

Sending Structured Data

Some client libraries provide helpers to send structured data while others depend on the application to perform any encoding and decoding and just take byte arrays for sending. The following example shows how to send JSON but this could easily be altered to send a protocol buffer, YAML or some other format. JSON is a text format so we also have to encode the string in most languages to bytes. We are using UTF-8, the JSON standard encoding.

Take a simple *stock ticker* that sends the symbol and price of each stock:

Go

```
nc, err := nats.Connect("demo.nats.io")
if err != nil {
    log.Fatal(err)
}
defer nc.Close()

ec, err := nats.NewEncodedConn(nc, nats.JSON_ENCODER)
if err != nil {
    log.Fatal(err)
}
defer ec.Close()

// Define the object
type stock struct {
    Symbol string
    Price  int
}

// Publish the message
if err := ec.Publish("updates", &stock{Symbol: "GOOG", Price: 1200}); err != nil {
    log.Fatal(err)
}
```

Java

```

class StockForJsonPub {
    public String symbol;
    public float price;
}

public class PublishJSON {
    public static void main(String[] args) {
        try {
            Connection nc = Nats.connect("nats://demo.nats.io:4222");

            // Create the data object
            StockForJsonPub stk = new StockForJsonPub();
            stk.symbol="GOOG";
            stk.price=1200;

            // use Gson to encode the object to JSON
            GsonBuilder builder = new GsonBuilder();
            Gson gson = builder.create();
            String json = gson.toJson(stk);

            // Publish the message
            nc.publish("updates", json.getBytes(StandardCharsets.UTF_8));

            // Make sure the message goes through before we close
            nc.flush(Duration.ZERO);
            nc.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

JavaScript

```
nc.publish("updates", JSON.stringify({ ticker: "GOOG", price: 2868.87 }));
```

Python

```
nc = NATS()
```

```
await nc.connect(servers=["nats://demo.nats.io:4222"])

await nc.publish("updates", json.dumps({"symbol": "GOOG", "price": 1200 }).encode())
```

C#

```
// dotnet add package NATS.Net
using NATS.Net;

await using var client = new NatsClient();

using var cts = new CancellationTokenSource();

Task process = Task.Run(async () =>
{
    // Let's deserialize the message as a UTF-8 string to see
    // the published serialized output in the console
    await foreach (var msg in client.SubscribeAsync<string>("updates", cancellationTok
cts.Token))
    {
        Console.WriteLine($"Received: {msg.Data}");
    }
});

// Wait for the subscription task to be ready
await Task.Delay(1000);

var stock = new Stock { Symbol = "MSFT", Price = 123.45 };

// The default serializer uses System.Text.Json to serialize the object
await client.PublishAsync<Stock>("updates", stock);

// Define the object
public record Stock {
    public string Symbol { get; set; }
    public double Price { get; set; }
}

// Output:
// Received: {"Symbol":"MSFT","Price":123.45}
```

Ruby

```
require 'nats/client'
require 'json'

NATS.start(servers:["nats://127.0.0.1:4222"]) do |nc|
  nc.publish("updates", {"symbol": "GOOG", "price": 1200}.to_json)
end
```

C

```
// Structured data is not configurable in C NATS Client.
```

```
### #1
[] cocobook-sync [] 2026-10-28 11:12:54 UTC
[] cocobook-sync [] 2026-10-28 11:12:54 UTC
```