**AATC\_Client.py**

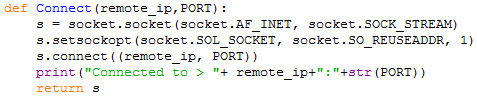
This module acts as an interface to the User portion of the server. Similarly to the AATC\_Drone.py and AATC\_Monitor.py modules this module simplifies the communication between the program and the server. It provides access to server’s commands via methods of the UserInterface object rather than requiring the user to manage connection objects, encryption and direct access to the sending of commands to the server. Instead another program only needs to call the function for that command and the result will be returned with built in handling of errors once setup. The program will allow exceptions to propagate while the initial connection is being established in order to inform the caller that setup did not complete successfully.

**Import**



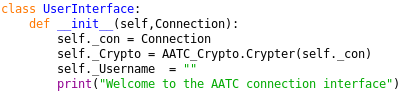
The module imports 5 modules. All are required for the correct functioning of the interface. The details are explained in another interface module.

**Connect**

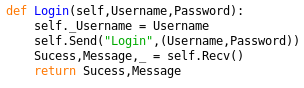


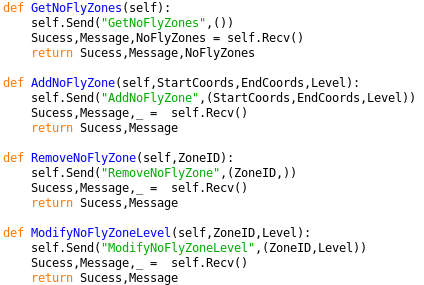
This function forms a connection between the server and the client with the socket module. The address is set to be reusable to allow the address to be reused if the previous connection was not cleanly shutdown. The function returns the created connection object once complete.

**UserInterface**

This class acts as an abstraction layer between the server and the client program. When instantiated the required argument is a connection object which has been connected to the server to which the user desires to be connected to. The object creates an AATC\_Crypto.Crypter object to manage the server validation and encryption with the connection object. The Username is set to the empty string and a message is displayed that the connection completed successfully.

Almost every method of the UserInterface takes a similar form. They take zero or more explicit arguments and return two or three variables, Success , Message and optionally Data. The method takes the variables, sends the relevant command and obtains the results. Finally it returns the information to the caller. If no data is returned ( returns into \_ , then that value will not be returned).

The login method takes two arguments, Username and Password. It sets the username to the Username of the UserInterface and sends the command and data to the server. The success and message are received and returned.

The GetNoFlyZone method obtains all the NoFlyZones for that server. It requires no arguments and returns the list of tuples of each row of data from the server.

The AddNoFlyZone method is used to add a NoFlyZone to the server, takes two string parameters Start and End Coords and one integer parameter Level , sends this information the the server and returns the Sucess and Message.

The RemoveNoFlyZone method is used to remove a NoFlyZone from the server, takes one integer parameter ZoneID and passes this to the server and returns the Sucess and Message as a response.

The ModifyNoFlyZoneLevel method is used to modify the level of a NoFlyZone ( it’s cost) ,takes two integer parameters ZoneID and Level, sends this information to the server and returns the Sucess and Message as a response.

AddDrone takes 6 parameters, 3 string parameters DroneName,DronePassword and Type, 2 integer parameters Speed and Range and one float parameter Weight. The method uses this information to register a new drone on the server and returns the message and response.

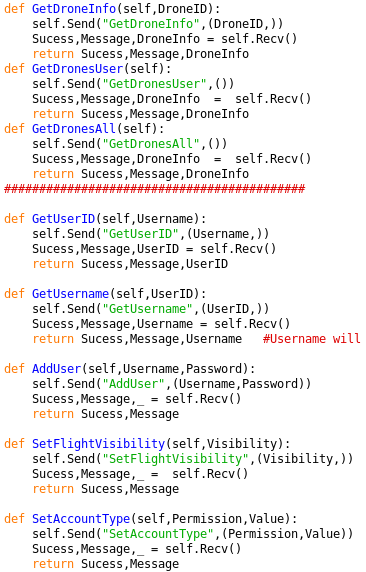
RemoveDrone takes one integer parameter DroneID, which is used to remove a drone from the server and returns the success and message.

GetDroneID ,used to obtain the droneID for a given drone name, takes one string parameter of the DroneName and sends this to the server and returns the success, response and the list containing the droneID.

GetDroneCredentials takes one integer parameter DroneID, sends this information and obtains the credentials for that drone, in the success, message and credentials variables returned to the user.

SetDroneCredentials takes two parameters, the integer DroneID and the string DronePassword. This information is used to set the drone credentials for the drone with the DroneID passed to the method and returns the success and message to the user.

CheckDroneOwnership is used to check if a UserID owns a DroneID. This takes two integer parameters UserID and DroneID and returns the success, message and data of this response.

The method GetDroneInfo is used to obtain information about a drone defined by the DroneID integer parameter passed to the function. The information is sent to the server and the success, message and DroneInfo responses are returned.

The Method GetDronesUser obtains all the information about the Drones belonging to a user. This method takes no parameters and returns the Sucess,Message and droneInfo response from the server.

The GetDronesAll method takes no parameters and is used to get all the information about the publicly displayed drones. This information is returned in the form Sucess,Message and DroneInfo.

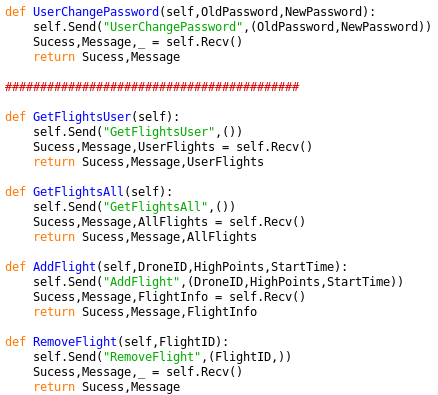
The method GetUserID takes one string parameter Username and returns the UserID for that username. It returns the Sucess, message and UserID for that username,

The method GetUsername takes one integer parameter UserID and obtains the Username linked to the passed UserID. The method then returns the Sucess, message and Username obtained from the server.

The method AddUser requires two string parameters Username and Password. This is used to add a new user to the server. This returns the success and messages responses from the server.

The SetFlightVisibility method takes one integer parameters Visibility , which is either 0 to set the visibility to private and 1 to set the visibility to public. This returns the success and message from the server.

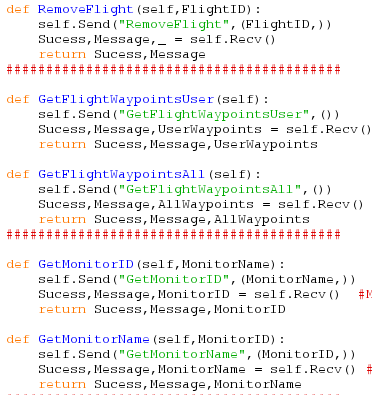
The SetAccountType method is used to change the account type of the account. The method takes two parameters , the string account type and the integer value. The account type is the account type to change and the value is the value it should set it to , normally 0 or 1. The response it returned.

The UserChangePassword method is used in order to alter the password of the user. This takes two string parameters, OldPassword and NewPassword. This is sent to the server and the response is returned.

GetFlightsUser obtains all the flights belonging to the user. This requires no parameters and the result is returned in the success, message and AllFlights variables.

GetFlightsAll obtains all the publicly available flights requiring no parameters and returning the success, message and AllFlights.

The AddFlight method is used to add new flights, requiring an integer DroneID, a list of strings containing tuple coordinates i.e. “(0.5,0.7,200)” and the integer strart time.



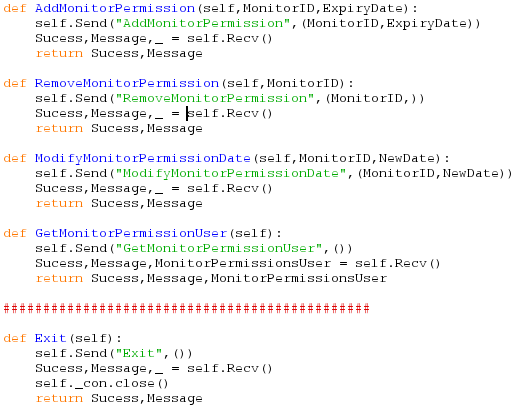
The RemoveFlight method takes the integer FlightID and removes the flight for that flightID if it exists and the user owns that flight. The method then returns the success and message response from the server.

The GetFlightWaypointsUser method obtains all flight waypoints belonging to flights belonging to the user. This is returned from the server in the form success, message and UserWaypoints.

GetFlightWaypointsAll is used to obtain all public waypoints on the server, taking no parameters and returning the success, message and AllWaypoints.

The GetMonitorID method is used to obtain the monitorID for the string MonitorName the user passes to the method. This method then returns the success, message and the monitorID result from the server.

The GetMonitorName method takes the integer monitorID and returns the monitor name of that monitor if it exists. This method then returns the success, message and monitor name result from the server.



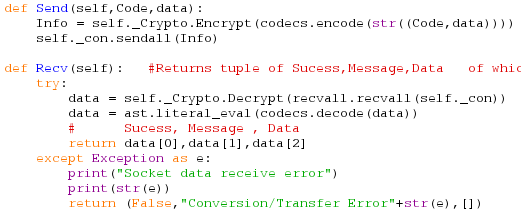
The AddMonitorPermission method takes two integer parameters MonitorID and ExpiryDate in order to add a new monitor permission. This returns the success and message responses from the server.

The RemoveMonitorPermission method is used to remove a monitor permission using one integer parameter MonitorID. This returns success and message based on the response.

The ModifyMonitorPermission method requires two integer parameters, MonitorID and NewDate in order to modify the monitor permission, changing the monitor permission expiry date to NewDate. This returns success and message based on the response.

The GetMonitorPermissionUser method is used to obtain all the monitor permissions belonging to that user. This is data is returned in the MonitorPermissionsUser variable.

The Exit command is used to close the connection between the user and the server gracefully, sending the exit command and closing the connection, returning success and message.

The send method takes two parameters, Code and Data. Code is the command the user wants to use and data is a tuple containing all additional information which is required to execute the command. The data is formed into a string, encoded into the binary format and encrypted and finally sent to the server.

The Recv method is used to receive a response from the server. The method waits for a response and is blocking. The method obtains the entire response using recvall , decrypts the response, decodes and returns the response seperated into the success, message and data tuples. Should an exception occur then the exception will be caught and a general warning message of a transfer error returned, as well as displaying the issue to the user.

The CreateUserInterface method is used to simply create user interface objects with default connection values. The method creates a socket connection to the IP /port combination passed as optional variables and creates a user interface object from this, returning the user interface.