# Message format

## Format

All messages must be sent using http POST. Every message except Users/RegisterUser/ must contain

*{*

*login:xxx*

*password:yyy*

*…*

*}*

All messages must be serialized to JSON.

All properties inside JSON begin with a capital letter. ID is written as it can be seen here.

## File embedding

Files must be embedded insIDe messages content using !<ID >! Tag. Any client must not allow sending !< or >! Tags. Subsequently when a message with !< tag is received it’s file should be fetched from the server using File Endpoints

# Communication

## Users

### Public interface IUsersRepository {

#### IEnumerable<User> GetUsers();

#### User GetUser(int ID);

#### User GetUserWithLogin(string login);

#### bool RemoveUser(int ID);

#### voID RegisterUser(User user);

#### bool IsLoginValID(string login, string password);

#### voID SetUser(User user);

}

### End points

Users/GetUsers

Users/GetUser/

Users/GetUserWithLogin/

Users/RemoveUser/

Users/RegisterUser/

Users/ValIDateLogin/

Users/SetUser/

## Relationships

### Public interface IRelationshipRepository {

#### IEnumerable<Relationship> GetRelForUser(int ID);

#### voID SetRel(Relationship rel);

#### bool RemoveRel(int ID);

#### voID AddRel(Relationship rel);

}

### End points

Relationships/GetRelForUser/

Relationships/SetRel/

Relationships/RemoveRel/

Relationships/AddRel/

## Rooms

### Public interface IRoomRepository {

#### IEnumerable<Room> GetRoomsWithUser(int IDUser);

Všechny místnosti kde user je

#### IEnumerable<User> GetUsersInRoom(int ID);

Všichni useři v místnosti

#### bool RemoveRoom(int ID);

#### voID AddRoom(Room room);

#### voID SetRoom(Room room);

#### voID AddUserToRoom(int IDUser, int IDRoom);

#### bool RemoveUserFromRoom(int IDUser, int IDRoom);

}

### End points

Rooms/GetRoomsWithUser/

Rooms/GetUsersInRoom/

Rooms/RemoveRoom/

Rooms/AddRoom/

Rooms/SetRoom/

Rooms/AddUserToRoom/

Rooms/RemoveUserFromRoom/

## Messages

### Public interface IMessageRepository {

#### IEnumerable<Message> GetMessagesInRoom(int ID);

#### IEnumerable<Message> GetMessagesInRoomSince(int ID, DateTime since);

All messages in a room since a date

#### IEnumerable<Message> GetNewMessagesForUser(int ID);

All messages sent to user after his last seen date

#### bool RemoveMessage(int ID);

#### voID SetMesssage(Message message);

#### voID AddMessage(Message message);

}

### End points

Messages/GetMessagesInRoom/

Messages/GetMessagesInRoomSince/

Messages/GetNewMessagesForUser/

Messages/RemoveMessage/

Messages/SetMessage/

Messages/AddMessage/

## Files

### public interface ICFileShareRepository {

#### CFile GetFile(int ID);

#### bool RemoveFile(int ID);

#### voID AddFile(CFile CFile);

}

### End points

CFiles/GetFile/

CFiles/RemoveFile/

CFiles/AddFile/

# Recomendations

When sending messages to server you can serialize anonymous objects, as to no needing to use too many classes. Example :

Serializer.Serialize( new { login=“Example“, password=“\*\*\*“, user=User });

Server never sends any objects that the client doesn’t also have so they can be easily deserialized

Deserializer.Deserialize<IEnumerable<Users>>(WebConnection.GetUsersInRoom(myRoom));

# Controller Documentation

## Template

<Controller name>

<Description>

<Endpoints>

## Users

Allows for adding, removing and changing users. I also user to register new users and valIDate user’s logins.

## Relationships

Used for modifying relationship statuses between users. Used for blocking users, adding friends and sending friends requests.

## Rooms

Facilitates communication space for users. Users can create, modify or delete rooms using this controller. Supports adding/removing users to/from rooms;

## Messages

Sends and fetches messages. Embedded content must be obtained and sent to the server using Files controller.

## Files

Allows sending and receiving files.

# Endpoint Documentation

## Template

<Endpoint name>

<Short description>

<Inputs>

<Outputs>

<Long description>

<Examples>

<Recommendations>

<See also>