Спецификация взаимодействия клиента и сервера.

1) Типы запросов клиента и их обработка
2) Формат сообщений, посылаемых с клиента на сервер
3) Формат ответа сервера

Типы запросов клиента к серверу:

- Функциональность клиента состоит в том, чтобы посылать запрос, принимать ответ и его выводить. Никакой обработки информации не нужно.
- Типы сообщений заданы в файлахGameEvent.as и GameEvent.h. ОБЯЗАТЕЛЬНО нужно соблюдать идентичность формата. Если добавились новые события в одном файле, нужно добавить ее в другом файле и добавить в спецификацию.
- Нужно учитывать, что везде может быть FAIL({Reason : string})
- Описание запросов:
  0. AUTH {vk_id:uint , Password:String } – запрос на авторизацию. Возможные варианты ответа:
     0. SUCCESS – все ок. если не существует, то создать и вернуть ему id.
     8. PLAYER_ALREADY_AUTHORIZED – уже есть подключение с данным игроком и новое подкючение создавать не стоит.
  1. GET_ONLINE_PLAYERS()
     ONLINE_PLAYERS {ids[id, vk_id, rating]}
  2. INVITE_TO_PLAY {opp_id: uint}– посылается запрос на opp_id о предложении сыграть от данного игрока. На клиент игрока посылается ответ GotInvitation.
     Ответа нет
  4. GOT_INVITE{uint opp_id} — запрос на игру.
     Ответа нет
  5. ACCEPT_INVITE{uint opp_id} – принять запрос на игру от opp_Id.
  6. REFUSE_INVITE{uint opp_id} – отказать запрос
  7. MOVE {from: String[2], to: String[2]}ход откуда куда.
  8. GAME_STATE {isValid: bool, whiteTurn:bool(), whites:[String[3]], blacks:[String[3]]}, fogCells: [String[2]]). // если isValid == false, то не смотрим на остальные сущности.
  9. GAME_FINISHED {STATUS : uint} //STATUS = Enum {WHITE_WON, BLACK_WON, DRAW}

Формат сообщений, посылаемых с клиента на сервер.[НИЗКОУРОВНЕВЫЙ]

Формат запросов будет выглядеть примерно так: [messageLength, messageCommand, {arguments}], где

- messageLength – ожидаемая длина сообщения в байтах. ~~Жизнь настолько сурова,~~ TCP/IP так устроен, что сообщения необязательно придут сразу же одним сообщением, поэтому обрабатывать запрос нужно только в том случае, когда у нас есть все данные.
- messageCommand – тип запрос клиента, см. Типы запросов
- arguments – аргументы запроса messageCommand. Должен приходить в виде объекта. Пример: {id:3371777, password:”PASSWORD”}. **Внимание!** Нужно строго придерживаться формата названий параметров!

Формат ответа сервера клиенту.[НИЗКОУРОВНЕВЫЙ]

Ответы сервера могут быть разными(начиная от числа, заканчивая массивами фигур), поэтому предлагаю следующий формат ответа: [messageLength, messageCommand, {answer}], где

- messageLength – длина ответа в байтах.
- respond - запрос | ответ на запрос. //const
- resultCode
- arguments – ответ. Приходит в виде массива данных.