role Auth::SCRAM::Client

Client side authentication using SCRAM

Table of Contents

```
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-scram;
```

Read and writable attributes

These attributes must be set before scram authentication is started.

client nonce

```
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
);
```

Normally it will not be needed to make one yourself. It was made available for testing purposes.

extension data

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

These variables are not yet used in this module.

Methods

start-scram

```
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. The method must be declared like this:

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

Its purpose is to send the provided $client-first-message to the server and to return the servers answer which is the servers first message.

- **mangle-password**. This method is optional. Some servers, like mongod, need extra manipulations to mangle the data. The username and password are normalized before calling. The method must be declared like:

```
method mangle-password (
  Str :$username, Str :$password, Str :$authzid
  --> Buf
)
```

When not defined, the following action is done instead

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

- **client-final**. The purpose of this method is to send the provided client final message to the server. The server returns a verification of its own identity after accepting the clients message. This message must be returned to the caller. Declare it as follows;

```
method client-final ( Str:D $client-final-message --> Str )
```

- **error**. The error method is called with a message whenever there is something wrong. This gives the object the opportunity to do last resort operations like closing ports etc. The procedure is then terminated after returning from the error method and the procedure will return the same message to the caller of start-scram.

```
method error ( Str:D $message )
```

- **cleanup**. This optional method is called when all ends successfully. After returning from this method it returns an empty string('') to the caller of start-scram

```
method cleanup()
```