BunWebSocketData

August 17, 2026

BunWebSocketData

Kind: Interface

Source: `src/adapter/bun/websocket.ts`

Part of: Adapter

`BunWebSocketData` stores per-connection metadata for the Bun WebSocket adapter. It keeps the WebSocket event handlers, the requested URL, and the negotiated protocol together so Bun callbacks can access connection context.

Properties

Property	Type
`events`	`WSEvents`
`url`	`URL`
`protocol`	`string`

Diagram

```
graph LR
  WS[Bun WebSocket connection] --> Data[BunWebSocketData]
  Data --> Events[events: WSEvents]
  Data --> URL[url: URL]
  Data --> Protocol[protocol: string]
  Events --> Handlers[WebSocket event handlers]
```

Usage

```
import type { BunWebSocketData } from "./adapter/bun/websocket";

const connectionData: BunWebSocketData = {
```

```
  events: {
    open(ws) {
      ws.send("Connected");
    },
    message(ws, message) {
      ws.send(`Received: ${message}`);
    },
  },
  url: new URL("wss://example.com/chat"),
  protocol: "chat",
};

// Pass connectionData as WebSocket adapter context where required.
```

AI Coding Instructions

- Keep `events` bound to the `WSEvents` handlers expected by the WebSocket adapter.
- Store the full `URL` object in `url`; do not replace it with a string.
- Set `protocol` to the protocol selected for the connection, including an empty string when no protocol is negotiated.
- Preserve this data object when forwarding Bun WebSocket callbacks so handlers receive the correct connection context.

How it works

`BunWebSocketData` is a TypeScript interface for the data attached to a Bun server WebSocket by this adapter. It has three fields: `events: WSEvents`, `url: URL`, and `protocol: string`. [src/adapter/bun/websocket.ts:24-28]

- `events` holds optional `onOpen`, `onMessage`, `onClose`, and `onError` callbacks. The callback WebSocket context is typed as `WSContext<unknown>` because `BunWebSocketData` uses `WSEvents` without a type argument. [src/helper/websocket/index.ts:14-19]
- `url` is the parsed request URL. During an upgrade, the adapter sets it with `new URL(c.req.url)`. [src/adapter/bun/websocket.ts:61-68]
- `protocol` is set to the first value from the `sec-websocket-protocol` request header after splitting on commas and trimming whitespace; it is an empty string when that header is absent. [src/adapter/bun/websocket.ts:65-68]

The adapter passes a `BunWebSocketData` object as the `data` option to `server.upgrade`. [src/adapter/bun/websocket.ts:49-55] [src/adapter/bun/websocket.ts:61-69] If no Bun

server is available from the context, the upgrade path throws `TypeError('env has to include the 2nd argument of fetch.')`. [src/adapter/bun/websocket.ts:49-59]

When creating a `WSContext` for a connected socket, the adapter reads `ws.data.url` and `ws.data.protocol` into the context, and exposes the Bun socket itself as `raw`. [src/adapter/bun/websocket.ts:33-45] The `WSContext` constructor copies a non-null URL into a new `URL` object and maps a missing protocol to `null`; this adapter supplies both fields from `BunWebSocketData`. [src/helper/websocket/index.ts:70-77]

The Bun WebSocket handler reads `ws.data.events` on `open`, `close`, and `message`, then calls the corresponding callback only when it is present. [src/adapter/bun/websocket.ts:76-103] `onOpen` receives an `Event('open')`; `onClose` receives a `CloseEvent` containing Bun's `code` and `reason`; and `onMessage` receives a `MessageEvent` whose data is either the incoming string or `message.buffer`. [src/adapter/bun/websocket.ts:77-102] Although `WSEvents` declares `onError`, this handler has no error method and does not invoke `onError`. [src/helper/websocket/index.ts:14-19] [src/adapter/bun/websocket.ts:76-104]