CloudFrontEdgeEvent

August 19, 2026

CloudFrontEdgeEvent

Kind: Interface

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

Part of: Adapter

`CloudFrontEdgeEvent` represents the event payload received by the Lambda@Edge adapter. It groups CloudFront event records under `Records`, allowing the handler to access request, response, and distribution data from each `CloudFrontEvent`.

Properties

Property	Type
`Records`	`CloudFrontEvent[]`

Diagram

```
graph LR
  Edge[CloudFrontEdgeEvent] --> Records[Records]
  Records --> Event[CloudFrontEvent]
  Event --> Request[CloudFront Request]
  Event --> Response[CloudFront Response]
```

Usage

```
import type { CloudFrontEdgeEvent } from './handler'

export const handleEdgeEvent = (event: CloudFrontEdgeEvent) => {
  for (const record of event.Records) {
    const request = record.cf.request

    console.log(request.uri)
```

```
  }
}
```

AI Coding Instructions

- Treat `Records` as the source of CloudFront event data and iterate over its entries when processing events.
- Read request and response fields from each record's `cf` property rather than assuming event data exists at the root level.
- Keep Lambda@Edge request and response mutations compatible with the CloudFront event shape.
- Preserve the `CloudFrontEdgeEvent` type at adapter boundaries so handlers receive typed event records.

How it works

`CloudFrontEdgeEvent` is an exported TypeScript interface for the Lambda@Edge event shape accepted by this adapter. It declares one required property, `Records`, containing an array of CloudFront event records. [src/adapter/lambda-edge/handler.ts:77-79]

Each record has a `cf` object containing:

- `config`: distribution domain name, distribution ID, event type, and request ID. [src/adapter/lambda-edge/handler.ts:62-75]
- `request`: client IP, headers, HTTP method, query string, URI, and optional body and origin data. [src/adapter/lambda-edge/handler.ts:41-54]
- Optional `response`: headers, status, and optional status description. [src/adapter/lambda-edge/handler.ts:56-60] [src/adapter/lambda-edge/handler.ts:69-75]

The Lambda@Edge `handle()` adapter accepts this type as its event argument. It reads only `Records[0]`: it passes that record’s config, request, and optional response into the Hono request environment, then constructs a Fetch `Request` from that record’s request data. [src/adapter/lambda-edge/handler.ts:120-145] [src/adapter/lambda-edge/handler.ts:164-195]

For request construction, the adapter takes the first `host` header value when present, otherwise the distribution domain name; combines it with the request URI and optional query string; appends every declared request-header value; and derives a body from the record’s optional body data. [src/adapter/lambda-edge/handler.ts:164-189] Base64

body data is decoded, while `GET` and `HEAD` requests have no body. [src/adapter/lambda-edge/handler.ts:198-212]

`getConnInfo()` also expects this event in the context bindings and maps `Records[0].cf.request.clientIp` to the remote address. [src/adapter/lambda-edge/conninfo.ts:5-15]

The interface is re-exported from the Lambda@Edge adapter entry point. [src/adapter/lambda-edge/index.ts:6-14]

There is no runtime validation of the event structure in the shown adapter code. In particular, it directly indexes `Records[0]`, so a usable runtime event must contain a first record with the nested fields read by the handler or connection-info helper. [src/adapter/lambda-edge/handler.ts:128-141] [src/adapter/lambda-edge/handler.ts:164-179] [src/adapter/lambda-edge/conninfo.ts:11-15]