Voting Server- Version 1.0

Protocol Version 1

Table of Contents

1. Introduction

The Voting Server (VS) is a server that allows users to be able vote for a specific item in an active poll. There is a monitor that is able to create polls. A poll is active for a specified amount of time and only registered users or monitors are able to vote in the active poll. To become a register user an anonymous user registers and once the user is registered they become a registered user and can start voting in active polls. Below is an Authorization Matrix that lists which users have access to what commands and the definition of each command.

2. Basic Operation

The server host starts the VS (Voting Server) by listening on an assigned port. When a client wants to connect it establishes a TCP connection with the host. When the connection is established, the VS server sends the version number of the server currently running. From then the client issues commands and the server responds to them. The connection is active until the user issues a QUIT command or the connection is lost.

Messages are sent back and form from the client and server by having the parameter name followed by a semicolon then the data for that parameter followed by a carriage return. A message is terminated by two carriage returns. A carriage return is denoted by "<CR>".

```
Example:
      command:LOGIN<CR>
      user:Andrew@sdsu.edu<CR>
      pass:hello<CR><CR>
```

A value that will be supplied by the client or server is specified as "<value>". Below is the same example as the above command but with the values for user and pass will be filled in by the client or server depending on who is sending the message.

```
Example:
      command:LOGIN<CR>
      user:<email><CR>
      pass:<password><CR><CR>
```

Commands in the VS server are uppercase. All data sent is ASCII text and dates are specified as MM-DD-YYYY format.

3. Authorization Matrix

List of commands and the users that are allowed to execute the commands. Details about each command are explained in detail in the Commands section.

	Anonymous	Registered	Monitor	Admin
LOGIN	X			
POLL-STATUS	X	X	X	
POLL-TOTALS	X	X	X	
REGISTER	X			
LIST-OPEN		X	X	
VOTE		X	X	
VOTE-STATUS		X	X	
LIST-CLOSED		X	X	
ADD-POLL			X	
ADD-MONITOR				X
QUIT	X	X	X	X

Each message received from the server contains a return code. If there is an error processing the request then the message only contains the return code value. Each message is terminated with two carriage returns.

4. Connecting

The connection to the server is statefull. When a user connects to the server the user is assumed to be an anonymous user. An anonymous user can send a LOGIN command or a REGISTER command. Once the user issues a LOGIN or REGISTER command then the user becomes a registered, monitor or admin user depending on the type of user who logged in. When a connection is established the server sends the following:

```
version:1.0<CR><CR>
```

Signifying that the version of the server is one point zero. A version 1.10 is greater than a version 1.1.

5. Commands – Below are the following commands that are available for the Voting Server.
   a. LOGIN
   b. POLL-STATUS
   c. POLL-TOTALS
   d. REGISTER
   e. LIST-OPEN
   f. VOTE
   g. VOTE-STATUS
   h. LIST-CLOSED
   i. ADD-POLL
   j. ADD-MONITOR
   k. QUIT

LOGIN

In order to become a registered, monitor or admin user you have to send a login command. Upon first connecting to the system you are an anonymous user and only have access rights that an anonymous user has.

CLIENT	SERVER
command:LOGIN<CR> email:<email><CR> pass:<password><CR> <CR>	return-code:<return-code><CR> <CR>

```
Return-code
      0-Success
      100-Invalid login attempt
      125-Server error
      127-User already logged in

Example:
Client:
      command:LOGIN<CR>
      user:andrew@sdsu.edu<CR>
      pass:hello<CR><CR>
Server:
      return-code:0<CR><CR>
```

POLL-STATUS

Requests the status of the polls/elections, that is if they are open or closed.

CLIENT	SERVER
command:VOTE-STATUS<CR> <CR>	return-code:<return-code><CR> poll-id:<poll-id><CR> label:<label><CR> status:<status><CR> ... poll-id:<poll-id><CR> label:<label><CR> status:<status><CR> <CR>

```
Return-code
      0  - Success
      125 – Server error

poll-id
      The unique id identifying this poll
label
      The label describing the poll
status
      open
      closed

Example:
Client:
      command:VOTE-STATUS<CR><CR>
Server:
      return-code:0<CR>
      poll-id:100<CR>
      label:Vote for your favorite programming language<CR>
      status:open<CR>
      poll-id:120<CR>
      label:Vote for your favorite scripting language<CR>
      status:closed<CR><CR>
```

POLL-TOTALS

Requests the total for a given poll/election. Anonymous users can only view totals for an active poll. Registered users can view totals for active and closed polls.

CLIENT	SERVER
Command:POLL-TOTALS<CR> id:<id><CR> <CR>	return-code:<return-code><CR> <option1>:<total1><CR> <option2>:<total2><CR> ... <optionN>:<totalN><CR><CR>

```
Return-code
      0  - Success
      125 – Server error
      130 – Access denied, not authorized for given poll.

Example:
Client:
      command:VOTE-TOTAL<CR>
      id:125<CR><CR>
Server:
      return-code:0<CR>
      c++:100<CR>
      java:245<CR>
      C#:82<CR>
      assembly language:14<CR><CR>
```

REGISTER

The register command registers an anonymous user. A successful registration requires that the email and nickname be unique in the system. If the registration is successful the users session changes from anonymous user to registered user and has access rights according to the authorization matrix.

CLIENT	SERVER
command:REGISTER<CR> user:<email><CR> pass:<password><CR> nick:<nickname><CR> <CR>	return-code:<return-code><CR> <CR>

```
Return-code
      0  - Success
      120- Invalid email
      121- Duplicate email
      122- Duplicate nickname
      125- Server error
      126- User not authorized

Example:
Client:
      command:REGISTER<CR>
      user:andrew@sdsu.edu<CR>
      pass:hello<CR>
      nick:andrew<CR><CR>
Server:
      return-code:0<CR><CR>
```

LIST-OPEN

Requests all the open polls/elections. Each poll has the following attributes: id (unique), label, description, start-date, end-date and options. Options is a comma separated list of items that a user can vote on.

CLIENT	SERVER
command:LIST-OPEN<CR> <CR>	return-code:<return-code><CR> poll-id:<poll-id><CR> label:<label><CR> description:<description><CR> start-date:<start-date><CR> end-date:<end-date><CR> options:<options><CR> ... poll-id:<poll-id><CR> label:<label><CR> description:<description><CR> start-date:<start-date><CR> end-date:<end-date><CR> options:<options><CR> <CR>

Return-code
- 0 - Success
- 125- Server error
- 126- User not authorized

Example:

Client:
```
      command:LIST-OPEN<CR><CR>
```
Server:
```
      return-code:0<CR>
      poll-id:120<CR>
      label:Vote for your favorite programming language<CR>
      description:Poll to determine which is the favorite programming
      language for January<CR>
      start-date:01-12-2007<CR>
      end-date:01-31-2007<CR>
      options:c++,java,cobol,fortran,assembly language<CR>
      poll-id:121<CR>
      label:Vote for your favorite scripting language<CR>
      description: Poll to determine which is the favorite scripting language
      for January<CR>
      start-date:01-12-2007<CR>
      end-date:01-31-2007<CR>
      options:javascript,actionscript,php,perl<CR><CR>
```

VOTE

Cast a vote for a particular poll. If the user already voted for the given poll then the new vote replaces the old one.

CLIENT	SERVER
command:VOTE<CR> poll-id:<poll-id><CR> option:<option> <CR>	return-code:<return-code><CR> <CR>

Return-code

0	- Success
125	- Server error
126	- User not authorized
144	- Invalid vote, option does not exist

Example:
Client:
```
      command:VOTE<CR>
      poll-id:120<CR>
      option:java<CR><CR>
```
Server:
```
      return-code:0<CR><CR>
```

VOTE-STATUS

Request the vote status for a given poll. That is, if the user has voted in the poll and which option was the vote applied to. If the user voted in a poll then the option is the item the user voted on. If the user did not vote in the given poll then option is blank.

CLIENT	SERVER
command:VOTE-STATUS<CR> poll-id:<poll-id><CR> <CR>	return-code:<return-code><CR> option:<option><CR> <CR>

Return-code

0	- success
125	- Server error
126	- user not authorized

Example of a user who has already voted for java:

Client:
```
      command:VOTE-STATUS<CR>
      poll-id:120<CR>
      <CR>
```
Server:
```
      return-code:0<CR>
      option:java<CR><CR>
```

Example of a user who has not voted:

Client:
```
      command:VOTE-STATUS<CR>
      poll-id:125<CR>
      <CR>
```
Server:
```
      return-code:0<CR>
      option:<CR><CR>
```

LIST-CLOSED

Requests all the closed polls/elections. Each poll has the following attributes: poll-id (unique), label, description, start-date, end-date and options.

Options contains a comma separated list of items that a user voted on and the total for that item. The format for options is specified below:
<options>=<option-name1>:<option-total1>,…,<option-nameN>:<option-totalN>

CLIENT	SERVER
command:LIST-CLOSED<CR> <CR>	return-code:<return-code><CR> poll-id:<poll-id><CR> label:<label><CR> description:<description><CR> start-date:<start-date><CR> end-date:<end-date><CR> options:<options><CR> ... poll-id:<poll-id><CR> label:<label><CR> description:<description><CR> start-date:<start-date><CR> end-date:<end-date><CR> options:<options><CR> <CR>

Return-code
- 0 - Success
- 125 – Server error
- 126 – User not authorized

Example:
Client:
```
command:LIST-CLOSED<CR><CR>
```
Server:
```
return-code:0<CR>
poll-id:120<CR>
label:Vote for your favorite programming language.<CR>
description: Poll to determine which is the favorite programming
language for January<CR>
start-date:01-01-2007<CR>
end-date:01-31-2007<CR>
options:c++:10,java:40,cobol:4,assembly language:2<CR>
poll-id:130<CR>
label:Vote for your favorite scripting language.<CR>
description: Poll to determine which is the favorite scripting language
for January<CR>
start-date:01-01-2007<CR>
end-date:01-31-2007<CR>
options:javascript:105,perl:80,actionscript:35<CR><CR>
```

ADD-POLL

Adds a poll/election. Each poll has the following attributes: poll-id (unique), label, description, start-date, end-date and options. Two polls with the same label can not be active at the same time. A poll is said to be active if the current date is between the start-date and end-date. A poll with the same label can not have a start-date or end-date that falls within another poll’s start-date and end-date that share equal label names.

CLIENT	SERVER
command:ADD-POLL<CR> label:<label><CR> description:<description><CR> start-date:<start-date><CR> end-date:<end-date><CR> options:<options><CR> <CR>	return-code:<return-code><CR> poll-id:<poll-id><CR> <CR>

Return-code

While most of these errors can be checked on the client side, the server still checks that the dates are valid.

```
0  - Success
140- Active poll with current label already exists.
141- End-Date is prior to start-date.
142- End-Date has already passed no voting can take place.
143- Start-Date has to be set to current-date or some future date.
125– Server error
126– User not authorized
```

label

The name of the poll. Maximum of 255 characters.

description

Maximum of 1024 characters.

start-date

The date the poll will be active. MM-DD-YYYY format

end-date

The last day the poll will be active. MM-DD-YYYY format

options

contains a comma separated list of items that a user voted on and the total for that item. The format for options is specified below: <options>=<option-name1>,…,<option-nameN>. The maximum length for each individual option is 255 characters.

Example:

Client:

```
command:ADD-POLL<CR>
label:Vote for your favorite programming language.<CR>
description: Poll to determine which is the favorite programming
language for January<CR>
start-date:01-01-2007<CR>
end-date:01-31-2007<CR>
options:c++,java,cobol,assembly language<CR><CR>
```

Server:

```
return-code:0<CR>
poll-id:120<CR><CR>
```

ADD-MONITOR

Adds a monitor to the system. Only a system administrator can add a monitor to the system. Email and nickname are unique. No two users can be registered having the same email or nickname.

CLIENT	SERVER
command:ADD-MONITOR<CR> email:<email><CR> pass:<password><CR> nick:<nickname><CR> <CR>	return-code:<return-code><CR> <CR>

Return-code
- 0 - Success
- 120- Invalid email
- 121- Duplicate email
- 122- Duplicate nickname
- 126- User not authorized

email
Email used to login to the system. Maximum length 255 characters.

password
Password used to authenticate. Maximum length 24 characters.

nickname
Maximum length 20 characters.

QUIT

Closes the connection.

CLIENT	SERVER
command:QUIT<CR><CR>	Return-code:0<CR><CR>

Return-code
- 0 - success
- 125- server error

6. Error Codes

```
  0- Success
100- Invalid login attempt
120- Invalid email
121- Duplicate email
122- Duplicate nickname
125- Server error
126- User not authorized
127- User already logged in
130- Access denied, not authorized for given poll
140- Active poll with current label already exists
141- End-Date is prior to start-date
142- End-Date has already passed no voting can take place
143- Start-Date has to be set to current-date or some future date
144- Invalid Vote, option does not exist
145- Invalid Command
```

An error code of 125 that is Server error means that there was a problem with the server processing the command. If the server uses a database to store the data it could be that the database is down or some other type of error has occurred. In this case it is up to the server administrator to fix this type of errors.

7. Sample Session

Below is a sample session that does the following:

1. User connects
2. The anonymous user registers
3. The registered user lists the open polls.
4. The registered user votes for a given item in a poll.
5. The registered user requests a vote total for the poll he voted on.
6. The registered user checks what item he voted on.
7. The registered user votes again on the same poll but on a different item.
8. The registered user requests a vote total for the poll he voted on.
9. The registered sends a quit command.

```
Client:
      Open tcp connection
Server:
      version:1.0<CR><CR>
Client:
      command:REGISTER<CR>
      user:andrew@sdsu.edu<CR>
      pass:hello<CR>
      nick:andrew<CR><CR>
Server:
      return-code:0<CR><CR>
Client:
      command:LIST-OPEN<CR><CR>
Server:
      return-code:0<CR>
      poll-id:120<CR>
      label:Vote for your favorite programming language<CR>
      start-date:01-12-2007<CR>
      end-date:01-31-2007<CR>
      options:c++,java,cobol,fortran,assembly language<CR>
      poll-id:121<CR>
      label:Vote for your favorite programming language<CR>
      start-date:01-12-2007<CR>
      end-date:01-31-2007<CR>
      options:c++,java,cobol,fortran,assembly language<CR><CR>
Client:
      command:VOTE<CR>
      poll-id:120<CR>
      option:cobol<CR><CR>
Server:
      return-code:0<CR><CR>
Client:
      command:VOTE-TOTAL<CR>
      id:120<CR><CR>
Server:
      return-code:0<CR>
      c++:100<CR>
      java:245<CR>
      cobol:82<CR>
      fortran:10<CR>
      assembly language:14<CR><CR>
Client:
      command:VOTE-STATUS<CR>
      poll-id:120<CR><CR>
Server:
      return-code:0<CR>
      option:cobol<CR><CR>
Client:
      command:VOTE<CR>
      poll-id:120<CR>
      option:java<CR><CR>

Server:
      return-code:0<CR><CR>
Client:
      command:VOTE-TOTAL<CR>
      id:120<CR><CR>
Server:
      return-code:0<CR>
      c++:100<CR>
      java:246<CR>
      cobol:81<CR>
      fortran:10<CR>
      assembly language:14<CR><CR>
Client:
      command:QUIT<CR><CR>
Server:
      Return-code:0<CR><CR>
      Closes tcp connection
Client:
      Closes tcp connection
```