

## G3 Banking API

### MessageType Enum:

<i>CONNECT_CLIENT</i>	- Used to tell the server an ATM machine has connected.
<i>CONNECT_TELLER</i>	- Used to tell the server a Teller machine has connected.
<i>LOGIN_REQ</i>	- Request server to verify login credentials.
<i>USER_INFO_REQ</i>	- Requests server for specific Users info.
<i>LOGS_REQ</i>	- Requests server for logs on a specified user's associated accounts.
<i>ACCOUNT_INFO</i>	- Requests server for info on specific account. Also used to notify client or teller machine of similar data.
<i>LOG_INFO</i>	- Requests server log info.
<i>DONE</i>	- Notifies client or teller of a completed loop.
<i>SUCCESS</i>	- Notifies client or teller of a successful action.
<i>FAIL</i>	- Notifies client or teller of a failed action.
<i>DEPOSIT</i>	- Requests server to deposit n amount of money into x account.
<i>WITHDRAW</i>	- Requests server to withdraw n amount of money from x Account.
<i>ADD_USER</i>	- Requests server to add a user to x account.
<i>REMOVE_USER</i>	- Requests server to remove a user from x account.
<i>TRANSFER</i>	- Requests server to Transfer funds from account A to account B.
<i>STATUS_CHANGE</i>	- Requests Server for a status change on x account.
<i>MAKE_ACCOUNT</i>	- Requests server to add an account to an existing user.
<i>LOGOUT</i>	- Requests server to

### Message Object:

The message object has two types of constructors. One for passing data when money is involved and the other for passing non-fund related messages.

```
Message(MessageType type, String data, float funds)
Message(MessageType type, String data)
```

### Connecting to Server:

- 1) When connecting to the server either a client or teller needs to connect. Any other type of connection will be automatically disconnected immediately.

```
Message(MessageType.CONNECT_CLIENT, null);
Message(MessageType.CONNECT_TELLER, null);
```

The funds and data fields aren't necessary.

Returning message type will be either SUCCESS or terminated connection.

- 2) After successful connection a message needs to be sent to the server to verify a user login.

```
Message(MessageType.LOGIN, data);  
String data = <username> + "\n" + <password>;
```

Returning message type will be either SUCCESS or FAIL establishing a completed connection to the server.

### Requests to Server:

Message Types and Formatting:

#### LOGOUT

- To Server:

```
Message(MessageType.LOGOUT, null);
```

- From Server:

```
Message(MessageType.LOGOUT, "User Logged Out")
```

#### DEPOSIT

- To Server:

```
Message(MessageType.DEPOSIT, account, amount);  
String account = <account name>;  
float amount = amount of funds to be deposited
```

- From Server:

```
Message(MessageType.SUCCESS, null); // Funds got deposited

Message(MessageType.FAIL, "Insufficient Funds") // Value of funds isn't greater
                                                than 0;

Message(MessageType.FAIL, "Invalid User") // User does not have permission to
                                          access account.

Message(MessageType.FAIL, "Invalid Account") // Account doesn't exist.
```

## WITHDRAW

- To Server:

```
Message(MessageType.DEPOSIT, account, amount);
String account = <account name>;
float amount = amount of funds to be withdraw.
```

- From Server:

```
Message(MessageType.SUCCESS, null); // Funds got withdrawn.

Message(MessageType.FAIL, "Insufficient Funds") // Not enough funds in account
                                                to withdraw.

Message(MessageType.FAIL, "Invalid User") // User does not have permission to
                                          access account.

Message(MessageType.FAIL, "Invalid Account") // Account doesn't exist.
```

## ADD\_USER

- To Server:

```
Message(MessageType.ADD_USER, data) // Adds user to specified account.
String data = [account] + "\n" + [user];
```

- From Server:

```
Message(MessageType.SUCCESS,data); // Confirms user has been added to
                                   specified account.
String data = "User " + [user] + " added to account" + [account];

Message(MessageType.FAIL,"User already attached") // User already exist on
                                                Account

Message(MessageType.FAIL,"Invalid Account") // Account doesn't exist.
```

## REMOVE\_USER

- To Server:

```
Message(MessageType.REMOVE_USER, data) // Removes user to specified account.
String data = [account] + "\n" + [user];
```

- From Server:

```
Message(MessageType.SUCCESS,data); // Confirms user has been removed from
                                   specified account.
String data = "User " + [user] + " removed from account" + [account];

Message(MessageType.FAIL,"User not attached") // User does not exist on
                                                account

Message(MessageType.FAIL,"Invalid Account") // Account doesn't exist.
```

## TRANSFER

- To Server:

```
Message(MessageType.SUCCESS,data,amount); // Transfers funds from account1 to
                                           Account2
String data = [account1] + "\n" + [account2];
float amount = The amount of funds to be transferred from one account to
               another.
```

- From Server:

```
Message(MessageType.FAIL,"Invalid account: "+[account1])) // account1 doesn't
                                                    exist.

Message(MessageType.FAIL,"Invalid account: "+[account1])) // account2 doesn't
                                                    exist.

Message(MessageType.SUCCESS,"Transfer Successful") // Funds are successfully
                                                    Transferred

Message(MessageType.FAIL,"Insufficient Funds") // Not withdraw from first
                                                    account exceeds account
                                                    minimum.
```

## USER\_INFO\_REQ

- To Server:
- From Server:

## LOGS\_REQ

- To Server:

```
Message(MessageType.LOG_INFO, account);
String account = account name.
```

- From Server:

```
Message(MessageType.LOG_INFO,logData) // Sends back data about actions that
                                     happened on a users accounts
String logData = [user] + "\n" + [action] + "\n" + [amount] + "\n" + [date];

Message(MessageType.DONE,"") // Used to flag when all logs have been sent

Message(MessageType.FAIL,"Invalid Account") // Account doesn't exist
```

## ACCOUNT\_INFO

- To Server:

```
Message(MessageType.ACCOUNT_INFO, user)
String user = name of the current user logged in.
```

- From Server:

```
Message(MessageType.ACCOUNT_INFO,account.getName() + "\n" +
account.getStatus(),account.getBalance())

Message(MessageType.ACCOUNT_INFO,data) // Returns account name, status, and
Balance
data = [name] + "\n" + [status] + "\n" + [balance]; // Account info.

Message(MessageType.FAIL,"Access Denied") // User does not have permission to
view account info.

Message(MessageType.FAIL,"Invalid Account") // Account doesn't exist.
```