

Authenticating with a Token



- [Connecting with a Token](#)
 - [Connecting with a Token in the URL](#)
-

Tokens are basically random strings, much like a password, and can provide a simple authentication mechanism in some situations. However, tokens are only as safe as they are secret so other authentication schemes can provide more security in large installations. It is highly recommended to use one of the other NATS authentication mechanisms.

For this example, start the server using:

```
nats-server --auth mytoken
```

The code uses localhost:4222 so that you can start the server on your machine to try them out.

Connecting with a Token

Go

```
// Set a token
nc, err := nats.Connect("127.0.0.1", nats.Name("API Token Example"), nats.Token("mytoken"))
if err != nil {
    log.Fatal(err)
}
defer nc.Close()

// Do something with the connection
```

Java

```
Options options = new Options.Builder()
    .server("nats://demo.nats.io:4222")
    .token("mytoken") // Set a token
```

```
        .build();
Connection nc = Nats.connect(options);

// Do something with the connection

nc.close();
```

JavaScript

```
const nc = await connect({
  port: ns.port,
  token: "aToK3n",
});
```

Python

```
nc = NATS()

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

# Do something with the connection.
```

C#

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

await using var client = new NatsClient(new NatsOpts
{
    Url = "127.0.0.1",
    Name = "API Token Example",
    AuthOpts = new NatsAuthOpts
    {
        Token = "mytoken"
    }
});
```

Ruby

```
NATS.start(token: "mytoken") do |nc|
  puts "Connected using token"
end
```

C

```
natsConnection      *conn      = NULL;
natsOptions          *opts      = NULL;
natsStatus           s          = NATS_OK;

s = natsOptions_Create(&opts);
if (s == NATS_OK)
    s = natsOptions_SetToken(opts, "mytoken");
if (s == NATS_OK)
    s = natsConnection_Connect(&conn, opts);

(...)

// Destroy objects that were created
natsConnection_Destroy(conn);
natsOptions_Destroy(opts);
```

Connecting with a Token in the URL

Some client libraries will allow you to pass the token as part of the server URL using the form:

```
nc nats://token@server:port
```

Again, once you construct this URL you can connect as if this was a normal URL.

Go

```
// Token in URL
nc, err := nats.Connect("mytoken@localhost")
if err != nil {
```

```
        log.Fatal(err)
    }
    defer nc.Close()

    // Do something with the connection
```

Java

```
Connection nc = Nats.connect("nats://mytoken@localhost:4222");//Token in URL

// Do something with the connection

nc.close();
```

JavaScript

```
// JavaScript doesn't support tokens in urls use the `token` option
```

Python

```
nc = NATS()

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

# Do something with the connection.
```

C#

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

await using var client = new NatsClient(new NatsOpts
{
    // .NET client doesn't support tokens in URLs
    // use Token option instead.
    AuthOpts = new NatsAuthOpts
    {
        Token = "mytoken"
    }
});
```

Ruby

```
NATS.start("mytoken@127.0.0.1:4222") do |nc|
  puts "Connected using token!"
end
```

C

```
natsConnection      *conn      = NULL;
natsOptions          *opts      = NULL;
natsStatus           s          = NATS_OK;

s = natsOptions_Create(&opts);
if (s == NATS_OK)
    s = natsOptions_SetURL(opts, "nats://mytoken@127.0.0.1:4222");
if (s == NATS_OK)
    s = natsConnection_Connect(&conn, opts);

(...)

// Destroy objects that were created
natsConnection_Destroy(conn);
natsOptions_Destroy(opts);
```

📄 #1

📄 cocobook-sync 📄 2022-01-19 01:57:33 UTC

📄 cocobook-sync 📄 2024-10-28 14:13:14 UTC