CloudFrontRequest

August 17, 2026

CloudFrontRequest

Kind: Interface

Source: `src/adapter/lambda-edge/handler.ts`

Part of: Adapter

`CloudFrontRequest` represents an incoming request handled by the Lambda@Edge adapter. It carries request routing data, headers, body metadata, client IP information, and origin details for request processing.

Properties

Property	Type
`clientIp`	`string`
`headers`	`CloudFrontHeaders`
`method`	`string`
`querystring`	`string`
`uri`	`string`
`body`	`{ inputTruncated: boolean action: string encoding: string data: string }`
`origin`	`CloudFrontOrigin`

Diagram

```
graph LR
  Request[CloudFrontRequest]
  Request --> ClientIp[clientIp: string]
  Request --> Headers[headers: CloudFrontHeaders]
  Request --> Method[method: string]
  Request --> Query[querystring: string]
  Request --> Uri[uri: string]
```

```
  Request --> Body[body]
  Request --> Origin[origin: CloudFrontOrigin]

  Body --> Truncated[inputTruncated]
  Body --> Action[action]
  Body --> Encoding[encoding]
  Body --> Data[data]
```

Usage

```ts
import type { CloudFrontRequest } from "./handler";

function inspectRequest(request: CloudFrontRequest): CloudFrontRequest {
  const path = request.querystring
    ? `${request.uri}?${request.querystring}`
    : request.uri;

  console.log({
    method: request.method,
    path,
    clientIp: request.clientIp,
  });

  if (request.body.inputTruncated) {
    console.warn("Request body was truncated by CloudFront");
  }

  return request;
}
```

AI Coding Instructions

- Keep `uri` and `querystring` separate when reading or rewriting request paths.
- Preserve the `body` fields together when forwarding or transforming request content.
- Check `body.inputTruncated` before assuming `body.data` contains the full request body.
- Treat `headers` and `origin` as CloudFront-shaped values when integrating with Lambda@Edge handlers.

How it works

`CloudFrontRequest` is an exported TypeScript interface for the `cf.request` object within each Lambda@Edge event record. `CloudFrontEvent.cf.request` is typed as

`CloudFrontRequest`, and `CloudFrontEdgeEvent` contains an array of those records. src/adapter/lambda-edge/handler.ts:41-54 src/adapter/lambda-edge/handler.ts:69-79

- It requires `clientIp`, `headers`, `method`, `querystring`, and `uri`, all as strings except `headers`. src/adapter/lambda-edge/handler.ts:41-46
- `headers` maps header names to arrays of `{ key, value }` string pairs, allowing multiple values for one header name. src/adapter/lambda-edge/handler.ts:10-17
- Its optional `body` has `inputTruncated`, `action`, `encoding`, and `data` fields. src/adapter/lambda-edge/handler.ts:47-52
- Its optional `origin` is either an S3 origin or a custom origin. The S3 form includes authentication method, headers, domain, path, and region; the custom form includes headers, domain, timeouts, path, port, protocol, and SSL protocols. The union excludes having both forms at once. src/adapter/lambda-edge/handler.ts:19-39 src/adapter/lambda-edge/handler.ts:53

The Lambda@Edge adapter reads only the first record's request when constructing the Fetch API `Request`. It selects the host from the first `host` header value, falling back to the distribution domain name, combines it with `uri` and a nonempty `querystring`, then creates an HTTPS URL. src/adapter/lambda-edge/handler.ts:164-170

For headers, the adapter iterates every request-header entry and appends every array element's value to a `Headers` object. src/adapter/lambda-edge/handler.ts:172-175 A test demonstrates that three `x-forwarded-for` values become the comma-separated value observed through the Hono request header API. src/adapter/lambda-edge/handler.test.ts:93-121

For the body, the adapter omits it when `body` is absent, when `body.data` is falsy, or when `method` is exactly `GET` or `HEAD`. src/adapter/lambda-edge/handler.ts:198-207 When `body.encoding` is exactly `base64`, it decodes `body.data` into a `Uint8Array`; otherwise, it passes `data` through as a string. src/adapter/lambda-edge/handler.ts:208-211 For a retained body, it calculates the byte length and overwrites the request's `content-length` header with that length. src/adapter/lambda-edge/handler.ts:181-195

The original `CloudFrontRequest` is also passed to `app.fetch()` as the `request` binding, without copying or transforming it. src/adapter/lambda-edge/handler.ts:128-141 A callback may return that request object as the handler result; the handler returns the first callback result, and throws the first callback error. src/adapter/lambda-edge/handler.ts:126-145 src/adapter/lambda-edge/handler.test.ts:226-241

The adapter contains no explicit validation of `CloudFrontRequest` fields before directly accessing the first record, its request headers, method, URI, and query string. src/adapter/lambda-edge/handler.ts:124-145 src/adapter/lambda-edge/handler.ts:164-179

Relationships

- IMPORTS → `decodeBase64`
- IMPORTS → `encodeBase64`