ALBProxyEvent

August 19, 2026

ALBProxyEvent

Kind: Interface

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

Part of: Adapter

`ALBProxyEvent` represents the request data passed from an Application Load Balancer to the Lambda handler. It carries HTTP method, path, headers, query parameters, body content, encoding state, and ALB request context for adapter-level request handling.

Properties

Property	Type
`httpMethod`	`string`
`headers`	`Record<string, string
`multiValueHeaders`	`Record<string, string[]
`path`	`string`
`body`	`string
`isBase64Encoded`	`boolean`
`queryStringParameters`	`Record<string, string
`multiValueQueryStringParameters`	`{ [parameterKey: string]: string[] }`
`requestContext`	`ALBRequestContext`

Diagram

```
graph LR
  ALB[Application Load Balancer] --> Event[ALBProxyEvent]
```

```
  Event --> Method[httpMethod]
  Event --> Path[path]
  Event --> Headers[headers and multiValueHeaders]
  Event --> Query[queryStringParameters and multiValueQueryStringParameters]
  Event --> Body[body and isBase64Encoded]
  Event --> Context[requestContext]
  Event --> Handler[Lambda handler]
```

Usage

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

function readRequest(event: ALBProxyEvent) {
  const contentType = event.headers["content-type"];
  const tags = event.multiValueQueryStringParameters.tags ?? [];
  const body = event.body ?? "";

  return {
    method: event.httpMethod,
    path: event.path,
    contentType,
    tags,
    body,
    isEncoded: event.isBase64Encoded,
  };
}
```

AI Coding Instructions

- Treat `body` as nullable and handle the absence of a request body before parsing it.
- Check `isBase64Encoded` before interpreting body content.
- Preserve both single-value and multi-value header and query parameter maps when adapting the event to another request format.
- Do not assume header names use a specific letter case; read the keys provided by the ALB event.
- Pass `requestContext` through when downstream code needs ALB request metadata.

How it works

`ALBProxyEvent` is an exported TypeScript interface for the Lambda event shape handled as an Application Load Balancer (ALB) request. It is one member of the

`LambdaEvent` union. src/adapter/aws-lambda/handler.ts:23-27 src/adapter/aws-lambda/handler.ts:84-97

Its required fields are:

- `httpMethod: string` and `path: string`, used as the method and path of the constructed `Request`. src/adapter/aws-lambda/handler.ts:86-89 src/adapter/aws-lambda/handler.ts:526-532
- `body: string | null` and `isBase64Encoded: boolean`; a truthy body is Base64-decoded when `isBase64Encoded` is true, otherwise UTF-8 encoded, then assigned as the request body and used to set `content-length`. src/adapter/aws-lambda/handler.ts:90-91 src/adapter/aws-lambda/handler.ts:333-339
- `requestContext: ALBRequestContext`, whose declared shape contains `elb.targetGroupArn: string`. src/adapter/aws-lambda/handler.ts:96 src/adapter/aws-lambda/types.ts:149-153

It can also contain optional single-value or multi-value headers and query-string parameter maps. src/adapter/aws-lambda/handler.ts:87-95

At runtime, an event is treated as an `ALBProxyEvent` when `requestContext` has its own `elb` property; this ALB test runs before the API Gateway v2 and Lattice tests. src/adapter/aws-lambda/handler.ts:625-644

For an ALB event, request-header conversion gives `multiValueHeaders` precedence over `headers`. Multi-value entries are joined with `;` and assigned as one header value; otherwise truthy values from `headers` are assigned. Header values containing non-ASCII characters are percent-encoded before assignment. src/adapter/aws-lambda/handler.ts:13-21 src/adapter/aws-lambda/handler.ts:503-524

The query string prefers `multiValueQueryStringParameters` when present. Each multi-value entry is rendered as repeated `key=value` pairs, while the single-value map is rendered as one `key=value` pair per truthy entry; these keys and values are not encoded by the ALB processor. src/adapter/aws-lambda/handler.ts:534-558

Request creation derives the domain from `requestContext.domainName` when available, then the `host` single-value header, then the `host` multi-value header; it constructs an HTTPS URL from that domain, the ALB path, and the generated query string. `ALBRequestContext` itself declares only `elb.targetGroupArn`, so the header fallback is the declared ALB-event route to a domain value. src/adapter/aws-lambda/handler.ts:301-342 src/adapter/aws-lambda/types.ts:149-153

When `handle()` processes such an event, it passes the converted request, the original event, its request context, and the optional Lambda context to `app.fetch()`. src/adapter/aws-lambda/handler.ts:251-274 If request creation throws, it logs the error and returns an ALB-formatted result with a `400` response for `TypeError`, or a `500` response otherwise. src/adapter/aws-lambda/handler.ts:255-266

Response formatting chooses `multiValueHeaders` when the incoming event has a truthy `multiValueHeaders`; otherwise it chooses `headers`. Response bodies are Base64-encoded for binary content types or non-`identity` content encodings, and are text otherwise. src/adapter/aws-lambda/handler.ts:344-385 For `set-cookie`, ALB output stores all cookies in `multiValueHeaders['set-cookie']` in multi-value mode, but stores only the first cookie in `headers['set-cookie']` otherwise. src/adapter/aws-lambda/handler.ts:388-400 src/adapter/aws-lambda/handler.ts:572-578