# Protocol

## Introduction

Building a protocol that will allow you to build a message saving service. Allow the client and serves to communicate with each other

The TMP (Twitter Message Protocol) will provide the user with a message saving service. The user will be able to:

* Log on to access the messages on the server
* Upload messages to the server
* Download and display all the user’s messages on the server
* Log off the server

The protocol will allow the client to communicate with the server to:

* Check if the user is in the server
* To upload messages to the server
* To download messages from the server

## Log On To Server

**Message Format:**

* Message: Logon
* Description: The user can log into the server with a username and password
* Code/Number: 100
* Message Parameters: username, password (both string)
* Response Message
  + Code: 110
  + Text: login successful
* Response Message
  + Code: 120
  + Text: login unsuccessful

**Message Functionality:**

Implementing the Connection message on the server as a function called login()

* save the username and password
* is a folder ! exists for username create it
* return 110: login successful
* if can’t create folder return 120

**Pseudocode**:

**Sequence diagram:**

Server

Client

Login 100 name password

110 login successful

120 login unsuccessful

## Upload

**Message Format:**

* Message: Upload
* Description: The user can upload a string message of max 160 characters to the server
* Code/Number: 200
* Message Parameters: message (string)
* Response Message
  + Code: 210
  + Text: upload successfully
* Response Message
  + Code: 220
  + Text: upload unsuccessfully

**Message Functionality:**

Implementing the Connection message on the server as a function called upload()

* save the message to the server
* return 210: message successfully uploaded
* if error occurs when uploading return 220

**Pseudocode**:

**Sequence diagram**

Server

Client

Upload 200 message

210 upload successful

220 upload unsuccessful

## Download

**Message Format:**

* Message: Download
* Description: The user can download and display all their messages on the server
* Code/Number: 300
* Message Parameters: none
* Response Message
  + Code: 310
  + Text: download successful
* Response Message
  + Code: 320
  + Text: download unsuccessful

**Message Functionality:**

Implementing the Connection message on the server as a function called download()

* Download all messages on the server for the user
* Display all the messages to the user
* return 310: download successful
* if an error occurs when downloading the messages return 320

**Pseudocode**:

**Sequence diagram**

Server

Client

Download 300

310 download successful

320 download unsuccessful

## Log Off Of Server

**Message Format:**

* Message: Logoff
* Description: The user can log off the server
* Code/Number: 400
* Message Parameters: none
* Response Message
  + Code: 410
  + Text: logoff successful
* Response Message
  + Code: 420
  + Text: logoff unsuccessful

**Message Functionality:**

Implementing the Connection message on the server as a function called logoff()

* The user should be logged off the server
* return 410: logoff successful
* if error occurs when logging off the server return 420

**Pseudocode**:

**Sequence diagram**

Server

Client

Logoff 400

410 logoff successful

420 logoff unsuccessful