ClientNats

August 18, 2026

ClientNats

Kind: Class

Source: `packages/microservices/client/client-nats.ts`

Part of: Microservices

`ClientNats` is a NATS-backed client responsible for establishing and managing a connection to a NATS broker. It initializes serialization, publishes messages and events, manages subscriptions/status updates, and exposes access to the underlying NATS client through `unwrap()`.

Extends: `ClientProxy`

Methods

Method	Signature	Returns
`close`	`close()`	`void`
`connect`	`connect()`	`Promise<any>`
`createClient`	`createClient()`	`Promise<Client>`
`handleStatusUpdates`	`handleStatusUpdates(client: Client)`	`void`
`on`	`on(event: EventKey, callback: EventCallback)`	`void`
`unwrap`	`unwrap()`	`T`
`createSubscriptionHandler`	`createSubscriptionHandler(packet: ReadPacket & PacketId, callback: (packet: WritePacket) => any)`	`void`
`publish`	`publish(partialPacket: ReadPacket, callback: (packet: WritePacket) => any)`	`() => void`
`dispatchEvent`	`dispatchEvent(packet: ReadPacket)`	`Promise<any>`

Method	Signature	Returns
`initializeSerializer`	`initializeSerializer(options: NatsOptions['options'])`	`void`
`initializeDeserializer`	`initializeDeserializer(options: NatsOptions['options'])`	`void`
`mergeHeaders`	`mergeHeaders(requestHeaders: THeaders)`	`void`

Properties

Property	Type
`logger`	`any`
`natsClient`	`Client
`connectionPromise`	`Promise
`statusEventEmitter`	`any`

Where it refuses work

- `ClientNats` stops the work with `Error` when `!this.natsClient` — “Not initialized. Please call the "connect" method first.”.
- `ClientNats` stops the work with an early return when `this.connectionPromise`.
- `ClientNats` stops the work with an early return when `error`.
- `ClientNats` stops the work with an early return when `rawPacket?.length === 0`.
- `ClientNats` stops the work with an early return when `message.id && message.id !== packet.id`.
- `ClientNats` stops the work with an early return when `isDisposed || err`.

When something fails

- `ClientNats` handles failure in 2 places: it logs it and continues in 1, and turns it into a return value in 1.

Diagram

```
graph LR
  App[Application Code] --> ClientNats[ClientNats]
  ClientNats --> Serializer[Serializer Initialization]
  ClientNats --> Connection[createClient / connect]
```

```
  Connection --> NATS[NATS Broker]
  ClientNats --> Publish[publish / dispatchEvent]
  ClientNats --> Subscribe[createSubscriptionHandler / on]
  Connection --> Status[handleStatusUpdates]
  ClientNats --> RawClient[unwrap]
```

Usage

```
import { ClientNats } from './client-nats';

const client = new ClientNats({
  servers: ['nats://localhost:4222'],
});

async function start() {
  await client.connect();

  // Register a handler for messages received on a subject.
  client.on('orders.created', async (message) => {
    console.log('Received order event:', message);
  });

  // Dispatch an event to NATS.
  await client.dispatchEvent('orders.created', {
    orderId: 'order-123',
    customerId: 'customer-456',
  });

  // Access the underlying NATS client when lower-level APIs are needed.
  const natsClient = client.unwrap();
  console.log('Connected to NATS:', !!natsClient);
}

start().catch(console.error);

// Close the connection during application shutdown.
// await client.close();
```

AI Coding Instructions

- Call `connect()` before publishing events, registering subscriptions, or accessing the underlying client with `unwrap()`.
- Keep message payloads compatible with the configured serializer; initialize or preserve serializer behavior when extending the client.
- Use `dispatchEvent()` for event-oriented publishing instead of bypassing the client with raw NATS calls unless lower-level functionality is required.

- Ensure `close()` is called during application shutdown to release subscriptions and NATS connection resources.
- Preserve status-update handling when changing connection logic so reconnects and broker state changes remain observable.

Relationships

- IMPORTS → `Logger`
- IMPORTS → `loadPackage`
- IMPORTS → `isObject`