# Message format

## Format

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

*{*

*Login:xxx*

*Password:yyy*

*…*

*}*

Password is a byte array of ASCII chatacters

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 !<UUID >! 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. UUID should be encoded using Base64.

# Communication

## Users

### Public interface IUsersRepository {

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

#### User GetUser(int ID);

#### User GetUserWithLogin(string UserLogin);

#### bool RemoveUser(int ID);

#### void RegisterUser(User User);

#### bool ValidateLogin(string Login, string Password);

#### void SetUser(User User);

}

### End points

[POST]User/GetUsers

[POST]User/GetUser/

[POST]User/GetUserWithLogin/

[DELETE]User/User/

[PUT]User/RegisterUser/

[POST]User/ValidateLogin/

[POST]User/User/

### Model

public int Id { get; set; }

public string FirstName { get; set; }

public string SecondName { get; set; }

public string Login { get; set; }

public byte[] Password { get; set; }

public string PictureUrl { get; set; }

public string Email { get; set; }

public DateTime DateRegistered { get; set; }

public DateTime DateLastLogin { get; set; }

public int Status { get; set; }

## Relationships

### Public interface IRelationshipRepository {

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

#### void SetRel(Relationship Relationship);

#### bool RemoveRel(int ID);

#### void AddRel(Relationship Relationship);

}Relationship

### End points

[POST]Relationship/GetRelForUser/

[POST]Relationship/Rel/

[DELETE]Relationship/Rel/

[PUT]Relationship/Rel/

### Model

public int Id { get; set; }

public int IdsourceUser { get; set; }

public int IdtargetUser { get; set; }

public int RelationType { get; set; }

public DateTime DateCreated { get; set; }

## Rooms

### Public interface IRoomRepository {

#### IEnumerable<Room> GetRoomsWithUser(int ID); // 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

[POST]Room/GetRoomsWithUser/

[POST]Room/GetUsersInRoom/

[DELETE]Room/Room/

[POST]Room/Room /

[PUT]Room/Room /

[PUT]Room/AddUserToRoom/

[DELETE]Room/RemoveUserFromRoom/

### Model

public int Id { get; set; }

public int Idcreator { get; set; }

public string Name { get; set; }

public bool OneOnOne { get; set; }

## 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

[POST]Message/GetMessagesInRoom/

[POST]Message/GetMessagesInRoomSince/

[POST]Message/GetNewMessagesForUser/

[DELETE]Message/Message/

[POST]Message/Message /

[PUT]Message/Message /

### Model

public int Id { get; set; }

public int Idsender { get; set; }

public int IdroomReceiver { get; set; }

public string Content { get; set; }

public bool Received { get; set; }

public bool Seen { get; set; }

public bool Edited { get; set; }

public bool Writing { get; set; }

public DateTime DateSent { get; set; }

public DateTime? DateReceived { get; set; }

public DateTime? DateRead { get; set; }

public DateTime? DateEdited { get; set; }

## CFiles

### public interface ICFileShareRepository {

#### CFile GetFile(byte[] UUID);

#### bool RemoveFile(byte[] UUID);

#### byte[] AddFile(byte[] Content, CFile CFile); // Returns UUID of file added

#### void SetFile(CFile CFile);

#### IEnumerable<CFile> GetFilesByUser(int ID);

#### IEnumerable<CFile> GetFilesInRoom(int ID);

#### byte[] GetFileContents(byte[] UUID);

#### IDictionary<byte[],byte[]> GetFilesContents(IEnumerable<byte[]> UUIDs); // Return format UUID:Contents

}

### End points

[POST]CFile/File/

[DELETE]CFile/File /

[PUT]CFile/File /

[PATCH]CFile/File /

[POST]CFile/GetFilesByUser/

[POST]CFile/GetFilesInRoom/

[POST]CFile/GetFileContents/

[POST]CFile/GetFilesContents/

### Model

public byte[] Uuid { get; set; }

public int Iduploader { get; set; }

public int Idroom { get; set; }

public string FileName { get; set; }

# 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>