RFC: 732

­­

[Protocol Name]

[Date]

Table of Contents

[**1. Introduction** 2](#_Toc161146659)

[**2. Protocol Objective** 3](#_Toc161146660)

[**3.** **Protocol Overview** 3](#_Toc161146663)

[**4. Protocol Specification** 3](#_Toc161146666)

[**4.1. Service Location** 3](#_Toc161146667)

[**4.2. Sequence of Inter-process Communication** 3](#_Toc161146668)

[**4.3. Representation and Interpretation of Data Exchanged** 4](#_Toc161146669)

[**4.4.** **Description of Message Formats** 4](#_Toc161146677)

[**4.5.** **Error Handling** 9](#_Toc161146678)

[**4.5.1.** **Handling Authorisation Errors** 9](#_Toc161146679)

[**4.6.** **Service Session Management** 10](#_Toc161146680)

[**5.** **Implementation of Functions** 10](#_Toc161146681)

[**5.1.** **Log On** 10](#_Toc161146687)

[**5.2.** **Upload Request** 10](#_Toc161146691)

[**5.3.** **Download Request** 11](#_Toc161146692)

[**5.4.** **Download All Requests** 11](#_Toc161146693)

[**5.5.** **Log Off** 11](#_Toc161146694)

[**6.** **Sequence Diagram** 11](#_Toc161146695)

[**7.** **Security Considerations** 11](#_Toc161146696)

[**8.** **Glossary** 12](#_Toc161146700)

[**9.** **Author** 12](#_Toc161146701)

[**10.** **References** 12](#_Toc161146702)

# **1. Introduction**

The protocol is intended to provide secure and efficient interactions between users (clients) and the server.

This document describes the [Protocol Name], a protocol designed for facilitating communication between clients and a server concurrently and securely.

# **2. Protocol Objective**

The objectives of the [Protocol Name] protocol are to enable users to log on to the server, upload and download messages, log off securely, and quit the application.

The protocol ensures concurrency, confidentiality, and reliability of data exchanged between clients and servers.

(Information Sciences Institute, University of Southern California, 1981)



# **Protocol Overview**

[Protocol Name] is a simple client-server protocol based on a request-response model. It supports secure communication using SSL/TLS encryption and concurrency using multithreading over TCP/IP.



# **4. Protocol Specification**

## **4.1. Service Location**

Clients connect to the server's IP address and port number.

Communication is secured using SSL/TLS encryption.

## **4.2. Sequence of Inter-process Communication**

* The client initiates a secure connection request to the server.
* The server listens for incoming client connections over SSL/TLS.
* Upon successful connection establishment, the client and server perform an SSL/TLS handshake to authenticate and establish a secure communication channel.
* Once the handshake is completed, the server responds to the client’s connection request.
* The client acknowledges the server's response over the secure connection.
* Communication continues bidirectionally over the established SSL/TLS connection until session termination. Session termination can occur due to termination by either party.

## **4.3. Representation and Interpretation of Data Exchanged**

Requests and responses follow a predefined format agreed upon by both client and server.

Plain text data format is used for encoding game commands(requests), responses, and other data over the secure connection.



## **Description of Message Formats**

The message formats are in plain text, making them easy to interpret by both the client and server. Each message includes a " Message type" field to indicate the action requested, along with any other necessary parameters. The server processes these messages according to the protocol and responds to the client accordingly, responses are also in string format.

* + 1. **Login Request Message Format**

**Description:** The user can login to the server with a username and password.

The login request message format includes the following elements:

**Parameters:**

Message type: LOGIN

Username: [username]

Password: [password]

**Example:**

+---------------------+

| **LOGIN user password** |

+---------------------+

**Response Message OK:**

Code: 101 - Success

Text: “Login successful! Welcome to [ protocol name ]”

**+-----------------------------------+**

**| 101 Login successful! |**

**| Welcome to [ protocol name ] |**

**+-----------------------------------+**

**Response Message Error:**

Code: 102 - Unauthenticated

Text: “Login unsuccessful. Please try again. Check logs for details”

**+-----------------------------+**

**| 102 Login unsuccessful |**

**| Please try again |**

**| Check logs for details |**

**+-----------------------------+**

**Response Message Error:**

Code: 103 – Invalid format

Text: “Invalid credentials format. Must be ‘username password’”

**+-----------------------------------+**

**| 103 Invalid credentials format |**

**| Please try again |**

**| Check logs for details |**

**+-----------------------------------+**

* + 1. **Upload Request Message Format**

**Description**: The user can upload a message to the server once authenticated.

The upload request message format includes the following elements:

**Parameters:**

Message type: UPLOAD

Message: [ message content ]

**Example:**

**+--------------------------+**

**| UPLOAD This is a message |**

**+--------------------------+**

**Response Message OK:**

Code: 201 - Success

Text: “Upload successful. Message ID: [id]”

**+-----------------------+**

**| 201 Upload successful |**

**| Message Id: [id] |**

**+-----------------------+**

**Response Message Error:**

Code: 202 – Empty message

Text: “Upload unsuccessful. Attempted to upload an empty message.”

**+-----------------------------+**

**| 202 Upload unsuccessful |**

**| Attempted to upload |**

**| null message |**

**+-----------------------------+**

* + 1. **Download Request Message Format**

**Description**: The user can download a specific message from the server once authenticated.

The download request message format includes the following elements:

**Parameters:**

Message type: Download

Message ID: [ ID of message to download ]

**+-----------------+**

**| DOWNLOAD 12345 |**

**+-----------------+**

**Response Message OK:**

Code: 301 - Success

Text: “Download successful. Message ID: [id] [message]”

**+-----------------------------+**

**| 301 Download successful |**

**| Message ID: [id] |**

**[message]**

**+-----------------------------+**

**Response Message Error:**

Code: 302 – No message with that id

Text: “Download unsuccessful. Message id not found”

**+-----------------------------+**

**| 302 Download unsuccessful |**

**| Message id not found |**

**+-----------------------------+**

* + 1. **Download All Request Message Format**

**Description**: The user can download all available message from the server once authenticated.

The download all request message format includes the following elements:

**Parameters:**

Message type: Download\_all

**+-----------------+**

**| DOWNLOAD\_ALL |**

**+-----------------+**

**Response Message OK:**

Code: 401 - Success

Text: “Download of all messages successful. [messages]”

**+----------------------------+**

**| 401 Download of all |**

**| messages successful |**

**[messages]**

**+----------------------------+**

**Response Message Error:**

Code: 402 – HashMap empty

Text: “No messages available”

**+-----------------------------+**

**| 402 No messages available |**

**+-----------------------------+**

* + 1. **Logout Message Format**

**Description**: The user can log off from the server.

The logout request message format includes the following elements:

**Parameters:**

Message type: Logout

**+-----------+**

**| LOGOUT |**

**+-----------+**

**Response Message OK:**

Code: 501 - Success

Text: “Logout successful, see you again soon”

**+-----------------------------+**

**| 501 Logout successful |**

**| See you again soon |**

**+-----------------------------+**

* + 1. **Quit Message Format**

**Description**: The user can quit the program.

The quit request message format includes the following elements:

**Parameters:**

Message type: Quit

**+-----------+**

**| QUIT |**

**+-----------+**

**Response Message OK:**

Code: 502 -Success

Text: “Logout successful, see you again soon”

**+----------------------------------+**

**| 502 Program quit successfully |**

**| Goodbye |**

**+----------------------------------+**

## **Error Handling**

Error codes and messages above are defined for common error scenarios.

Both client and server handle exceptions and errors appropriately over the secure connection and log all error messages appropriately.

## **Service Session Management**

Upon connection establishment, a secure service session is initiated.

The server assigns a unique session identifier to each client session securely.

Session management includes maintaining session state and uploaded messages securely over the SSL/TLS connection.

Session clean-up is performed securely upon session termination.

# **Implementation of Functions**

The server implements functions for handling login, upload, download, download all messages, logout, and quit requests from clients.



## **Log On**

**login(String credentials)**

* If login successful return success message (101)
* If login !successful return error message (102)
* If credential format incorrect return error message (103)
* If exception log message



## **Upload Request**

**upload(String message)**

* Attempt to store message and message id to previously initialised HashMap
* If stored successfully return success message (201)
* If message empty return error message (202)
* If exception log message

## **Download Request**

**download(int id)**

* Get value from HashMap using id
* If successful return success message (301)
* If !successful return error (302)

## **Download All Requests**

**downloadAll ()**

* Get all messages from HashMap
* If successful return success message (401)
* If ! successful return error (402)

## **Log Off**

**logout()**

* Perform logout operations
* If successful return success message (501)
* If exception log message

## **Quit**

**quit()**

* Perform quit operations
* If successful return success message (502)
* If exception log message

# **Sequence Diagram**

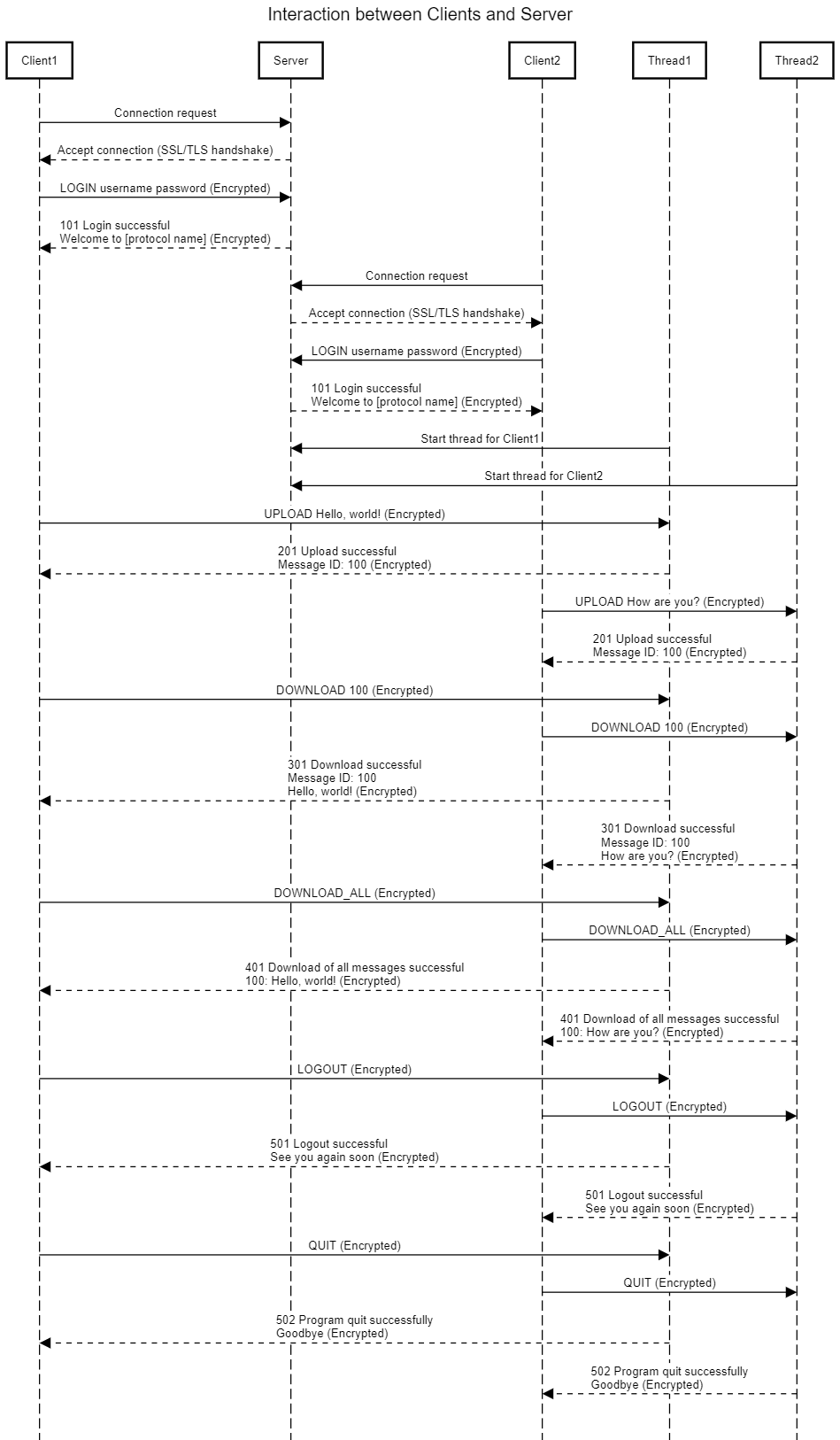


Fig 1 Created using sequencediagram.org

# **Security Considerations**

Communication between the client and server is encrypted using SSL/TLS, providing confidentiality and integrity for communication between server and client by encrypting the data being exchanged. SSL/TLS also enables mutual authentication via the handshake process. During this process, both parties verify each other’s identity, and a secure communication channel is established.



# **Author**

Deirdre Lee

Munster Technological University

Dept. of Computer Science

Distributed Computing

Kerry

# **References**

Information Sciences Institute, University of Southern California, 1981. *TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION.* [Online]   
Available at: https://datatracker.ietf.org/doc/html/rfc793  
[Accessed 4 March 2024].

Sengul, C. & Kirby, A., 2023. *Message Queuing Telemetry Transport (MQTT) and Transport Layer Security (TLS) Profile of Authentication and Authorization for Constrained Environments (ACE) Framework.* [Online]   
Available at: https://datatracker.ietf.org/doc/html/rfc9431  
[Accessed 4 March 2024].