

Authenticating with a Credentials File

The 2.0 version of NATS server introduced the idea of decentralized authentication based on [JSON Web Tokens \(JWT\)](#). Clients interact with this new scheme using a [user JWT](#) and corresponding [NKey](#) private key. To help make connecting with a JWT easier, the client libraries support the concept of a credentials file. This file contains both the private key and the JWT and can be generated with the `nsc` [tool](#). The contents will look like the following and should be protected because it contains a private key. This credentials file is unused and only for example purposes.

```
-----BEGIN NATS USER JWT-----
eyJ0eXAiOiJqd3QlLCJhbGciOiJlZDI1NTE5In0.eyJqdGkiOiJUVlNNTTEtTWkJBN01VWDNYQUxNUVQzTjRISUw1UkZGQU
9YNUtaUFhEU0oyWlAzNkVMNVJBImwiaWF0IjoxNTU4MDQ1NTYyLCJpc3MiOiJBQlZTQk0zVTQ1REdZRVVFQ0tYUVM3QkV0
SFdHN0tGUUVVEU1RFSEFKQVNPULBWW0JaNEhPSUtDSCIsIm5hbWUiOiJvbnVnYSIsInN1YiI6IlVEWEIyVk1MWFBBU0FKN1
pEVEtZTlE3UU9DRldTR0I0Rk9NWVFRMjVIUVdTQUY3WlFKRUJTVUNXIiwidHlwZSI6InVzZXIiLCJuYXRzIjp7InB1YiI6
e30sInN1YiI6e319fQ.6TQ2ilCdb6m2ZDiJuj_D_0ePGXFyN3Ap2DEm3ipcU5AhrWrNvneJryWrpgi_yuVWKo1UoD5s8bx
1mwypWVGFAA
-----END NATS USER JWT-----

***** IMPORTANT *****

NKEY Seed printed below can be used to sign and prove identity.
NKEYs are sensitive and should be treated as secrets.

-----BEGIN USER NKEY SEED-----
SUA0Y5JZ2WJKVR4U02KJ2P3SW6FZFNWE0IMAXF4WZEUNVQXXU0KGM55CYE
-----END USER NKEY SEED-----

*****
```

Given a creds file, a client can authenticate as a specific user belonging to a specific account:

Go

```
nc, err := nats.Connect("127.0.0.1", nats.UserCredentials("path_to_creds_file"))
if err != nil {
```

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

// Do something with the connection
```

Java

```
Options options = new Options.Builder()
    .server("nats://localhost:4222")
    .authHandler(Nats.credentials("path_to_creds_file"))
    .build();
Connection nc = Nats.connect(options);

// Do something with the connection

nc.close();
```

JavaScript

```
// credentials file contains the JWT and the secret signing key
const authenticator = credsAuthenticator(creds);
const nc = await connect({
  port: ns.port,
  authenticator: authenticator,
});
```

Python

```
nc = NATS()

async def error_cb(e):
    print("Error:", e)

await nc.connect("nats://localhost:4222",
                user_credentials="path_to_creds_file",
                error_cb=error_cb,
                )

# Do something with the connection
```

```
await nc.close()
```

C#

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

await using var client = new NatsClient("127.0.0.1", credsFile: "/path/to/file.creds");
```

C

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

s = natsOptions_Create(&opts);
if (s == NATS_OK)
    // Pass the credential file this way if the file contains both user JWT and seed.
    // Otherwise, if the content is split, the first file is the user JWT, the second
    // contains the seed.
    s = natsOptions_SetUserCredentialsFromFiles(opts, "path_to_creds_file", NULL);
if (s == NATS_OK)
    s = natsConnection_Connect(&conn, opts);

(...)

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

📄 #3

📄 cocobook-sync 📄 2026-10-28 11:11:11 UTC
📄 cocobook-sync 📄 2026-10-29 01:42:36 UTC