ApiGatewayRequestContext

August 17, 2026

ApiGatewayRequestContext

Kind: Interface

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

Part of: Adapter

`ApiGatewayRequestContext` describes API Gateway metadata attached to an AWS Lambda request. It carries account, API, domain, HTTP, authorization, identity, and request-path details used by the Lambda adapter.

Properties

Property	Type
`accountId`	`string`
`apiId`	`string`
`authorizer`	`{ claims?: unknown scopes?: unknown }`
`domainName`	`string`
`domainPrefix`	`string`
`extendedRequestId`	`string`
`httpMethod`	`string`
`identity`	`Identity`
`path`	`string`
`protocol`	`string`
`requestId`	`string`
`requestTime`	`string`
`requestTimeEpoch`	`number`

Property	Type
`resourceId`	`string`
`resourcePath`	`string`
`stage`	`string`

Diagram

```
graph LR
  Request[API Gateway Request] --> Context[ApiGatewayRequestContext]
  Context --> Account[accountId / apiId]
  Context --> Domain[domainName / domainPrefix]
  Context --> Http[httpMethod / path / protocol]
  Context --> Auth[authorizer claims and scopes]
  Context --> Identity[identity: Identity]
  Context --> RequestId[extendedRequestId]
```

Usage

```
import type { ApiGatewayRequestContext } from './types';

function getRequestDetails(context: ApiGatewayRequestContext) {
  return {
    requestId: context.extendedRequestId,
    method: context.httpMethod,
    path: context.path,
    host: context.domainName,
    claims: context.authorizer.claims,
    identity: context.identity,
  };
}
```

AI Coding Instructions

- Treat `authorizer.claims` and `authorizer.scopes` as `unknown`; validate or narrow their types before reading values.
- Read `identity` through the imported `Identity` type rather than duplicating its fields.
- Use `httpMethod`, `path`, and `protocol` when building request URLs or routing adapter inputs.
- Preserve `extendedRequestId` when logging request-specific errors or diagnostics.

- Do not assume `authorizer` contains claims or scopes; both fields are optional.

How it works

`ApiGatewayRequestContext` is an exported TypeScript interface representing the `requestContext` field of the adapter's API Gateway v1 proxy event (`APIGatewayProxyEvent`). That event type contains HTTP method, headers, path, body, query parameters, and this context object. [src/adapter/aws-lambda/handler.ts:63-82]

The interface requires API/account and request metadata: `accountId` , `apiId` , `domainName` , `domainPrefix` , `extendedRequestId` , `httpMethod` , `path` , `protocol` , `requestId` , `requestTime` , `requestTimeEpoch` , `resourcePath` , and `stage` . `resourceId` is optional. [src/adapter/aws-lambda/types.ts:85-105]

It also requires:

- `authorizer` , whose only declared fields are optional `claims` and `scopes` , each typed as `unknown` . [src/adapter/aws-lambda/types.ts:88-91]
- `identity` , containing required `sourceIp` and `userAgent` , plus optional AWS, Cognito, caller, organization, user, and client-certificate fields. [src/adapter/aws-lambda/types.ts:69-83] A client certificate, when present, includes PEM, subject/issuer distinguished names, serial number, and `notBefore` / `notAfter` validity strings. [src/adapter/aws-lambda/types.ts:58-67]

The Lambda adapter treats an event as API Gateway v1 when it is not identified as an ALB event, API Gateway v2 event, or Lattice v2 event; it then selects the v1 processor. [src/adapter/aws-lambda/handler.ts:625-637] The v1 event's HTTP request is built from top-level event fields such as `path` and `httpMethod` , rather than from this context's `path` or `httpMethod` . [src/adapter/aws-lambda/handler.ts:443-450]

When handling an event, `handle()` obtains `event.requestContext` without transforming it and passes it to `app.fetch()` as `requestContext` , alongside the original event and optional Lambda context. [src/adapter/aws-lambda/handler.ts:116-124] [src/adapter/aws-lambda/handler.ts:252-274] The streaming handler follows the same pattern. [src/adapter/aws-lambda/handler.ts:147-157]

`getConnInfo()` accepts this interface as one member of its Lambda request-context union. [src/adapter/aws-lambda/conninfo.ts:9-18] If the context has an `identity` property and `identity.sourceIp` is truthy, it returns that value as `remote.address` . [src/adapter/aws-lambda/conninfo.ts:45-53] The included test constructs a v1-shaped context and verifies that `identity.sourceIp` becomes the returned remote address. [src/adapter/aws-lambda/conninfo.test.ts:5-35]

This file declares only the interface shape; it contains no runtime validation, conversion, error handling, or side effects for `ApiGatewayRequestContext`. [src/adapter/aws-lambda/types.ts:85-105] The adapter's public AWS Lambda entry point re-exports the interface as a type. [src/adapter/aws-lambda/index.ts:6-14]