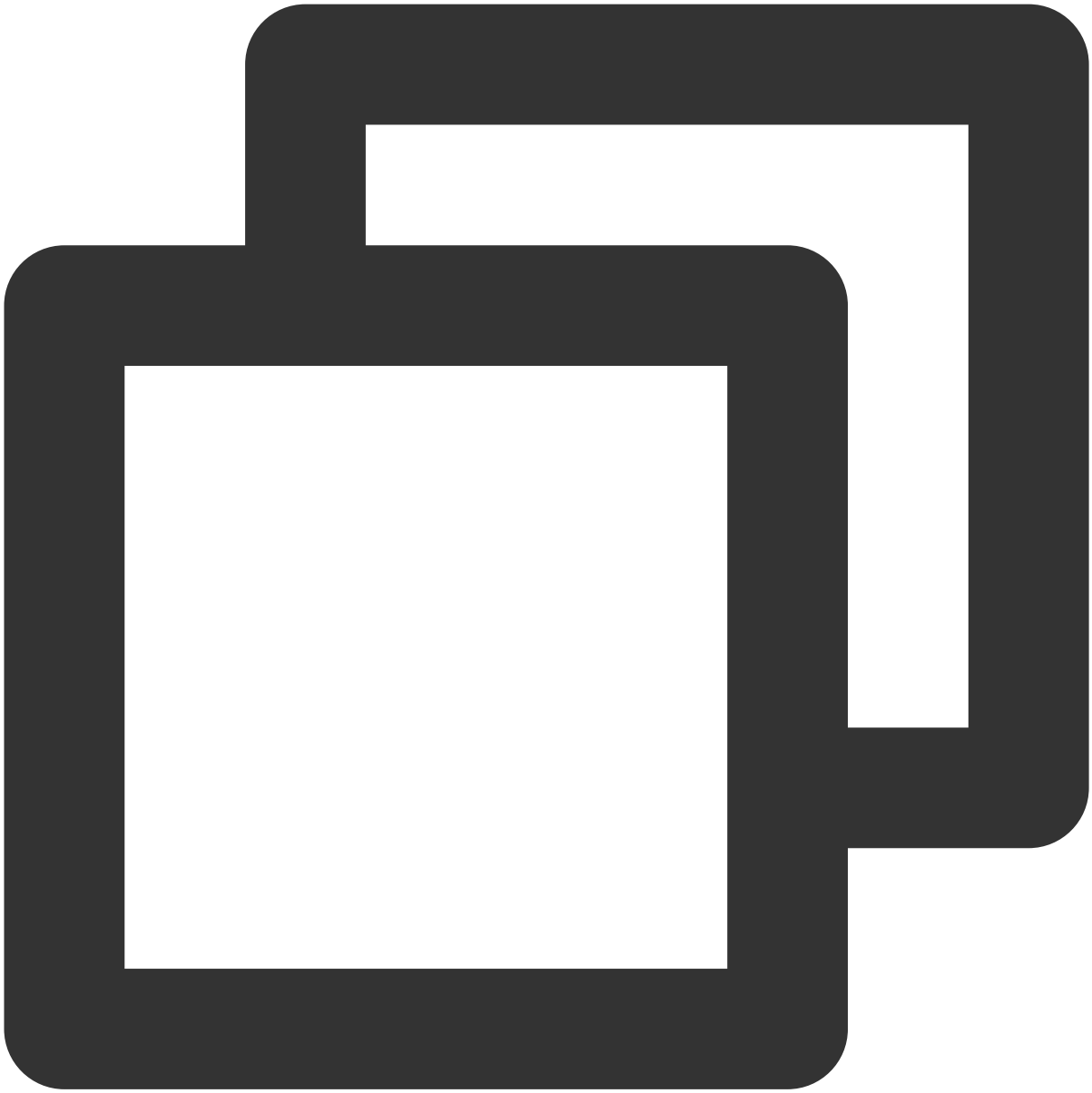
取消请求



```
void cancel()
```

getQuicNetStats

获取 QUIC 网络状态信息



```
QuicNetStats getQuicNetStats()
```

QuicRequest

请求参数

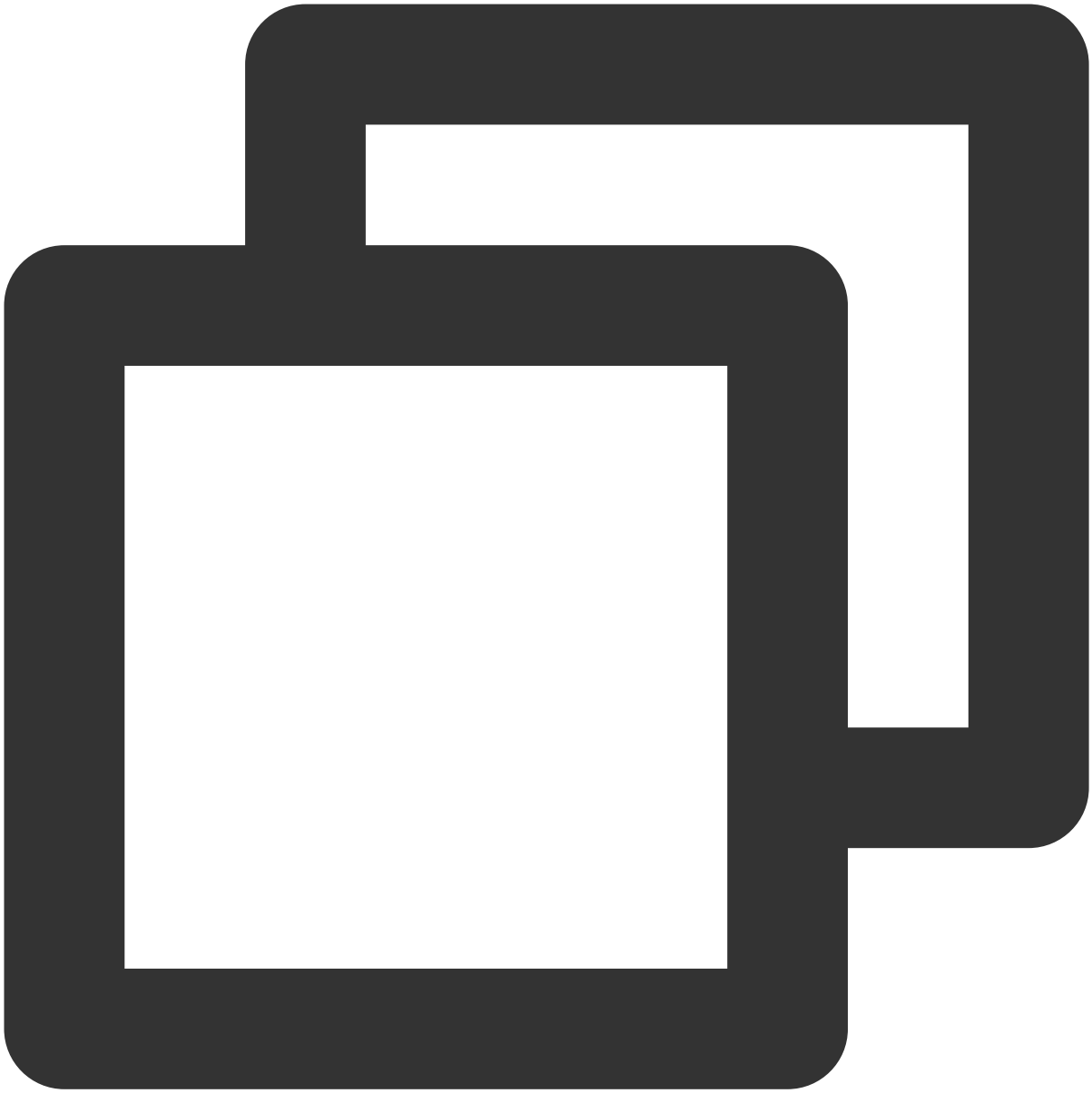
API	描述
Builder	请求参数构建

Builder

API	描述
setUrl	设置请求 Url
setIp	设置请求 IP
addHeader	添加请求 Header 信息
get	设置为 Get 请求
post	设置为 Post 请求
method	其它请求类型参数设置
build	创建 QuicRequest 对象

setUrl

设置请求 Url

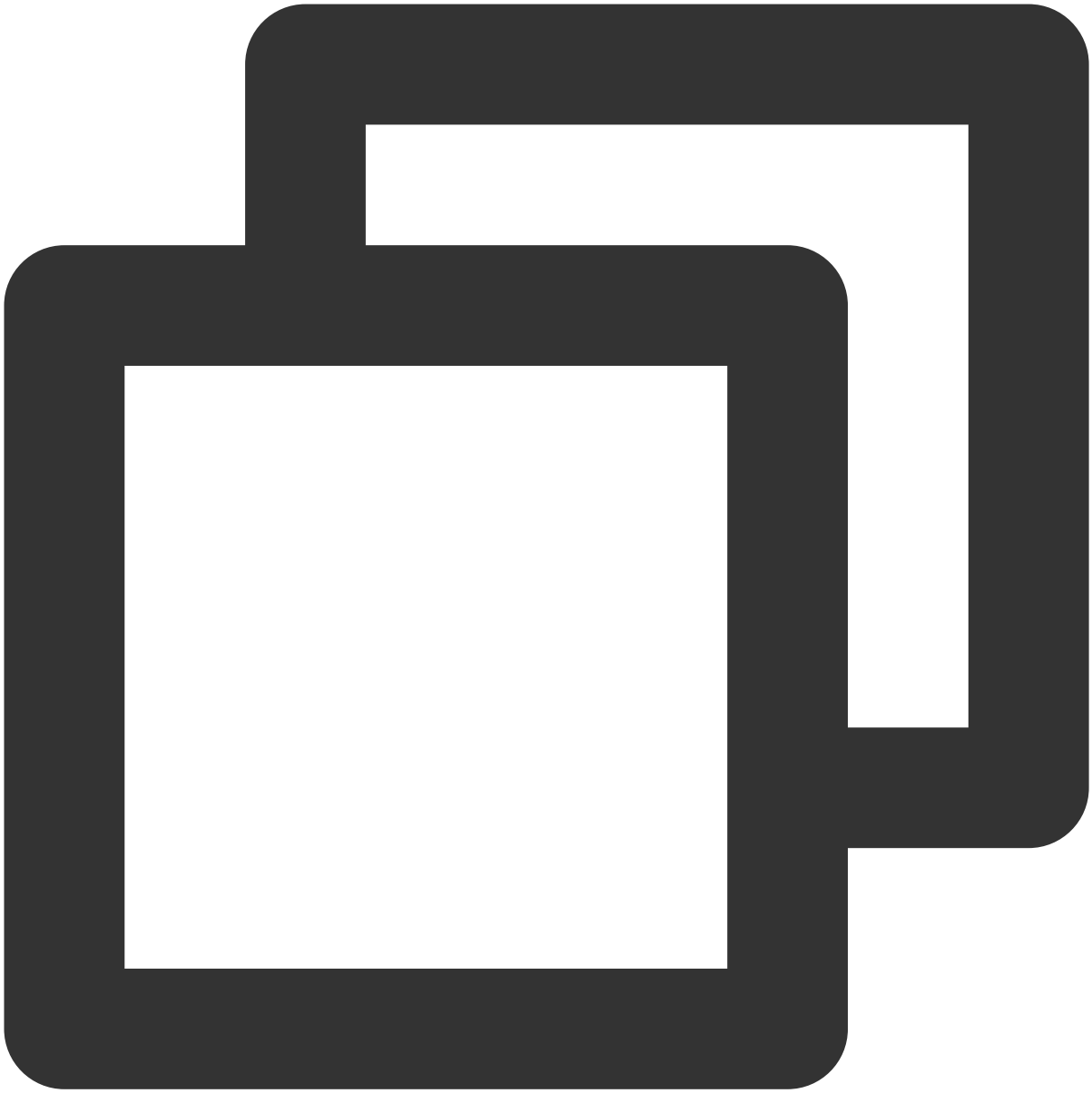


```
Builder setUrl(String url)
```

参数	描述
url	必填，请求 Url

setIp

设置请求 IP 地址

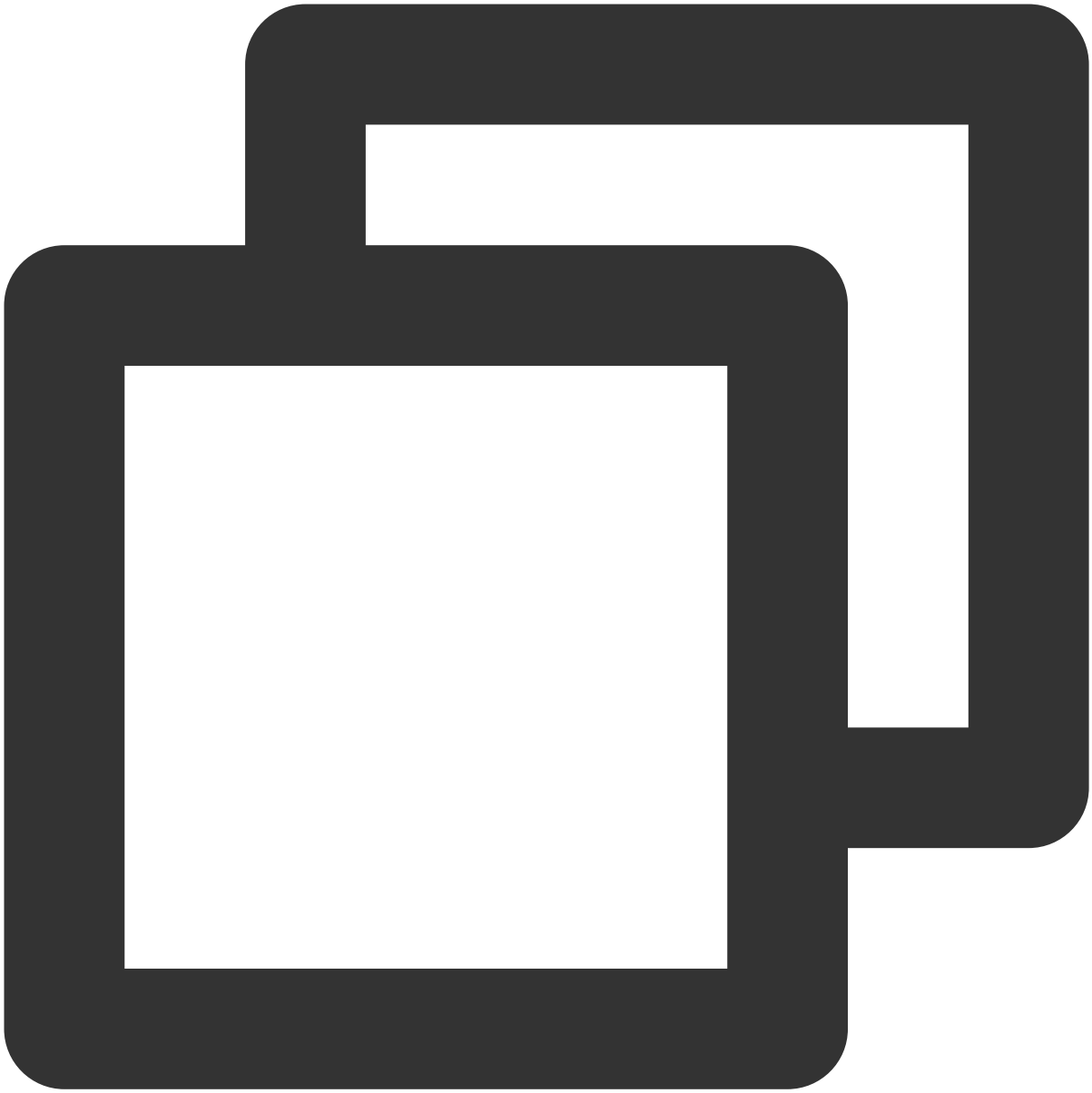


```
Builder setIp(String ip)
```

参数	描述
ip	可选，如果不采用默认 DNS 解析，可以设置解析后的 ip 地址

addHeader

添加请求 header 信息，key-value 形式

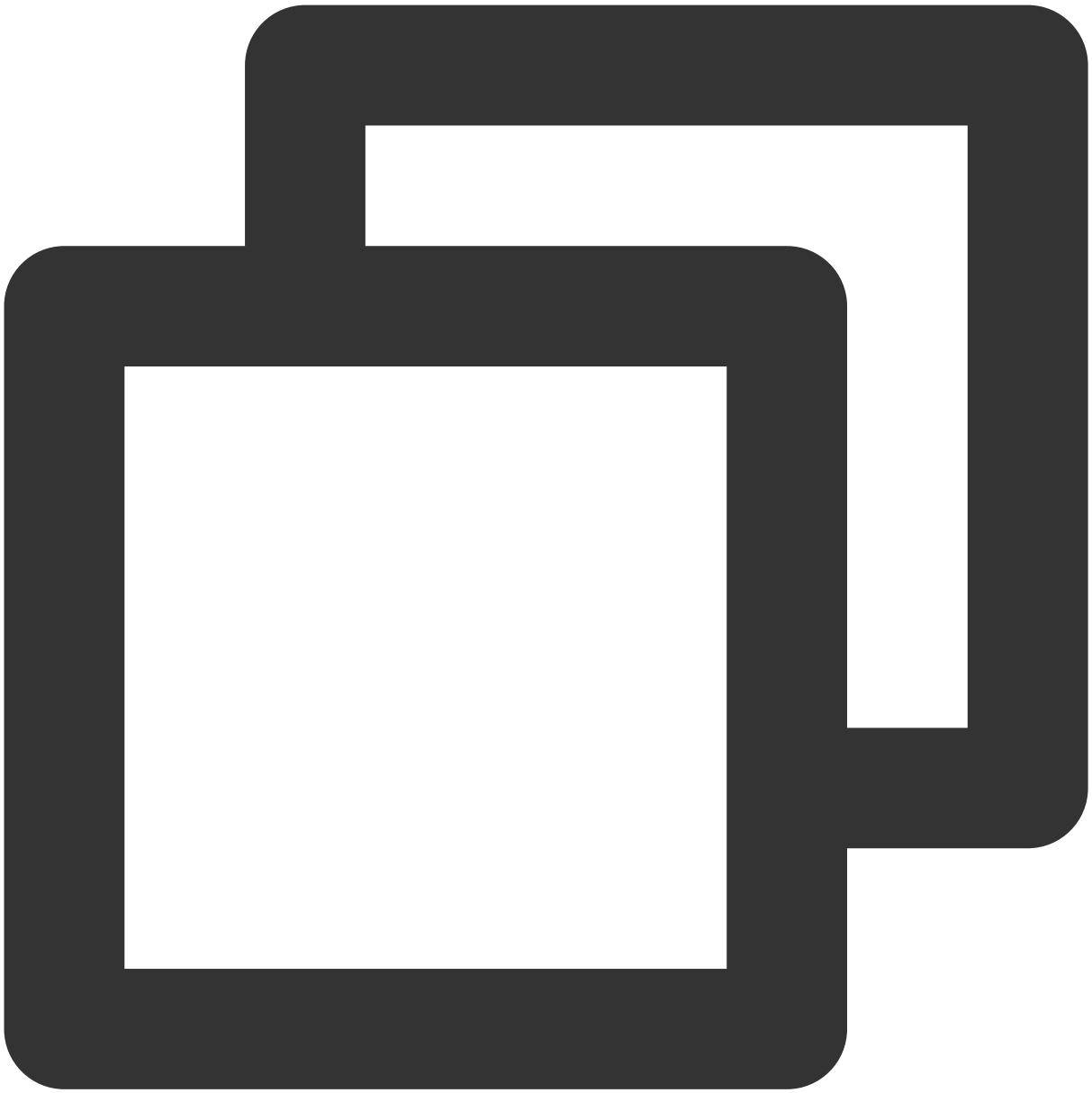


```
Builder addHeader(String key, String value)
```

参数	描述
key	Header 中的 key
value	Header 中的 value

get

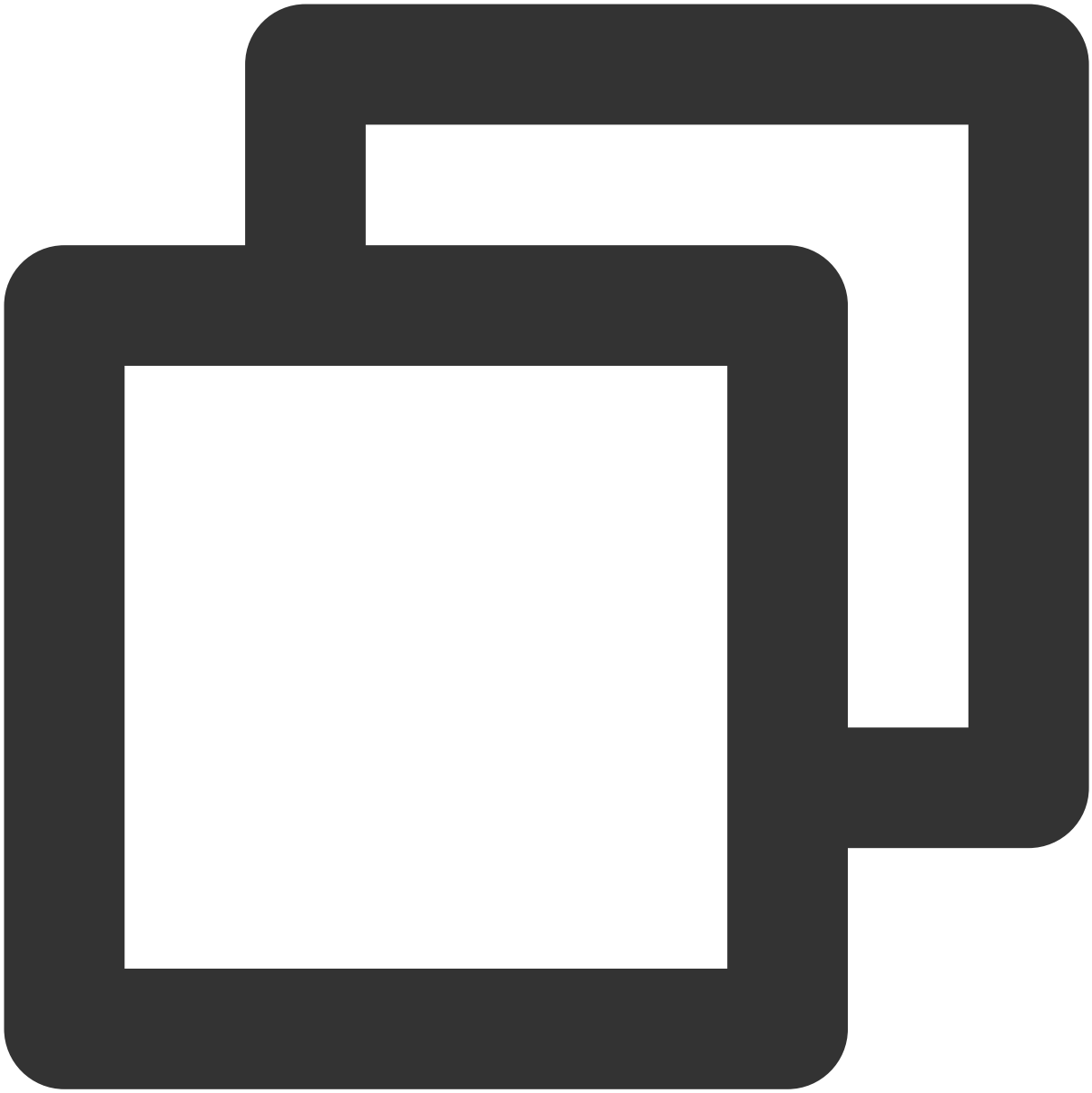
设置为 Get 请求



```
Builder get()
```

post

设置为 Post 请求

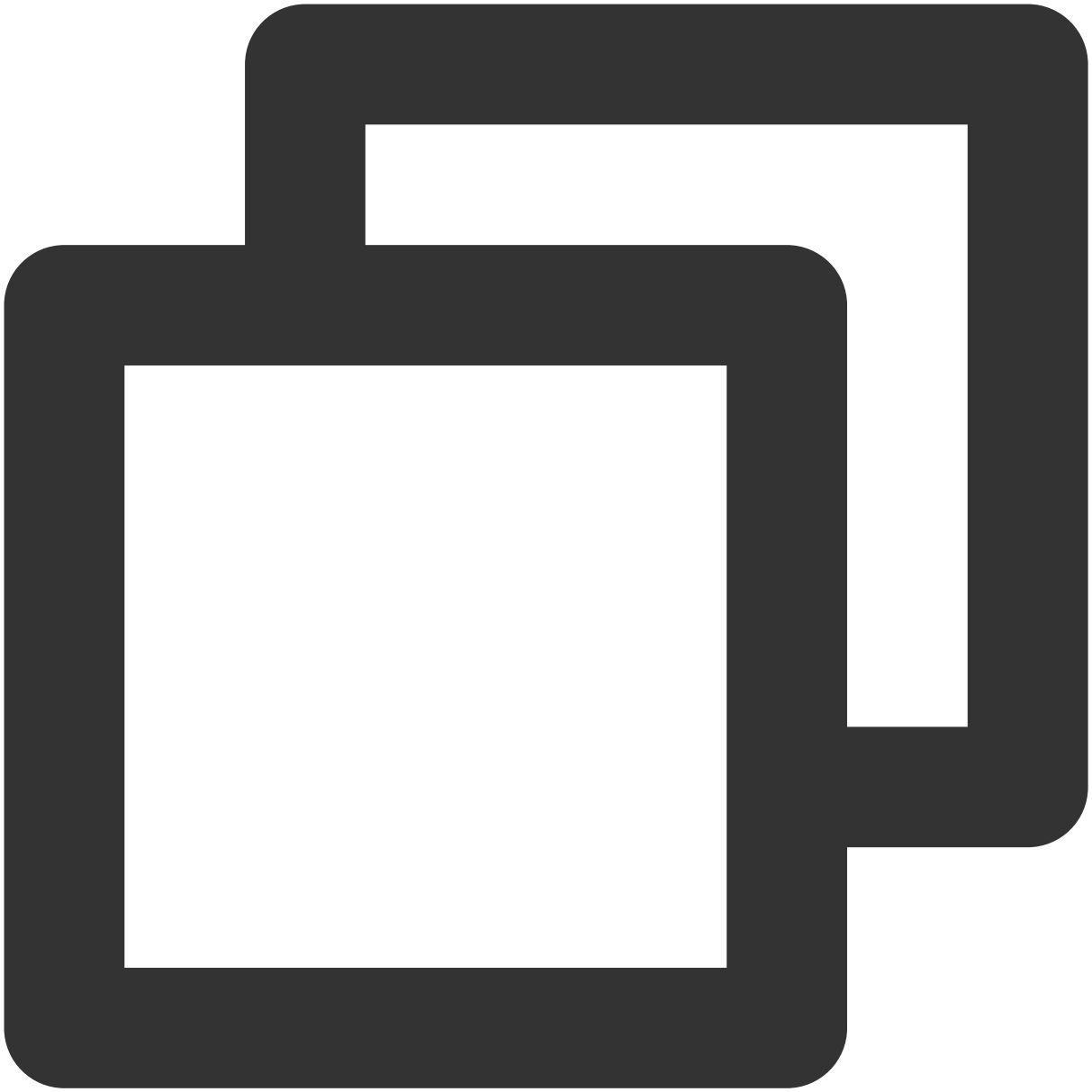


```
Builder post(RequestBody body)
```

参数	描述
body	body 数据

method

用于构建 Delete、Put 等请求

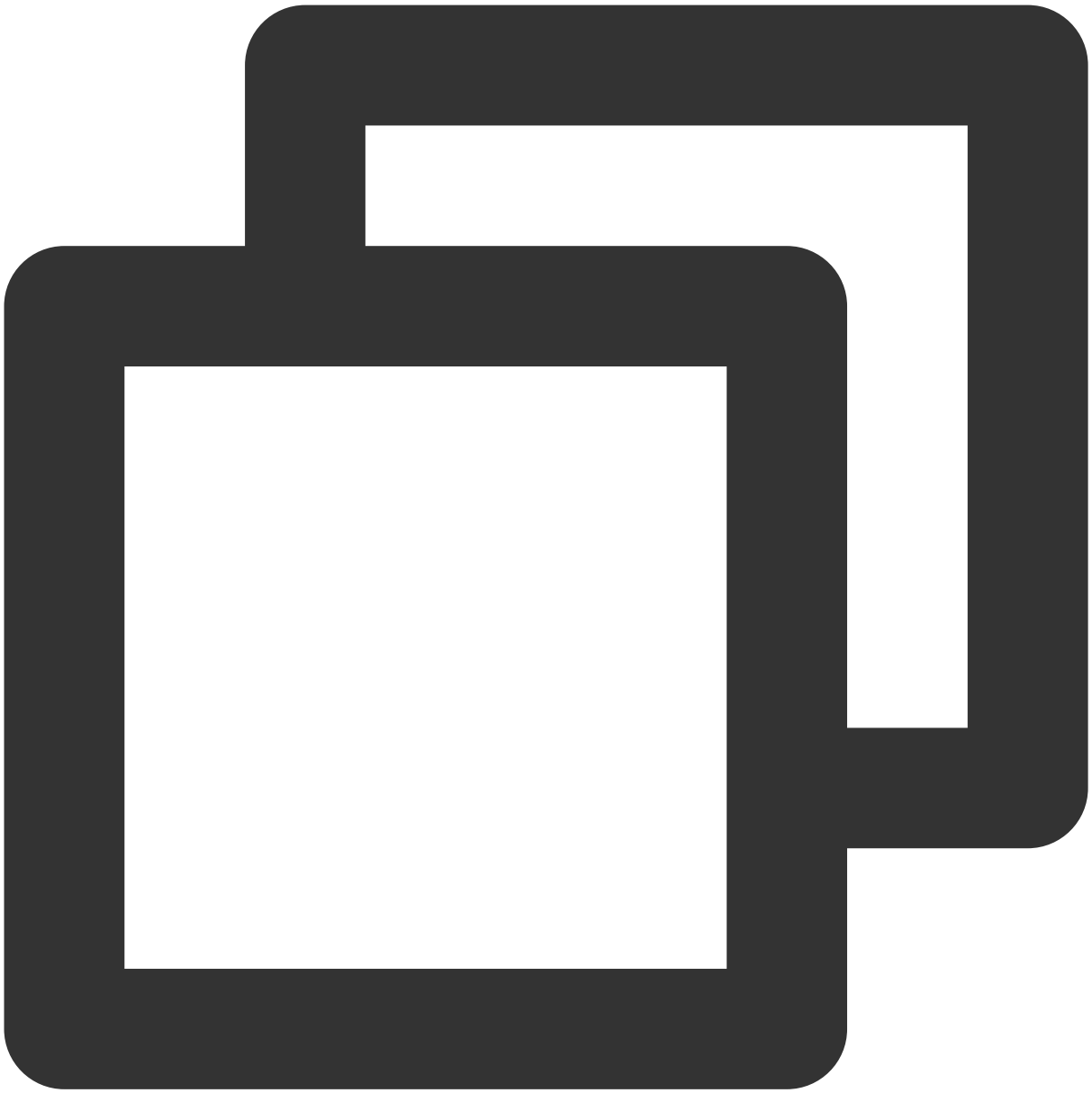


```
Builder method(String method, RequestBody body)
```

参数	描述
method	请求方式，可选：Put、Delete、head、patch
body	body 数据

build

创建 QuicRequest



```
QuicRequest build()
```

QuicResponse

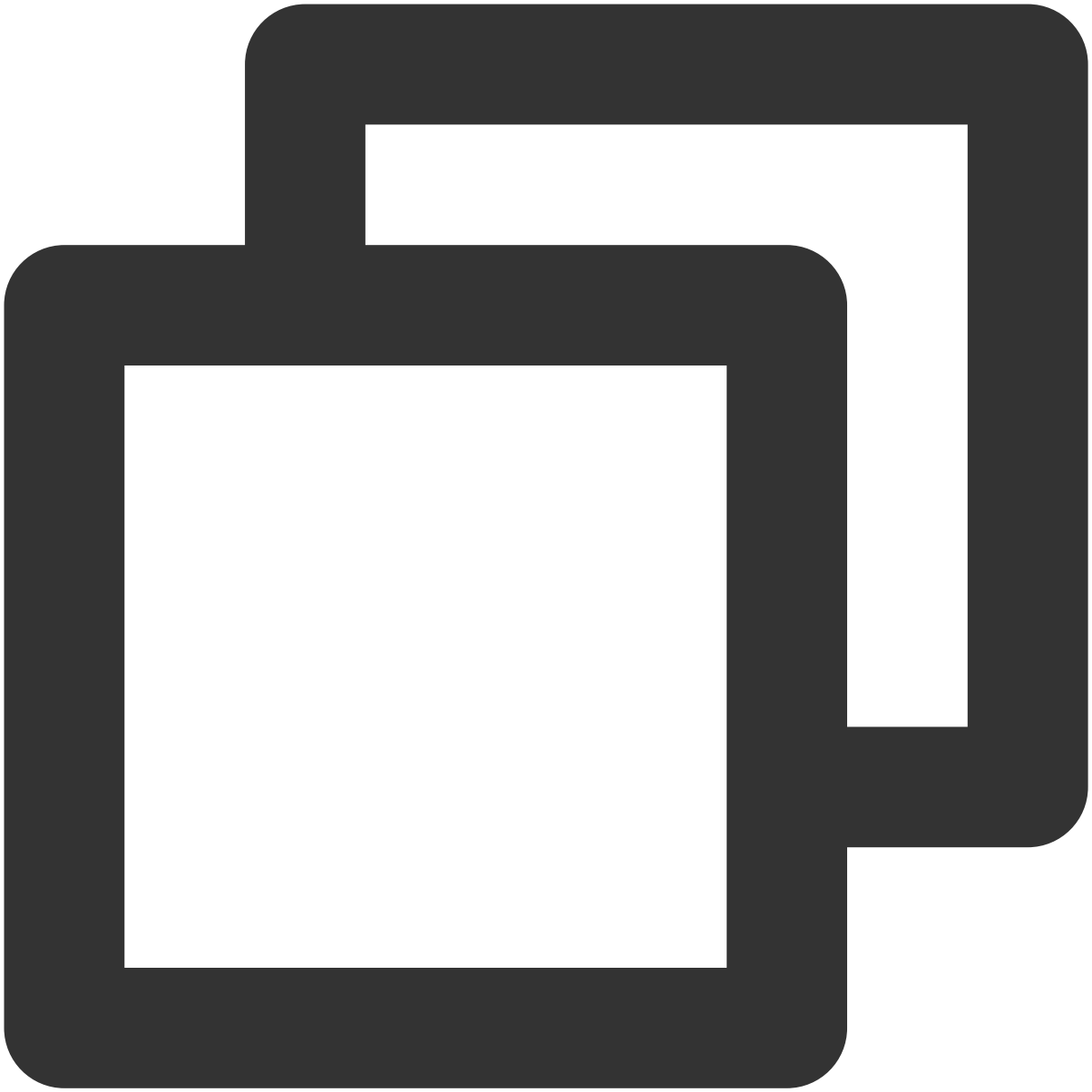
请求响应信息

API	描述

getCode	获取响应状态码
getHeaders	获取响应 Header 信息
getContentType	获取 Content-Type
getContentLength	获取 Content-Length
body	获取响应 body

getCode

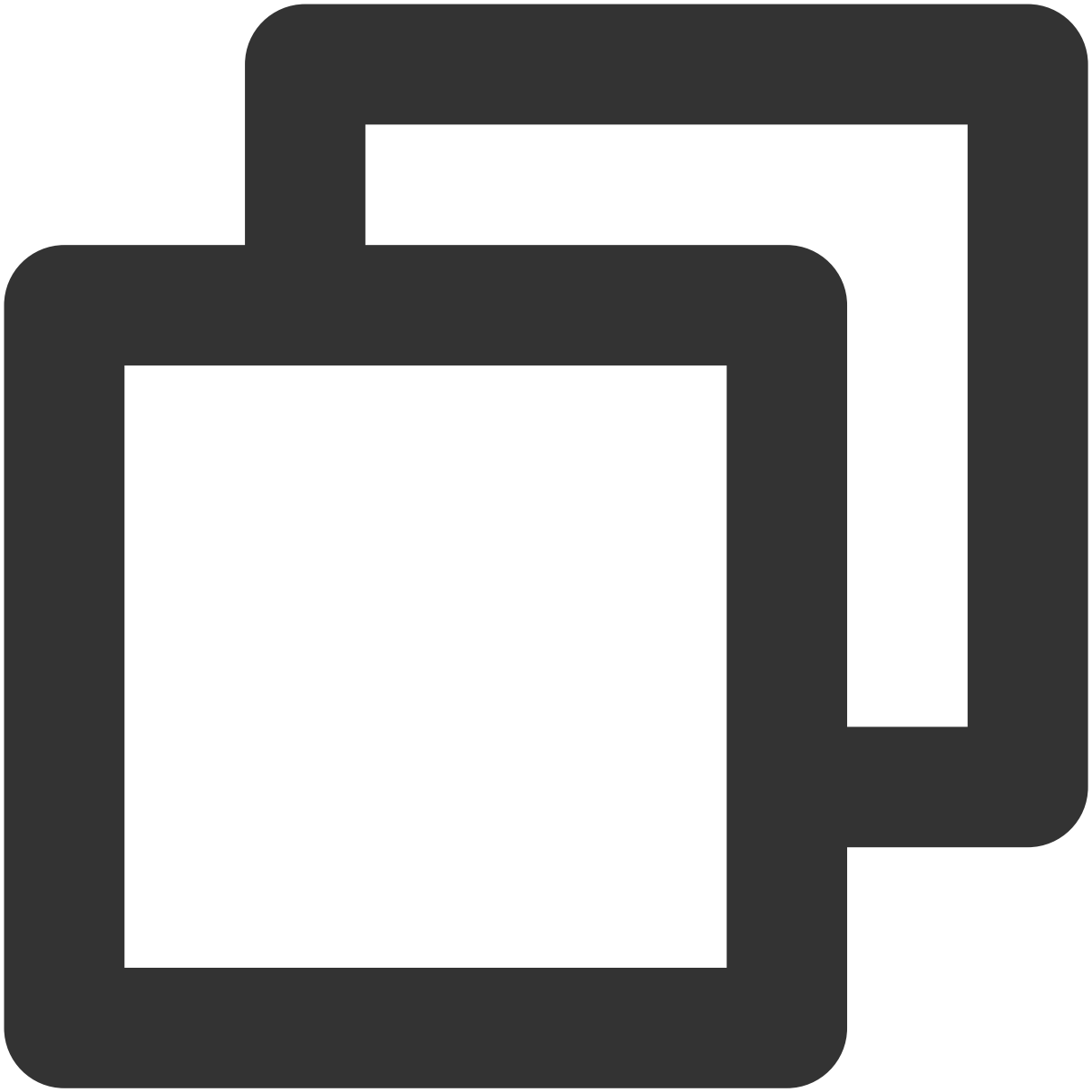
获取响应状态码



```
int getCode()
```

getHeaders

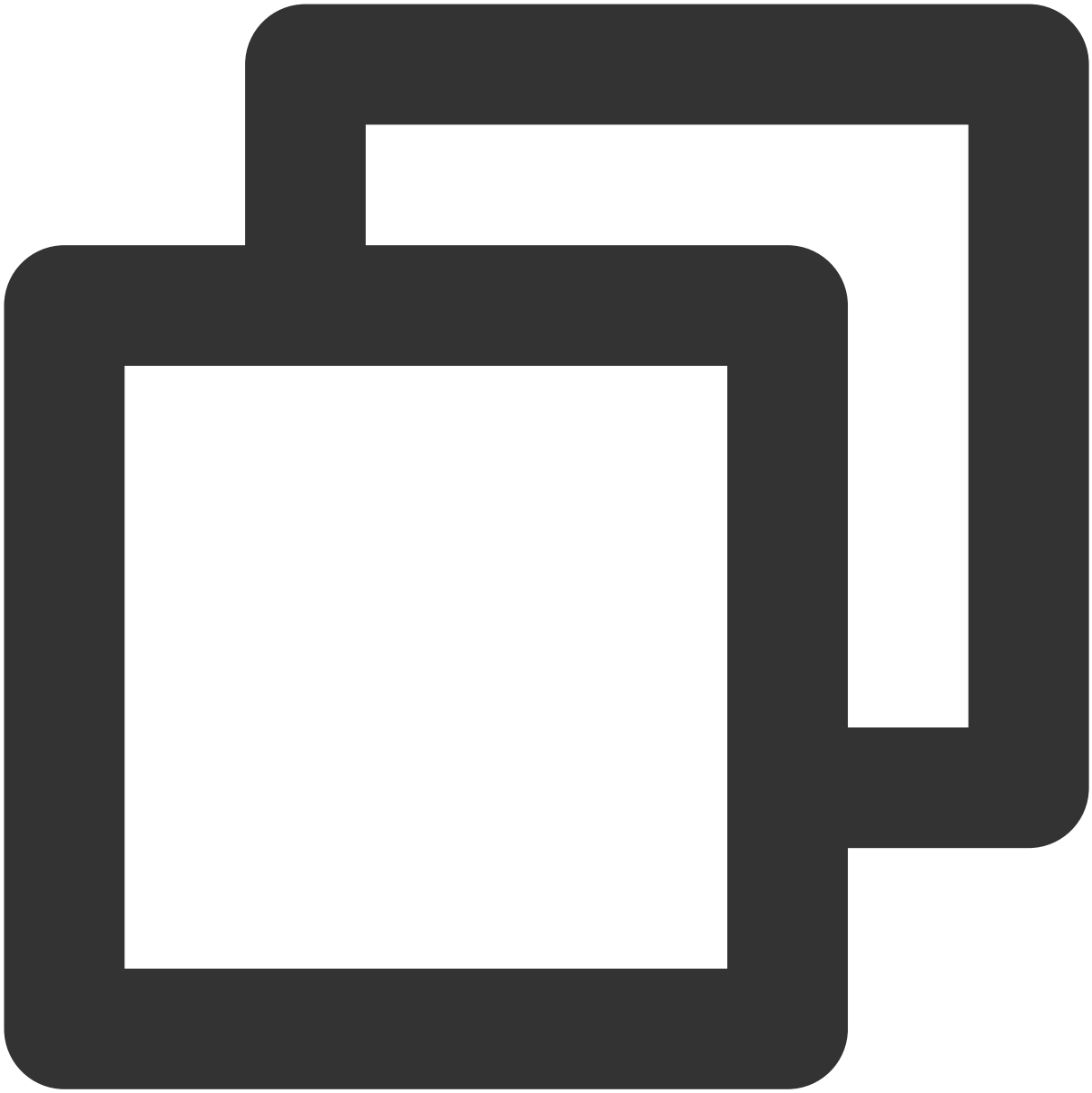
获取响应的 Header 信息，以列表形式返回



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

getContentType

获取响应的内容类型

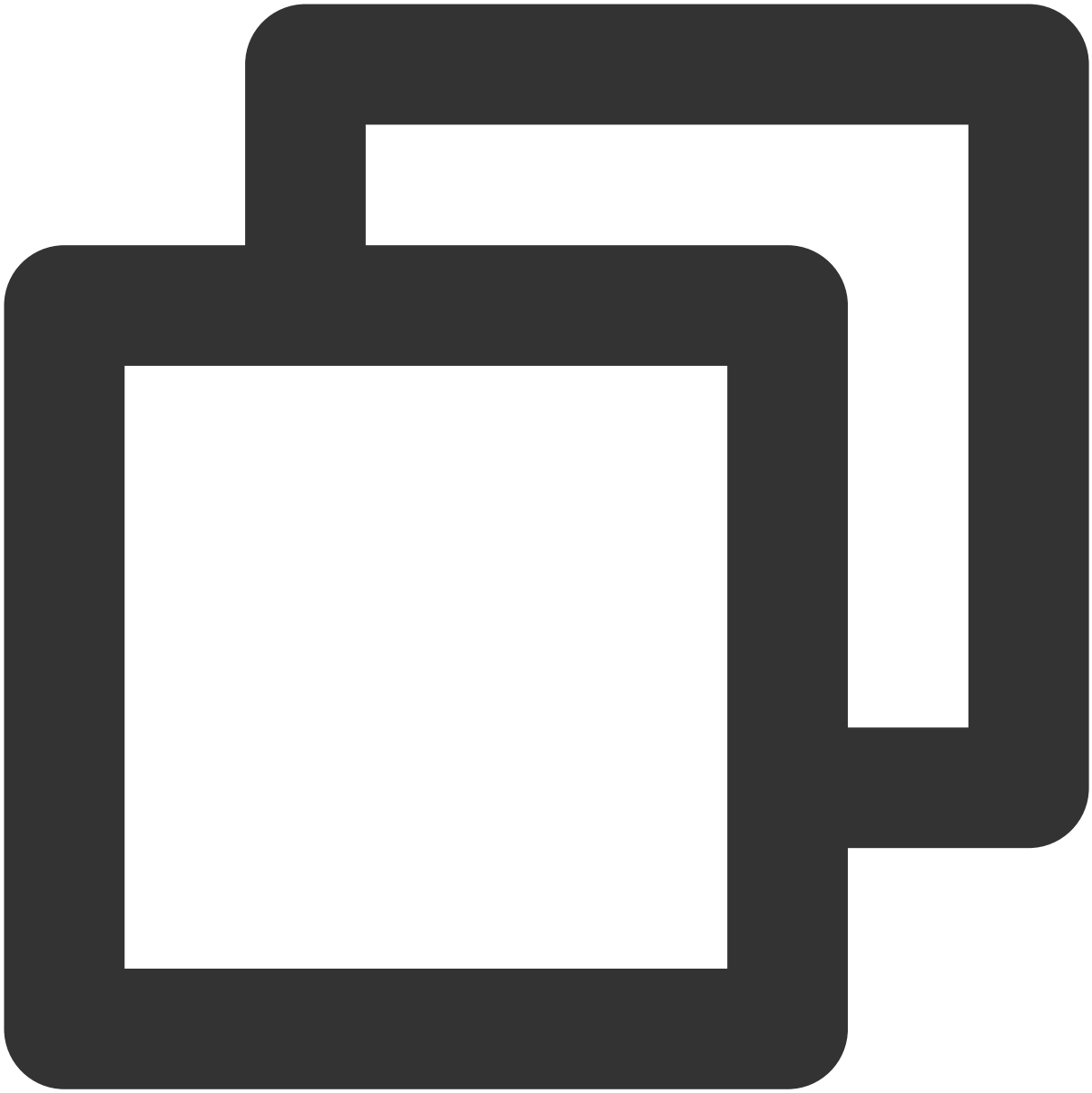


```
void setContentType(String contentType)
```

参数	描述
contentType	Header 中的 Content-Type 信息

getContentLength

获取内容长度

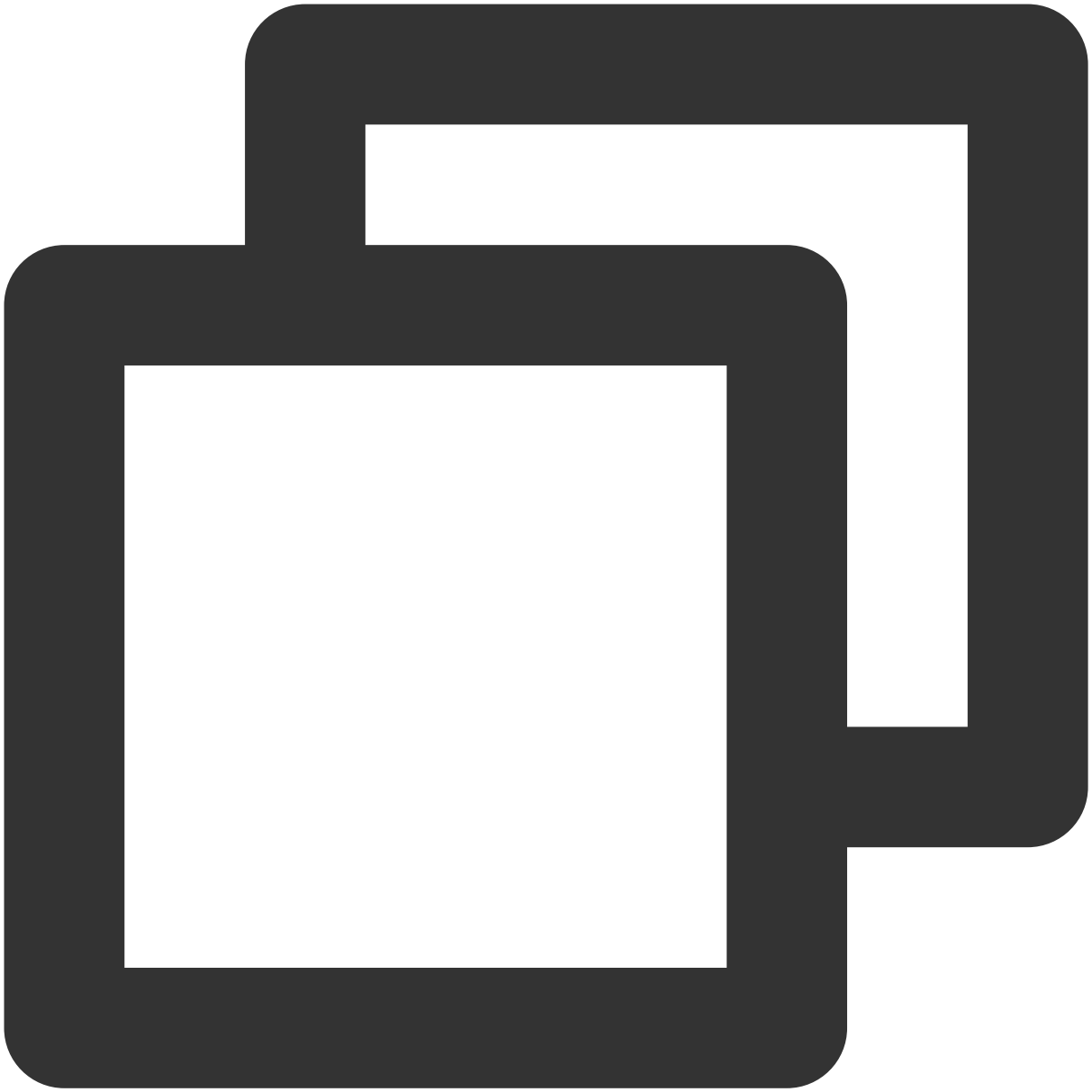


```
void setContentLength(long contentLength)
```

参数	描述
contentLength	Header 中的 Content-Length 信息

body

获取响应的 body 内容



```
ResponseBody body()
```

QuicCallback

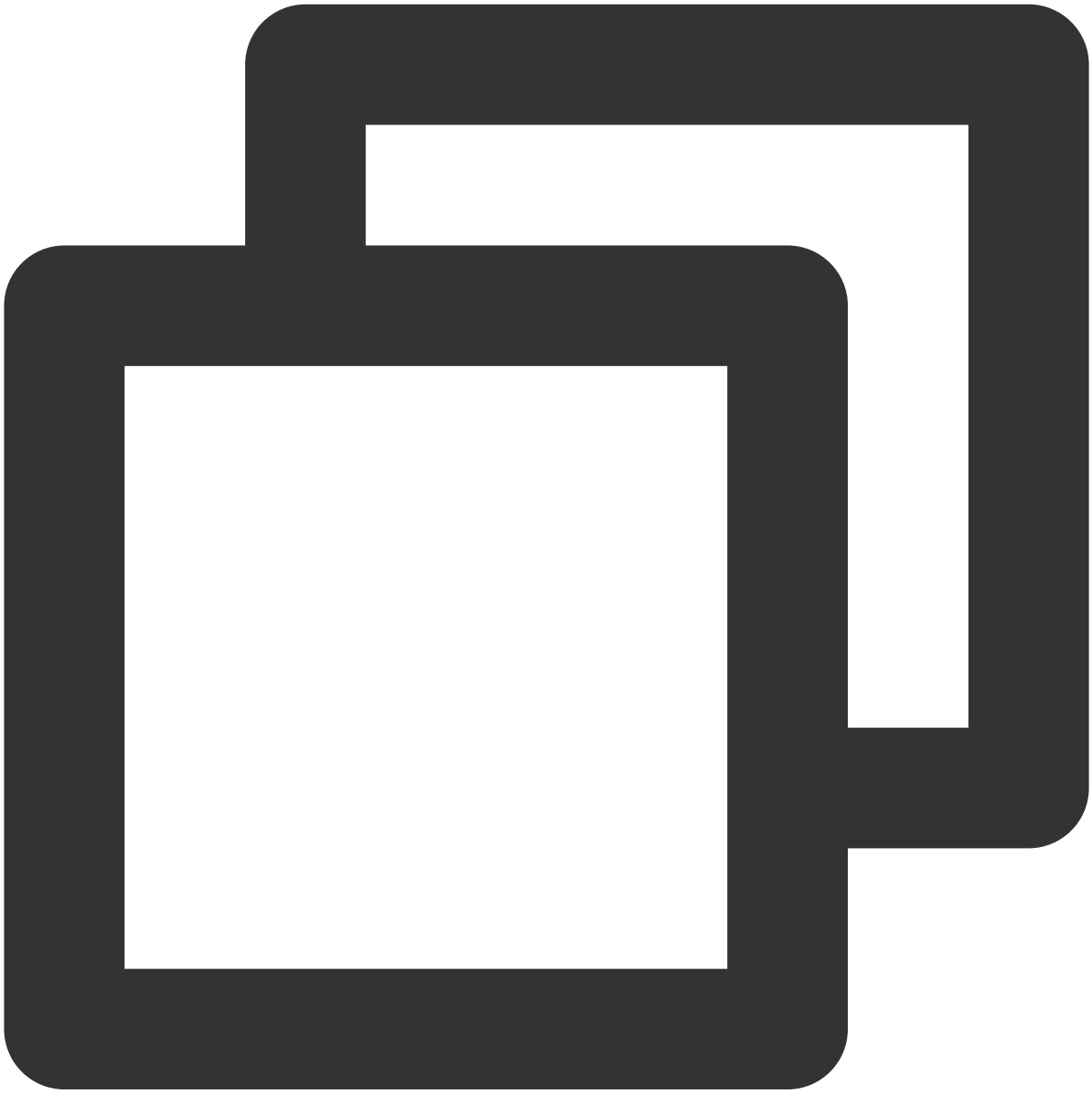
请求回调接口

API	描述
onResponse	请求成功回调

onFailed	请求失败回调

onResponse

请求正常响应时回调函数



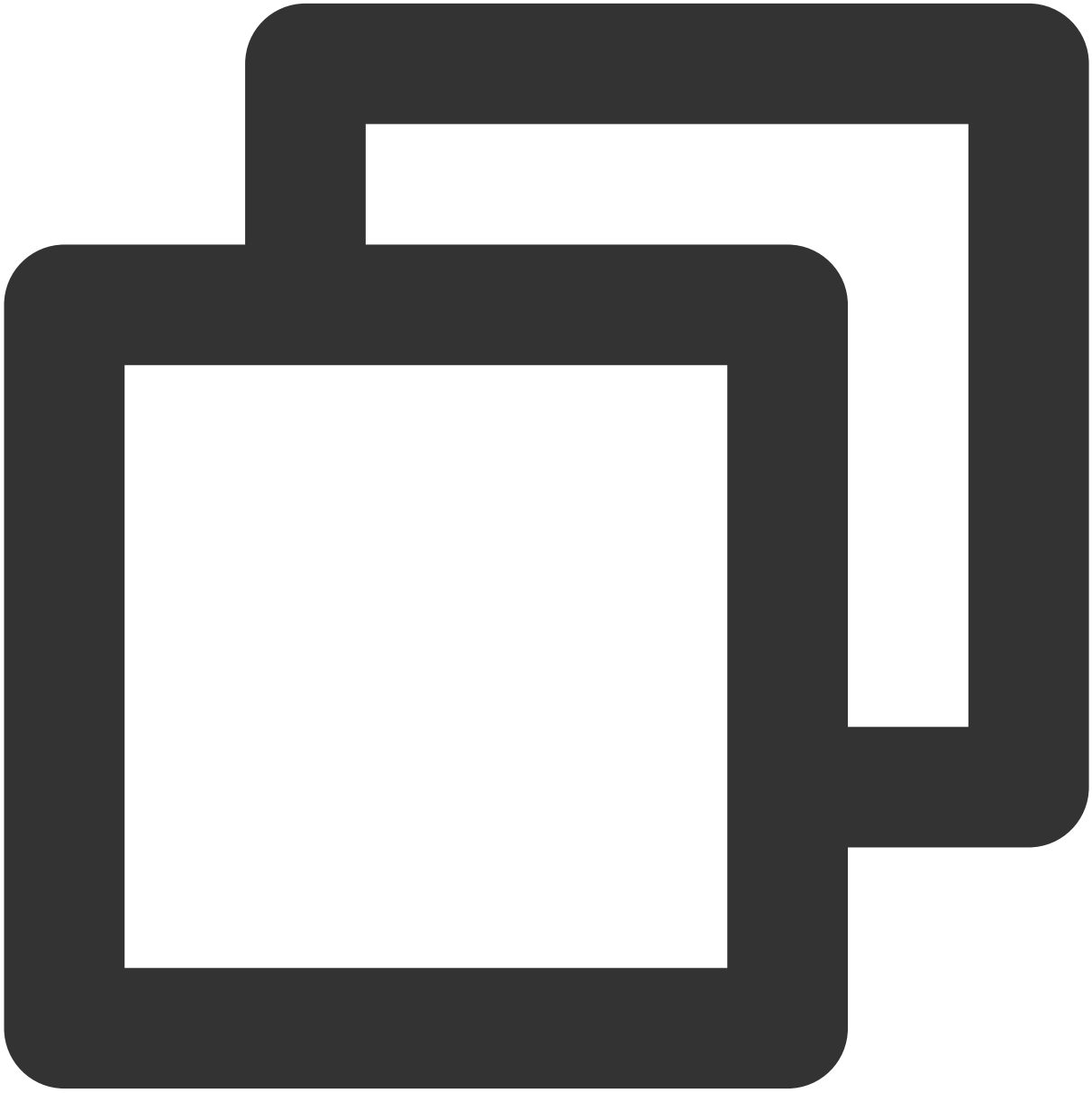
```
void onResponse(QuicCall call, QuicResponse response) throws IOException
```

参数	描述

call	用来管理 QUIC 请求，详见：QuicCall
response	QUIC 响应信息，详见：QuicResponse

onFailed

请求失败时的回调函数



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

参数	描述

call	用来管理 QUIC 请求，详见：QuicCall
errorCode	错误码
errorMsg	错误信息

QuicNetStats

可获取请求过程中的网络状态信息

API	描述
isValid	状态值是否有效
isQuic	是否 QUIC 请求
is0rtt	是否为 0-RTT 连接
isConnReuse	是否为连接复用
getConnectMs	获取连接耗时，单位：毫秒
getDnsMs	获取 dns 耗时，单位：毫秒
getDnsCode	获取 dns 错误码
getTtfbMs	获取首包耗时，单位：毫秒
getCompleteMs	获取请求完成时间（不含连接耗时），单位：毫秒
getSrttMs	获取滑动平均 rtt，单位：毫秒
getPacketsSent	获取发包量，单位：byte
getPacketsRetransmitted	获取重传包量，单位：byte
getBytesSent	获取发送字节数，单位：byte
getBytesRetransmitted	获取重传字节数，单位：byte
getPacketsLost	获取丢包量，单位：byte
getPacketsReceived	获取收包量，单位：byte
getBytesReceived	获取收到字节数，单位：byte
getStreamBytesReceived	获取 stream 层收到的字节数，单位：byte

iOS

最近更新时间：2023-07-26 15:31:53

API概览

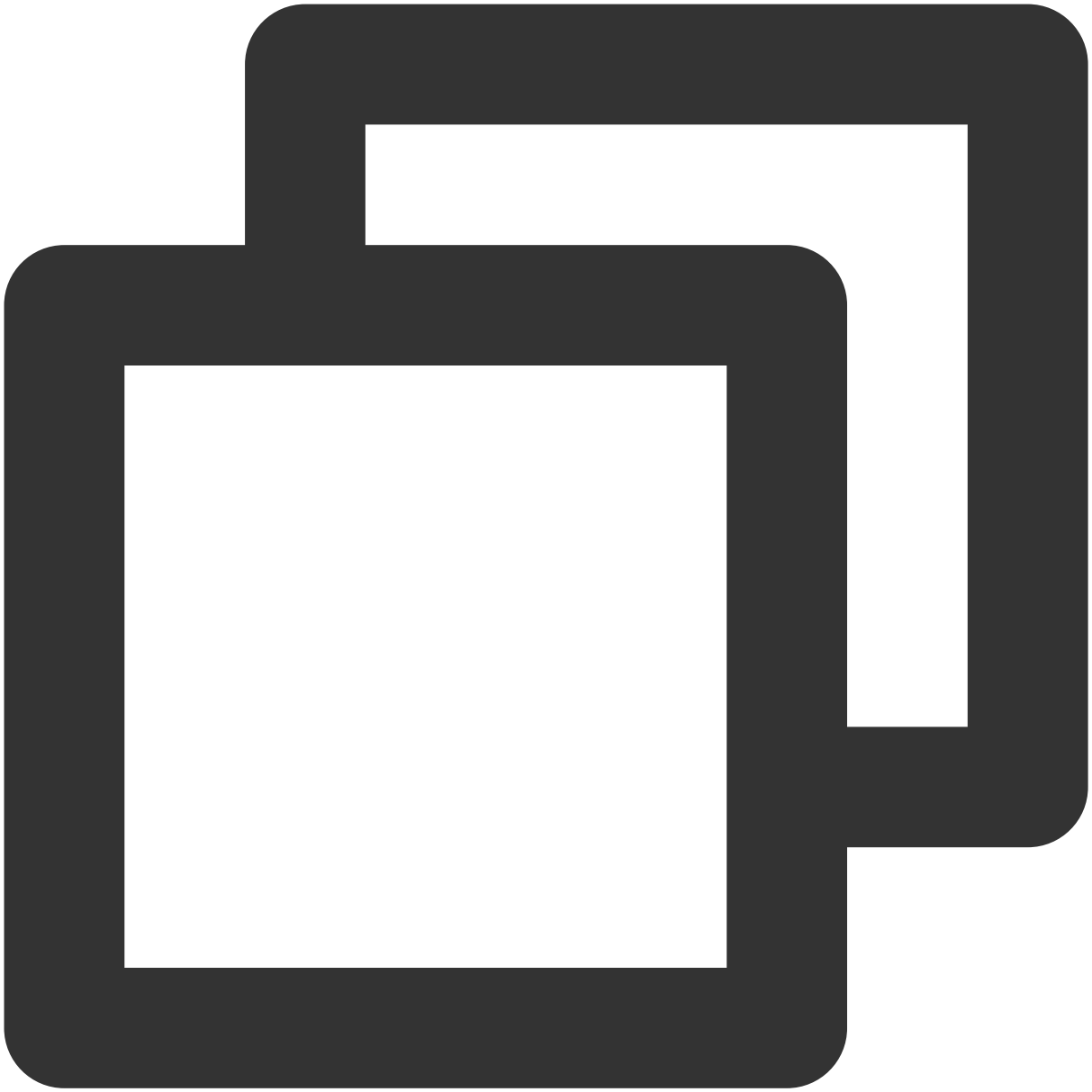
API	描述
TQUICHTTPSessionManager	会话管理接口
TQUICURLSessionConfiguration	QUIC 请求配置信息
TQUICURLSessionDataTask	请求任务管理接口
TQUICURLRequestSerialization	请求数据序列化接口
TQUICURLResponseSerialization	响应数据序列化接口

TQUICHTTPSessionManager

API	描述
manager	静态方法，构造 TQUICHTTPSessionManager
initWithSessionConfiguration	根据配置构造 TQUICHTTPSessionManager 实例，配置信息见 TQUICURLSessionConfiguration
initWithBaseURL	根据 Url 构造 TQUICHTTPSessionManager 实例
GET	发起 GET 请求
POST	发起 POST 请求

manger

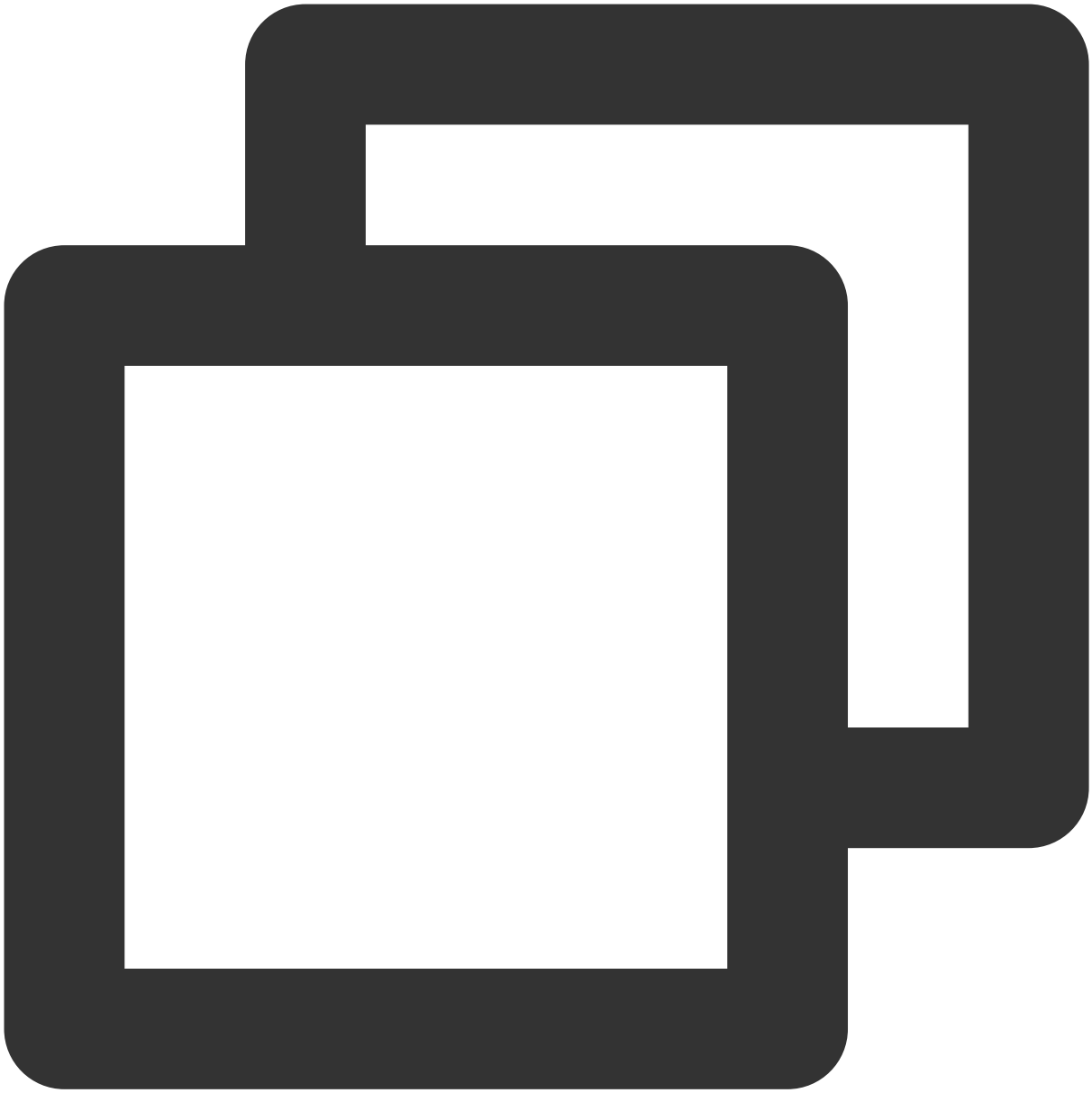
构造 TQUICHTTPSessionManager，使用默认 QUIC 配置



```
+ (instancetype)manager
```

initWithSessionConfiguration

创建
TQUICHTTPSessionManager
实例，并传入 QUIC 配置信息



```
- (instancetype)initWithSessionConfiguration:(nullable TQUICURLSessionConfiguration
```

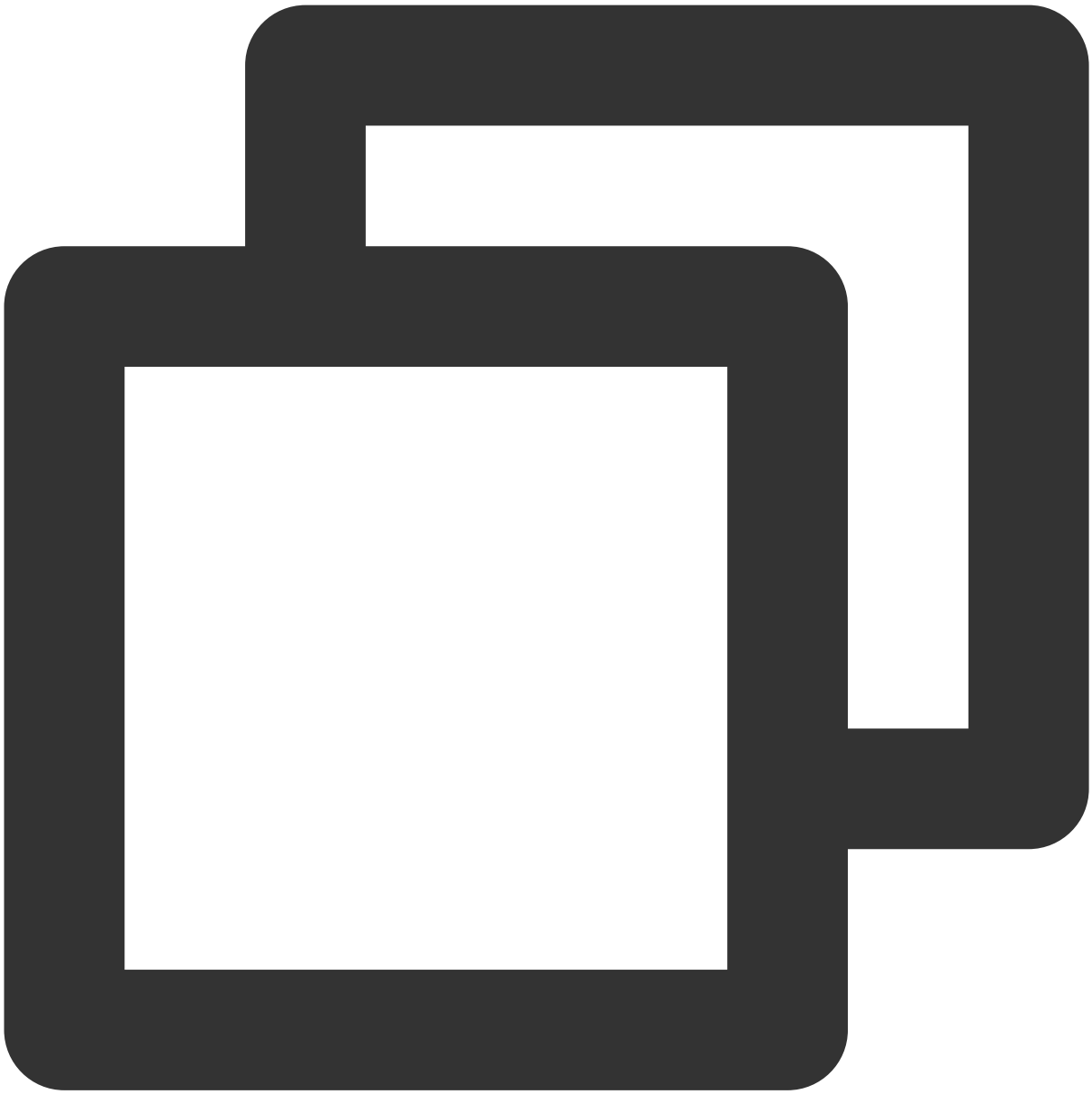
参数	描述
configuration	QUIC 请求配置，详细信息见：TQUICURLSessionConfiguration

initWithBaseURL

创建

TQUICHTTPSessionManager

实例，并传入url，配置请求参数在此 url 为基础上添加



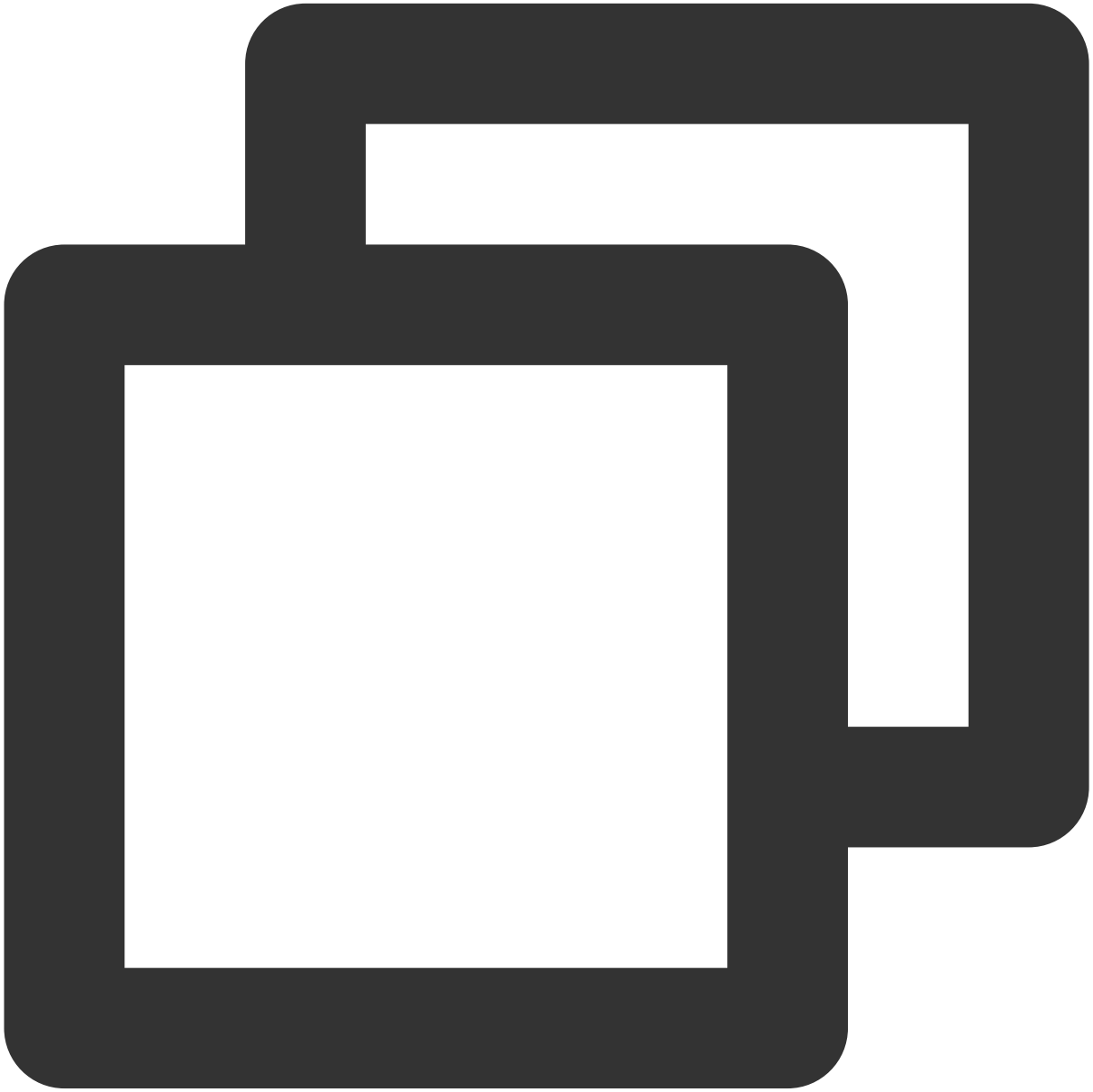
```
- (instancetype)initWithBaseURL:(nullable NSURL *)baseURL;

- (instancetype)initWithBaseURL:(nullable NSURL *)baseURL
            sessionConfiguration:(nullable TQUICURLSessionConfiguration *)configurat
```

参数	描述

baseURL	请求域名
configuration	QUIC 请求配置，详细信息见：TQUICURLSessionConfiguration

GET

创建 GET 请求



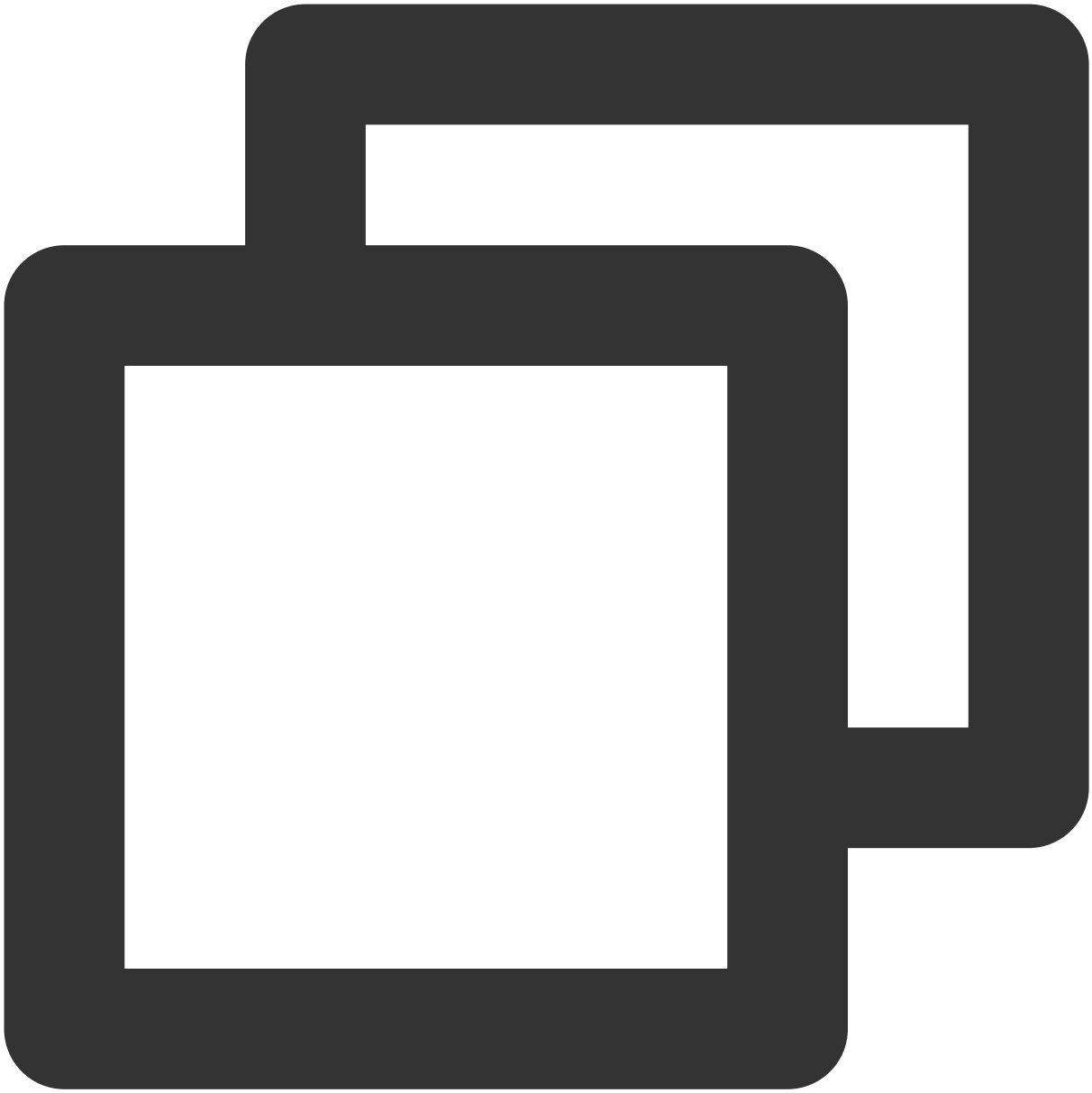
```
- (nullable TQUICURLSessionDataTask *)GET:(NSString *)URLString
```

```
                                parameters:(nullable id)parameters

                                   headers:(nullable NSDictionary <NSString *, NSStr

                            timeoutInterval:(NSTimeInterval)timeoutInterval

                           downloadProgress:(nullable TQUICURLSessionTaskDownloadProg

                                   success:(nullable TQUICURLSessionTaskSuccess)succ

                                   failure:(nullable TQUICURLSessionTaskFailure)fail
```

参数	描述
URLString	请求 Url
parameters	请求参数
headers	请求 Header 信息
timeoutInterval	连接超时时间
downloadProgress	下载进度回调
success	请求成功回调
failure	请求失败回调

POST

创建 POST 请求



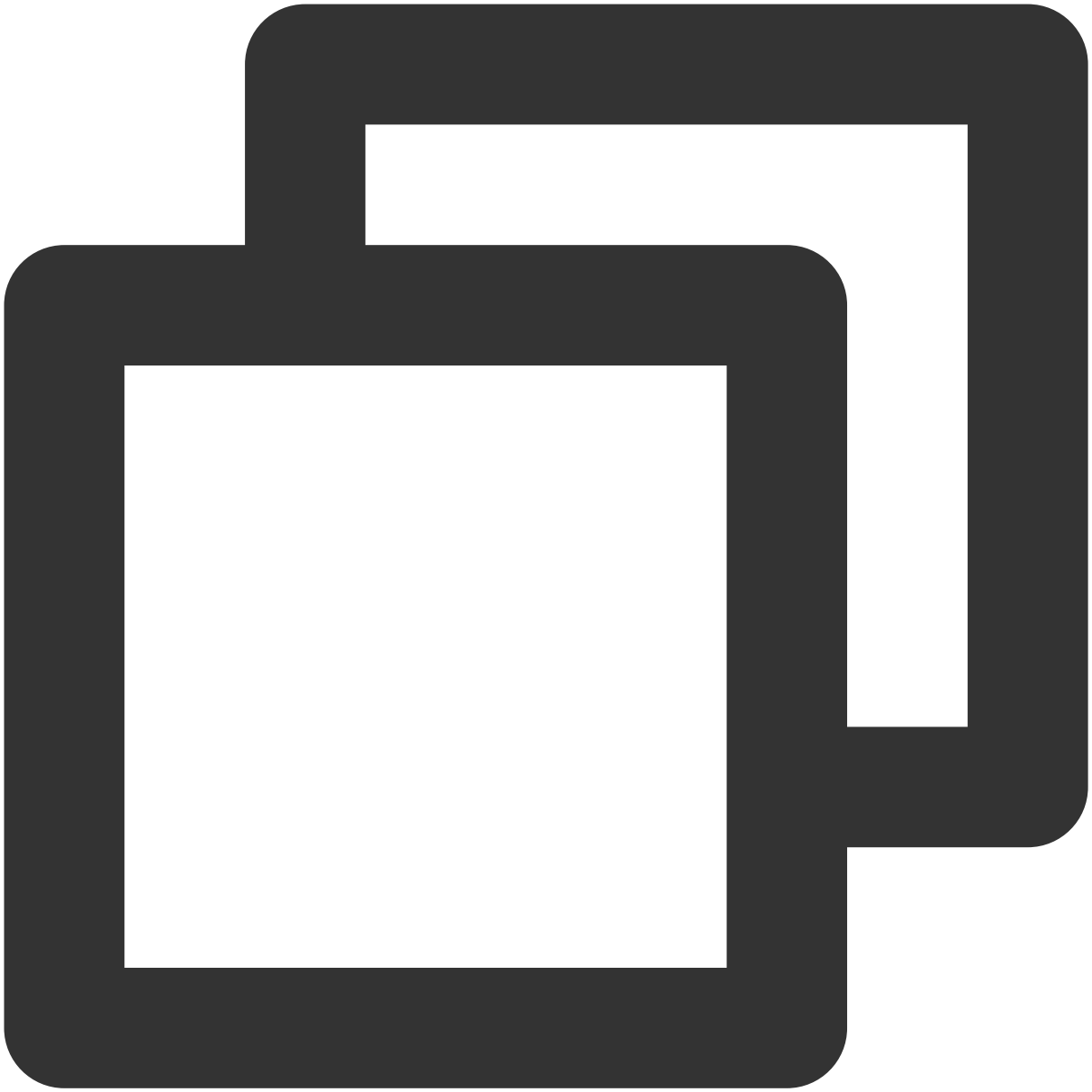
```
- (nullable TQUICURLSessionDataTask *)POST:(NSString *)URLString
                                parameters:(nullable id)parameters
                                   headers:(nullable NSDictionary <NSString *, NSSt
                           timeoutInterval:(NSTimeInterval)timeoutInterval
                            uploadProgress:(nullable TQUICURLSessionTaskUploadProgr
```

```
                                    success:(nullable TQUICURLSessionTaskSuccess)suc

                                    failure:(nullable TQUICURLSessionTaskFailure)fai
```

参数	描述
URLString	请求 Url
parameters	请求参数
headers	请求 Header 信息
timeoutInterval	连接超时时间
uploadProgress	上传进度回调
success	请求成功回调
failure	请求失败回调

TQUICURLSessionTaskSuccess

任务执行成功回调 block

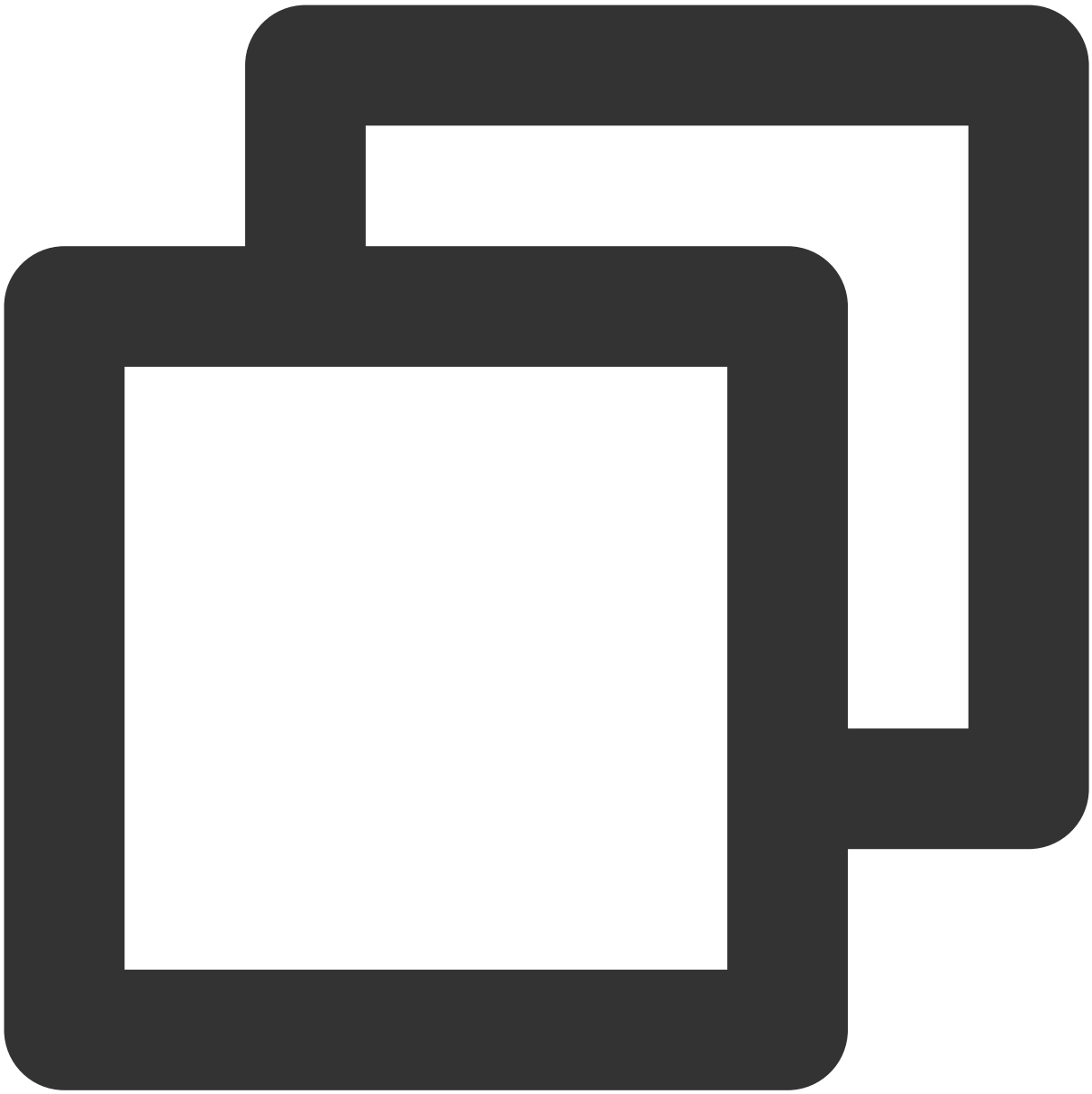


```
typedef void (^TQUICURLSessionTaskSuccess)(TQUICURLSessionTask *task, id _Nullable
```

参数	描述
task	请求任务
responseObject	响应数据

TQUICURLSessionTaskFailure

任务执行失败回调 block



```
typedef void (^TQUICURLSessionTaskFailure)(TQUICURLSessionTask * _Nullable task, NS
```

参数	描述
task	请求任务
error	错误信息

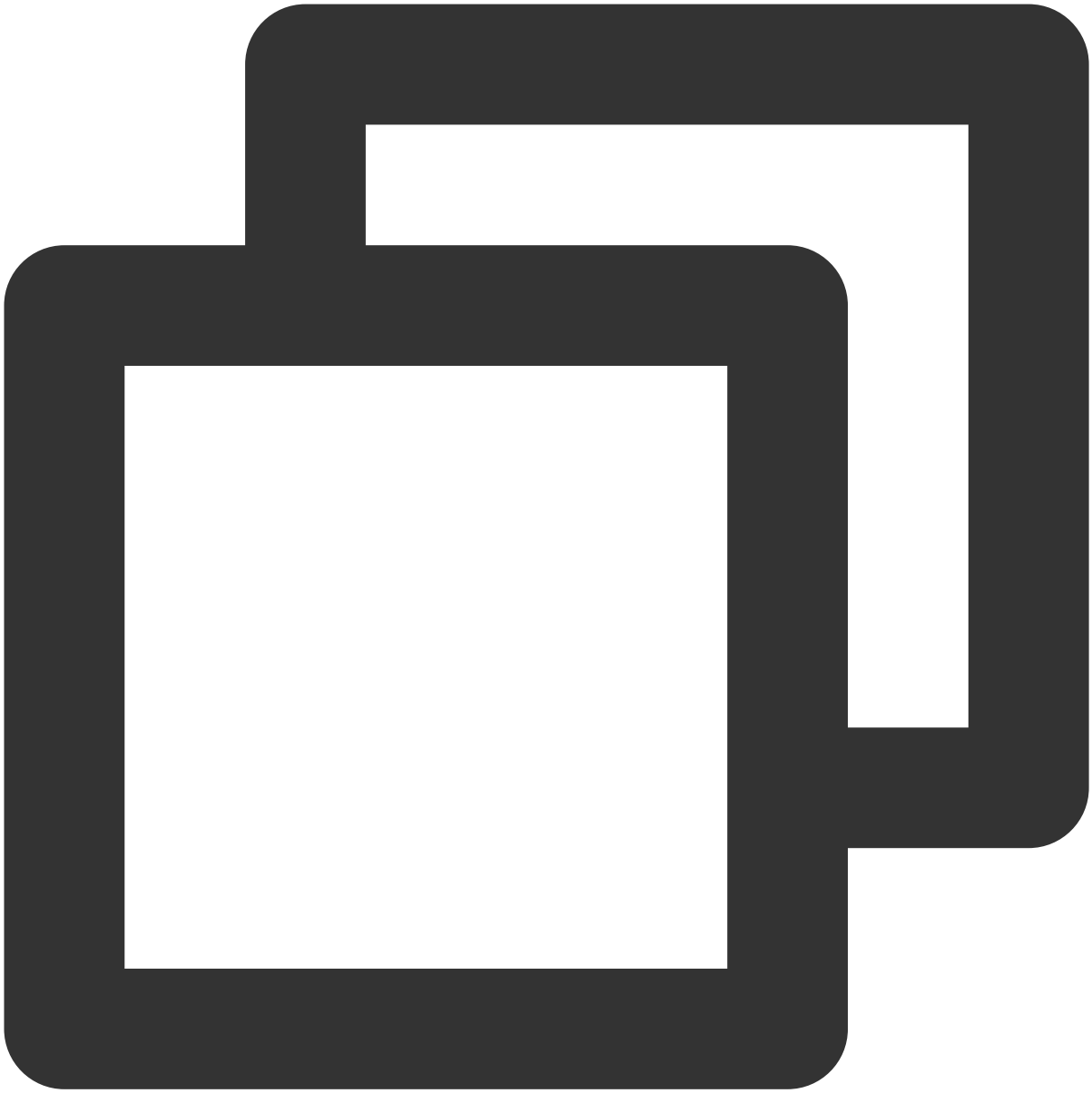
TQUICURLSessionManager

会话管理接口

API	描述
initWithSessionConfiguration	根据配置构造实例
dataTaskWithRequest	使用指定 NSURLRequest 参数发起请求
setTaskDidFinishCollectingMetricsBlock	配置请求完成后的数据统计回调
setTaskDidReceiveResponseBlock	设置接收响应结果回调 block

initWithSessionConfiguration

创建 TQUICURLSessionManager 实例，并传入 QUIC 配置信息

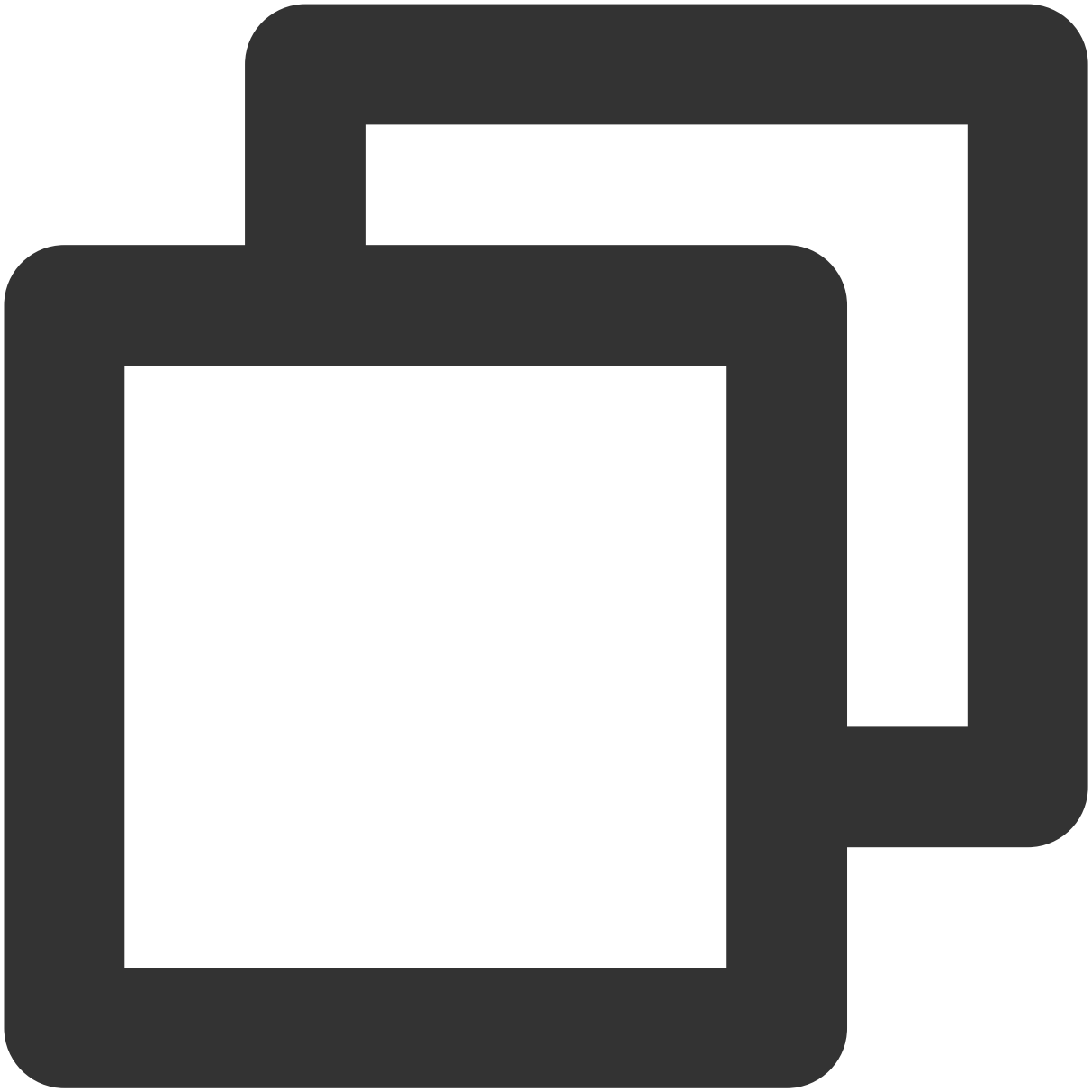


```
- (instancetype)initWithSessionConfiguration:(nullable TQUICURLSessionConfiguration
```

参数	描述
configuration	QUIC请求配置

dataTaskWithRequest

使用指定 NSURLRequest 参数发起请求



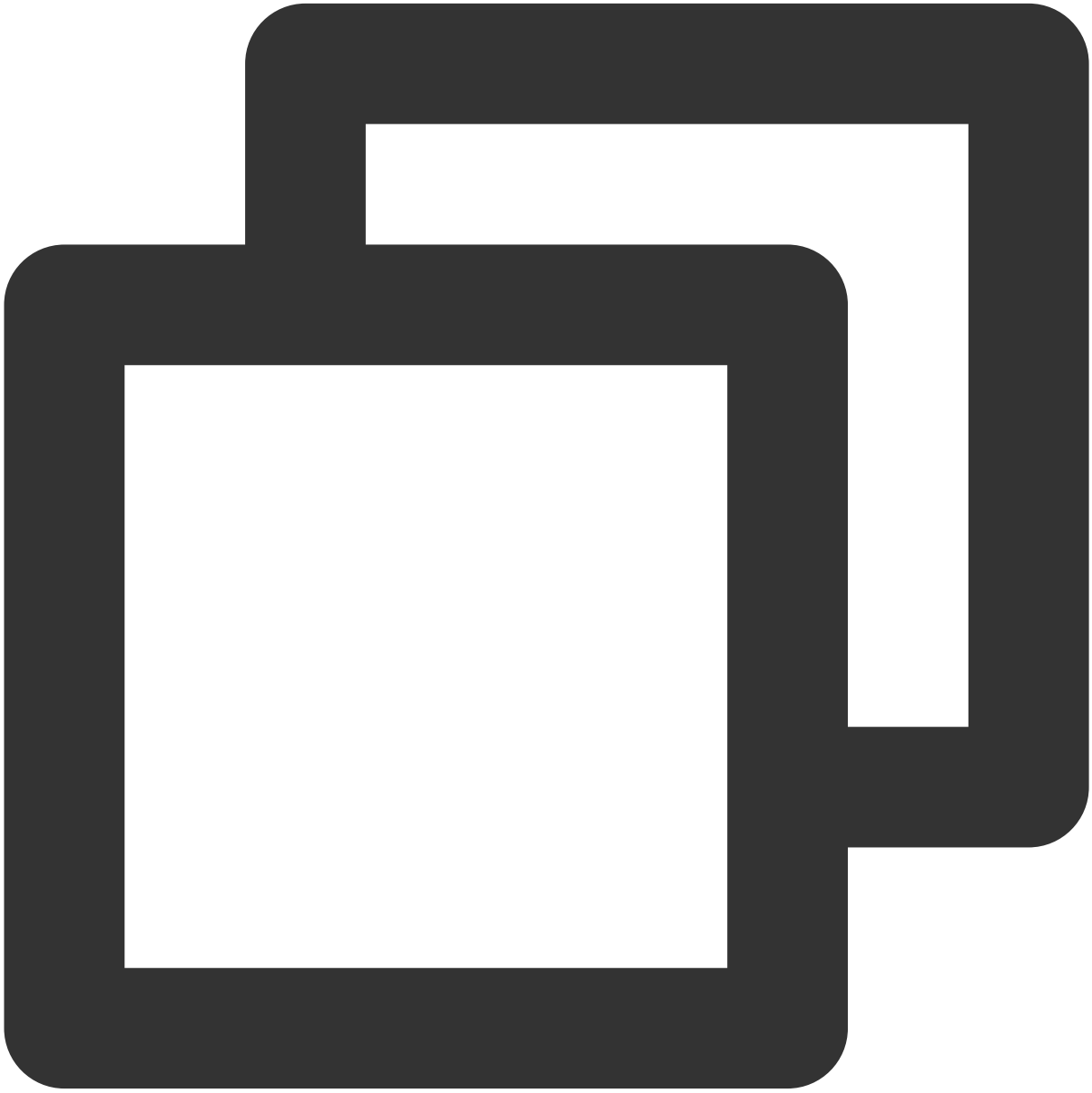
```
- (TQUICURLSessionDataTask *)dataTaskWithRequest:(NSURLRequest *)request
                                  uploadProgress:(nullable TQUICURLSessionTaskUploa
                                downloadProgress:(nullable TQUICURLSessionTaskDownl
                               completionHandler:(nullable TQUICURLSessionTaskCompl
```

参数	描述
request	请求配置，参考系统 NSURLRequest
uploadProgress	上传进度回调

downloadProgress	下载进度回调
completionHandler	请求完成回调

setTaskDidFinishCollectingMetricsBlock

配置请求完成后的数据统计回调



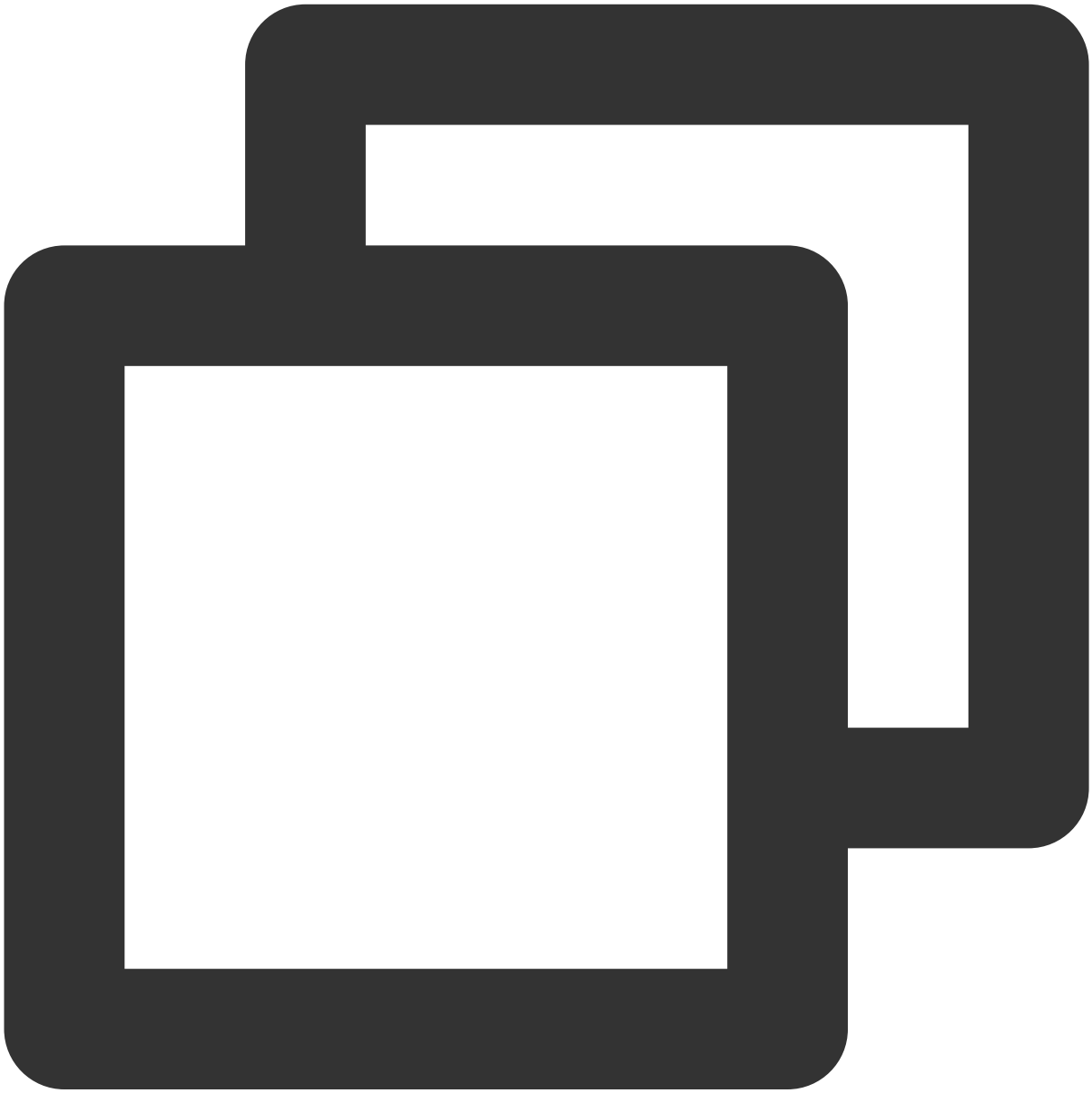
```
- (void)setTaskDidFinishCollectingMetricsBlock:(nullable TQUICURLSessionTaskDidFini
```

参数	描述

block	请求完成数据统计回调

setTaskDidReceiveResponseBlock

设置接收响应结果回调 block

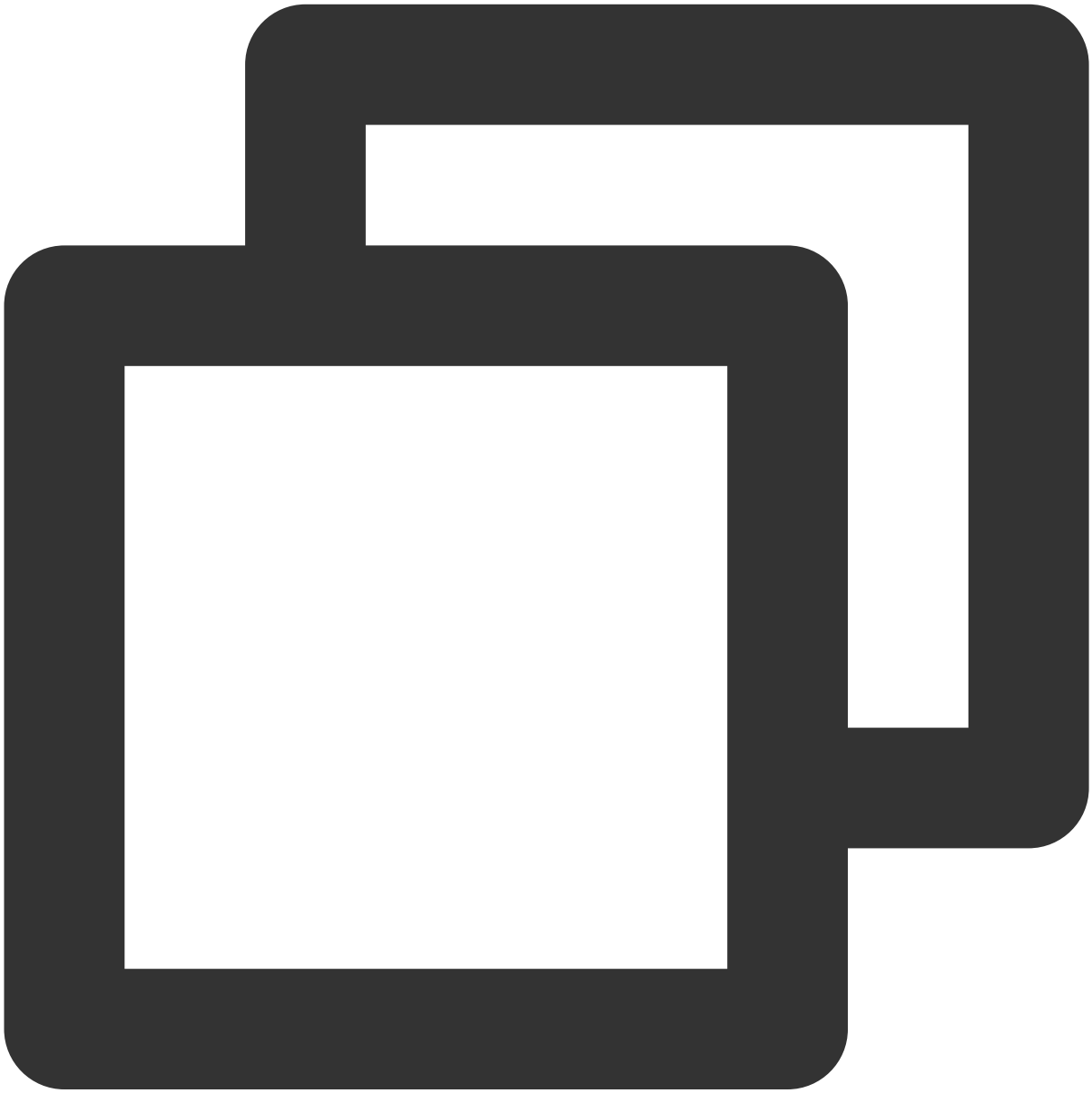


```
- (void)setTaskDidReceiveResponseBlock:(nullable TQUICURLSessionTaskDidReceiveRespo
```

参数	描述
block	接收到响应结果后的回调 block

TQUICURLSessionTaskDidReceiveResponseBlock

会话任务收到响应结果回调 block



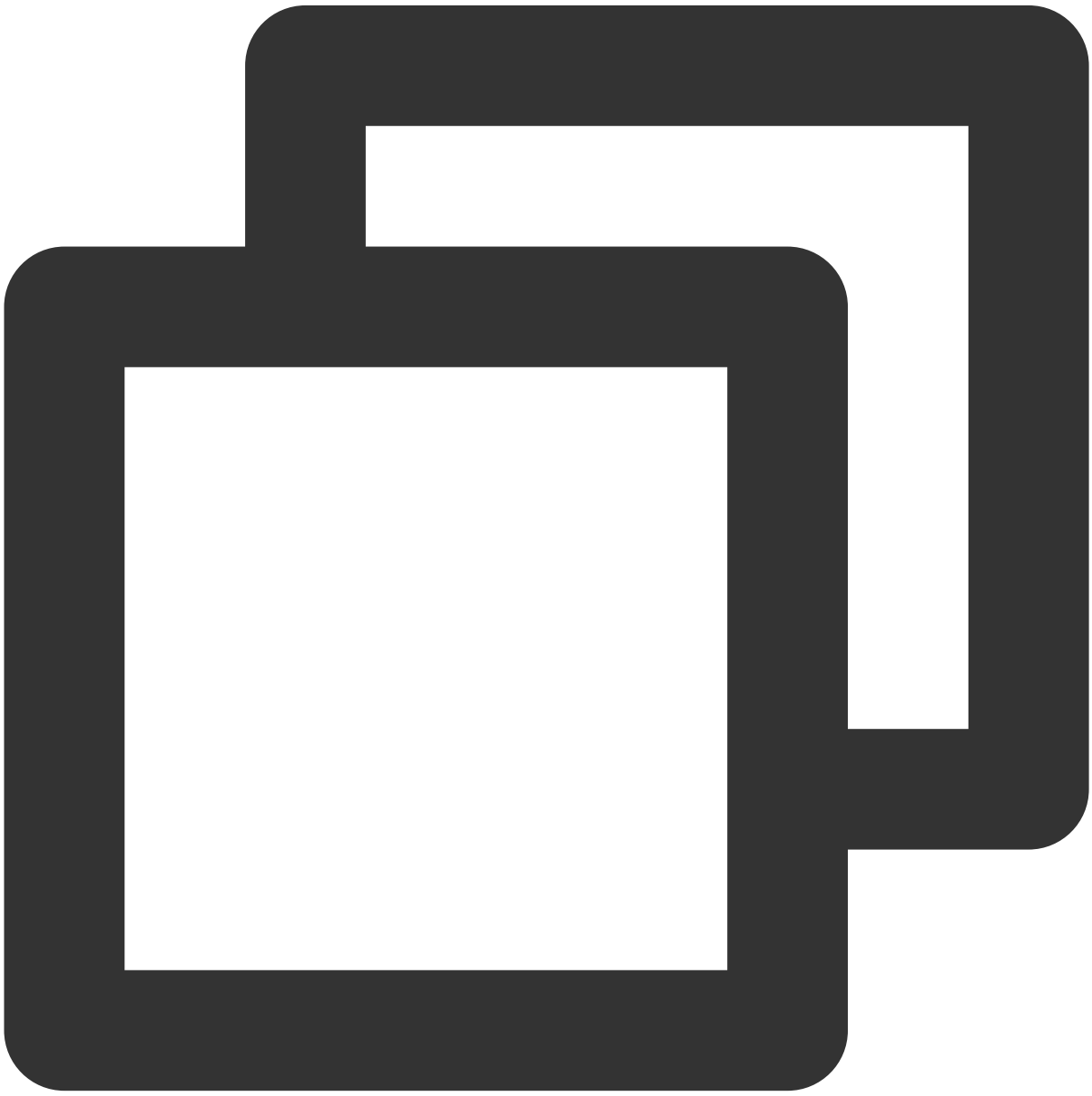
```
typedef void (^TQUICURLSessionTaskDidReceiveResponseBlock)(TQUICURLSession *session
```

参数	描述

session	会话管理类
task	任务管理类
response	响应结果

TQUICURLSessionTaskDownloadProgressBlock

下载进度回调 block

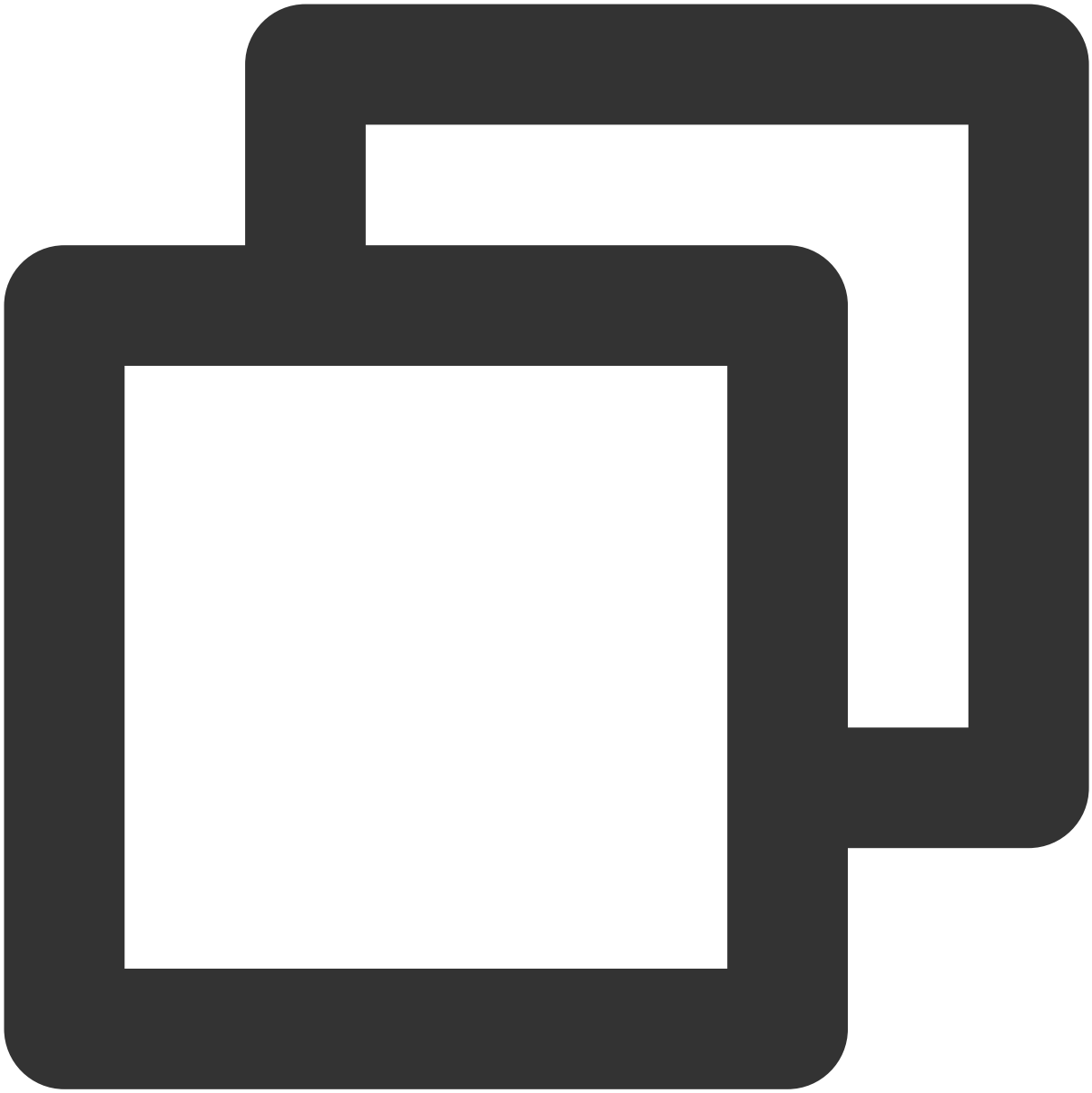


```
typedef void (^TQUICURLSessionTaskDownloadProgressBlock)(NSProgress *downloadProgre
```

参数	描述
downloadProgress	下载进度

TQUICURLSessionTaskUploadProgressBlock

上传进度回调 block

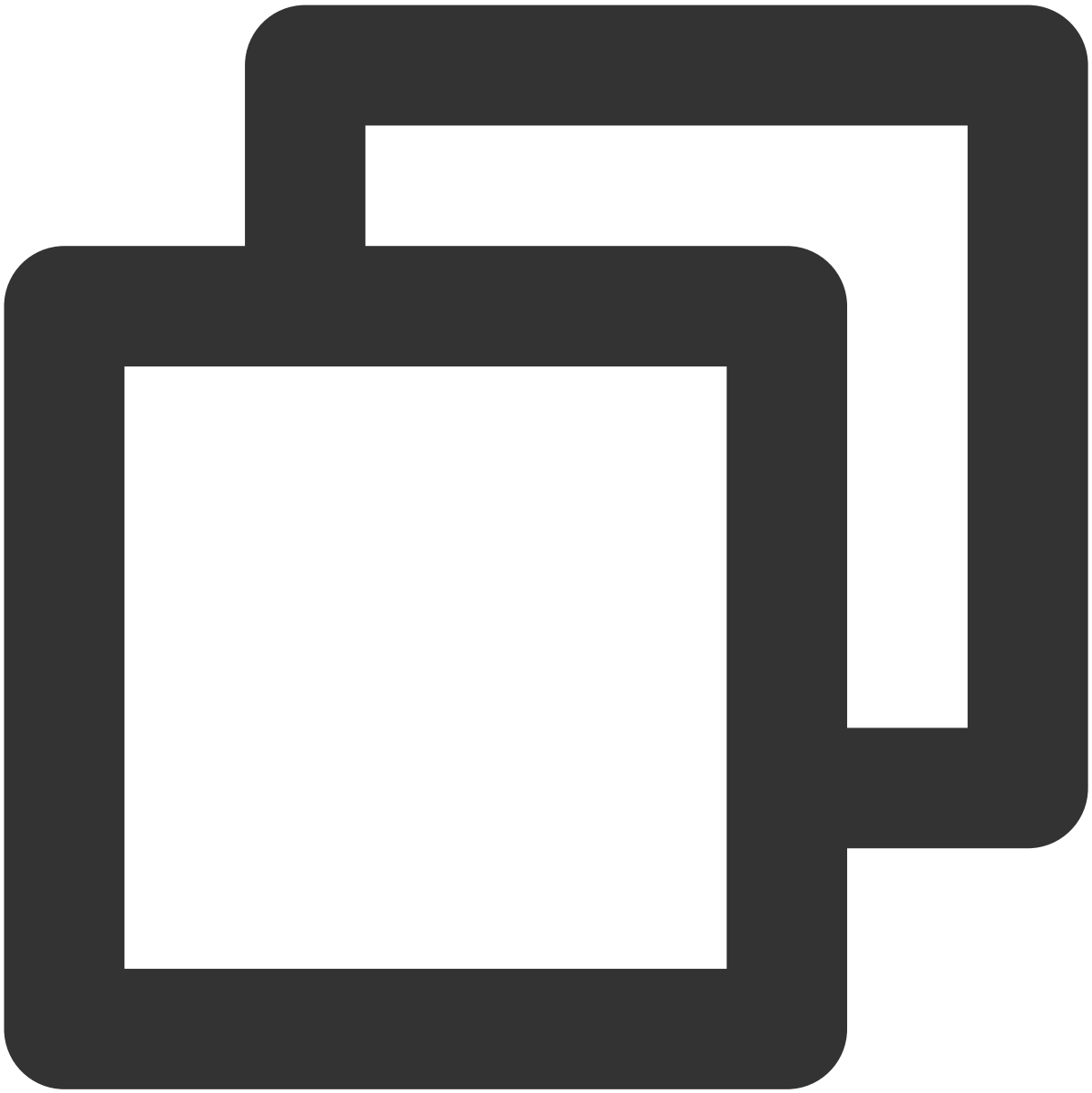


```
typedef void (^TQUICURLSessionTaskUploadProgressBlock)(NSProgress *uploadProgress)
```

参数	描述
uploadProgress	上传进度

QUICURLSessionTaskCompletionHandler

会话执行完成回调 block

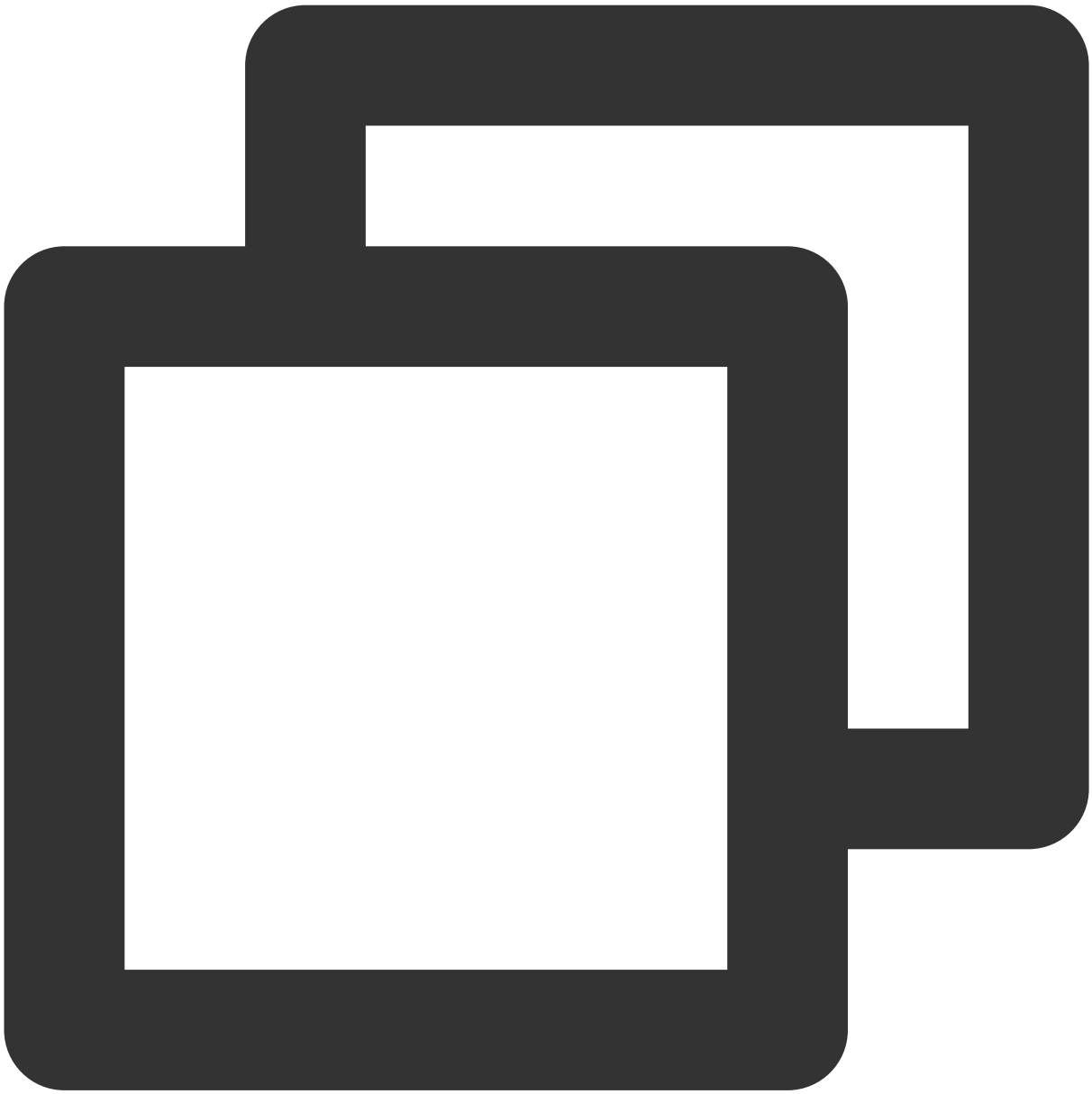


```
typedef void (^TQUICURLSessionTaskCompletionHandler)(NSURLResponse *response, id re
```

参数	描述
response	响应结果，参考系统 NSURLResponse
responseObject	响应数据
error	错误信息

TQUICURLSessionTaskDidFinishCollectingMetricsBlock

会话执行完成数据统计回调 block



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

参数	描述
session	请求会话管理
task	请求任务管理

metrics	网络状态统计

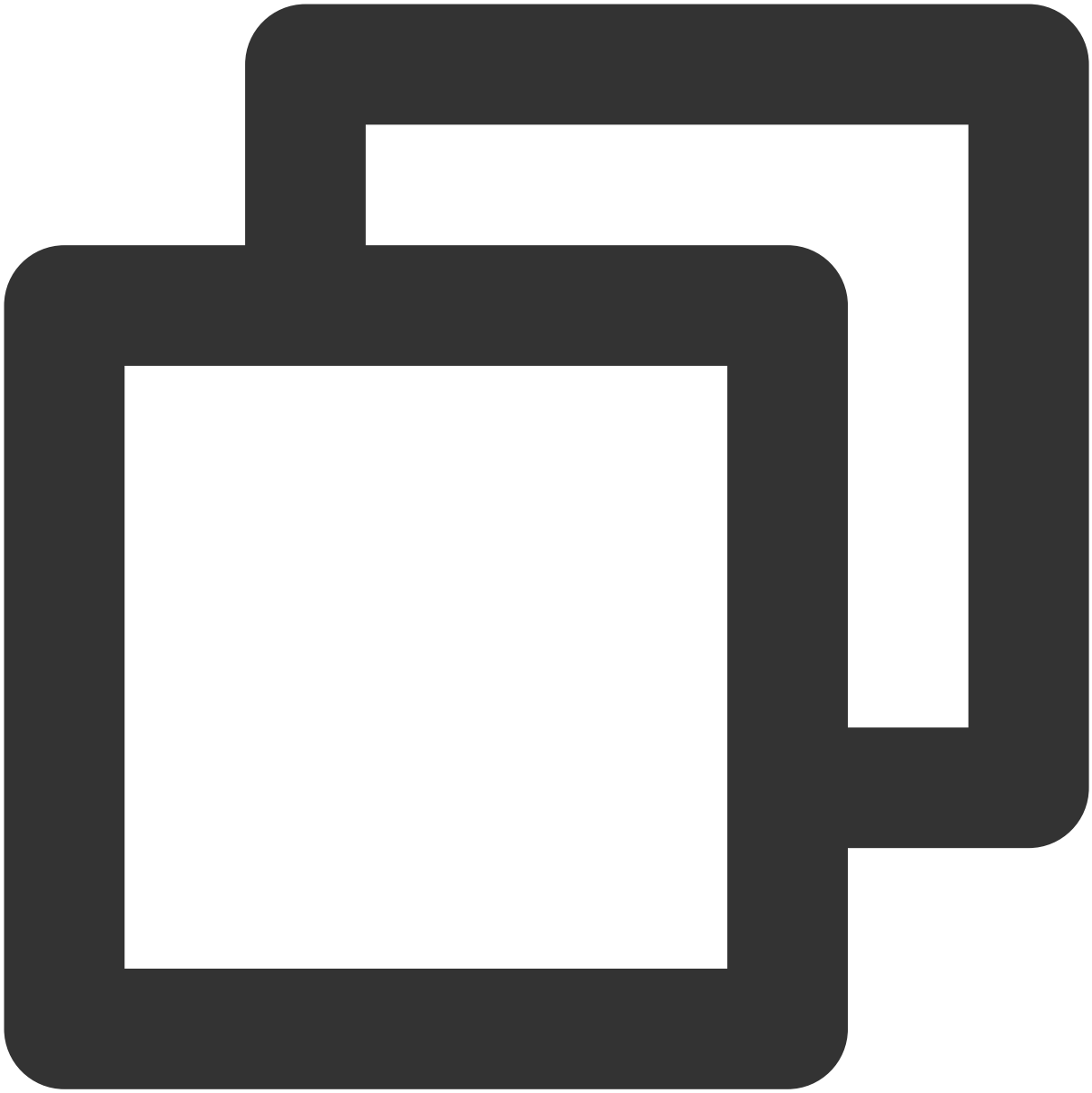
TQUICURLSessionConfiguration

QUIC 请求参数配置

成员变量	描述
quicVersion	设置QUIC版本号 支持的版本：Q043、Q046、Q050、Q051、draft-29、RFC-V1（RFC 9000） 取值： TQUICVersionQ043 (默认值) TQUICVersion046 TQUICVersion050 TQUICVersion051 TQUICVersionDraft29 TQUICVersionRFCV1
congestionType	支持的拥塞算法（CubicBytes、RenoBytes,、BBR、PCC、GCC） 取值： TQUICCongestionTypeBBR（默认值） TQUICCongestionTypeCubicBytes TQUICCongestionTypeRenoBytes TQUICCongestionTypePCC TQUICCongestionTypeGCC
connectTimeoutMillis	连接超时时间，单位：毫秒 默认值：60000
idleTimeoutMillis	空闲连接超时时间，影响连接复用，单位：毫秒 默认值：90000
ipv6Enabled	是否支持IPv6，可选 YES 或者 NO 默认值：NO
dnsParser	自定义 DNS 解析代理，参考：TQUICDNSParserDelegate

TQUICDNSParserDelegate

DNS 解析代理，可实现自定义 DNS 解析



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

参数	描述
hostName	域名，在请求过程中会回调域名，需要返回解析到的 IP 地址

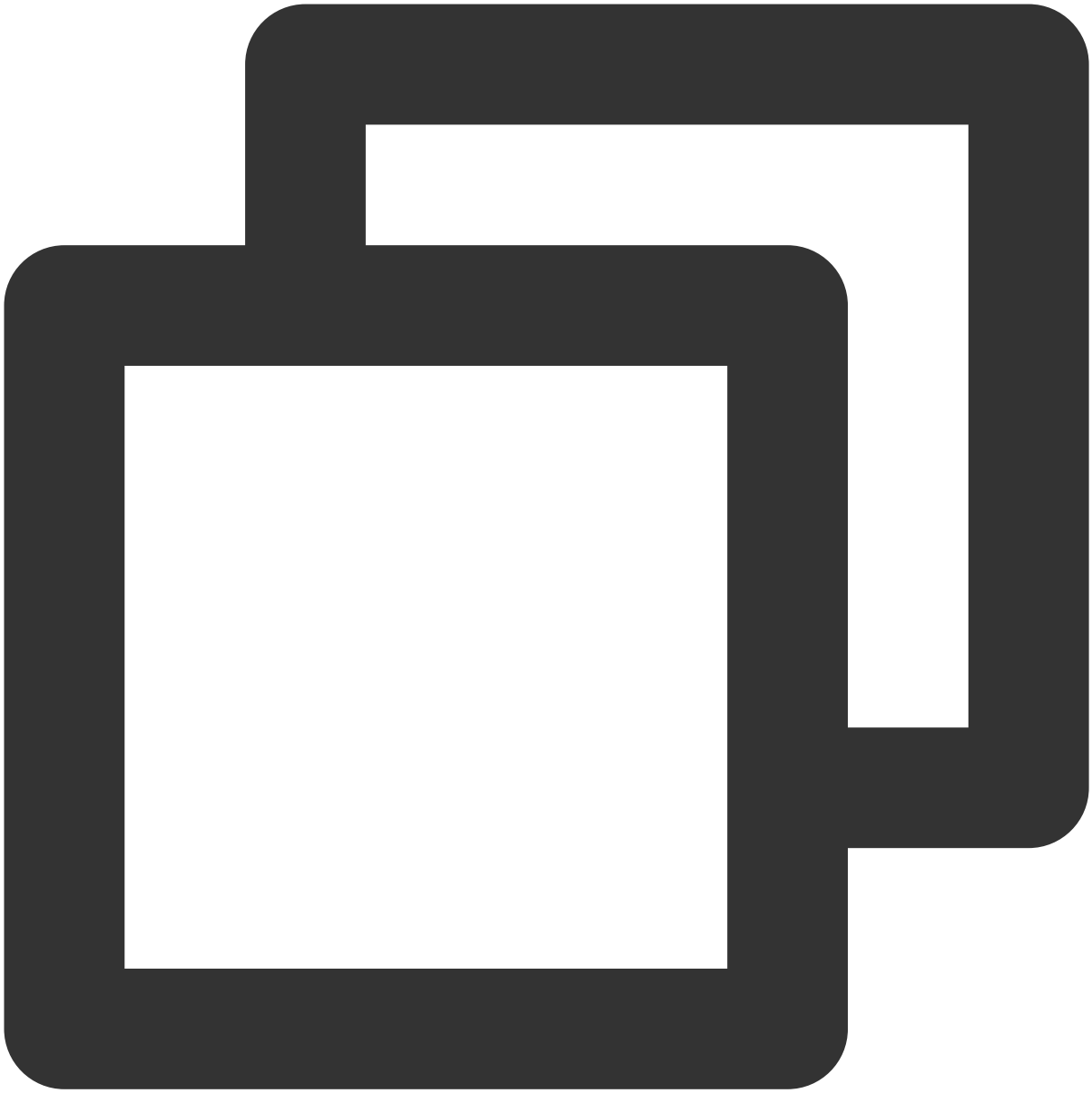
TQUICURLSession

请求任务管理接口

API	描述
sessionWithConfiguration	根据配置创建实例
sharedSession	根据默认配置创建实例，单例
dataTaskWithRequest	根据request参数创建请求任务

sessionWithConfiguration

根据配置创建实例

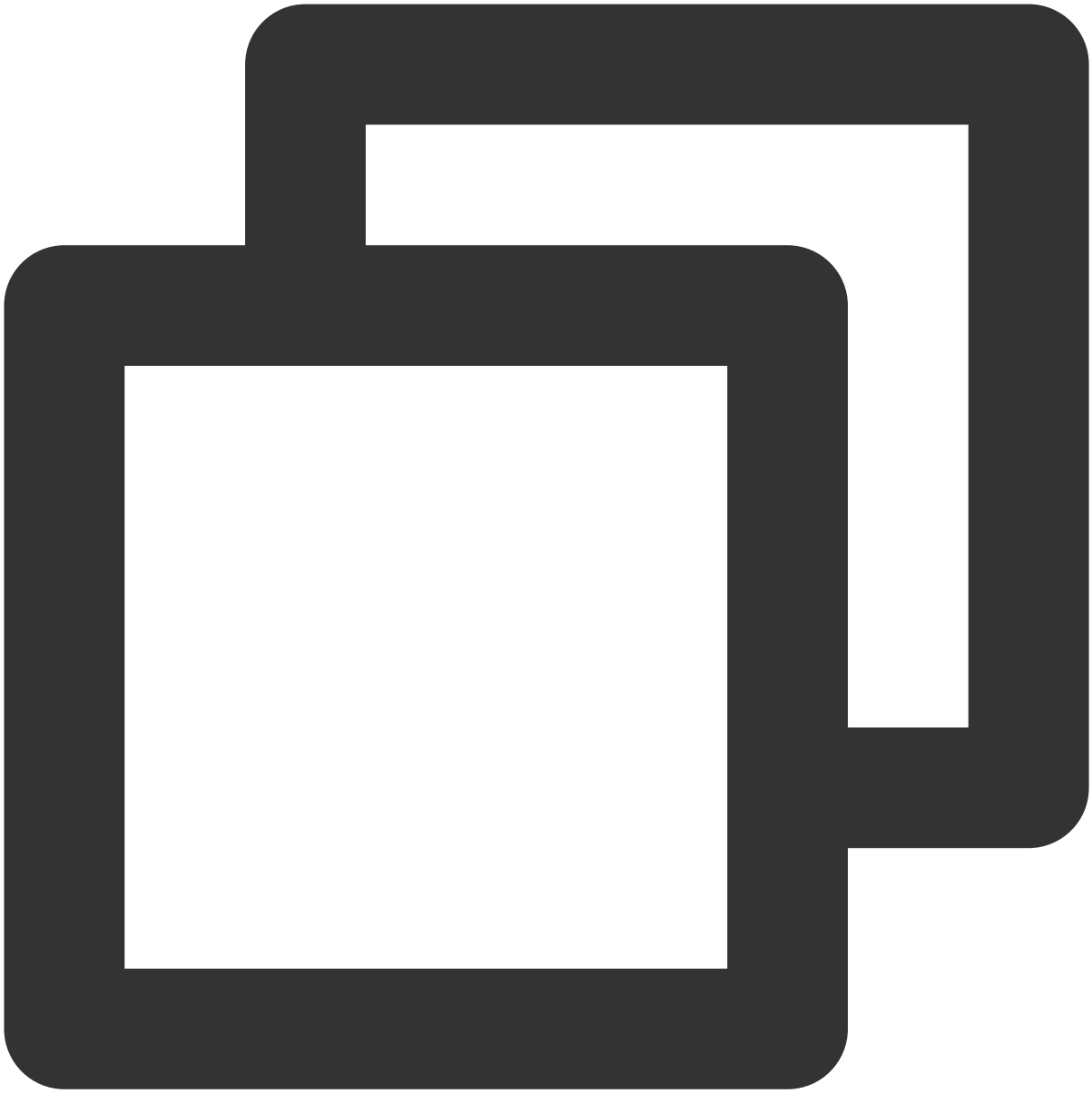


```
+ (instancetype)sessionWithConfiguration:(TQUICURLSessionConfiguration *)configurat
```

参数	描述
configuration	配置信息

sharedSession

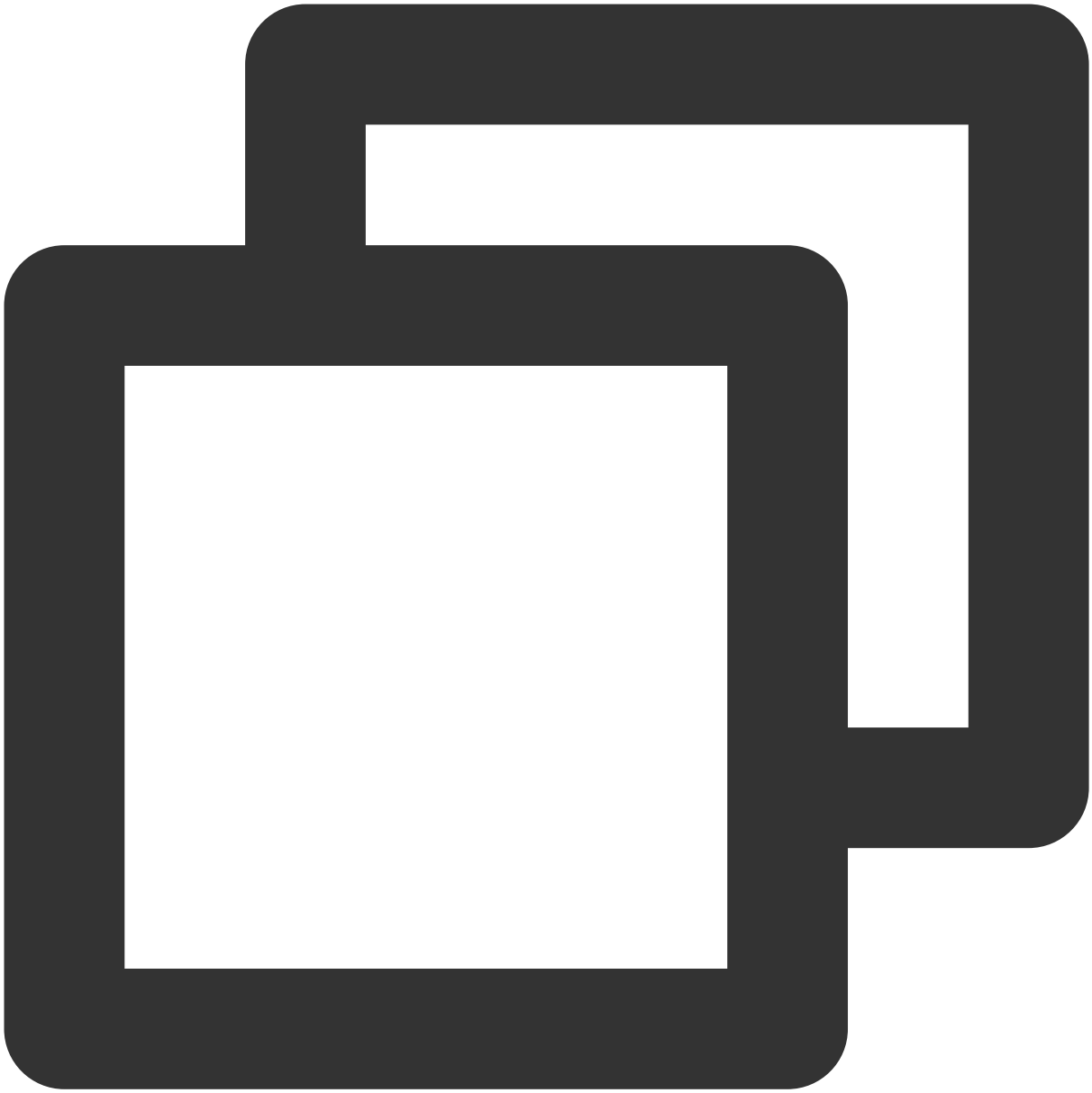
根据默认配置创建实例，单例



```
+ (instancetype)sharedSession
```

dataTaskWithRequest

根据 request 参数创建请求任务



```
- (nullable TQUICURLSessionDataTask *)dataTaskWithRequest:(NSURLRequest *)request
```

参数	描述
request	请求参数，参考系统 NSURLRequest

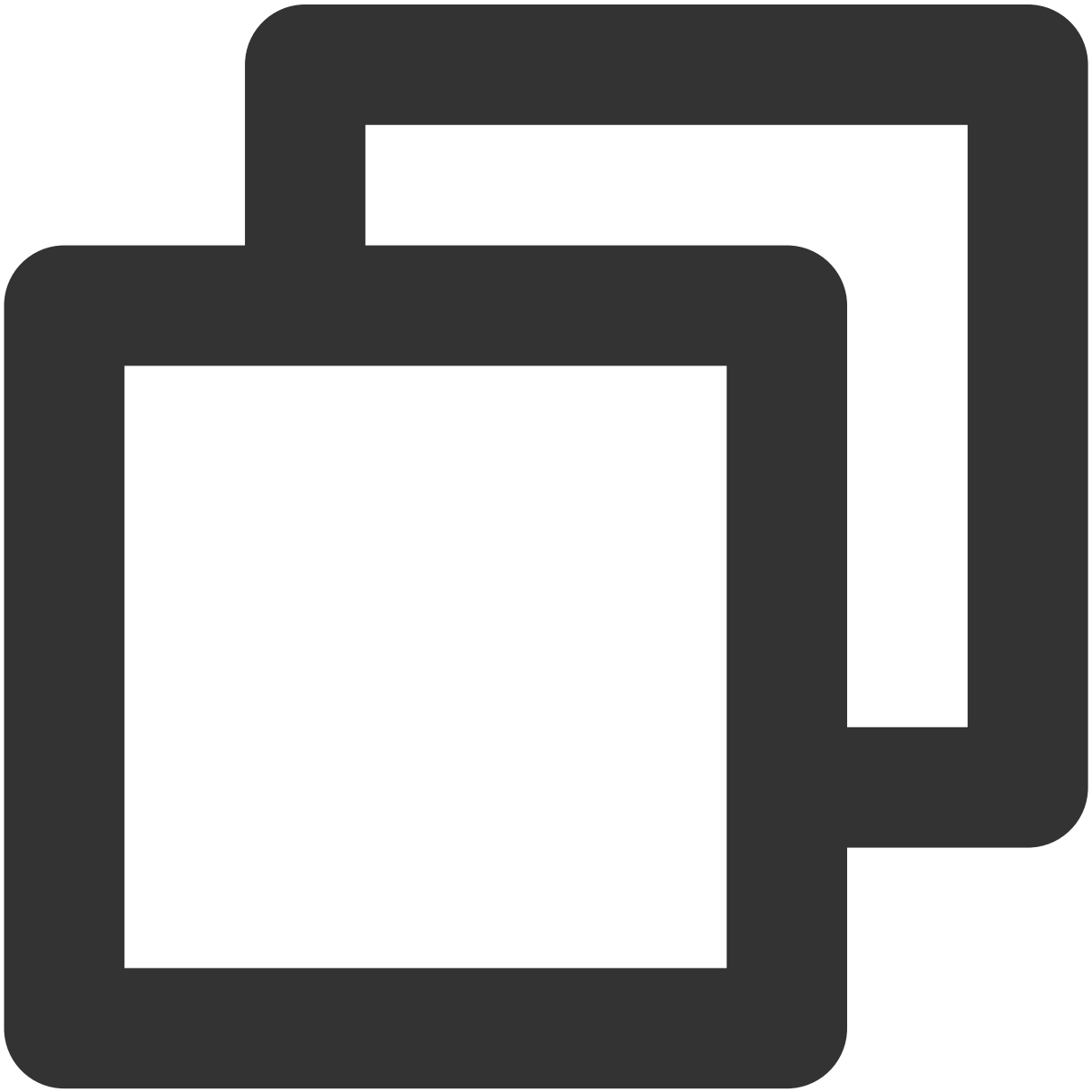
TQUICURLSessionTask

求任务管理接口

API	描述
resume	启动请求
cancel	取消请求

resume

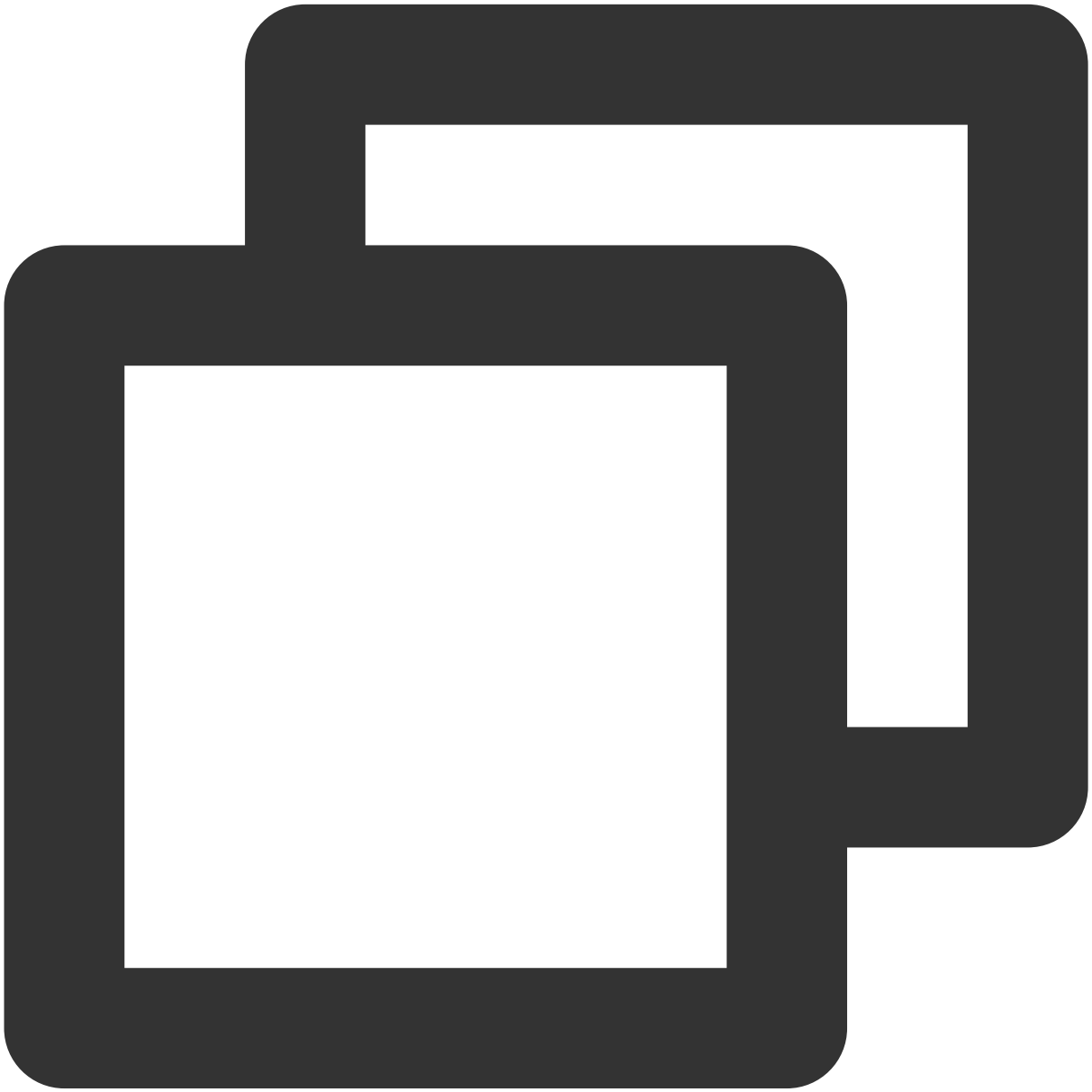
开始请求



```
- (void)resume
```

cancel

取消请求



```
- (void)cancel
```

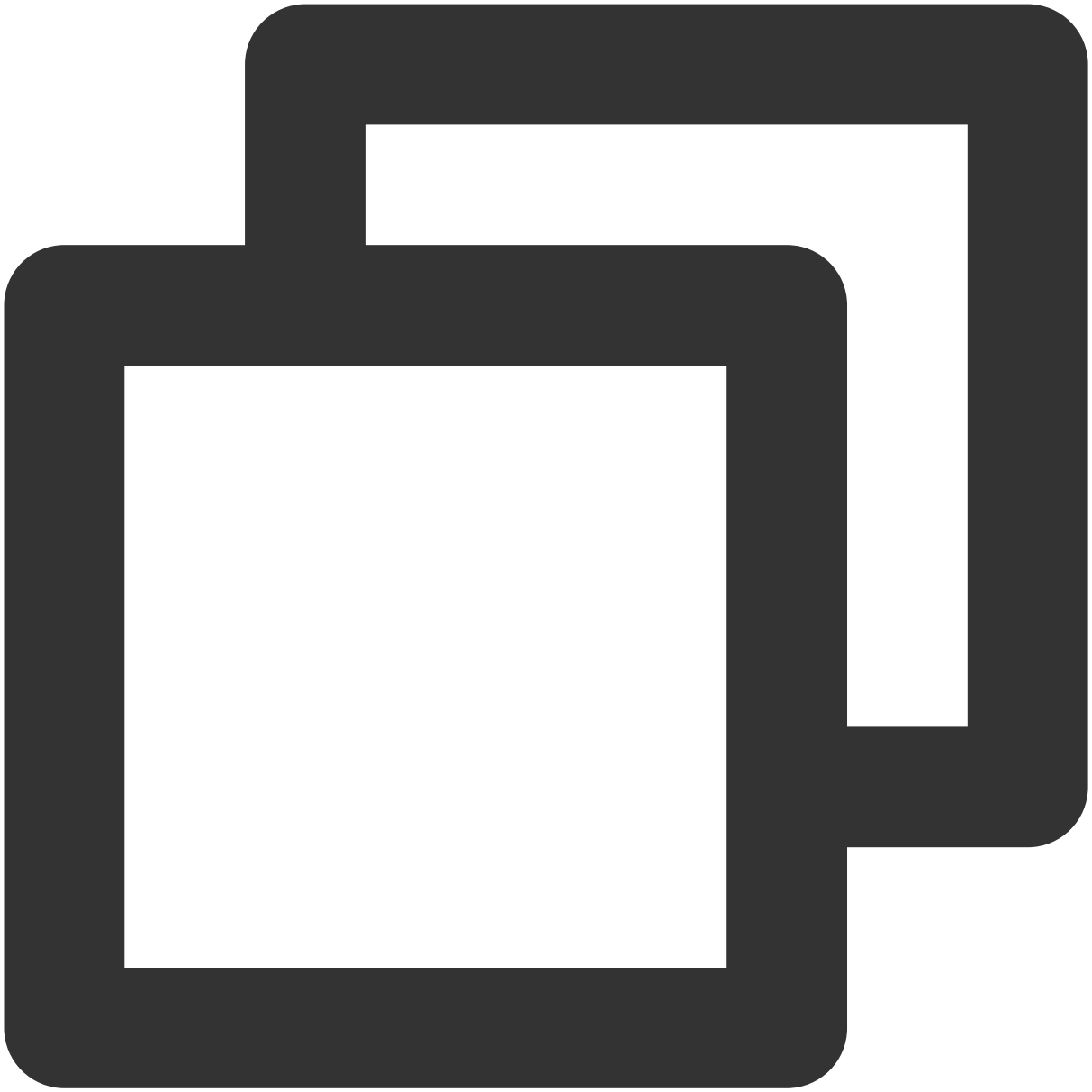
TQUICURLSessionDataTask

请求任务管理接口，扩展 TQUICURLSessionTask 协议

API	描述
resume	启动请求
cancel	取消请求

resume

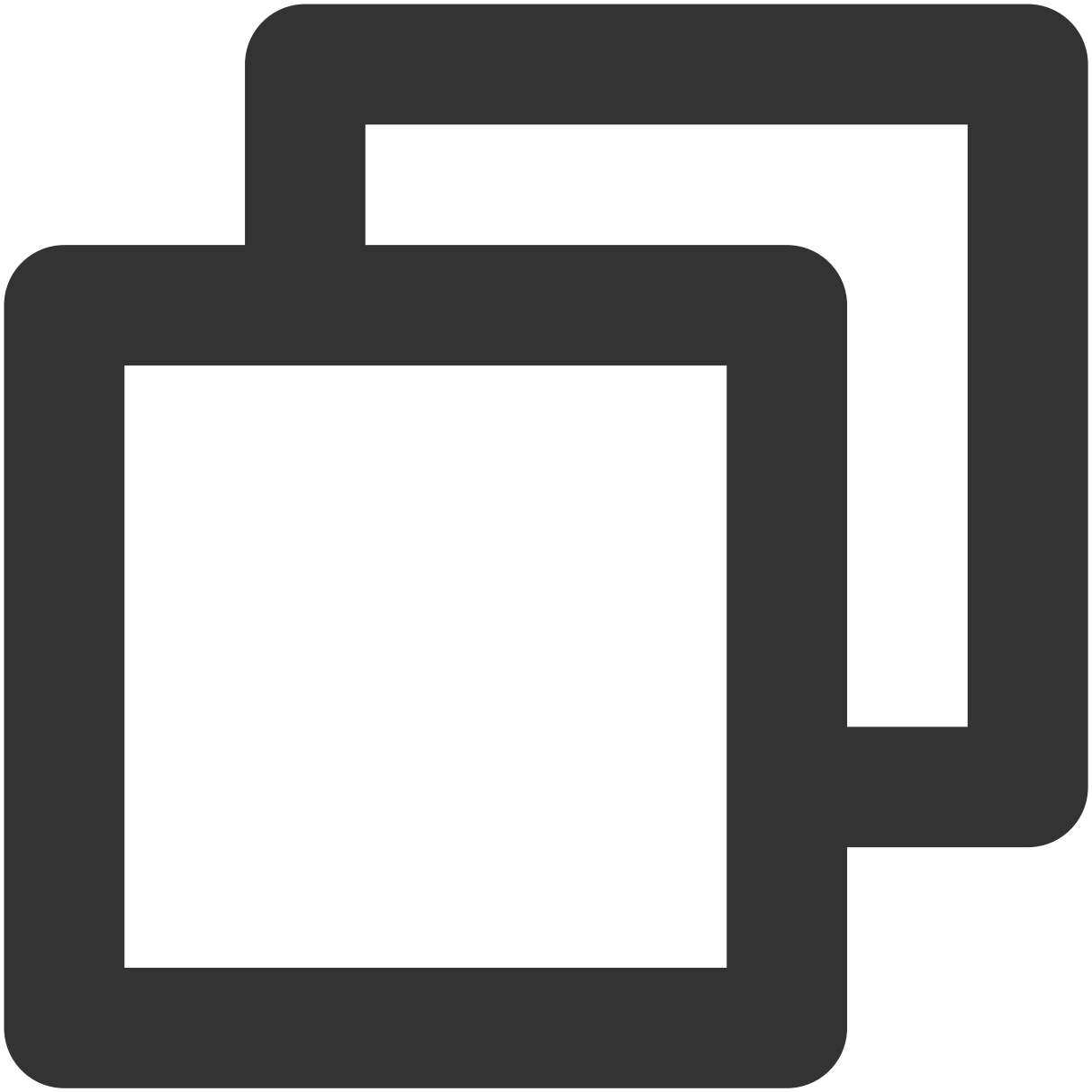
开始请求



```
- (void)resume
```

cancel

取消请求



```
- (void)cancel
```

TQUICURLRequestSerialization

API	描述
TQUICHTTPRequestSerializer	http 请求参数序列化

TQUICJSONRequestSerializer	JSON 请求参数序列化

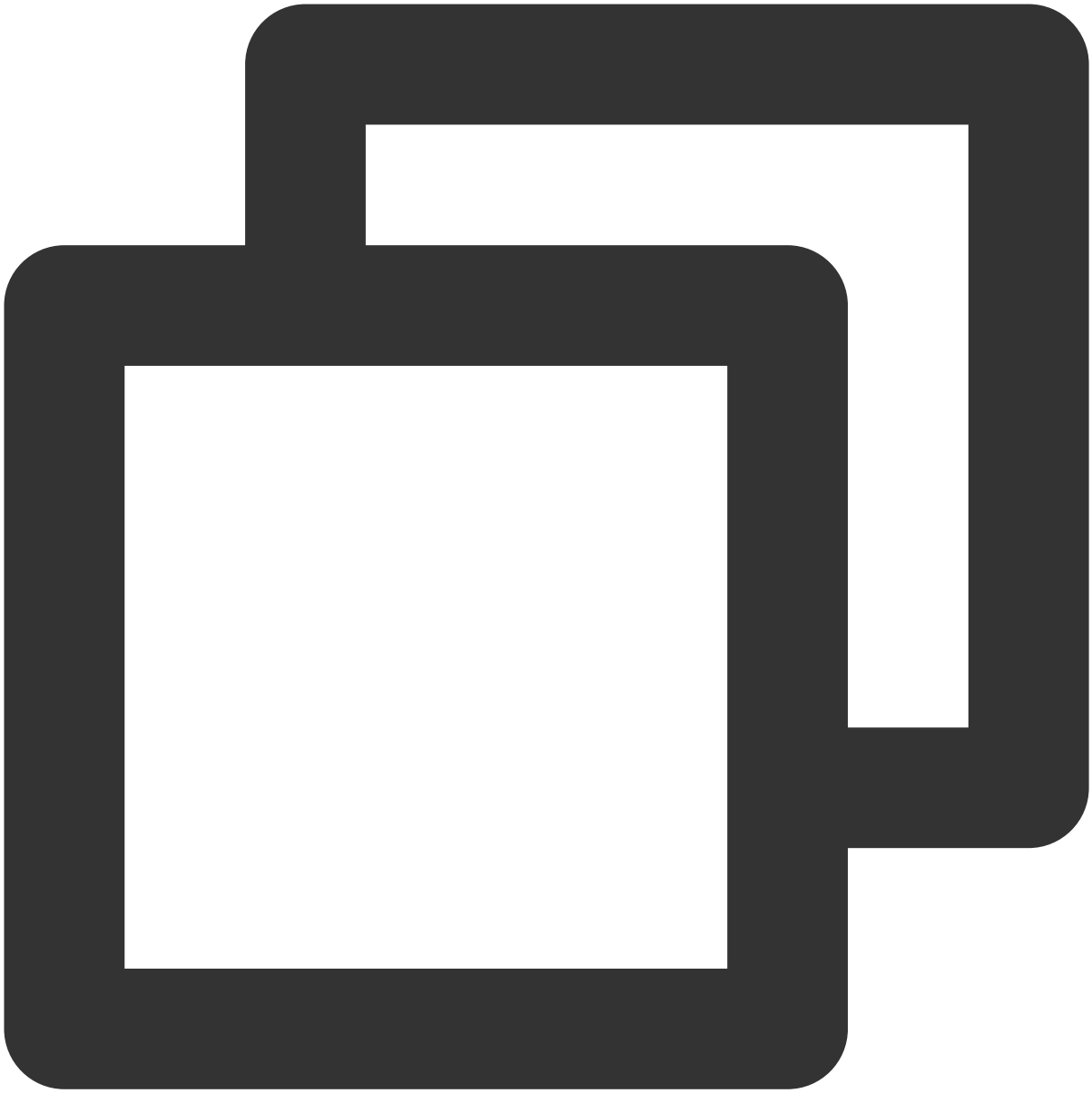
TQUICHTTPRequestSerializer

请求数据序列化接口

API	描述
serializer	实例化
setValue	设置 Header 值
valueForHTTPHeaderField	根据 Header 字段返回对应的值
requestWithMethod	创建 NSMutableURLRequest
multipartFormRequestWithMethod	创建 NSMutableURLRequest，用于传输流式数据

serializer

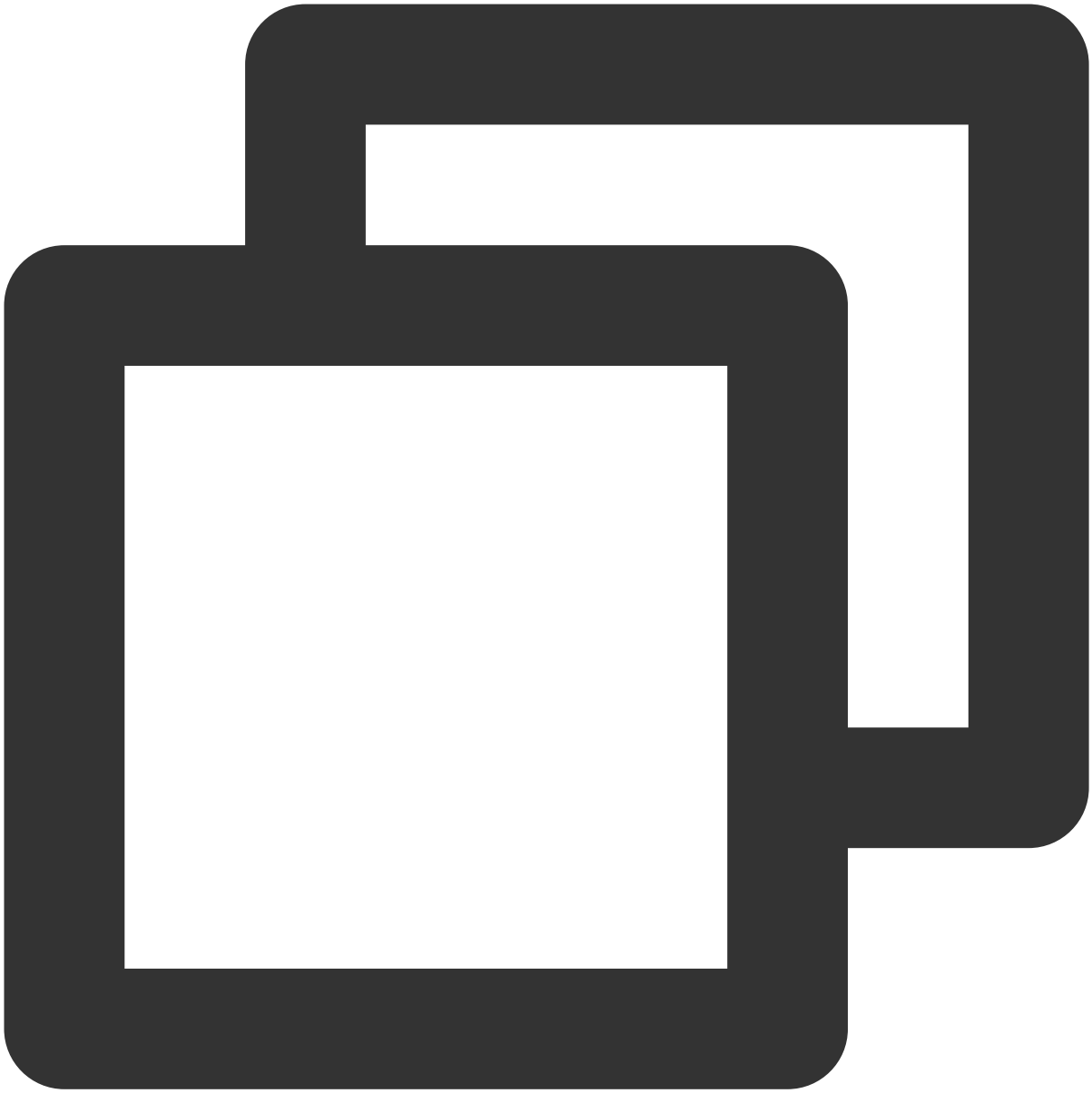
构造 TQUICHTTPRequestSerializer 实例



```
+ (instancetype)serializer
```

setValue

设置 Header 值

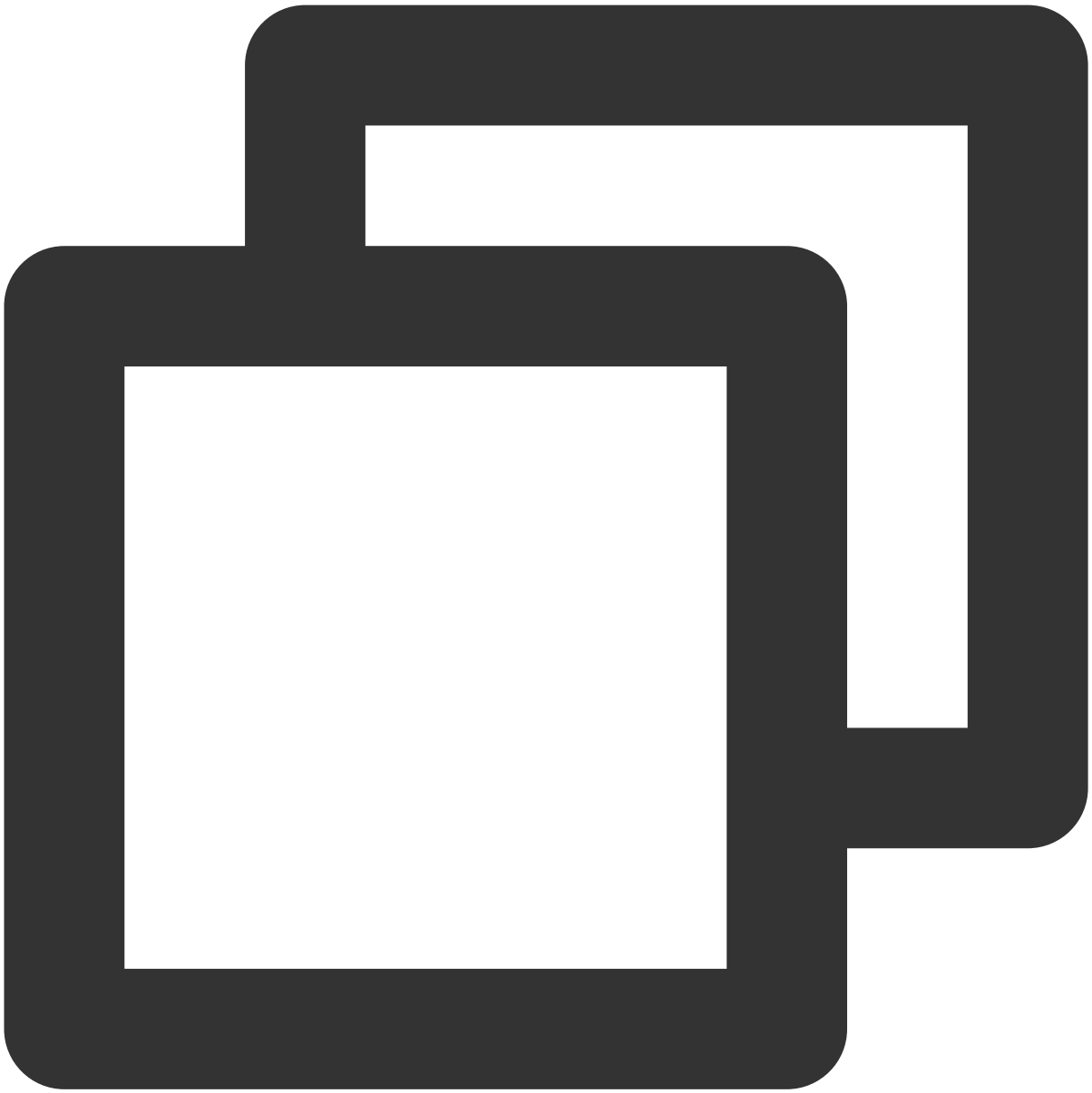


```
- (void)setValue:(nullable NSString *)value forHTTPHeaderField:(NSString *)field
```

参数	描述
value	Header 值
field	Header 字段

valueForHTTPHeaderField

根据 Header 字段返回对应的值

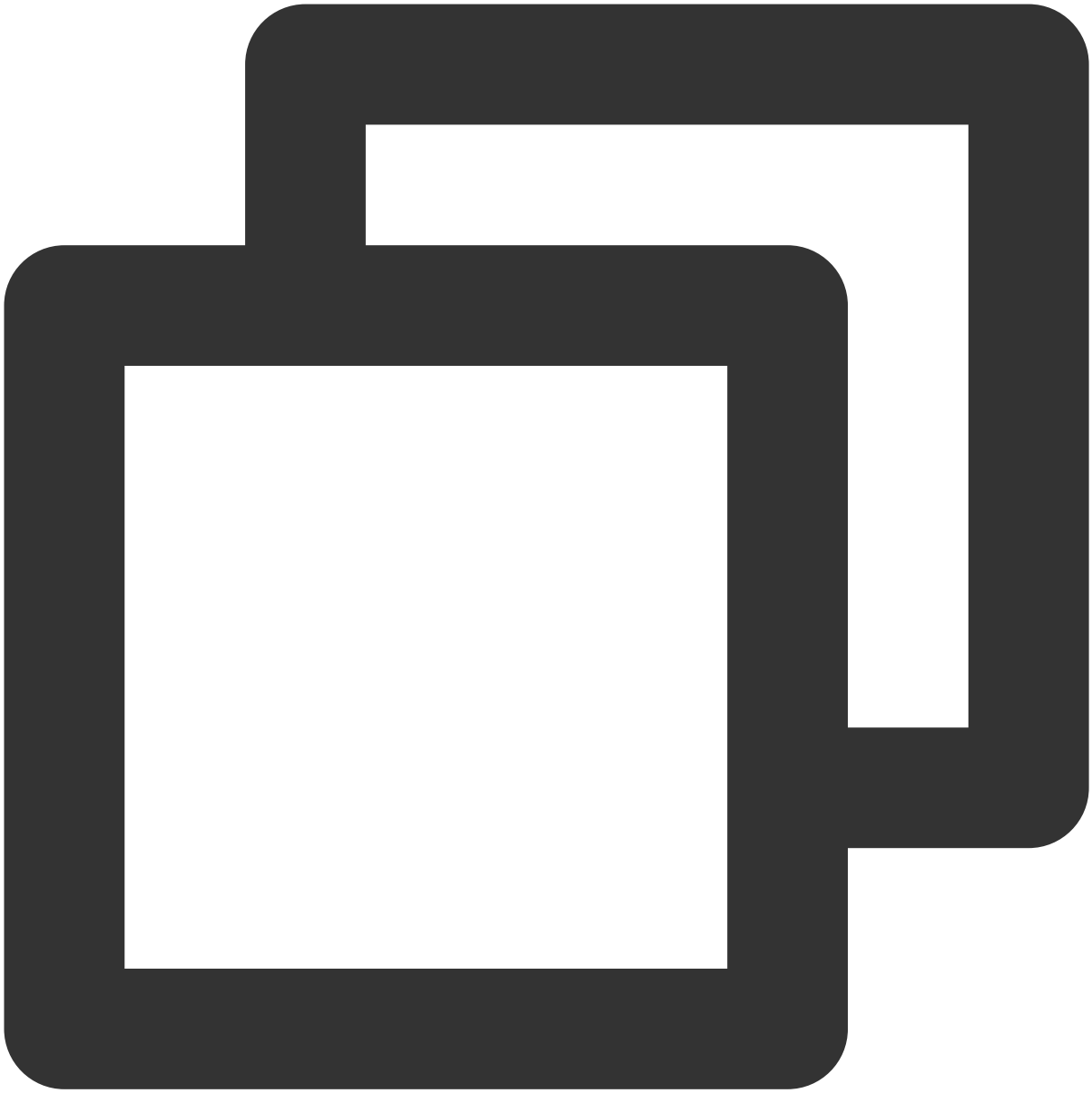


```
- (nullable NSString *)valueForHTTPHeaderField:(NSString *)field
```

参数	描述
field	Header 字段

requestWithMethod

创建 NSMutableURLRequest

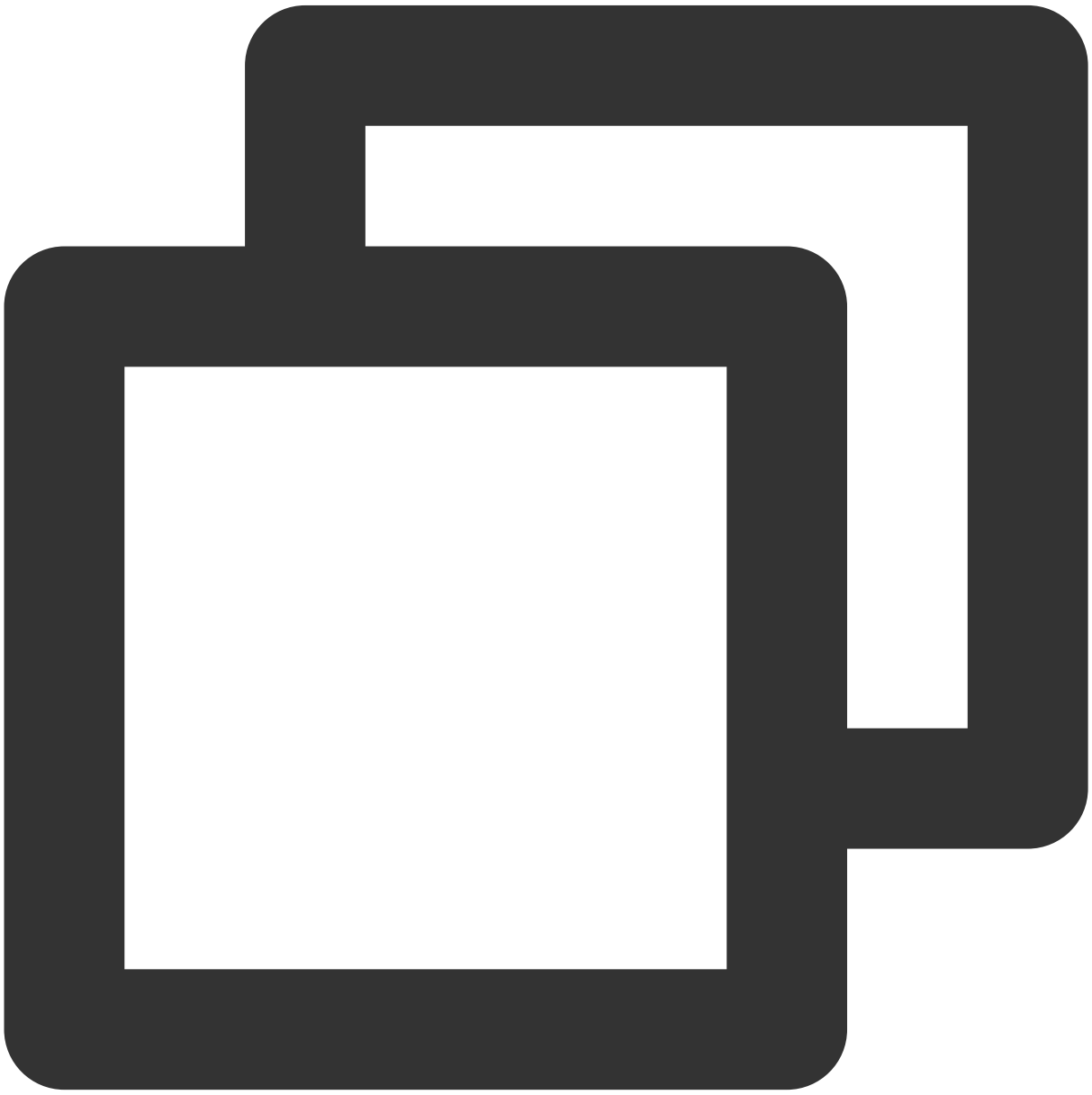


```
- (nullable NSMutableURLRequest *)requestWithMethod:(NSString *)method
                                          URLString:(NSString *)URLString
                                         parameters:(nullable id)parameters
                                              error:(NSError * _Nullable __autorele
```

参数	描述

method	设置请求方法，get、post、put、delete、head、patch 等 http 标准支持的方法
URLString	请求 Url
parameters	请求参数
error	错误信息，NSError

multipartFormRequestWithMethod



```
- (NSMutableURLRequest *)multipartFormRequestWithMethod:(NSString *)method
                                              URLString:(NSString *)URLString
                                             parameters:(nullable NSDictionary <NSS
                                 constructingBodyWithBlock:(nullable void (^)(id <TQUI
                                                  error:(NSError * _Nullable __auto
```

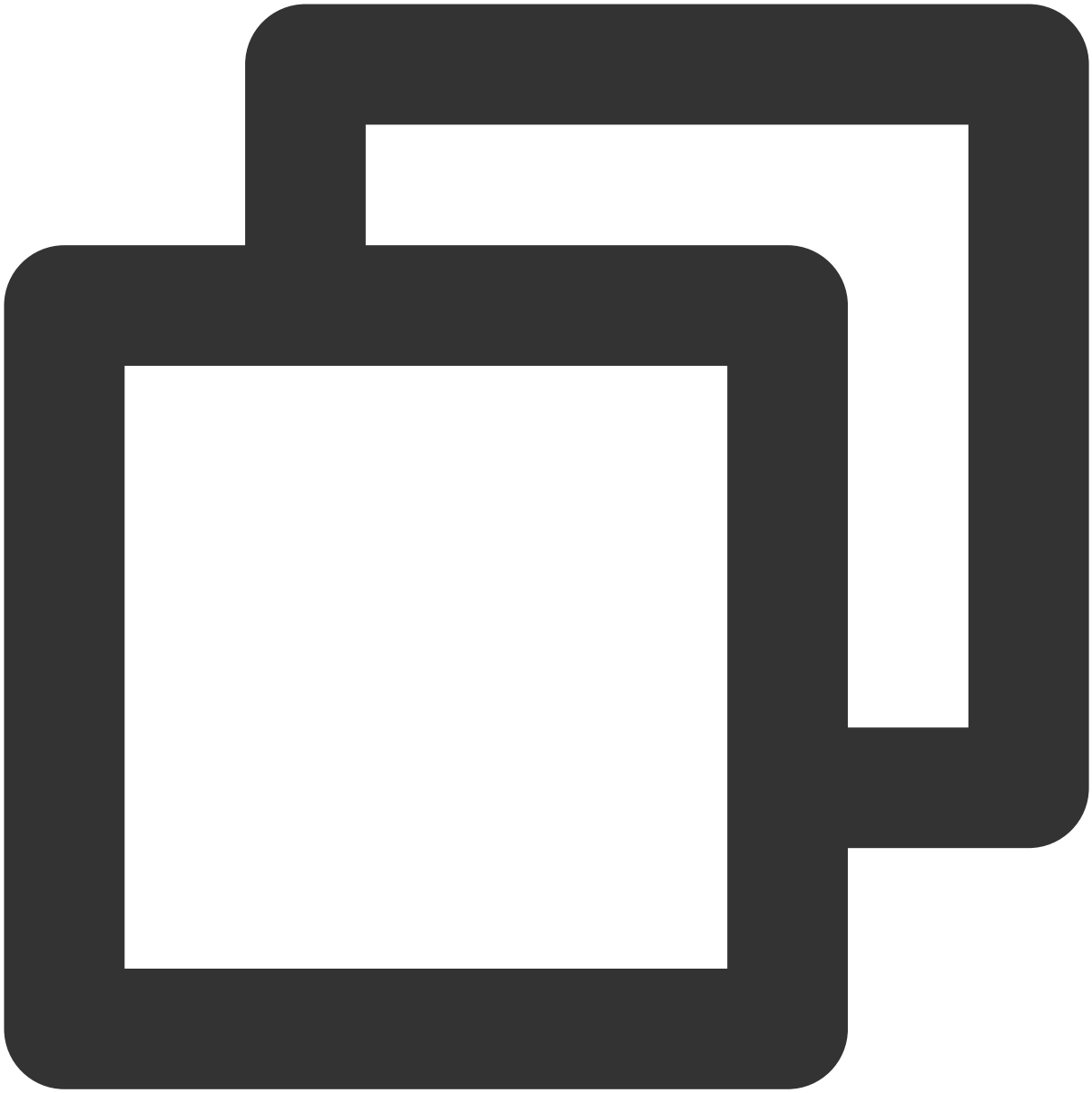
参数	描述
method	设置请求方法，get、post、put、delete、head、patch 等 http 标准支持的方法
URLString	请求 Url
parameters	请求参数
constructingBodyWithBlock	使用 Block 方式构建 body 内容
error	错误信息，NSError

TQUICJSONRequestSerializer

API	描述
serializerWithWritingOptions	创建实例

serializerWithWritingOptions

根据 JSON 序列化选项创建 TQUICJSONRequestSerializer 实例



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

参数	描述
writingOptions	根据 JSON 序列化选项创建实例，参考系统 NSJSONWritingOptions

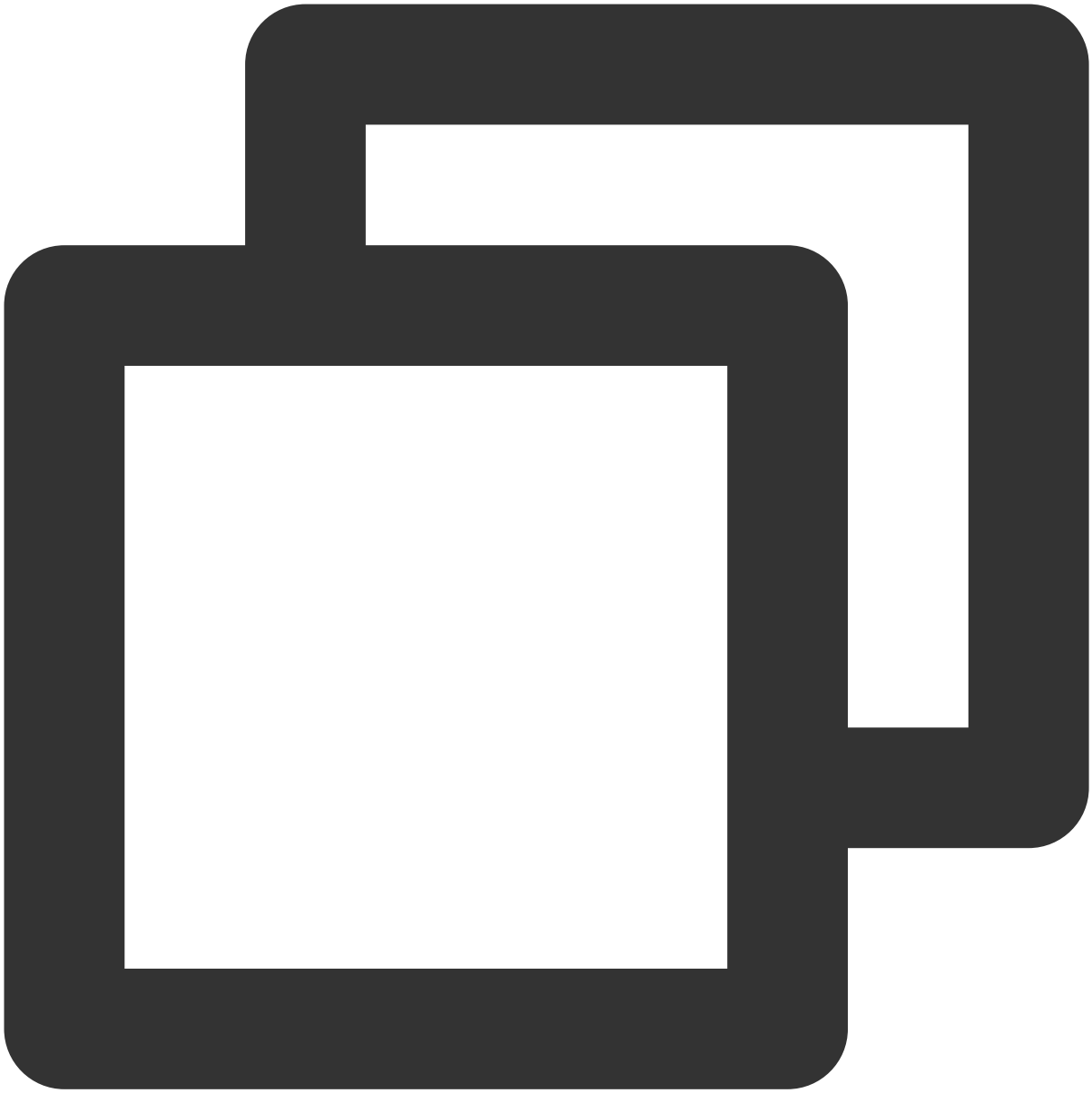
TQUICMultipartFormData

多分块表单数据上传

API	描述
appendPartWithFileURL	通过配置文件路径上传表单数据
appendPartWithInputStream	通过输入流上传表单数据
appendPartWithFileData	分块上传表单数据
appendPartWithFormData	添加数据分块
appendPartWithHeaders	把 Headers 和数据内容添加到表单
throttleBandwidthWithPacketSize	上传带宽限制

appendPartWithFileURL

通过指定文件路径上传表单数据

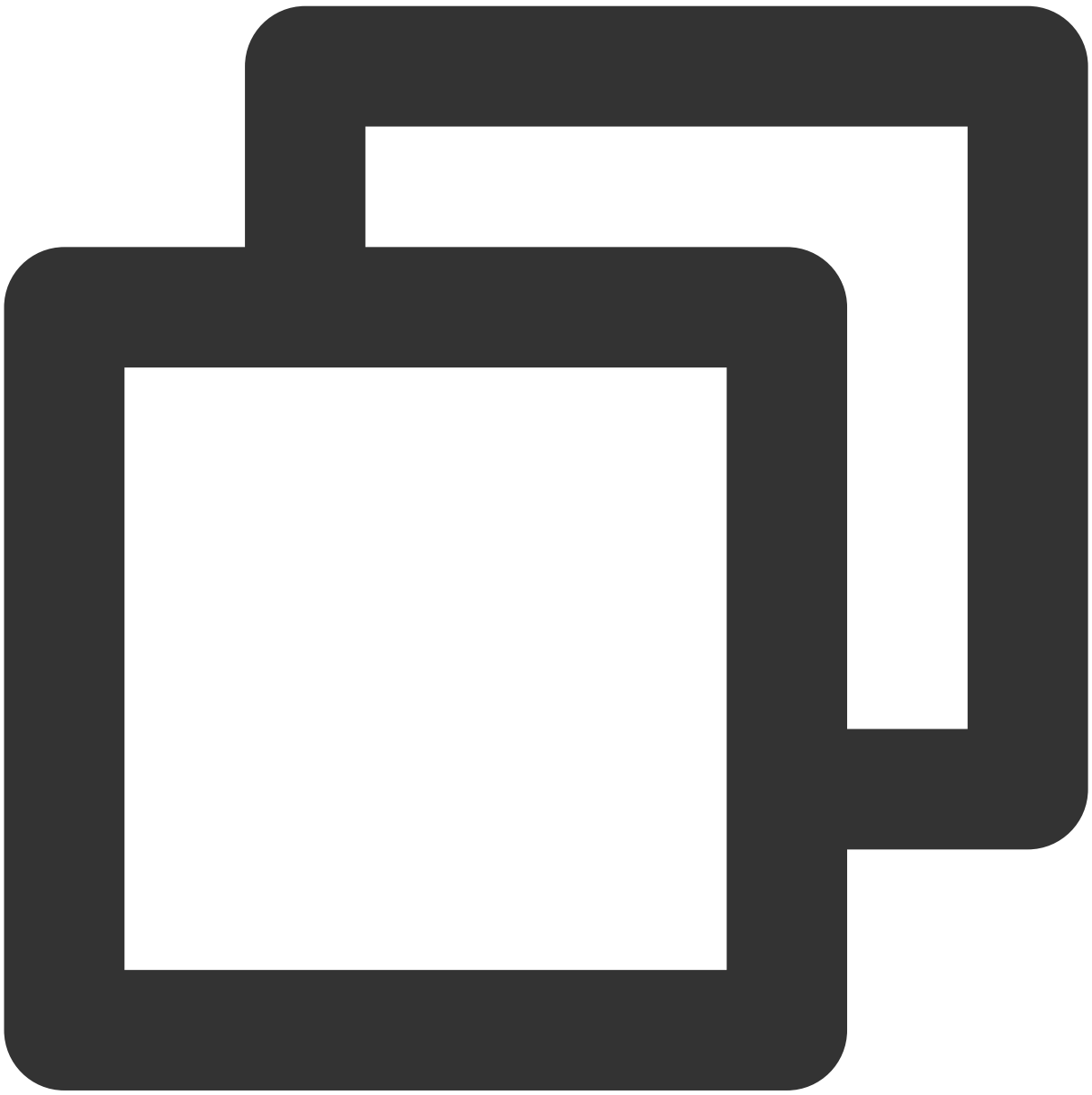


```
- (BOOL)appendPartWithFileURL:(NSURL *)fileURL
                         name:(NSString *)name
                        error:(NSError * _Nullable __autoreleasing *)error
```

参数	描述
fileURL	添加到表单的文件路径
name	关联文件名称
error	错误信息返回

appendPartWithInputStream

通过指定输入流上传数据

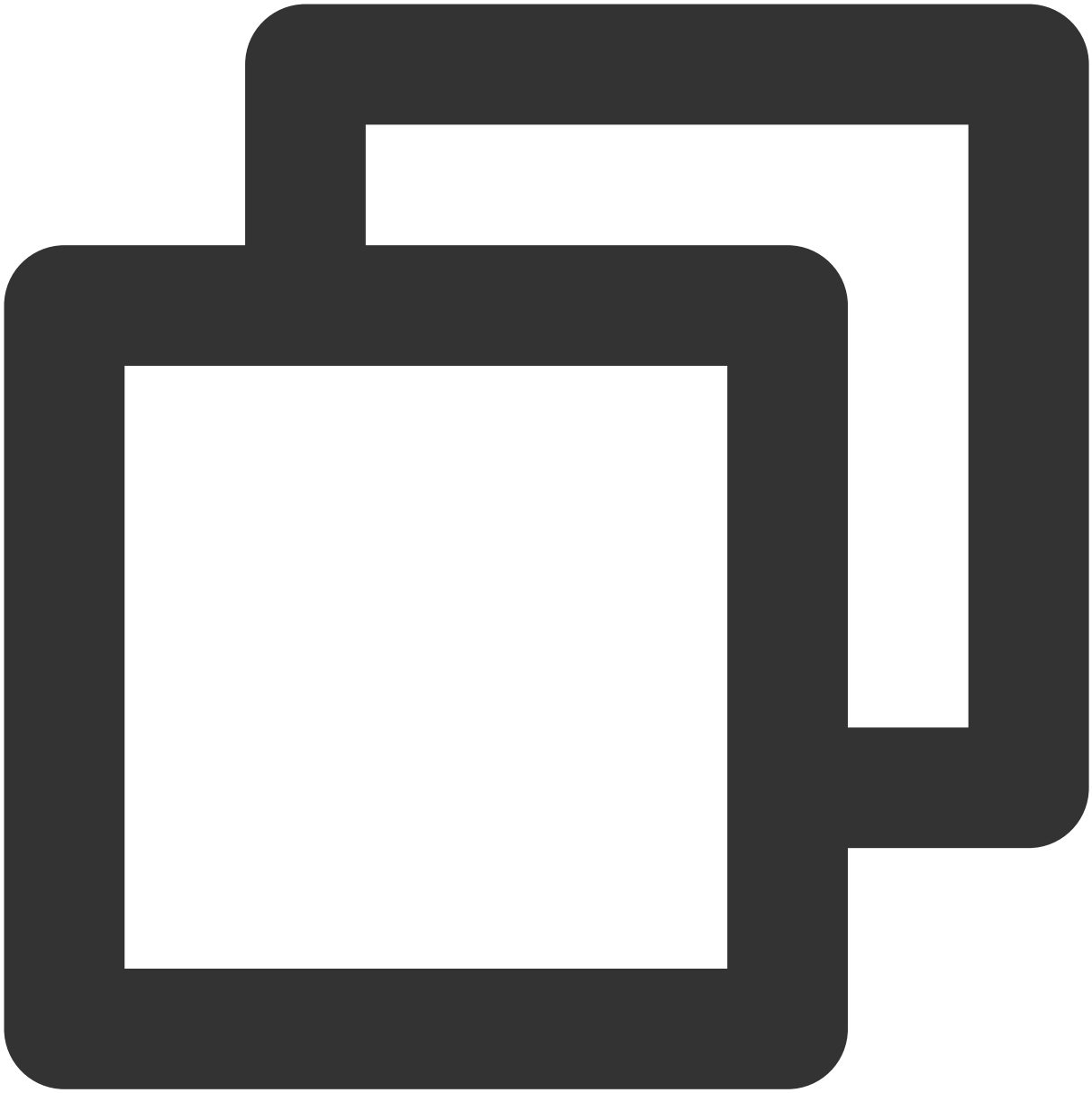


```
- (void)appendPartWithInputStream:(nullable NSInputStream *)inputStream
                             name:(NSString *)name
                         fileName:(NSString *)fileName
                           length:(int64_t)length
                         mimeType:(NSString *)mimeType
```

参数	描述
inputStream	输入流
name	关联输入流的名称
fileName	关联输入流文件名
length	流数据长度(单位 bytes)
mimeType	MIME Type（例如：image/jpeg，详细参考 HTTP 协议标准）

appendPartWithFileData

分块上传表单数据



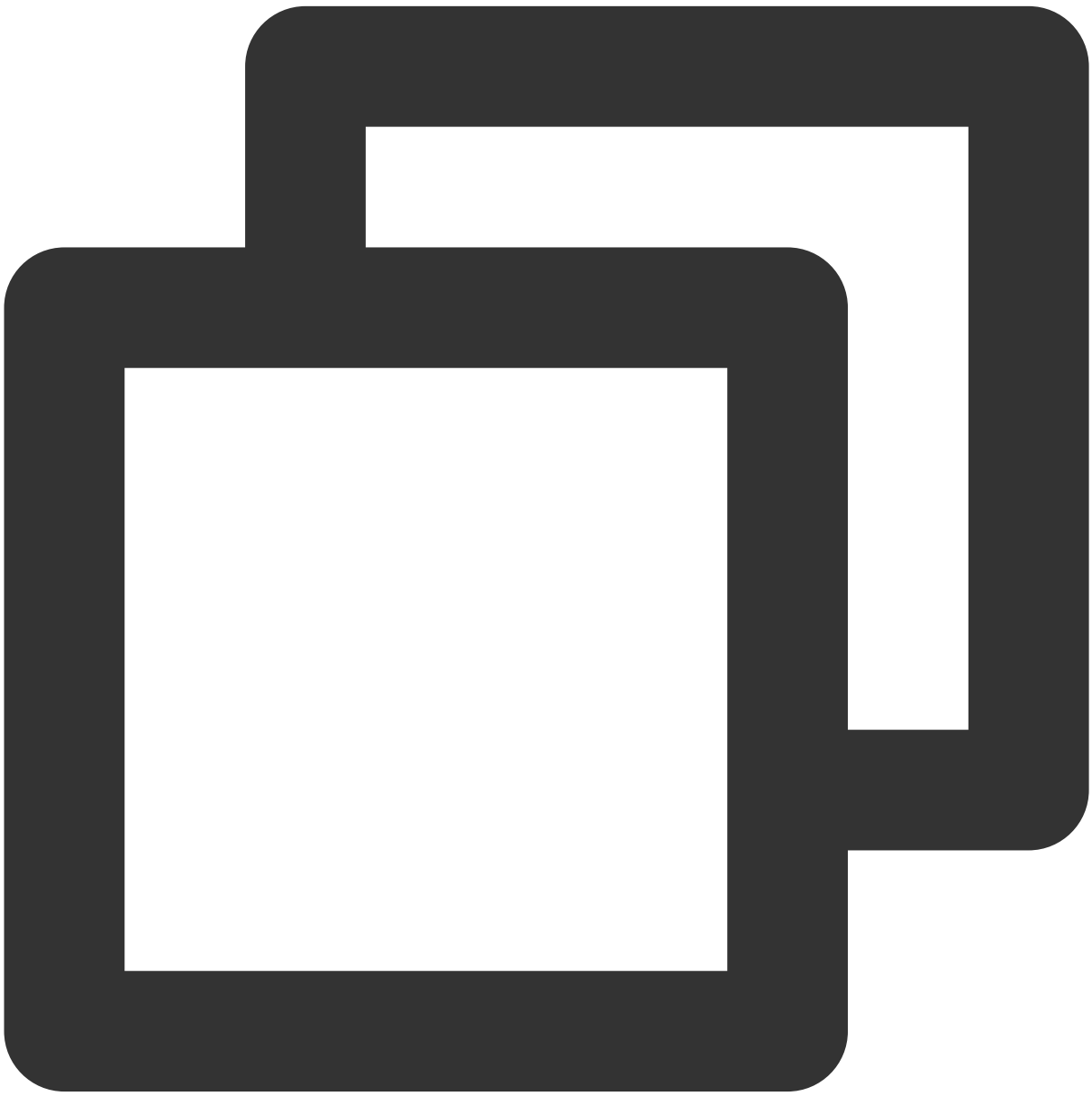
```
- (void)appendPartWithFileData:(NSData *)data
                          name:(NSString *)name
                      fileName:(NSString *)fileName
                      mimeType:(NSString *)mimeType
```

参数	描述
data	添加到表单的数据
name	关联数据的名称

fileName	关联数据文件名
mimeType	MIME Type（例如：image/jpeg，详细参考 HTTP 协议标准）

appendPartWithFormData

添加数据块

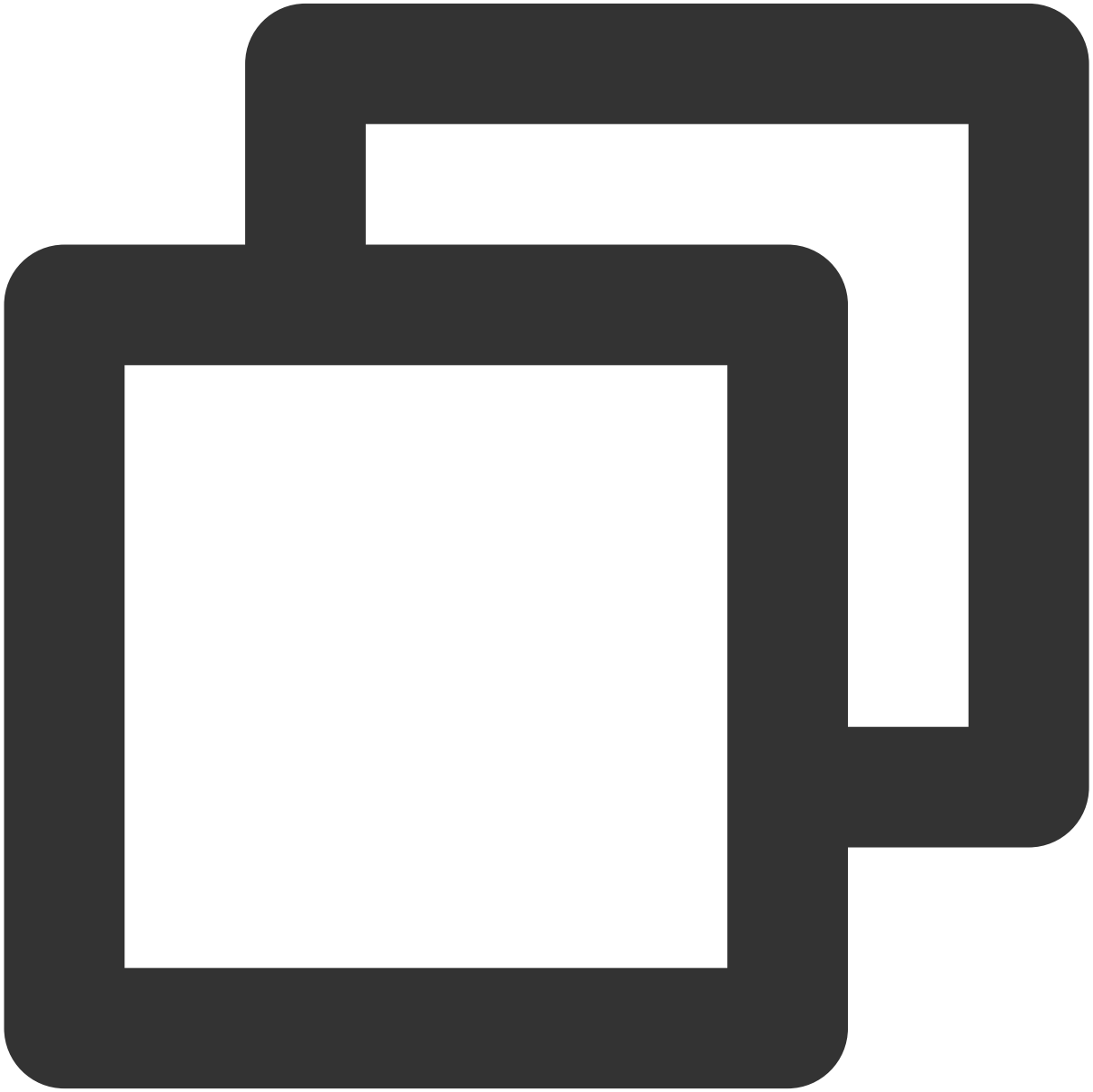


```
- (void)appendPartWithFormData:(NSData *)data
                          name:(NSString *)name
```


参数	描述
data	添加到表单的数据
name	关联数据名称

appendPartWithHeaders

把 Headers 和数据内容添加到表单



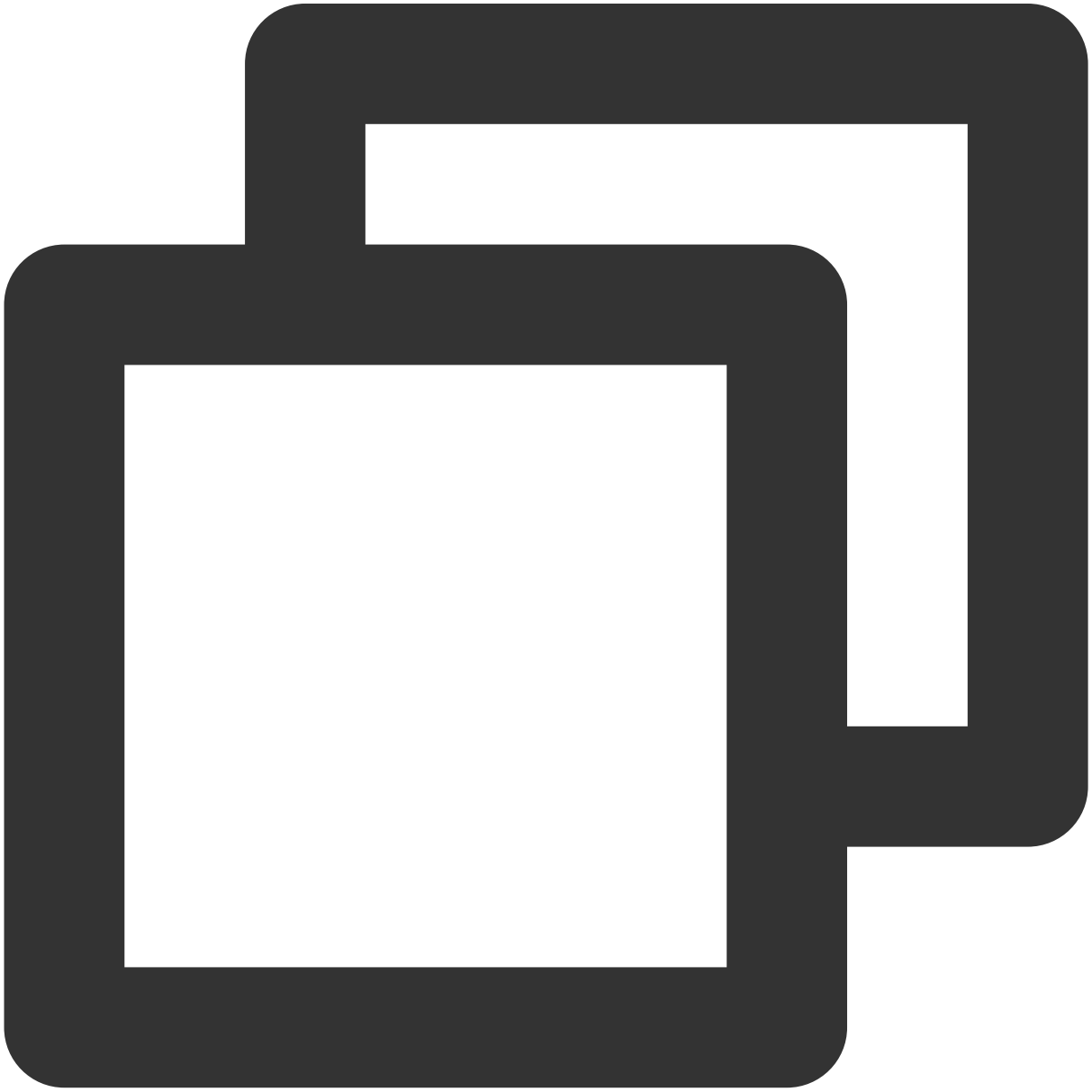
```
- (void)appendPartWithHeaders:(nullable NSDictionary <NSString *, NSString *> *)hea
```

```
                    body:(NSData *)body;
```

参数	描述
headers	要添加 headers 内容
body	添加到表单的数据

throttleBandwidthWithPacketSize

上传带宽限制



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

参数	描述
numberOfBytes	数据包最大值（单位：bytes，默认值：16kb）
delay	数据包读取时延（默认值：0）

TQUICURLResponseSerialization

响应数据序列化接口

API	描述
TQUICHTTPResponseSerializer	http 响应数据序列化
TQUICJSONResponseSerializer	JSON 响应数据序列化

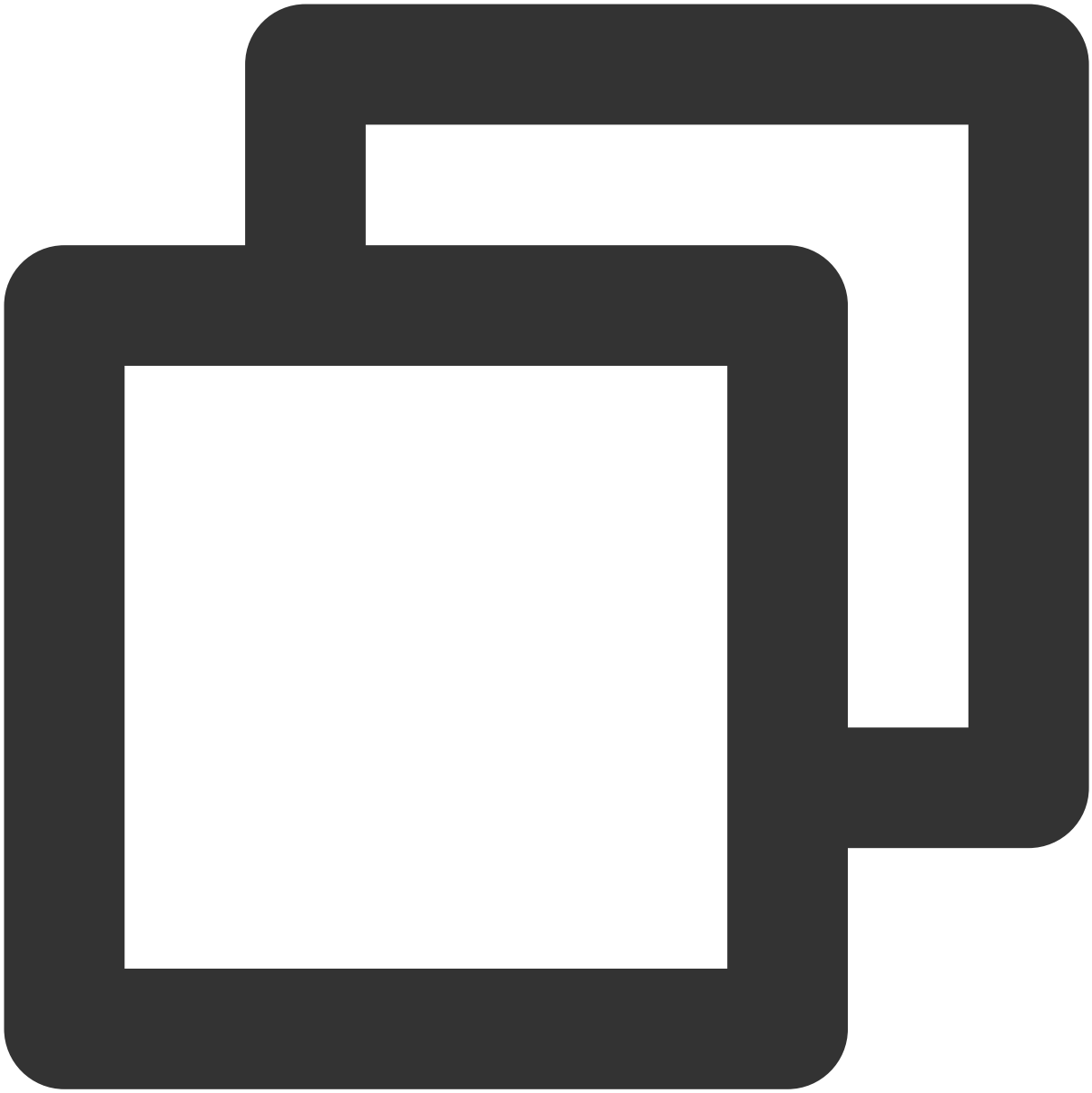
TQUICHTTPResponseSerializer

响应数据序列化接口

API	描述
serializer	实例化
validateResponse	检查响应数据是否合法

serializer

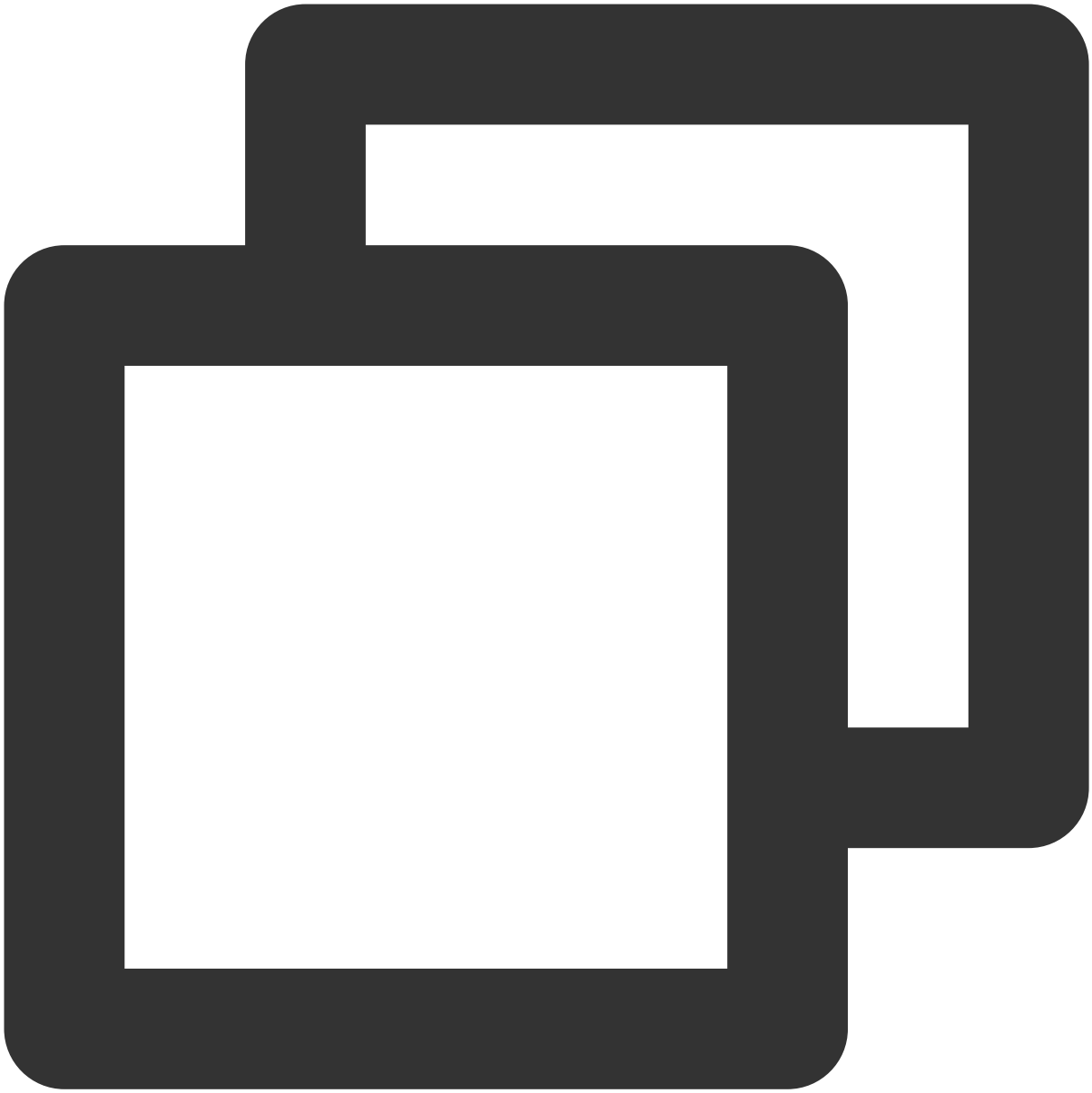
构造 TQUICHTTPResponseSerializer 实例



```
+ (instancetype)serializer
```

validateResponse

检查响应数据是否合法



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

参数	描述
response	要校验的返回结果，详见：NSHTTPURLResponse
data	返回的数据

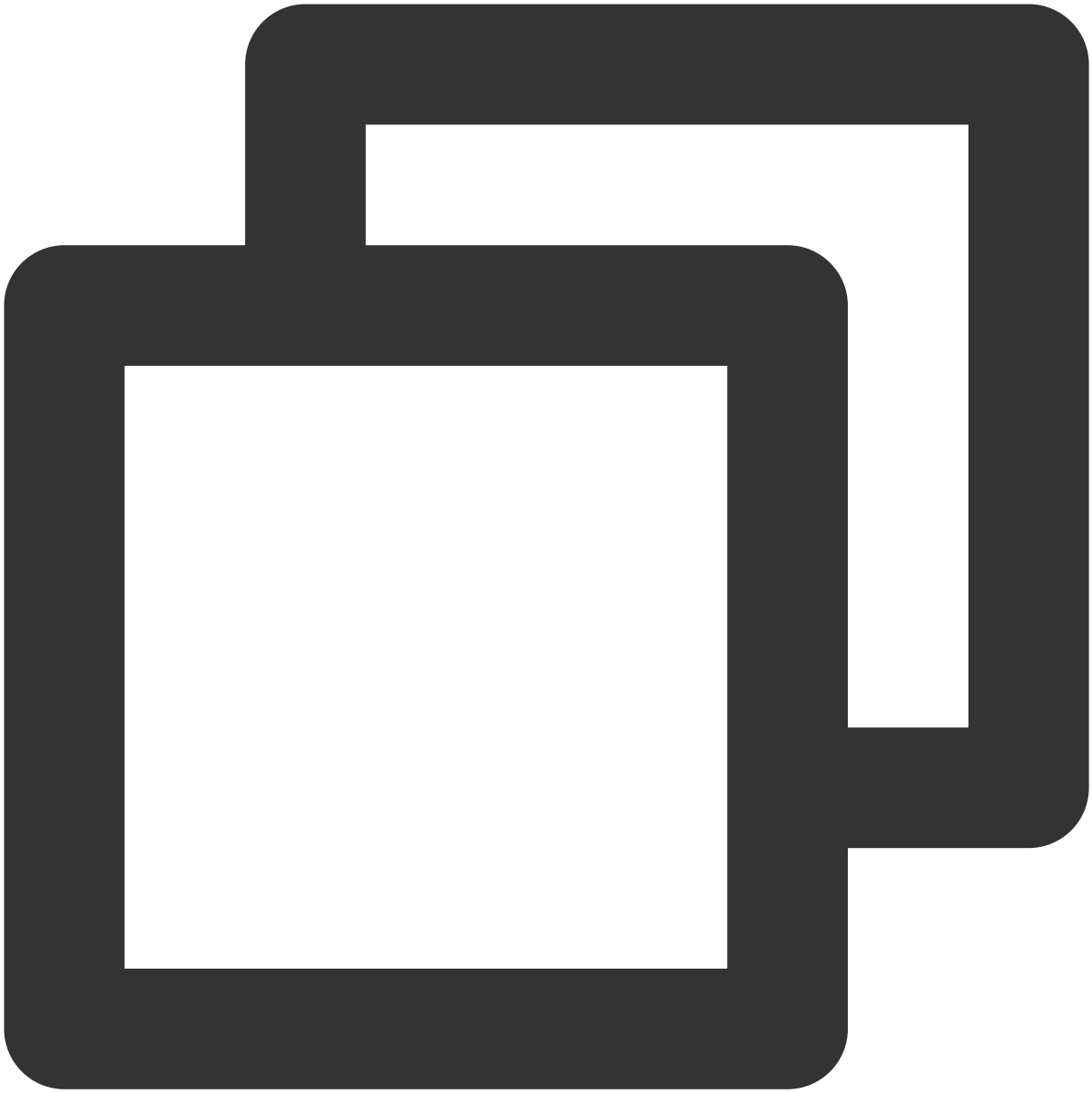
error	校验过程出错信息

TQUICJSONResponseSerializer

API	描述
serializerWithWritingOptions	创建实例

serializerWithWritingOptions

根据 JSON 序列化选项创建 TQUICJSONResponseSerializer 实例



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

参数	描述
serializerWithReadingOptions	根据 JSON 序列化选项创建实例，参考系统 NSJSONReadingOptions

TQUICURLSessionTaskMetrics

QUIC 网络状态统计，一次会话统计

属性	描述
transactionMetrics	数组，由于有重定向，一次会话会包含多次请求，每次请求的统计内容见：TQUICURLSessionTaskTransactionMetrics
taskInterval	任务耗时，从任务创建到执行完成的时间间隔
redirectCount	重定向次数

TQUICURLSessionTaskTransactionMetrics

QUIC 网络状态统计，一次请求统计

属性	描述
isValid	状态值是否有效
isQuic	是否 QUIC 请求
is0rtt	是否为 0-RTT 连接
isConnReuse	是否为连接复用
connectMillis	获取连接耗时，单位：毫秒
dnsMillis	获取 dns 耗时，单位：毫秒
dnsCode	获取 dns 错误码
ttfbMillis	获取首包耗时，单位：毫秒
completeMillis	获取请求完成时间（不含连接耗时），单位：毫秒
srttMicros	获取滑动平均rtt，单位：毫秒
packetsSent	获取发包量，单位：byte
packetsRetransmitted	获取重传包量，单位：byte
bytesSent	获取发送字节数，单位：byte
bytesRetransmitted	获取重传字节数，单位：byte
packetsLost	获取丢包量，单位：byte
packetsReceived	获取收包量，单位：byte
bytesReceived	获取收到字节数，单位：byte

streamBytesReceived	获取stream层收到的字节数，单位：byte

IPv6 访问

最近更新时间：2023-10-11 10:21:39

功能简介

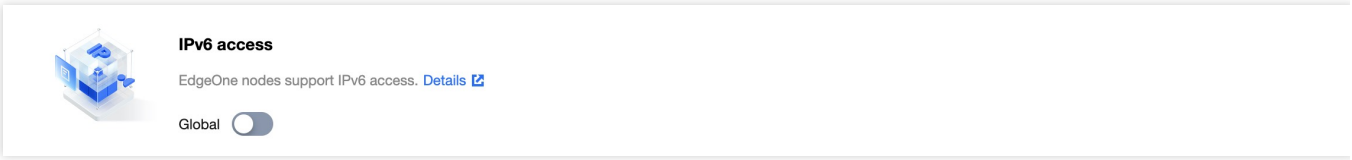
EdgeOne 支持一键开启 IPv6 访问，支持 IPv6 客户端以 IPv6 协议访问节点。
说明：
目前 EdgeOne 大部分节点已支持 IPv6 访问，您可联系我们确认具体资源覆盖情况。

使用场景

IPv6-only 网络环境：部分地区和组织可能已经在使用 IPv6-only 网络环境，这些网络环境中的设备可能无法直接访问基于 IPv4 的服务。通过启用 IPv6 访问功能，您可以确保这些客户端可以正常访问您的加速资源。
双栈网络环境：对于支持 IPv4 和 IPv6 的双栈网络环境，客户端可以根据网络条件自动选择使用 IPv4 或 IPv6 协议访问加速资源。在某些情况下，IPv6 连接可能比 IPv4 更快，因此启用 IPv6 访问功能可以帮助提高这些客户端的访问性能。
未来网络兼容性：随着 IPv4 地址资源逐渐枯竭，越来越多的网络和设备将采用 IPv6。通过启用 IPv6 访问功能，您可以确保您的加速服务在未来仍能与这些新兴网络和设备保持兼容。
政策和合规要求：某些地区或行业可能要求在服务中启用 IPv6 支持，以满足政策或合规要求。在这种情况下，开启 IPv6 访问功能可以帮助您满足这些要求。

操作步骤

1. 登录边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速** > **网络优化** ，进入网络优化详情页面。
3. 找到 IPv6 访问配置卡片，单击**全局开启**，即为站点全部域名开启 IPv6 访问。



最大上传大小

最近更新时间：2023-10-11 10:22:07

功能简介

最大上传大小即客户端用户单次请求中可以上传的数据的最大值。若超出设置的上限值，则 EdgeOne 会响应客户端 413（Request Entity Too Large）。

说明：

1. 仅 EdgeOne 企业版和标准版套餐支持关闭上传大小限制。
2. 最大上传大小限制默认开启，上限值为800MB。

使用场景

1. **大型文件上传：**对于涉及大型文件上传的业务，如在线视频平台、大型游戏分发、大数据分析等，您可以上调上传大小的上限值或直接关闭大小限制，让您的大文件上传无阻。
2. **防御恶意上传：** 对于用户互动频繁的业务，如社交媒体、论坛或博客等，可能会遇到恶意用户上传过大的文件，您可开启大小限制并调低上限值，防止过大的文件上传到源站，从而减轻源站的压力，防止潜在的安全风险，提升用户体验。
3. **节省传输流量：** 对于流量敏感的业务，如在线教育、在线会议、API 服务等，可能会希望通过限制上传文件的大小来节省流量，您可以开启大小限制并将上限值调整到一个较小的值，减少不必要的传输流量，降低流量成本。

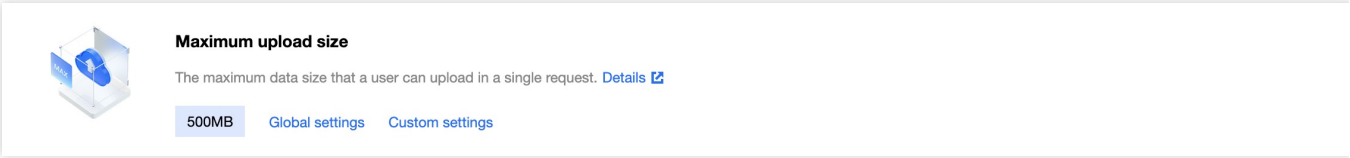
您可以通过灵活配置"最大上传大小"，满足各种业务场景的需要。

操作步骤

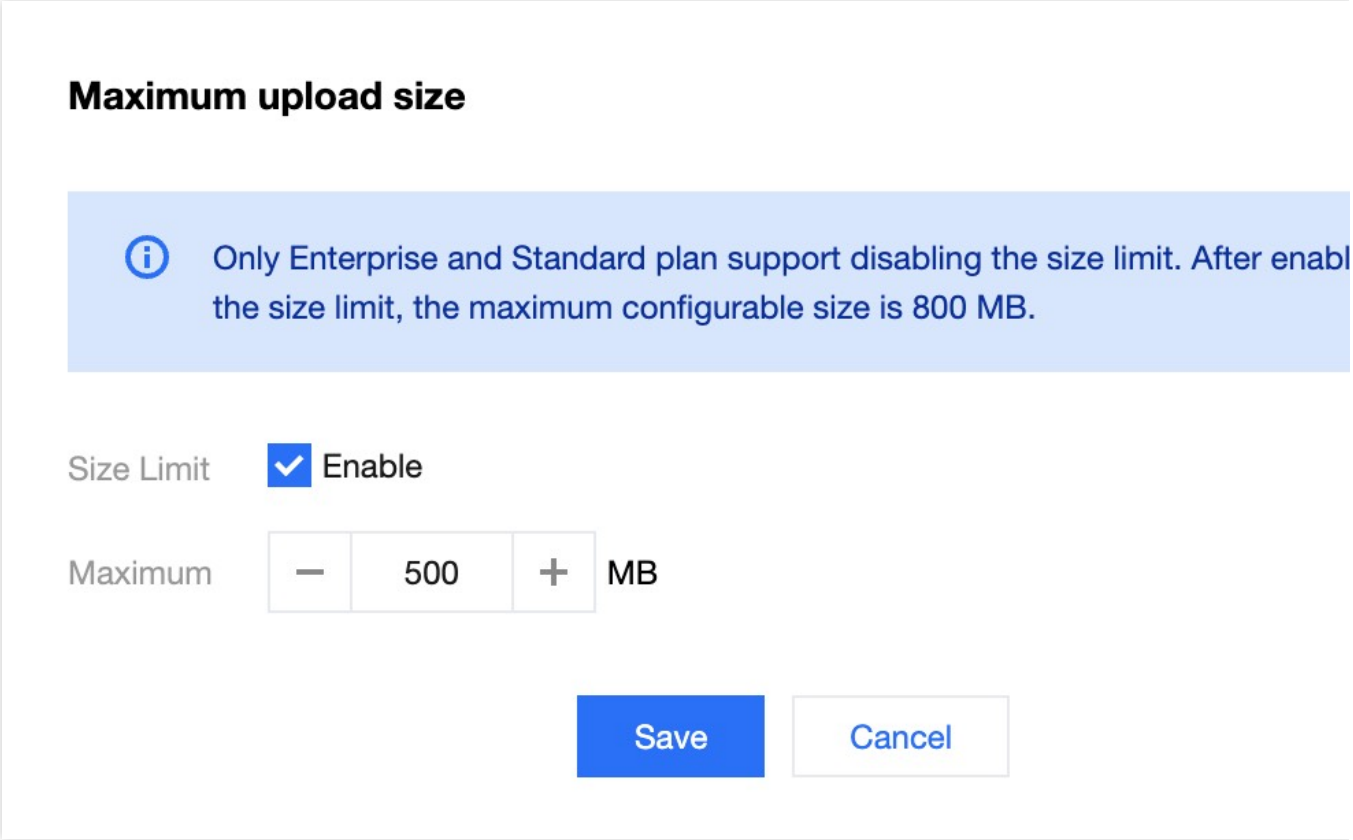
场景一：针对站点所有域名配置最大上传大小

若您需要对整个接入站点配置相同的最大上传大小，或作为站点级兜底配置，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速** > **网络优化** ，进入网络优化详情页面。
3. 找到**最大上传大小**配置卡片，单击**全局站点设置。**



4. 在弹窗内，勾选开启大小限制并修改上限值，单击**保存**后即可生效。



场景二：针对指定域名，路径或文件后缀等请求粒度配置最大上传大小

若您需要针对不同域名，路径或文件后缀等配置不同的最大上传大小，例如：针对 `example.com` 站点下的 `www.example.com` 域名配置最大上传大小为20 MB。可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**规则引擎。**
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
4. 在规则编辑页面，输入规则名称，选择 Host 为匹配类型，配置为 `www.example.com` 。
5. 单击**操作，**在弹出的操作列表内，选择操作为**最大上传大小**，单击

开启大小限制，上限值配置为 20 MB。

6. 单击**保存并发布**，即可完成该规则配置。

WebSocket

最近更新时间：2023-12-14 17:49:49

功能简介

EdgeOne 支持开启 WebSocket 协议访问，使用 WebSocket 协议使得服务端可主动向客户端推送数据。WebSocket 协议是基于 TCP 的一种持久化协议，它实现了客户端与服务器全双工（full-duplex）通信，允许服务器主动发送信息给客户端。在 WebSocket 协议之前，实现客户端和服务端双工通讯的 Web App 需要通过不断发送 HTTP 请求呼叫来进行询问，这导致了服务成本增加和效率低下的问题。由于具有全双工通信的优势，WebSocket 广泛应用于社交订阅、协同办公、行情播报、互动直播、在线教育、物联网等场景，能更好地节省服务器资源和带宽，并且能够更实时地进行通讯。

说明：

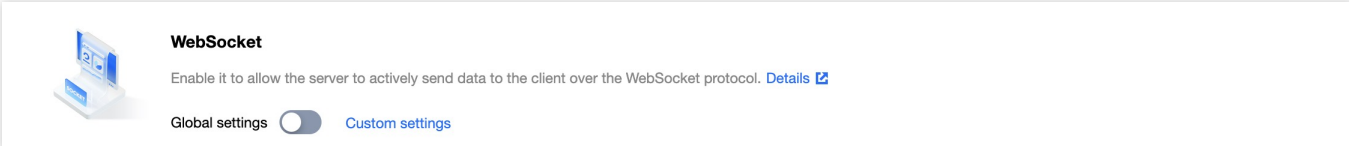
1. 目前仅支持基于 HTTP/1.1 的 WebSocket，不支持 HTTP/2 的 WebSocket。
2. 最大连接超时时长支持：300秒。

操作步骤

场景一：针对站点所有域名配置 WebSocket

若您需要对整个接入站点配开启/关闭 WebSocket，或作为站点级兜底配置，可参考以下步骤：

1. 登录边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速** > **网络优化** ，进入网络优化详情页面。
3. 找到 **WebSocket** 配置卡片，单击开关开启或关闭 WebSocket 功能,并配置最大连接超时时长。



开启状态：默认关闭，不支持 WebSocket 协议。开启后支持 WebSocket 协议。

最大连接超时时长：超时时间之内若没有数据收发，连接将被断开。

场景二：针对指定域名配置 WebSocket

若您需要针对不同域名配置 WebSocket，例如：针对 `example.com` 站点下的 `www.example.com` 域名配置 WebSocket。可参考以下步骤：

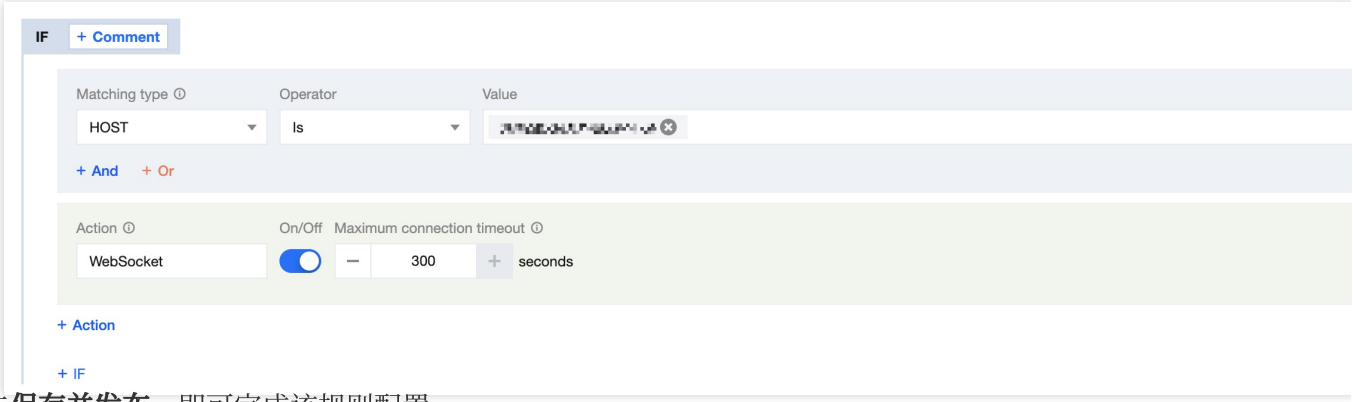
1. 登录 边缘安全加速平台 EO控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。

3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面在规则编辑页面，选择 Host 为匹配类型，配置为 `www.example.com` 。

4. 单击**操作，**在弹出的操作列表内，选择操作为**WebSocket**，点击**开关**开启，并配置最大连接超时时长。

说明：

最大连接时长：可配置1-300秒。



5. 单击**保存并发布**，即可完成该规则配置。

携带客户端 IP 头部回源

最近更新时间：2023-08-17 15:29:37

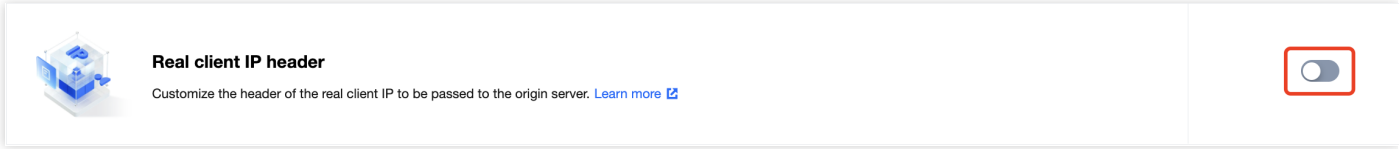
功能简介

支持自定义回源 HTTP 请求头部，携带真实客户端 IP 信息回源。

操作步骤

1. 登录 边缘安全加速平台控制台，在左侧菜单栏中，单击**站点加速** > **网络优化**。

2. 在网络优化页面，选择所需站点，单击  ，开启“真实客户端 IP 头部”功能。



3. 在弹窗中，自定义设置该头部的名称，例如 Tencent-Client-IP，单击**保存**。

携带客户端 IP 地理位置头部回源

最近更新时间：2023-08-17 15:29:57

功能简介

自定义头部携带客户端 IP 地理位置信息回源站。

> 说明：
>
> - 国家/地区维度，值采用两位字母国家/地区代码：ISO 3166-1 alpha-2 codes。
> - 暂不支持 IPv6。

操作步骤

1. 登录 边缘安全加速平台控制台，在左侧菜单栏中，单击**站点加速 > 网络优化**。

2. 在网络优化页面，选择所需站点，单击  ，开启“客户端 IP 地理位置”功能。

3. 在弹窗中，可自定义设置该头部的名称，或直接使用默认名称 EO-Client-IPCountry，单击**保存**。

开启 gRPC

最近更新时间：2023-10-11 10:23:00

EdgeOne 对 gRPC 的支持情况

EdgeOne 支持在控制台内开启 gRPC 协议支持（默认关闭），开启后，可同时支持 HTTP/HTTPS/gRPC 协议，根据用户请求协议自动适配，即请求 HTTP，则使用 HTTP 协议，请求 gRPC，则使用 gRPC 协议。

说明：

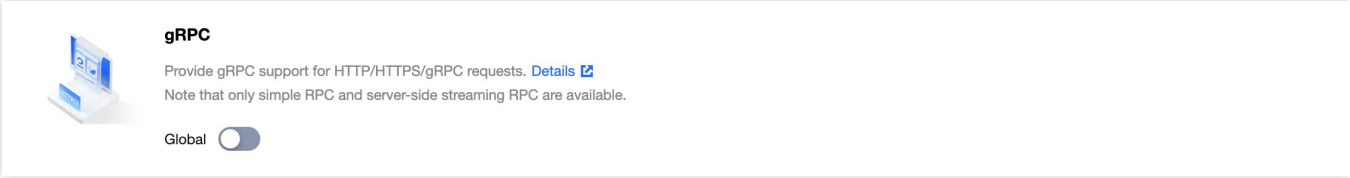
1. 当前仅支持 Simple RPC、Server-side streaming RPC 两种模式；
2. gRPC 是基于全链路 HTTP/2 实现的，因此开启 gRPC 的同时，请确保已开启 HTTP/2 访问及 HTTP/2 回源。

什么是 gRPC ？

gRPC (gRPC Remote Procedure Calls) 是 Google 发起的一个开源远程过程调用（Remote procedure call）系统。该系统基于 HTTP/2 标准设计，具备诸如双向流、流控、头部压缩、单 TCP 连接上的多复用请求等特性。

操作步骤

1. 登录边缘安全加速平台控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**，进入站点详情页面。
2. 在站点详情页面，单击**站点加速** > **网络优化**，找到 gRPC 模块。



3. 单击 gRPC 模块的“开关”，开启或关闭 gRPC 协议支持。

URL 重写
访问 URL 重定向

最近更新时间：2023-11-24 16:56:45

功能简介

EdgeOne 节点支持响应 3XX 状态码将客户端请求 URL 重定向到目标 URL。此功能可以将您业务场景中需要源站实现的 URL 重定向，改为直接由 EdgeOne 边缘节点构造并返回，减少回源的网络延时和源站生成 URL 重定向的性能消耗，提升客户端访问性能。

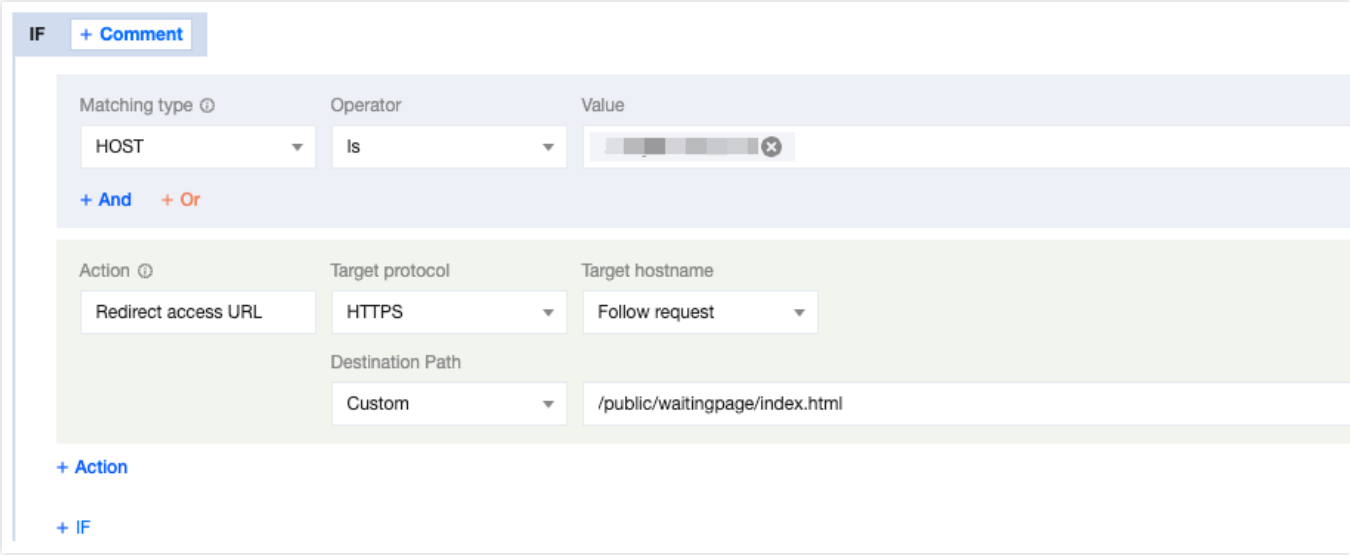
适用场景

以下为常见的访问 URL 重写适用场景：
网站迁移或重构：当网站进行迁移或重构时，可能会涉及到 URL结构的变化。为了保持旧链接的有效性，可以使用 URL 重定向将旧 URL 重定向到新 URL，从而确保用户和搜索引擎可以顺利访问新的资源。
地理位置或设备类型定向：根据用户的地理位置或设备类型，可以使用 URL 重定向将用户引导到不同的资源或页面。例如：针对移动设备用户提供专门优化的移动页面，或根据用户所在地区提供不同语言版本的页面。
临时维护或活动页面：当网站进行临时维护或举办特定活动时，可以使用 URL 重定向将用户引导到维护通知页面或活动页面，从而提升用户体验。

操作步骤

场景一：域名业务需要 302 跳转到临时维护页面

若您 `example.com` 站点下的 `www.example.com` 域名业务需要进行临时维护，希望将域名下的所有请求均 302 跳转至 `https://www.example.com/public/waitingpage/index.html`，可参考以下步骤：
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `www.example.com` 。
3.2 单击**操作**，在弹出的操作列表内，选择操作为 **访问 URL 重定向**。
3.3 配置访问 URL 重定向规则。可保持目标请求协议为HTTPS、目标 Hostname 为跟随请求，目标路径自定义为 `/public/waitingpage/index.html` 。
4. 完整的规则配置如下所示，单击**保存并发布**，即可完成该规则配置。



场景二：源站资源目录迁移，客户端请求 URL 需要保持不变

若您 `example.com` 站点下的 `www.example.com` 域名，所有的 jpg 图片资源需要由 `test` 目录迁移至 `newtest` 目录，但客户端请求 URL 需要保持不变，即仍访问 `test` 目录，可参考以下步骤：

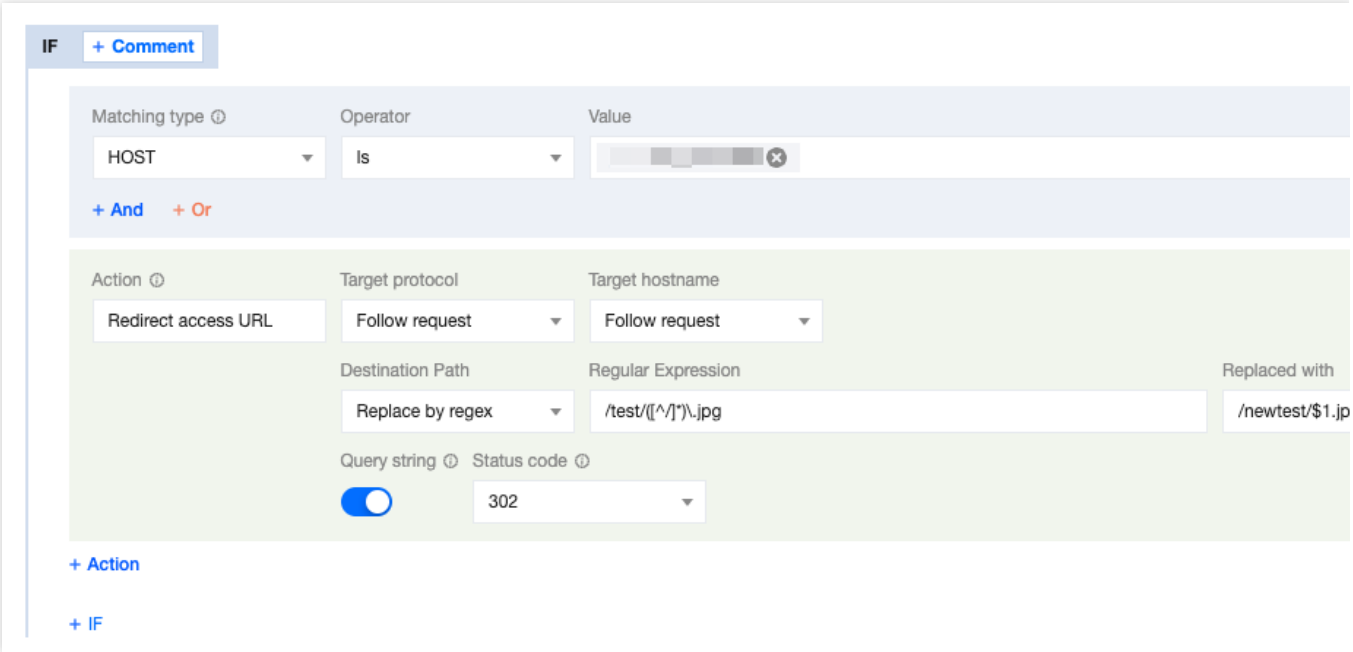
1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。

3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `www.example.com` 。

3.2 单击**操作**，在弹出的操作列表内，选择操作为 **访问 URL 重定向**。

3.3 配置访问 URL 重定向规则。可保持目标请求协议为跟随请求、目标 Hostname 为跟随请求。配置目标路径为正则替换，输入正则表达式 `/test/([^/]*)\\.jpg` 用于匹配目标路径，替换为 `/newtest/$1.jpg` 。

4. 完整的规则配置如下所示，单击**保存并发布**，即可完成该规则配置。



相关参考

访问 URL 重定向相关配置项说明如下：

配置项	说明
目标请求协议	目标重定向地址的请求协议，默认跟随请求协议，可支持指定跳转为 HTTP/HTTPS 协议。
目标 Hostname	目标重定向地址的 Hostname 部分，默认跟随请求域名，支持修改为自定义域名。例如：www.example.com。
目标路径	目标重定向地址的路径部分，提供三种模式选择： 跟随请求：默认配置，跟随请求的路径。 自定义：自定义一个完整路径，原请求路径替换为目标路径。例如 `/download` 。 正则替换：支持通过Google RE2正则表达式匹配和替换路径。同时支持以 `$num` 引用正则捕获组， `num` 代表组编号，最多支持 `$9` 。 例如：当前希望将路径 `/old-parh/1234` 替换为 `/new-path/1234` ，可以配置正则表达式为 `^/old-path/(\\d+)$` ，替换路径为 `/new-path/$1` ， `$1` 表示引用正则表达式中的第一个捕获分组，即路径中数字部分。
携带查询参数	是否携带原查询参数至目标 URL ，默认开启，即重定向后仍携带原查询参数。
状态码	选择重定向的响应状态码：302（默认），301，303 和 307。

回源 URL 重写

最近更新时间：2023-11-24 16:57:42

功能简介

客户端向 EdgeOne 节点发起请求，请求未命中节点缓存需要回源时，支持按照回源 URL 重写设定的规则，将请求重定向至源站的目标 URL，该功能不影响节点缓存。

适用场景

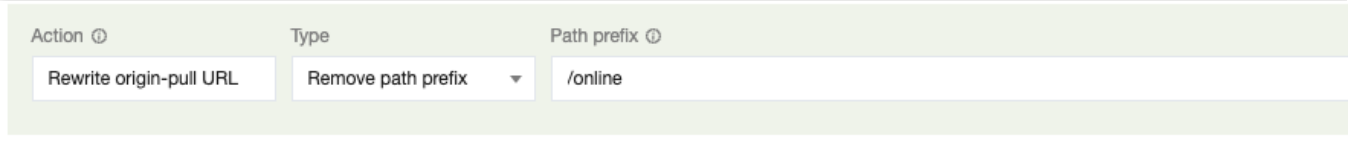
客户端访问的 URL 已经对外发布，不宜更改，而业务源站由于某些原因变更了源站上的 URL 路径；或者为了搜索引擎优化（SEO），客户端访问的 URL 和源站 URL 的路径并不一致。在这些场景下，通过设置回源 URL 重写规则，节点可以在不改变客户端访问 URL 的前提下，将回源的 URL 重写为源站上资源对应的实际 URL。

操作步骤

场景：客户端请求 URL 和对应的源站资源路径不一致

例如：客户端请求站点域名 `www.example.com` 的 URL 路径为 `http://www.example.com/online/index.html`，而此时文件目录已变更，回源时需要去掉目录前缀 `/online` 才可以获取到对应的文件资源，可参考以下步骤：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。
3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `www.example.com` 。
3.2 单击**操作**，在弹出的操作列表内，选择操作为 回源 **URL 重写**。
3.3 选择类型为移除路径前缀，路径前缀为 `/online`，配置如下：



4. 单击**保存并发布**，即可完成该规则配置。

相关参考

回源 URL 重写各配置项说明如下：

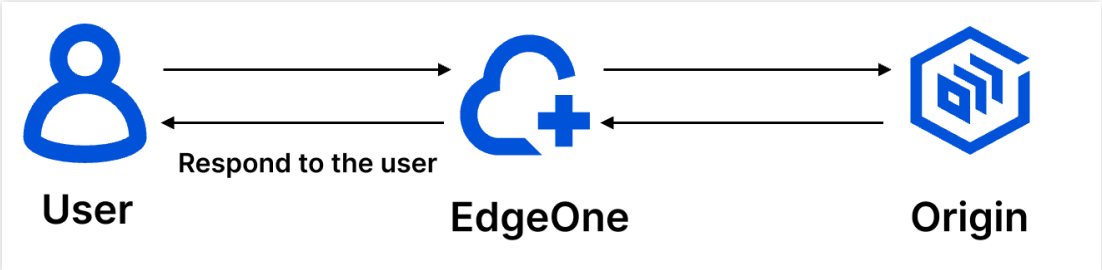
类型	说明
增加路径前缀	增加指定路径前缀至请求 URL Path。例如请求 URL为 `http://www.example.com/path0/index.html`，增加的路径前缀为 `/prefix`，则重写后的 URL 为 `http://www.example.com/prefix/path0/index.html`。
移除路径前缀	移除请求 URL 的指定路径前缀。例如请求 URL为 `http://www.example.com/path0/path1/index.html`，指定移除的路径前缀为 `/path0`，则重写后的 URL 为 `http://www.example.com/path1/index.html`。
替换完整路径	替换完整的请求 URL Path，例如请求 URL是 `http://www.example.com/path0/index.html`，替换完整路径为 `/new/page.html`，则重写后的 URL 是 `http://www.example.com/new/page.html`。

修改头部
修改 HTTP 节点响应头

最近更新时间：2023-10-13 14:22:34

功能简介

支持自定义设置/增加/删除 HTTP 节点响应头（节点响应客户方向），修改 HTTP 节点响应头不会影响节点缓存。



说明：

EdgeOne 已默认携带部分响应头，您无需配置，详见：EdgeOne 默认 HTTP 响应头。

场景：跨域头响应仅允许指定的域名访问页面资源

示例场景

若您的业务场景涉及跨域访问，当前业务域名为 `www.example.com` 的资源仅允许来自 `https://www.example.com` 的页面访问加速域名，可参考以下步骤。

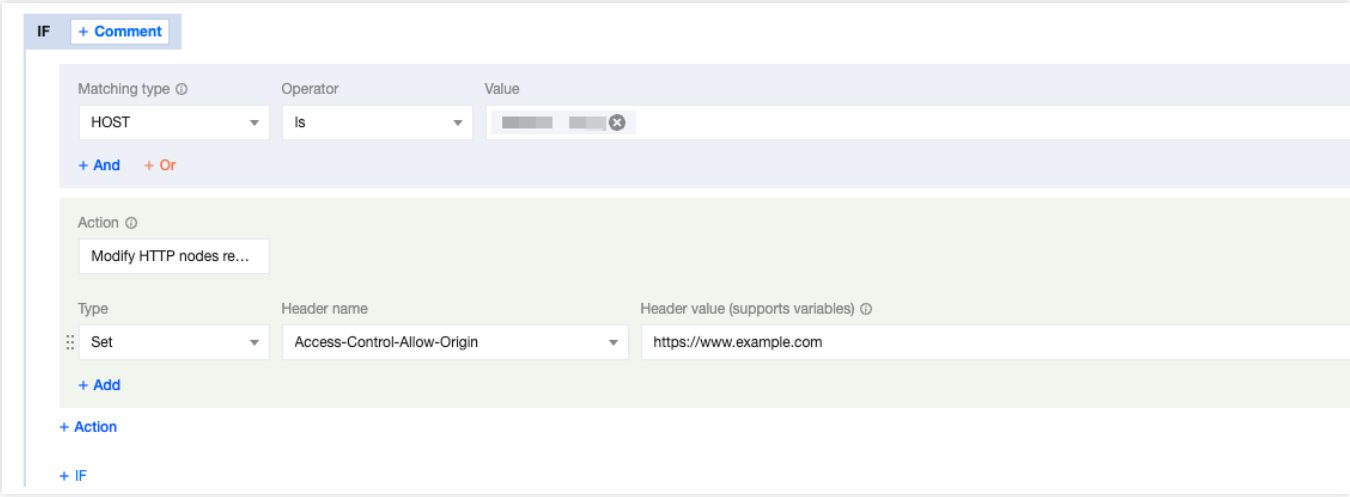
操作步骤

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。

3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `www.example.com` 。

3.2 单击**操作**，在弹出的操作列表内，选择操作为**修改 HTTP 节点响应头**。

3.3 选择类型为设置，头部名称为 `Access-Control-Allow-Origin` ，头部值设置为 `https://www.example.com` 。



4. 单击**保存并发布**，即可完成该规则配置。

相关参考

支持的类型说明：

类型	说明
设置	变更指定头部参数的取值为设置后的值，且头部唯一。注意：若指定头部不存在，则会增加该头部。
增加	增加指定的头部。注意：若头部已存在，仍会增加，不会覆盖已存在头部。
删除	删除指定的头部。

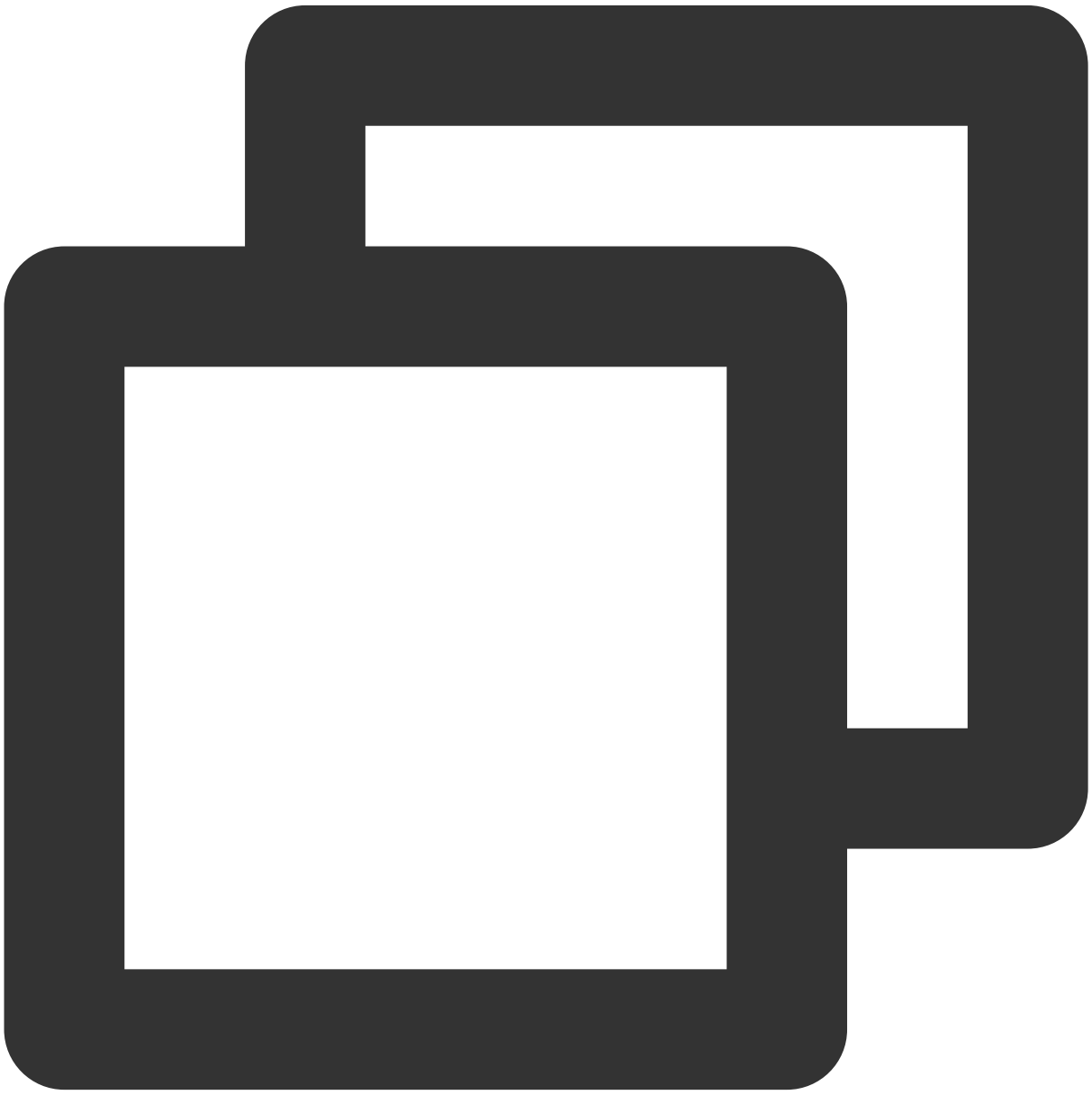
支持的头部类型说明：

头部类型	说明
自定义	支持修改自定义头部内容，填写说明如下： 名称：1 - 100 个字符，由数字0 - 9、字符a - z、A - Z，及特殊符 - 组成。 值：支持1 - 1000 个字符，不支持中文。
预设头部	支持以下预设头部修改： Access-Control-Allow-Origin：用于指定允许访问资源的源（域名），必须包含 http:// 或 https://。支持设置通配符 `*`，即允许被所有域请求。 Access-Control-Allow-Methods：用于设置跨域允许的 HTTP 请求方法，如 POST，GET，OPOTIONS。 Access-Control-Max-Age：用于指定 preflight 请求的结果在多少秒内有效，单位为秒。

	Content-Disposition：用于激活浏览器的下载弹窗，同时可以设置默认的下载的文件名。如：Content-Disposition：attachment;filename=FileName.txt Content-Language：用于定义页面所使用的语言代码。如：Content-Language: zh-CN

限制说明

同一个修改 HTTP 请求头操作中，可添加多条不同类型操作，最多30条，执行顺序为从上至下。
部分标准头部不支持修改，清单如下：



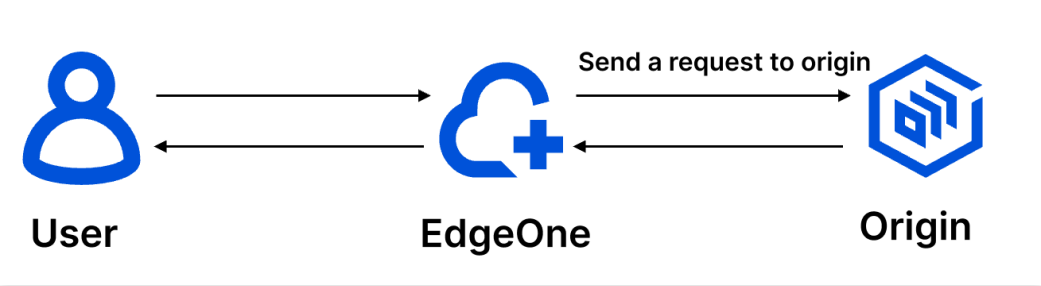
```
Accept-Ranges
Age
Allow
Authentication-Info
Cache-Control
Connection
Content-Encoding
Content-Length
Content-Location
Content-MD5
Content-Range
Content-Type
Date
Error
ETag
Expires
If-Modified-Since
Last-Modified
Meter
Proxy-Authenticate
Retry-After
Set-Cookie
Transfer-Encoding
Vary
WWW-Authenticate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

修改 HTTP 回源请求头

最近更新时间：2024-02-22 16:59:27

功能简介

Edgeone 节点回源时，若源站需要通过 HTTP 请求头获取某些特定的信息用于业务逻辑判断或者数据分析，例如：客户端设备类型、提供加速服务的厂商。可以通过自定义设置/增加/删除 HTTP 回源请求头（节点回源站方向）来实现，其中头部值支持变量，详情请参见 EdgeOne 预设变量。



说明：

EdgeOne 默认支持携带 `X-Forwarded-For` 和 `X-Forwarded-Proto` 回源，您无需再配置，详见：EdgeOne 默认回源 HTTP 请求头。

操作步骤

场景一：回源请求时携带设备类型信息至源站

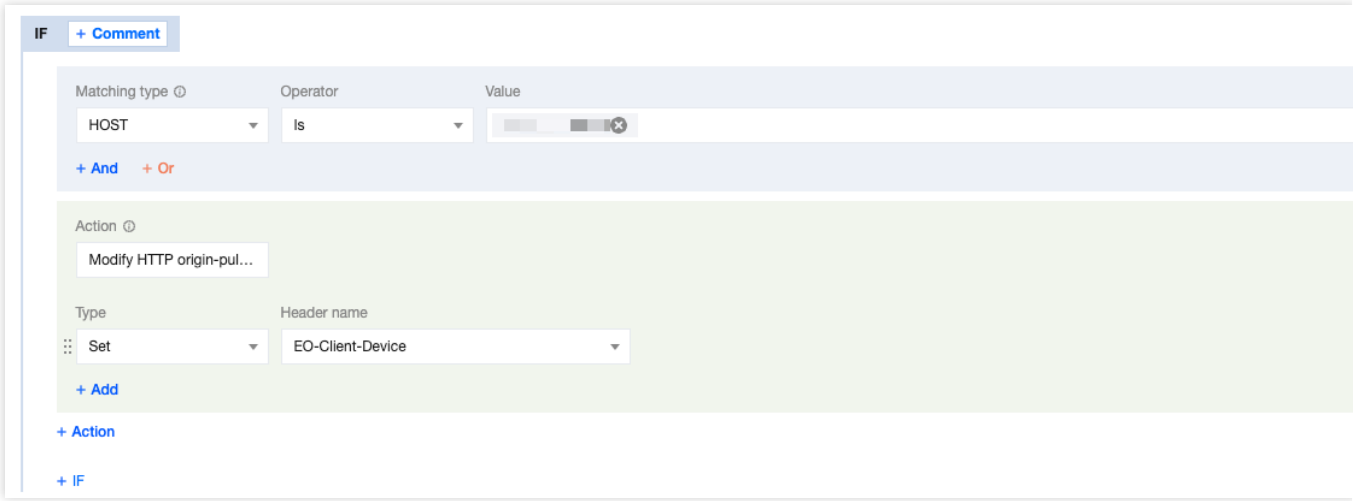
例如：若您希望当前的站点域名 `www.example.com` 在回源时，根据客户端请求携带的 `User-Agent` 头部值聚合为设备类型相关头部传递给源站，可参考以下步骤配置：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。

3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `www.example.com` 。

3.2 单击**操作**，在弹出的操作列表内，选择操作为修改 HTTP 回源请求头。

3.3 选择类型为增加，头部名称为 `EO-Client-Device` 的预设头部，配置结果如下。



4. 单击**保存并发布**，即可完成该规则配置。

场景二：回源请求时通过自定义头部传递加速域名至源站

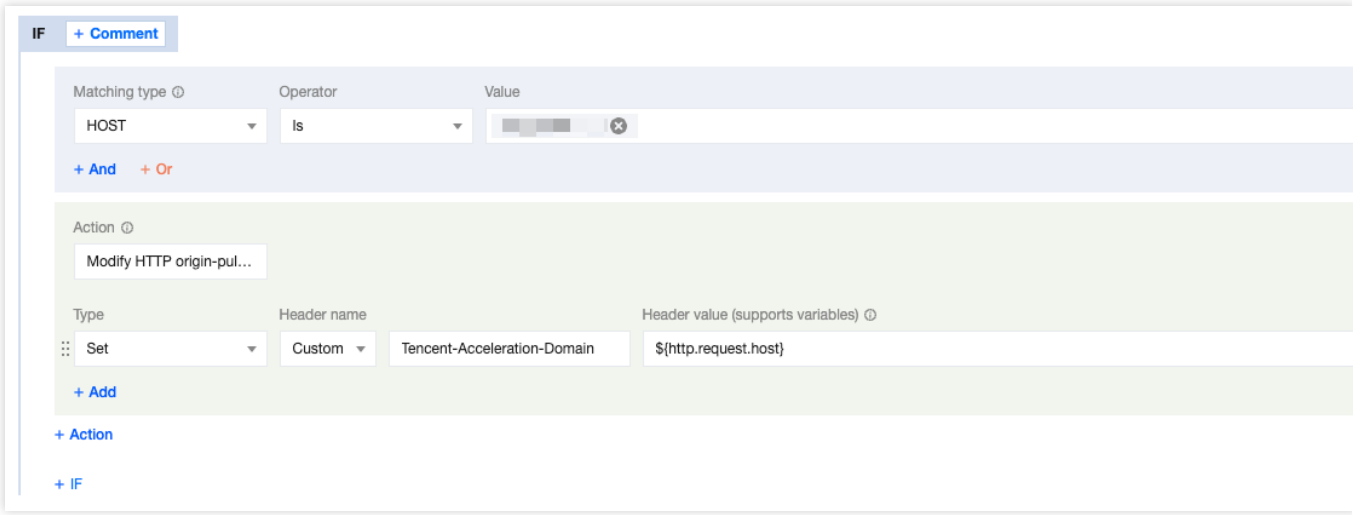
例如：若您当前的站点域名 `www.example.com` 已配置回源 Host 为其它域名，此时，您希望通过自定义头部 `Tencent-Acceleration-Domain` 将加速域名传递给源站，可参考以下步骤配置：

1. 登录 边缘安全加速平台控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。

3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `www.example.com` 。

3.2 单击**操作**，在弹出的操作列表内，选择操作为修改 HTTP 回源请求头。

3.3 选择类型为增加，头部名称为 `Tencent-Acceleration-Domain` ，头部值为 `${http.request.host}` ，配置结果如下。



4. 单击**保存并发布**，即可完成该规则配置。

相关参考

修改 HTTP 回源请求头支持的类型说明如下：

类型	说明
设置	变更指定头部参数的取值为设置后的值，且头部唯一。注意：若指定头部不存在，则会增加该头部。
增加	增加指定的头部。注意：若头部已存在，则会覆盖原有头部。
删除	删除指定的头部。

支持的头部名称说明：

头部类型	说明
自定义	自定义头部。 名称：1 - 100 个字符，由数字 0 - 9、字符 a - z、A - Z，及特殊符 - 组成。 值：1 - 1000 个字符，不支持中文。
预设头部	根据客户端 User-Agent 信息聚合的头部： 客户端设备类型：`EO-Client-Device` 取值：`Mobile`，`Desktop`，`SmartTV`，`Tablet` 或 `Others` 客户端操作系统：`EO-Client-OS` 取值：`Android`，`iOS`，`Windows`，`MacOS`，`Linux` 或 `Others` 客户端浏览器类型：`EO-Client-Browser` 取值：`Chrome`，`Safari`，`Firefox`，`IE` 或 `Others`

限制说明

同一个修改 HTTP 请求头操作中，可添加多条不同类型操作，最多30条，执行顺序为从上至下。

自定义错误页面

最近更新时间：2024-01-02 10:29:24

功能简介

EdgeOne 支持当源站响应指定错误状态码时，返回302状态码跳转到指定的自定义页面内，帮助您的站点在响应异常时，通过自定义的错误页面来告知用户当前的网站状态，避免用户在请求出错时无法确定具体的原因和处理方式。

说明

此功能为回源错误状态码的重定向，不支持 Token 鉴权、Web 防护规则等访问控制策略产生的状态码重定向。

操作步骤

例如：当前您自建的电商服务网站通过 `shop.example.com` 提供在线支付能力，如果在用户访问高峰期，可能出现源站过载，响应503状态码，为了避免用户流失，希望自定义一个错误页面 `https://www.example.com/error.html` 将用户引导至其它电商平台内下单购买。您可以参照步骤操作：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**规则引擎**。
3. 在规则引擎管理页面，单击**创建规则**，进入新规则的编辑页面。以当前场景为例，可按照如下步骤操作：

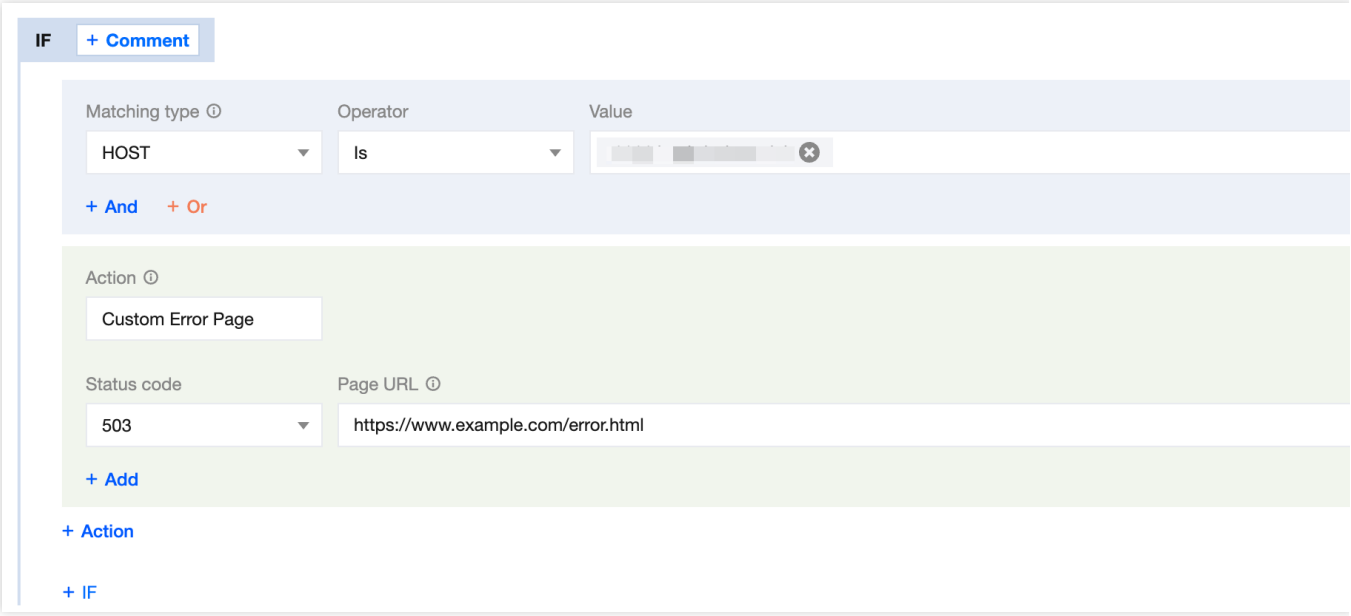
3.1 在规则编辑页面，匹配类型选择为 HOST 等于 `shop.example.com` 。

3.2 单击**操作**，在弹出的操作列表内，选择操作为**自定义错误页面**。

3.3 配置自定义错误页面，选择状态码为 503，页面 URL 输入 `https://www.example.com/error.html` 。相关配置项说明如下：

配置项	说明
状态码	指定源站响应的错误状态码： 4XX：400, 403, 404, 405, 414, 416, 451 5XX：500, 501, 502, 503, 504
页面地址	指定错误页面地址，例如：`https://www.example.com/custom-page.html`

4. 完整的规则配置如下所示，单击**保存并发布**，即可完成该规则配置。

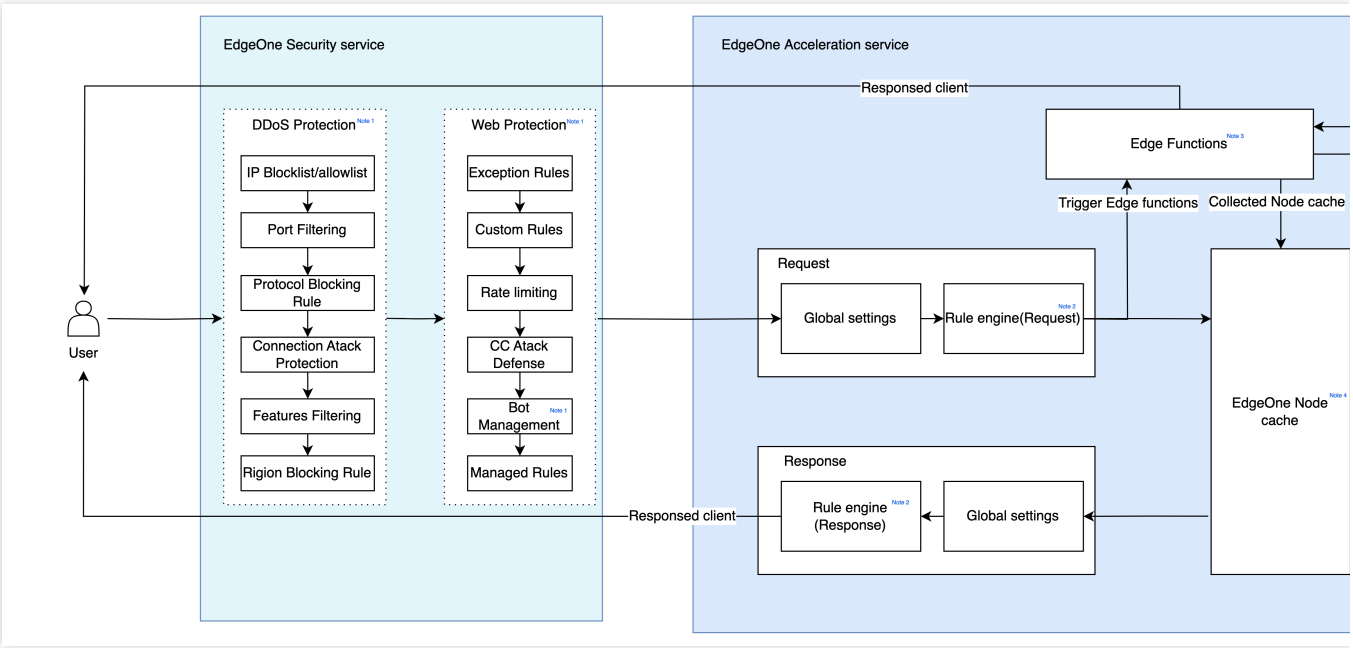

IF
+ Comment
Matching type
Operator
Value
HOST
Is
+ And
+ Or
Action
Custom Error Page
Status code
Page URL
503
https://www.example.com/error.html
+ Add
+ Action
+ IF

请求与响应行为
请求处理顺序

最近更新时间：2023-08-30 15:13:48

本文介绍了客户端向 EdgeOne 发起请求后，平台内所配置的各模块规则对该请求的处理顺序，帮助用户了解规则的生效顺序及影响，以保障配置的规则能按照期望的效果来工作。

请求处理顺序



用户发起请求后，请求将按照如上顺序进行处理，例如：用户同时在规则引擎和边缘函数中配置了修改 HTTP 头部设置，因为边缘函数后处理，则最终依照边缘函数的处理结果生效。

1. 安全服务模块内请求可能触发多种规则处置，相关处置顺序说明如下：
DDoS 防护的请求处理顺序仅针对已购买四层代理的独立 DDoS 防护用户生效，详情请参见 DDoS 防护概述。
Web 防护内包含 Bot 管理模块，Web 防护模块内请求处理顺序，详情请参见 Web 防护请求处理顺序。Bot 管理模块内的规则生效顺序，详情请参见 Bot 管理概述。
如果请求触发 EdgeOne 安全防护模块内的安全策略，且该策略为拦截、丢弃并拉黑、封禁 IP 等处置方式，则该请求将被拒绝。
2. 规则引擎内规则生效优先级均高于全局站点设置规则，并且规则引擎内的规则生效顺序为下方规则优先级更高，详情请参见 规则引擎概述。

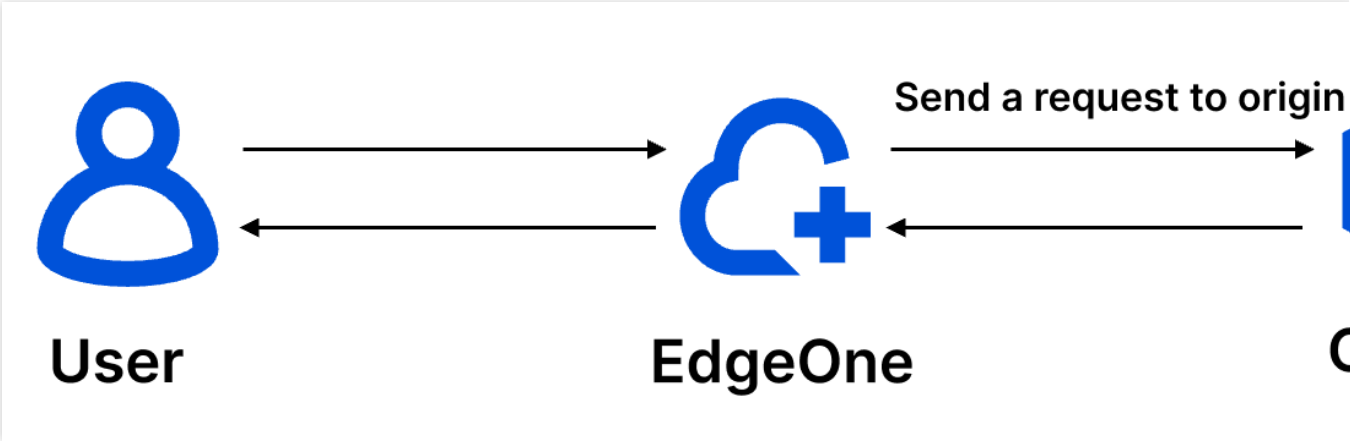
3. 当请求触发边缘函数规则时，请求将由边缘函数进行处理，边缘函数可通过子请求来访问第三方服务、EdgeOne 的缓存内容或者回客户源站，也可以直接响应客户端请求。
4. 在请求至 EdgeOne 节点缓存时，如果当前节点没有缓存，则会继续回源，如果节点命中缓存，则不会继续触发后续的回源规则，直接返回相应资源给用户。

EdgeOne 默认 HTTP 回源请求头

最近更新时间：2023-12-18 15:23:58

概述

默认情况下， EdgeOne 在回源请求时，将透传客户端的所有请求头部，同时携带由 EdgeOne 自定义的默认请求头部回源。如果您还需要对回源的 HTTP 头部进行增删改配置，可参见 修改 HTTP 回源请求头。



默认 HTTP 回源请求头介绍

以下为 EdgeOne 在回源时，默认增加的 HTTP 请求头及含义。

EO-Connecting-IP

`EO-Connecting-IP` 记录了与 EdgeOne 建立连接的客户端请求 IP 地址。如果该请求未经过任何代理服务器，则该头部 IP 即为真实客户端 IP 地址，如果请求经过代理服务器，则该头部 IP 值指代理服务器的 IP 地址。

X-Forwarded-For

`X-Forwarded-For` 用于记录代理服务器和真实客户端 IP 地址。当用户请求经过多跳到达 EdgeOne 边缘节点中时，可通过该头部来查看真实的客户端 IP 地址以及到达 EdgeOne 边缘节点的前序代理服务器地址。该头部取值如下：
如果发送到 EdgeOne 的请求中携带有 `X-Forwarded-For` 头部，该头部已记录了最原始的访问客户端 IP 地址，则 EdgeOne 会将到达 EdgeOne 边缘节点的前序代理服务器 IP 地址追加到头部值。假设与 EdgeOne 边缘节点建连

的前序代理服务器 IP 地址为 `10.1.1.1`，且请求时携带 `X-Forwarded-For: 192.168.1.1`（原始客户端 IP），则回源请求头取值为 `X-Forwarded-For: 192.168.1.1,10.1.1.1`。
如果发送到 EdgeOne 边缘节点的请求中没有 `X-Forwarded-For` 头部，则 EdgeOne 将在回源请求时，增加 `X-Forwarded-For` 头部，该头部取值为与 EdgeOne 边缘节点建连的前序代理服务器 IP 地址，取值与 `EO-Connecting-IP` 头部相同。
更多详情请参见 X-Forwarded-For。

X-Forwarded-Proto

`X-Forwarded-Proto` 用于记录客户端的请求协议，取值为当前客户端发起请求所使用的 HTTP 协议，头部取值有：

`X-Forwarded-Proto：http`

`X-Forwarded-Proto：https`

`X-Forwarded-Proto：quic`

更多详情请参见 X-Forwarded-Proto。

CDN-Loop

`CDN-Loop` 用于记录当前请求经过 EdgeOne 边缘加速节点的次数，主要用于平台防止请求环路。当客户端请求每重复经过 1 次 EdgeOne 的节点时，`CDN-Loop` 的次数则加1，并标记到请求头中，当请求头的 Loops 数值达到 ≥ 16时，则节点将拒绝请求并响应423状态码。

该头部格式示例：`CDN-Loop: TencentEdgeOne; loops=3`。

EO-LOG-UUID

EO-LOG-UUID 代表了当前请求的唯一标识符，该头部主要用于当出现访问异常时，通过该头部值匹配用户请求的全链路日志来定位问题。

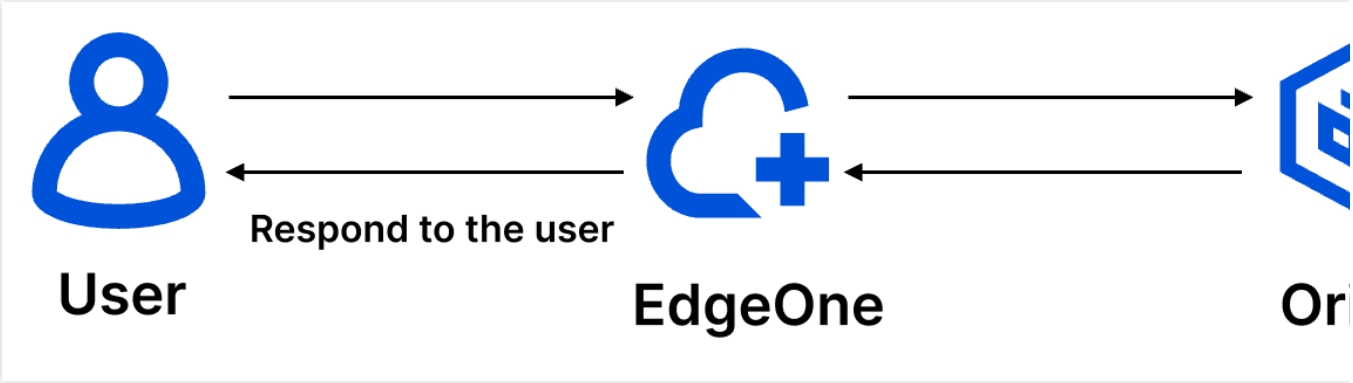
该头部格式示例：`EO-LOG-UUID: 4105283880544427145`。

EdgeOne 默认 HTTP 响应头

最近更新时间：2024-01-02 10:52:52

概述

默认情况下， EdgeOne 会透传源站的响应头部给客户端，除非客户有自定义 HTTP 头部增删改配置。如下将介绍由 EdgeOne 定义的响应头部，这些头部会默认响应给客户端。



默认 HTTP 响应头介绍

以下为 EdgeOne 在响应客户端请求时默认携带的 HTTP 响应头介绍。

EO-Cache-Status

`EO-Cache-Status` 用于标识当前客户端发起的请求是否命中缓存，取值有：

`EO-Cache-Status：HIT` ：表示请求资源在 EdgeOne 节点命中缓存且缓存未过期，直接由节点响应用户请求。

`EO-Cache-Status：MISS` ：表示请求资源在 EdgeOne 节点未命中缓存，或者命中缓存，但缓存过期，节点回源校验，源站文件有更新响应 200 状态码，节点需要回源站获取资源。

`EO-Cache-Status：RefreshHit` ：表示请求资源在 EdgeOne 节点命中缓存，但缓存过期，节点回源校验，源站文件无更新响应 304 状态码，节点继续用缓存响应用户请求。

Server

用于标识服务器名称。头部值取决于 Web Server 是基于什么服务搭建的。默认情况下，如果源站的 HTTP 响应头中包含该头部，则透传该头部至客户端，如果源站没有响应该头部，则 EdgeOne 节点将新增该头部，取值 `Server：TencentEdgeOne` 。更多详情请参见 Server。

腾讯云常见的源站类型响应 Server 值如下：
源站为腾讯云 COS 时：`Server: tencent-cos`。
源站为腾讯云 CVM 时：`Server: nginx`、`Server: Apache`、`Server: tomcat`、`Server: Microsoft-IIS`。
源站为腾讯云 CLB 时：`Server: openresty`。

Date

`Date` 头部取值为 EdgeOne 节点服务器当前时间。更多详情请参见 Date。
例如：`Date: Sat, 07 Jan 2023 14:15:52 GMT`。

Connection

用于标识客户端和服务器通信时对于长链接如何进行处理。 默认情况下，如果源站 HTTP 响应头中包含该头部，则透传该头部至客户端，如果源站没有响应该头部，EdgeOne 将根据以下情况，新增该头部：
如果当前请求使用 HTTP/2 或者 QUIC 则不添加此头部。
如果当前请求使用 HTTP1.0 且没有开启 keepalive，则该头部设置为：`Connection:close`。
源站响应头中不包含 `content-length` 且与 `transfer-encoding` 头部，则该头部设置为：`Connection:close`。
其他情况下，该头部设置为 `Connection:keepalive`。
更多详情请参见 Connection。

Alt-Svc

`Alt-Svc` 全称为“Alternative-Service”，该头部列举了当前站点备选的访问方式列表。一般用于在提供 QUIC 等新访问协议支持的同时，实现向下兼容。若域名开启 HTTP/3（QUIC）访问，则 EdgeOne 会默认在 HTTP 响应头中增加该头部。更多详情请参见 Alt-Svc。

EO-LOG-UUID

EO-LOG-UUID 代表了当前请求的唯一标识符，该头部主要用于当出现访问异常时，通过该头部值匹配用户请求的全链路日志来定位问题。

该头部举例如下：`EO-LOG-UUID: 4105283880544427145`。