This project includes two parts, the server program and the client program.

The server will define some functions that will actually perform the transactions, and exports those functions as APIs for others via the Internet. The client then imports those APIs, make the call like they are defined locally, and interpret the results (message code or real data) returned from the server program as it is the return value from the local function.

Different from actually native local function, the design of this project uses the common communication code for the returned value instead of just return void. If it were the true local function, the most common routine is to return nothing and print out the error message directly. Since this routine would not work if the message is printed out in server side, to solve this issue, the commonly understandable code for the error message is utilized. We don’t return the message string directly based on the principal of optimizing the network traffic load.

The sketch on how RMI project works is in the graph below:



For improvements, the multi-thread feature in the server program is a must. Right now there is only a single thread in the server, which is sufficient since in this simple example all the transactions will not take noticeable amount of time. But in real life case, due to the possible bottleneck of the I/O operations and database speed, the client requests might congest. So implementing the thread pool and use each thread to serve an individual client will improve the system throughputs significantly.