



role Auth::SCRAM::Client

Client side authentication using SCRAM

Table of Contents

- 1 [Synopsis](#)
- 2 [Read and writable attributes](#)
- 2.1 [client nonce](#)
- 2.2 [extension data](#)
- 3 [Methods](#)
- 3.1 [init](#)
- 3.2 [start-scram](#)

```
unit package Auth;  
class SCRAM::Client { ... }
```

Synopsis

```
# User defined class with the task of communicating with a  
# server for the authenticating process  
class MyClient {  
  
    submethod BUILD {  
        # Establish server connection  
    }  
  
    method client-first ( Str:D $client-first-message --> Str ) {  
        # Send $client-first-message to server and return server  
        # response as server first message  
    }  
  
    method client-final ( Str:D $client-final-message --> Str ) {  
        # Send $client-final-message to server and return server  
        # response as server final message  
    }  
  
    method error ( Str:D $message --> Str ) {  
        # Errors? nah ... (Famous last words!)  
    }  
}
```

```
# Initialize SCRAM with above class
my Auth::SCRAM $sc .= new(
  :username<user>,
  :password<pencil>,
  :client-object(MyClient.new),
);

my Str $error = $sc.start-scam;
```

Read and writable attributes

These attributes must be set before scram authentication is started.

client nonce

Defined as

```
has Int $.c-nonce-size is rw = 24;
has Str $.c-nonce is rw;
```

Define a nonce. The result must be a hexadecimal string of the proper size generated by a base64 encoding operation. When not set, the class will makeup one of the default length of 24 octets and encode in base64.

```
#!/c-nonce = encode-base64(
  Buf.new((for ^#!/c-nonce-size { (rand * 256).Int })),
  :str
);
```

extension data

Defined as

```
has Str $.reserved-mext is rw;
has Hash $.extensions is rw = %();
```

These variables are not yet used in this module.

Methods

init

Defined as

```
method init (
  Str:D :$username!, Str:D :$password!, Str :$authzid,
  :$client-object!
)
```

Initialize the process. Method is called from BUILD in Auth::SCRAM. User should not call this function!

start-scram

Defined as

```
method start-scram( --> Str ) {
```

Start authentication. An error message is returned when an error is encountered. When successful, it returns an empty string ("). \$client-first-message must be defined when a server object is provided to the new() method.

The calls to the user provided client object are as follows;

- **client-first.** This method must return the servers first message. Definition is

```
method client-first ( Str $client-first-message --> Str )
```

- **mangle-password.** Optional method to do some mangling stuff on the given arguments. The username and password are normalized before calling. Definition is

```
method mangle-password (
  Str :$username, Str :$password, Str :$authzid,
  Auth::SCRAM :$scram-obj
  --> Buf
)
```

When not defined, the following action is done

```
my Buf $mangled-password = Buf.new($password.encode);
```