ErrorPayload

August 18, 2026

ErrorPayload

Kind: Interface

Source: `packages/websockets/exceptions/base-ws-exception-filter.ts`

Part of: Websockets

`ErrorPayload` defines the standardized error message shape emitted by the WebSocket exception handling layer. It identifies the payload as an error, provides a human-readable message, and preserves the underlying `Cause` for clients or downstream handlers that need additional context.

Properties

Property	Type
`status`	`'error'`
`message`	`string`
`cause`	`Cause`

Diagram

```
graph LR
  A[WebSocket handler throws an error] --> B[Base WebSocket Exception Filter]
  B --> C[ErrorPayload]
  C --> D[status: "error"]
  C --> E[message: string]
  C --> F[cause: Cause]
  C --> G[WebSocket client receives error event]
```

Usage

```ts
import type { ErrorPayload } from './base-ws-exception-filter';

function createErrorPayload(
  message: string,
  cause: Cause,
): ErrorPayload {
  return {
    status: 'error',
    message,
    cause,
  };
}

const payload = createErrorPayload(
  'Unable to join the room.',
  {
    code: 'ROOM_NOT_FOUND',
    details: { roomId: 'room-123' },
  },
);

// Example: emit through a WebSocket server or gateway
client.emit('exception', payload);
```

AI Coding Instructions

- Always set `status` to the literal value `'error'`; do not use arbitrary status strings.
- Keep `message` safe and client-friendly; avoid exposing stack traces, secrets, or internal database details.
- Populate `cause` with the original structured error context so exception filters can preserve error metadata.
- Use `ErrorPayload` consistently for WebSocket error emissions rather than creating endpoint-specific error shapes.
- Ensure WebSocket clients handle the `status`, `message`, and `cause` fields when processing exception events.

How it works

`ErrorPayload<Cause = { pattern: string; data: unknown }>` is an exported TypeScript interface for the non-object WebSocket exception payload shape. Its default `cause` type contains a string `pattern` and `unknown` `data`. [base-ws-exception-filter.ts:11-24]

- `status` is required and is limited to the literal value `'error'`. [base-ws-exception-filter.ts:15]

- `message` is required and is a string. [base-ws-exception-filter.ts:17-19]
- `cause` is optional and has the generic `Cause` type. [base-ws-exception-filter.ts:20-23]
- The interface is exported again through the WebSocket exceptions index. [exceptions/index.ts:1]

`BaseWsExceptionFilter` constructs an `ErrorPayload<unknown>` for a `WsException` whose `getError()` result is not an object, setting `status` to `'error'` and `message` to that result. It emits this payload on the client's `'exception'` event. [base-ws-exception-filter.ts:76-92] `WsException.getError()` returns the value passed to its constructor, whose declared type is `string | object`; therefore this payload path is for string-valued `WsException` errors. [ws-exception.ts:3-6, ws-exception.ts:24-26]

For exceptions that are not `WsException` instances, the filter constructs an `ErrorPayload<unknown>` with `status: 'error'` and the message `'Internal server error'`, then emits it on `'exception'`. [base-ws-exception-filter.ts:72-74, base-ws-exception-filter.ts:100-110, core/constants.ts:3-8] If the exception is not an `IntrinsicException`, this unknown-error path also logs the exception. [base-ws-exception-filter.ts:112-115]

A `cause` is attached only when both a cause argument exists and `includeCause` is enabled. The filter defaults `includeCause` to `true` and defaults `causeFactory` to a function returning `{ pattern, data }`; a configured `causeFactory` determines the attached cause value. [base-ws-exception-filter.ts:51-55, base-ws-exception-filter.ts:88-90, base-ws-exception-filter.ts:106-108] The `catch` method obtains the WebSocket pattern and data from the arguments host and passes them as the cause input. [base-ws-exception-filter.ts:57-65]

A `WsException` carrying an object does **not** get wrapped as an `ErrorPayload`; the filter emits that object directly on `'exception'`. [base-ws-exception-filter.ts:77-81]

Relationships

- IMPORTS → `ArgumentsHost`
- IMPORTS → `IntrinsicException`
- IMPORTS → `Logger`
- IMPORTS → `WsExceptionFilter`
- IMPORTS → `isObject`
- IMPORTS → `MESSAGES`