VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IMAP Client – Manual

19th October, 2024 Tomáš Dolák

Contents

1 Assignment

The project assignment in the ISA (Network Applications and Network Administration) subject was to create IMAP4rev1 (according to RFC3501), which will be able to communicate only over TCP/IP as well as using SLL/TLS - IMAPS. The following program downloads emails from the defined server in the argument of the program call and saves them in the output directory. If the path to the output repository specified by the argument does not exist, the program creates it on this path.

2 Implementation Details

The principles of object-oriented programming have been applied to the program code. The program is divided into logical classes such as the `ClientConfig` class - which takes care of the configuration of the resulting IMAP client based on the input arguments of the program, which it also processes. On the result of the configuration, either an instance of the `NonSecureImapClient` class is created, which mediates communication, classically only over TCP/IP, or an instance of the `SecureImapClient` class, which also uses SSL/TLS in addition to TCP/IP. Both classes `NonSecureImapClient` and `SecureImapClient` inherit basic properties from `BaseImapClient`, which provides features that are common to both derived classes, such as generating a TAG, finding the value of the current TAG or translating a hostname to an IPv4 address.

2.1 Implementation of Client

The NonSecureImapClient and SecureClient classes are characterized by their very similar behavior, at the beginning when the class is instantiated they receive information like `MailBox`, `OutputDirectory`, `HeadersOnly` and `NewOnly` as input parameters. These parameters are used to define further behavior of the program and their description is given in table below.

Parameter	Description
MailBox	Mailbox from which emails will be downloaded
OutputDirectory	Specifies where downloaded emails will be stored
HeadersOnly	Only header of the emails will be downloaded
NewOnly	Only unseen emails will be downloaded

Then, from the user's point of view, the classes only need to call the Run method with the parameters server address, port login and password. This method interacts with the server and its behaviour is as follows.

2.2 Program's Features

The program also contains some of the extra features, the first of which is that the client saves a special `.uidvalidity` file with the value `UIDVALIDITY` when downloading to a specific output directory for the first time, this value is used in case the user wants to repeatedly download files to the same `output directory` and have synchronized mailboxes. On each subsequent run, the `UIDVALIDITY` value is checked to see if it is the same locally (in this `.uidvalidity` file) and on the server, if the values are different, it is a sign that the structure of the mailbox on the server has changed (i.e. the emails have been removed or moved to another mailbox, etc.) and the output directory needs to be purged and the emails downloaded again. This is done to ensure that the locally downloaded emails are always synchronized with those on the server.

The client always needs to have a specific `output directory` to which it will download emails from the IMAP server, it can happen that the user specifies his/her own location, but the folder on the given path does not exist (for example `-o /Desktop/email/my_mailbox/`) in this case the client is able to create the folder on the given path (for example `/Desktop/email/my_mailbox/`) and store the emails in it.

2.3 Program Flow

The behaviour of the program has already been mentioned, this chapter contains a more detailed description and the program flow diagram shows the behaviour of the client in graphical form.

The program parses the program arguments at the beginning - find out how the client should be configured. Next, a client is created according to the request, which establishes a connection to the IMAP server using only `TCP/IP` or a combination of `TCP/IP` with `SSL/TLS`. If the connection is established, it sends a `LOGIN` command to the server, which should log the user in. In case of successful login, the SELECT command selects the `mailbox` from which the user wants to download emails (note: the default is `INBOX`), next, the `UIDVALIDITY` value is verified against the local copy (if the local copy does not exist in the `output directory`, it is created). Then all `UIDs` related to the mailbox are retrieved using the `FETCH` command (the collection of retrieved `UIDs` can be influenced by this parameter if the `'-n'` argument is used). After the set of `UIDs` is obtained, the email download sequence starts, the emails are downloaded one by one and the rest of the communication with the IMAP client is removed. Finally, the email is stored in the output directory specified by the `'-o'` argument and the client repeats this process with the next email with a UID from the set of pending emails.

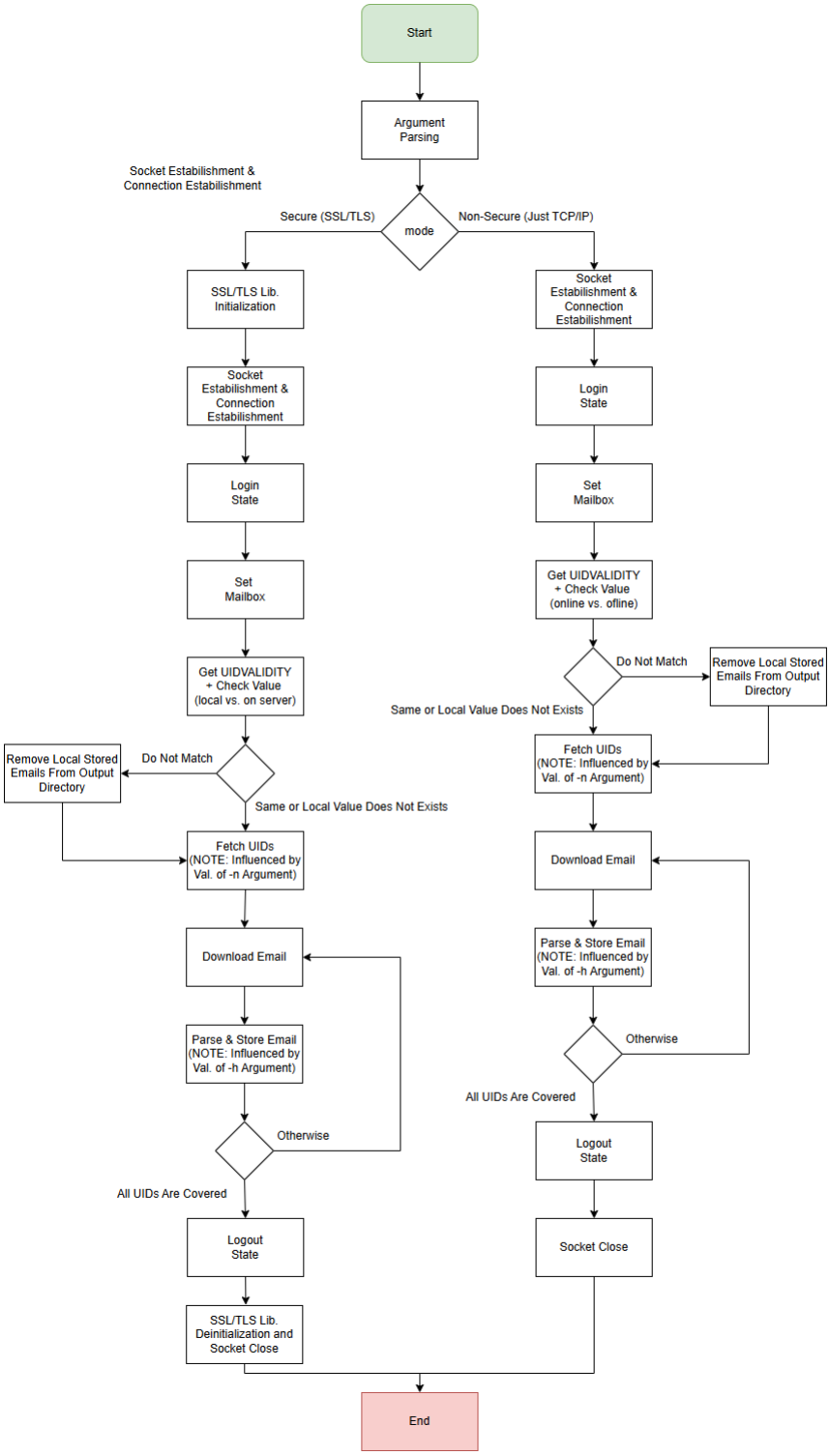

Figure 1: Program Flow Diagram

2.4 Classes

Here Will Be Description.

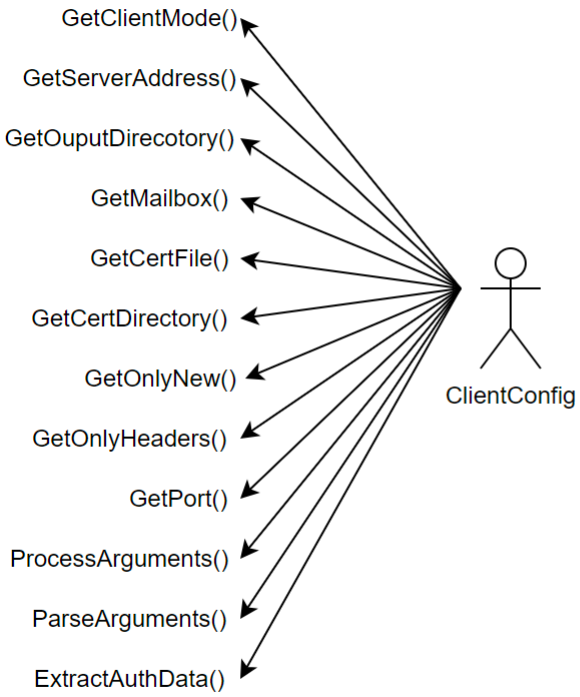

Figure 2: Program Flow Diagram

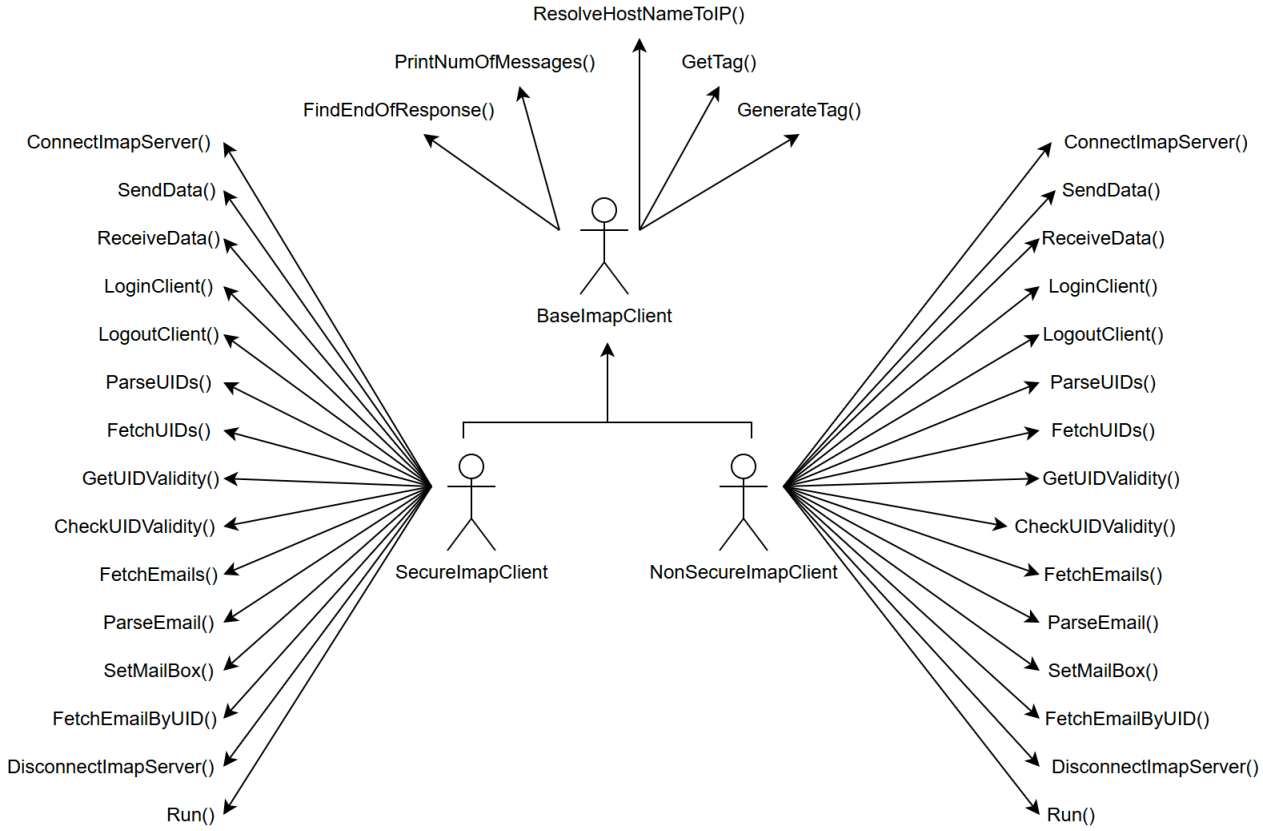

Figure 3: Program Flow Diagram

3 Testing

The `imapcl` program has also been tested. The testing was performed in two ways, the first one was tested using the local implementation of the IMAP server, the second way was chosen to test on foreign IMAP servers provided by the school such as eva or merlin.

3.1 Testing of Local IMAP Servers

To ensure the correct operation of the program, `imapcl` was tested using a local implementation of the IMAP server. The local IMAP server is available in the repository in the tests folder under the name `imap_server.py`. The server is implemented in Python using the `os`, `socket`, `threading`, `subprocess`, `time`, and `signal` libraries and handles all requests from the client implementation such as `LOGIN`, `FETCH` (fetch support involves responding to specific requests such as `'tag' UID 'x' FETCH BODY[HEADER]` or `'tag' UID 'x' FETCH BODY[TEXT]` where `'tag'` is specific tag of the request and `'x'` is UID of the email), `SELECT`, and `LOGOUT`.

3.2 Testing of Foreign IMAP Servers

As a complementary method, testing on foreign IMAP servers provided by the school, such as eva or merlin, was chosen to ensure compatibility outside the local environment and to demonstrate the usability of the program.

Testcase	Server	Expected Output	Program Output
Compilation	eva	Successful compilation.	Successful compilation.
Auth. file parsing	eva	Auth. file successfully parsed	Auth. file successfully parsed.
Calling `./imapcl outlook.office365.com -p 143 -a PATH/auth_file.txt -b INBOX -o PATH/output/`	eva	Downloaded whole emails (including header and body), note: number emails on server was 1739.	Downloaded 1739 from server with headers and bodies.
Calling `./imapcl outlook.office365.com -p 143 -a PATH/auth_file.txt -b INBOX -o PATH/output/` with changed value of `.uidvalidity` file	eva	Downloaded whole emails (including header and body) and updated value in `.uidvalidity` file, note: number emails on server was 1739	Downloaded 1739 from server with headers and bodies. Updated valued of `.uidvalidity` file.
Calling `./imapcl outlook.office365.com -p 143 -a PATH/auth_file.txt -b INBOX -o PATH/output/` and path `PATH/output/` does not exists	eva	Downloaded whole emails (including header and body) and created output directory in `PATH/output/` file, note: number emails on server was 1739	Downloaded 1739 from server with headers and bodies. in output directory `PATH/output/`.
Calling `./imapcl outlook.office365.com -T -a PATH/auth_file.txt -b INBOX -o PATH/output/` with changed value of `.uidvalidity` file	eva	Downloaded whole emails (including header and body) and updated value in `.uidvalidity` file thru SSL/TLS on port 993, note: number emails on server was 1739	Downloaded 1739 from server with headers and bodies. Updated valued of `.uidvalidity` file. Used SSL/TLS on default port 993.
Compilation	merlin	Successful Compilation	Successful Compilation
Auth. file parsing	merlin	Auth. file successfully parsed	Auth. file successfully parsed.

Testcase	Server	Expected Output	Program Output
Calling `./imapcl eva.fit.vutbr.cz -p 143 -a PATH/auth_file.txt -b INBOX -o PATH/output/`	merlin	Downloaded whole emails (including header and body), note: number emails on server was 2349.	Downloaded 2349 from server with headers and bodies.
Calling `./imapcl eva.fit.vutbr.cz -p 143 -h -a PATH/auth_file.txt -b INBOX -o PATH/output/`	merlin	Downloaded email only headers, note: number emails on server was 2349.	Downloaded 2349 email headers from server.
Calling `./imapcl eva.fit.vutbr.cz -p 143 -n -a PATH/auth_file.txt -b INBOX -o PATH/output/`	merlin	Downloaded email only new emails (headers + bodies), note: number emails on server was 56.	Downloaded 56 new emails from server with headers and bodies.
Calling `./imapcl eva.fit.vutbr.cz -p 143 -h -n -a PATH/auth_file.txt -b INBOX -o PATH/output/`	merlin	Downloaded email only new emails (only headers), note: number emails on server was 56.	Downloaded 56 headers of new emails from server.
Calling `./imapcl eva.fit.vutbr.cz -g 143 -h -n -a PATH/auth_file.txt -b INBOX -o PATH/output/`	merlin	Wrong argument `-g`, help on terminal should be printed.	Printed help to the user on terminal.
Calling `./imapcl eva.fit.vutbr.cz -p 143`	merlin	A small number of arguments for calling the program. The error message should be displayed to the user.	Displayed the error message that informs user a small number of arguments for calling the program that were used.

4 Return Codes

The program is designed in such a way that in case of an error or failure, the user is always informed about the situation that has occurred. In case of success the program always returns the return value 0 (respectively `SUCCESS`), in other cases the program gives the return values according to the table below.

Name	Data Type	Value	Description
OUTPUT_DIR_NOT_CREATED	int	-7	Output Directory Could Not Be Created
UIDVALIDITY_FILE_NOT_FOUND	int	-6	.uidvalidity File Not Found
UIDVALIDITY_FILE_ERROR	int	-5	Error With .uidvalidity File (Invalid Format, Out of Range)
CREATE_CONNECTION_FAILED	int	-4	Failed to Create Connection With IMAP Server
SSL_CERT_VERIFICATION_FAILED	int	-3	SSL Certificate Verification Failed
FETCH_EMAIL_FAILED	int	-2	Fetching Email By UID Failed
SUCCESS	int	0	Operation Was Successful
NO_IP_ADDR_FOUND	int	1	No IPv4 Address Was Found
PARSE_ARGUMENTS_FAILED	int	2	Parsing Program's Arguments Failed
PARSE_CREDENTIALS_FAILED	int	3	Parsing Credentials Failed
SERVER_UNKNOWN_RESPONSE	int	4	Server Sent an Unknown Response
TRANSMIT_DATA_FAILED	int	5	Transmission of Data Failed
RECEIVE_DATA_FAILED	int	6	Reception of Data Failed
RESPONSE_NOT_FOUND	int	7	Expected Server's Response Was Not Found
PARSE_BY_REGEX_FAILED	int	8	Parsing of Regular Expression Failed
NON_UIDS_RECEIVED	int	9	No UIDs Were Received From The IMAP Server
CONTINUE_IN_RECEIVING	int	10	Continue Receiving More Data
UNDEFINED_STATE	int	11	Undefined State Encountered
UID_VALIDITY_ERROR_IN_RECV	int	14	Unable to Receive UIDVALIDITY From The Server
REMOVAL_OF_EMAILS_FAILED	int	15	Failed to Remove Emails When UIDVALIDITY Does Not Match
BAD_RESPONSE	string	"Bad Response :("	Error During Receiving of Server's Response