Simple Transfer and Exchange Protocol (STEP) v1.0

1. Introduction

The Simple Transfer and Exchange Protocol (STEP) is a stateless, application-layer, request/response protocol that defines a simple way to exchange data and file.

STEP is a TCP-based protocol, which guarantees reliable data transmission. JSON (JavaScript Object Notation) is exploited as the unique data representation format, which is simple and human-readable.

The STEP servers provide services for data and file uploading, storage, downloading and deleting. The corresponding clients can firstly get authorization and operate public or private data and file on the server.

2. Terminology and Core Concepts

a) Connections, Clients and Servers

STEP is a C/S protocol that operates over a reliable transport layer (TCP) connection [1, 2]. The port that the server listens on must be 1379. Therefore, both IP address of the server and the port (1379) must be used for the connection. The connection keeps alive until the client closes the connection.

A STEP client is a program that establishes a connection to a server for the purpose of sending one or more continuous STEP requests, while a STEP server is a program that accepts connections to serve STEP requests by sending responses.

b) Messages

The messages are exchanged across a TCP connection. The client sends requests message with “operation”, “direction” and other compulsory fields, while the server responds to a request by sending a response message with “operation” and “direction” too. The response message and request message share the same format.

The details are described in Section 3.

3. Message

In order to simplify the definition of STEP messages, the request and response messages share the same format and definition. Some reserved fields are used to indicate the communication direction.

a) Format

The message consists of three parts: length of JSON data part (binary, compulsory), length of Binary data part (binary, compulsory), JSON data (text, compulsory) and file part (binary, optional).

The first 4 bytes (N) are used for representing the length of the JSON data, which means the maximum

length of the following JSON data can be 2^32.

The following 4 bytes (M) are used for representing the length of the Binary data, which means the maximum length of the following Binary data can be 2^32.

The following N bytes are used to represent the JSON data. As they are text-based, the data should be decoded and parsed using the rule of JSON.

The binary file data (file / block) will be extended to the end of the message.

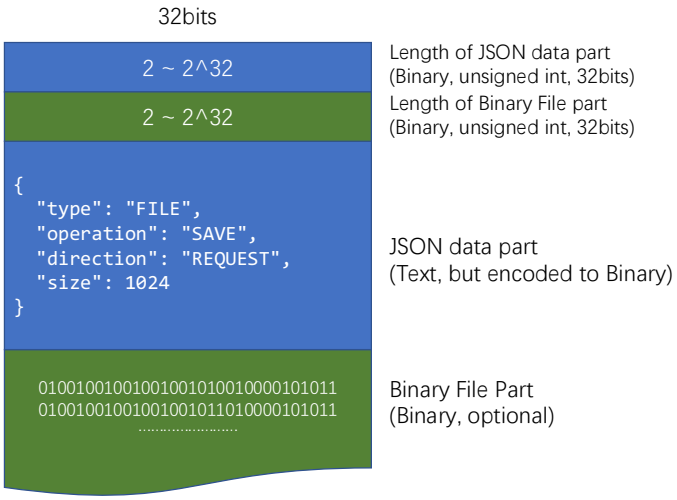

Fig 1. STEP message format

b) Reserved Fields of JSON part

1) type: [FILE, DATA, AUTH]

The type can be FILE and DATA.

2) operation: [SAVE, DELETE, GET, UPLOAD, DOWNLOAD, BYE, LOGIN]

For authorization, the operation can be LOGIN.

The “type” has to be “AUTH”. “username” is your student_id (string type) while the “password” is md5(student_id). The response includes the “token”.

For DATA, the operations can be SAVE, DELETE and GET.

SAVE: save the data. The server will reply with the key. The client can also indicate the key using the reserved field “key”. If not, the server will reply with a random key. The key can be any non-existed string. If the key is used, the server will reply error with a specific code.

GET: get the data using the key.

Delete: delete the data using the key

For FILE, the operations can be SAVE, DELETE, GET, UPLOAD and DOWNLOAD.

SAVE: start a file storage request with the “key” and “size” of the file. The key could be defined by the client. If not, the server will reply with a random key. The server will reply with the file upload plan, which includes the “key”, “block_size” and “total_block”. For a file, the key can be the filename with the extensive name. E.g., foo_bar.jpg. The key can be any non-existed string. If the key is used, the server will reply error with a specific code.

DELETE: delete the file by the key.

GET: start a file download request by key. The server will reply with the file download plan, which includes the “key”, “size” (total file size), “block_size”, “total_block” and “md5” of the file.

UPLOAD: upload a file block or the whole file using the “key”, and “block_index”. When the server receives the whole file, the “md5” will be replied to the client to check the completeness.

DOWNLOAD: download a file block or the whole file using the “key”, and “block_index”. The binary file block will be replied.

3) direction: [REQUEST, RESPONSE]
4) status: [200, 400+]

   Section 4 will describe the details.
5) md5:

   md5 value of the file.
6) size: 1 ~ 4294967296

   File size or block size …
7) block_index: 0 ~ N-1 (N is the total block number)
8) block_size: 1 ~ 65536
9) total_block:

   Total block number for uploading or downloading
10) key: <string>
11) token: <string>

    The token will be obtained by authorization.

c) Data Fields of JSON part:

   Any other non-reserved fields.

d) Binary Content

   For files or file blocks.

4. Status Codes

a) Successful

200: OK

b) Error

400: Compulsory field is missing.

401: Password error for login.

402: The key is existing.

403: No token or token is wrong.

404: Cannot find the key.

405: The block_index is over the maximum block number.

406: The size for uploading does not match the required block_size.

407: Wrong direction.

408: Wrong operation.

409: Type is not allowed.

410: Field is missing.

Reference

[1] Nottingham, M. "HTTP Semantics." (2022).

[2] Fielding, Roy, et al. "RFC2616: Hypertext Transfer Protocol--HTTP/1.1." (1999).