Mindspot Indonesia

Melisa's Messaging Protocol

Technical Document

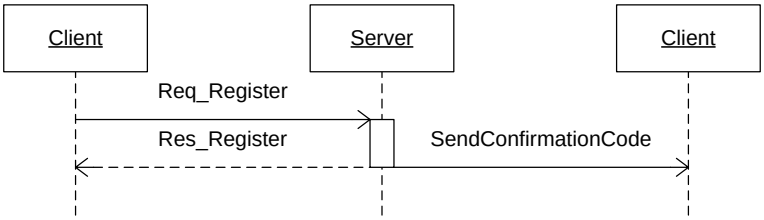
Version 1.0.0

This document includes protocols and standards used by melisa to perform messaging. Highly confidential document and only for internal purposes.

Change Log

Date	Revise By	Revision
2016-07-16	Riza Anshari	• Change chapter's titles. • Add how to stop following someone chapter.
2016-07-11	Riza Anshari	• Added Req_RequestToFollow chapter. • Added Req_GetListFollowing chapter. • Added Req_GetListFollower chapter.
2016-06-17	Riza Anshari	• Added Req_UserProfileUpdate chapter. • Added `ErrorUserMustAuthorizedFirst` and `ErrorEmailIsMandatory` in Req_UserProfileUpdate protocol. • Added Req_GetListCity chapter. • Added Req_GetListCountry chapter.
2016-06-11	Riza Anshari	• Added two new error codes in Req_Register (`ErrorInvalidUsername and ErrorInvalidEmailAddress`) and redefined the others error codes value. • Added request id for Req_Register. • Default value of user type id is `0`. • Add not be used yet flag to `Req_GetListServiceStationAround`.
2016-06-10	Riza Anshari	• Added Req_RegisterConfirm chapter. • Enhanced the user register mechanism with the new concept.
2016-03-26	Riza Anshari	• Changed nomenclature from user id to user identifier in error code description of login request. • Remove `username` from Req_GetListServiceStationInside's input parameter. This request does not need this parameter actually. • Added AuthorizationKey to Req_GetListServiceStationInside's input parameter. • Added `AuthorizationKey` to `ResponseLogin` data structure. • Added some brief about `ServiceStationId` in input parameter chapter.
2016-03-25	Riza Anshari	• Added some brief of Req_Register chapter. • Fixed register sequence diagram. `SendEmail` message was changed to `SendConfirmationCode`. • Added document cover. • Added change logging page. • Initial version.

User Registration



All users must register before using the mobile application. Clients will send the `Req_Register` message to server if they want to register a new user. If registration process succeeded, server will send a confirmation code to client. Server can be delivered this code by sending an email containing confirmation link or by sending a SMS via GSM network.

Request ID

The request ID for `Req_Register` (in integer) is `100001`.

Input parameters

The data sent for registration request follows the following data structure:

```
class RequestRegister
{
    string[25] Username;
    string[64] Password;
    int UserTypeId;
    string[100] Email;
    double Latitude;
    double Longitude;
    string[100] ServiceStationName;
}
```

`Username`, this parameter contains information about the identifier expected by user to be used. This input parameter is mandatory and should be filled with letter [A - Z, a – z] and/or number [0 – 9] and/or period [.] and/or underscore [_].

`Password`, this parameter contains information about expected password. This parameter can be blank but client application should prevent the user to add a blank password.

`UserTypeId`, contain the code of user type. If server found this parameter is not defined, it will set this parameter to `0`.

`Email`, contain email address information. This is mandatory field.

`Latitude`, contain the latitude of reference service station. It should be filled with blank or not to be sent if `UserTypeId` parameter was defined as regular user.

`Longitude`, contain the longitude of service station that associated with user that identified by the username parameter. It should be filled with blank or not to be sent if `UserTypeId` parameter was defined as regular user.

`ServiceStationName`, this parameter contains the name of service station that associated with user that identified by the `username` parameter. It should be blank or not to be sent if `UserTypeId` parameter was defined as regular user.

Response

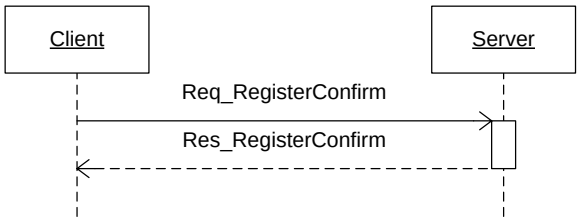
Server will set a response for registration request follow this following data structure:

```
class ResponseRegister
{
    int ErrorCode;
}
```

Error codes

Error Code	Label	Description
0	ErrorSuccess	Error code returned when user has registered successfully.
-1	ErrorInvalidUsername	Error code returned when user gives an invalid or empty username.
-2	ErrorUsernameAlreadyExist	Error code returned when given username is already used by other user.
-3	ErrorInvalidEmailAddress	Error code returned when given email address has invalid format.
-4	ErrorEmailAlreadyExist	Error code returned when given email is already used by other user.
-99	ErrorUnknown	Error code sent if unknown error occurred.

User Registration Confirmation



Send `Req_RegisterConfirm` to server for sending confirmation code back to server.

Request ID

Request ID for `Req_RegisterConfirm` is (in integer) `100002`.

Input Parameters

The data sent to confirm the registration process is follow the data structure below:

```
class RequestRegisterConfirm
{
    string[25] Username;
    string[5] ConfirmationCode;
}
```

`Username`, this parameter contains information about the user identifier to be confirmed.

`ConfirmationCode`, this parameter contains information about confirmation code sent by server and will be sent back by client to server.

Response

Server will send the response after receiving `Req_RegisterConfirm` request. The response data will follow as following data structure:

```
class ResponseRegister
{
    int ErrorCode;
}
```

Error Codes

Error Code	Label	Description
0	ErrorSuccess	Error code returned when user has confirmed successfully.
-1	ErrorUserNotFound	Error code returned when given username cannot be found within the system.
-2	ErrorUserAlreadyRegistered	Error code returned when given username has already registered in the system.
-3	ErrorInvalidConfirmationCode	Error code returned when given confirmation code does not match with the given username.

-99	`ErrorUnknown`	Error code sent if unknown error occurred.

Get List of Service Station Around

Send `Req_GetListServiceStationAround` to server for retrieving list of service station around the given position.

This protocol will not be used until melisa system has sufficient service station data to handle the request itself. Client must make a request directly to google map server instead.

Request ID

Request ID for `GetListServiceStationAround` is `100003`.

Input parameters

The data sent to request the service station around given position follow the following data structure.

```
class RequestListServiceStationAround
{
    float Latitude;
    float Longitude;
}
```

`Latitude`, determine the latitude in degree.

`Longitude`, determine the longitude in degree.

Response

Server will send the response data follow the following data structure.

```
class ResponseListServiceStationAround
{
    int ErrorCode;
    List<EntryServiceStationAround> ServiceStations;
}
```

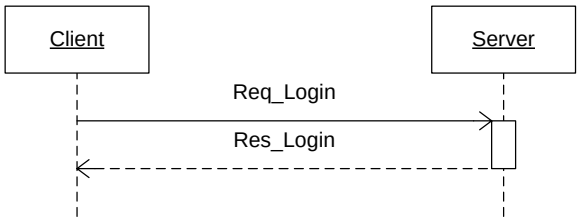
`ErrorCode`, server will send error code as follow:

Code	Label	Description
`0`	`ErrorSuccess`	Returned if server processed the request successfully.
`< 0`		Server will return negative integer if error occurred during process.

`ServiceStations`, if process succeeded, server will send the list of service station entry. Otherwise, this parameter will have null. The entry of service station must follow the following data structure.

```
class EntryServiceStationAround
{
    bigint ServiceStationId;
    string[100] ServiceStationName;
    string[100] ServiceStationAddress;
    Int64 ServiceStationCityId;
    string[50] ServiceStationCityName;
    smallint ServiceStationCountryId;
    string[50] ServiceStationCountry;
    string[10] ServiceStationPostalCode;
}
```

Login



`Req_Login` is sent by clients to server to obtain the authorization for entering the system. Melisa's login does not use or maintain the session id. It means that every client logged will be deemed as existing instance in the system until they perform logout process.

Please note that a user can only login once at the same time using same username. System will disconnect the current logged user connection if there is another connection trying to connect to server using the same username.

Request ID

Request ID for `Req_Login` is `100004`.

Input parameters

The data sent by client to do request login follows the following data structure.

```
class RequestLogin
{
    string[100] Identifier;
    string[64] Password;
}
```

`Identifier`, define the user identifier to login. It can be username or email.

`Password`, define the password of user.

Response

Server will return the response data follow the following data structure.

```
class ResponseLogin
{
    int ErrorCode;
    string[13] AuthorizationKey;
    EntryUserProfile UserProfile;
}
```

`ErrorCode`, returned by server follow the table as following.

Code	Label	Description
0	ErrorSuccess	Returned if server processed the request successfully.
-1	ErrorIdentifierNotFound	Error code returned if server detects user identified by the identifier parameter has never been registered before.

-2	ErrorInvalidPassword	Error code returned password does not matched with given user identifier.
-99	ErrorUnknown	Error code for unknown error.

AuthorizationKey, if login process succeeded, server will fill this parameter with 13 bytes length string. Every request that require the login process must send this key together with the other parameters needed. Otherwise if login process failed, this parameter will be set as empty.

UserProfile, define the profile data of logged user. If login succeeded, server will return user profile. Otherwise, server will return null. Profile data must have the data structure as follow.

```
class EntryUserProfile
{
    Int64 UserId;
    string[25] Username;
    string[50] FirstName;
    string[50] MiddleName;
    string[50] LastName;
    int GenderId;
    string[50] GenderName;
    string[100] Address;
    Int64 CityId;
    string[100] CityName;
    smallint CountryId;
    string[100] CountryName;
    string Flag;
    string[10] PostalCode;
    datetime BirthPlace;
    datetime BirthDate;
    string[20] MobilePhoneNumber;
    string[100] Email;
    string[100] AlternateEmail;
    string[255] WebSite;
    string[255] Notes;
    string Picture;
}
```

Get List of Service Station Inside

Send `Req_GetListServiceStationInside` message to fetch the service stations inside the circle with configured radius. By default, radius is configured as far as 5000 meters.

Request ID

Request ID for `Req_GetListServiceStationInside` is (in integer) `100005`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request must follow the following data structure.

```
class RequestListServiceStationInside
{
    string[32] Key;
    float Latitude;
    float Longitude;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. This parameter is mandatory. See how to perform login process.

`Latitude`, define the latitude in degree. This parameter is mandatory.

`Longitude`, define the longitude in degree. This parameter is mandatory.

Response

Server will send response to client follow the following data structure.

```
class ResponseListServiceStationInside
{
    int ErrorCode;
    int Count;
    List<EntryServiceStationInside> ServiceStations;
}
```

`ErrorCode`, server will return the error code as following.

Code	Label	Description
0	ErrorSuccess	Returned if server processed the request successfully.

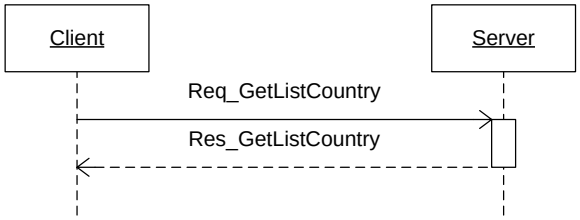
-1	ErrorBadPosition	Error code returned when latitude and/or longitude has/have bad value.
-98	ErrorUserMustBeAuthorizedFirst	Error code returned when user tried to request in logout state.
-99	ErrorUnknown	Unknown error.

Count, server will fill this field with size of ServiceStations field.

ServiceStations, if server succeeded to execute the request, server will return the list of service station entry. Service station entry has structure as following data structure.

```
class EntryServiceStationInside
{
    bigint ServiceStationId;
    float Latitude;
    float Longitude;
    string[100] ServiceStationName;
    string[100] ServiceStationAddress;
    Int64 ServiceStationCityId;
    string[50] ServiceStationCityName;
    smallint ServiceStationCountryId;
    string[50] ServiceStationCountryName;
}
```

Get County List



Client should send `Req_GetListCountry` to ask server returning the list of country around the world.

Request ID

Request ID for `Req_GetListCountry` is (in integer) `100008`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request must follow the structure as follow.

```
class RequestGetListCountry
{
    string[13] Key;
    int SortOption = 0;
    int IsWithFlag = 0;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. See how to perform the login process.

`SortOption`, define the type of sorting be applied to the country list response data. This parameter should contain one of these following options.

Sort Option	Label	Description
0	`SortOptionCountryNameAsc`	Country list response data will be ordered by the country name in ascending.
1	`SortOptionCountryNameDesc`	Country list response data will be ordered by the country name in descending.
2	`SortOptionCountryIdAsc`	Country list response data will be ordered by the country id in ascending.
3	`SortOptionCountryIdDesc`	Country list response data will be ordered by the country id in descending.

`IsWithFlag`, define the option of the request. If this parameter set to `1`, server will return the list of country including its flag image. Default value of this parameter is `0`.

Response

Server will send response to client following format as below.

```
class ResponseGetListCountry
{
    int ErrorCode;
    int Count;
    List<EntryCountry> Countries;
}
```

`ErrorCode`, server will return the error code as follow:

Code	Label	Description
0	`ErrorSuccess`	Returned if server processed the request successfully.
-98	`ErrorUserMustBeAuthorizedFirst`	Error code returned when user tried to request in logout state.
-99	`ErrorUnknown`	Unknown error.

`Count`, server will fill this field with size of the `Countries` parameter.

`Countries`, if server succeeded to execute the request, server will send the list of country entries. A Country entry follows the data structure as below.

```
class EntryCountry
{
    string[5] CountryId;
    string[100] CountryName;
    int FlagId;
    blob Flag;
}
```

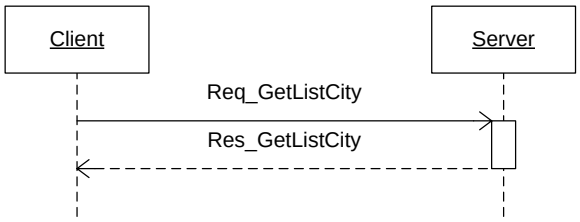
`CountryId`, this field will fill with identifier of country.

`CountryName`, this field will fill with country name corresponding to the `CountryId` field.

`FlagId`, this field will fill with identifier of the flag corresponding to `CountryId` field.

`Flag`, server will fill this field with country flag image if `IsWithFlag` input parameter is set when the request performed. Flag image returned in base64 encoding.

Get City List



`Req_GetListCity` is sent by client to ask server to return the list of cities in specified country.

Request ID

Request ID for `Req_GetListCity` is (in integer) `100007`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request is follow as format below:

```
class RequestGetListCity
{
    string[13] Key;
    string[5] CountryId;
    Int64 CityId;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. See how to perform the login process.

`CountryId`, define the identifier of country that its city will be requested to server. The list of country and its identifier can be obtained by requesting to server. This parameter is mandatory. See how to perform the request to get the country list.

`CityId`, define the identifier of city that will be fetched. Server will return the list that only contains an entry corresponding to the city which it identifier specified by this parameter. If this parameter is not specified, then server will return a list of all cities that are in the country specified by the `CountryId` parameter.

Response

Server will send the response to client following the format as below:

```
class ResponseGetListCity
{
    int ErrorCode;
    int Count;
    List<EntryCity> Cities;
}
```

`ErrorCode`, server will return the error code as follow:

Code	Label	Description
0	`ErrorSuccess`	Returned if server processed the request successfully.
-1	`ErrorCountryIdNotFound`	Error code returned if user gives an unknown country id.
-98	`ErrorUserMustBeAuthorizedFirst`	Error code returned when user tried to request in logout state.
-99	`ErrorUnknown`	Unknown error.

`Count`, server will fill this field with the size of `Cities` parameter.

`Cities`, if server succeeded to execute the request, server will send the list of city entries. A City entry follows the structure as format below.

```
class EntryCity
{
    Int64 CityId;
    string[100] CityName;
    string[5] CountryId;
    string[100] CountryName;
    double Latitude;
    double Longitude;
}
```

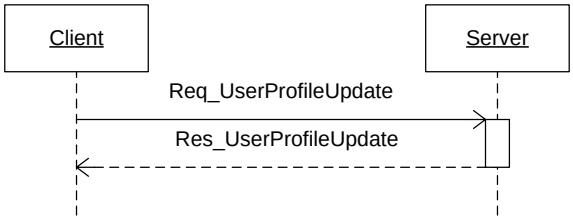
`CityId`, server will fill this field with identifier of the city.

`CityName`, this field will contain the name of city corresponding to the `CityId` field.

`Latitude`, this field will contain the latitude of city corresponding to the `CityId` field.

`Longitude`, this field will contain the longitude of city corresponding to the `CityId` field.

Update User Profile



To update the user profile, client can sent `Req_UserProfileUpdate`.

Request ID

Request ID for `Req_UserProfileUpdate` is (in integer) `100006`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request must follow the format below:

```
class RequestUserProfileUpdate
{
    string[13] Key;
    string[50] FirstName;
    string [50] MiddleName;
    string[50] LastName;
    int GenderId;
    string[100] Address;
    Int64 CityId;
    string[10] PostalCode;
    string[10] BirthPlace;
    date BirthDate;
    string[20] MobilePhoneNumber;
    string[100] Email;
    string[100] AlternateEmail;
    string[255] Website;
    string[255] Notes;
    mediumblob Picture;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. This parameter is mandatory. See how to perform the login process.

`FirstName`, define the first name of user.

`MiddleName`, define the middle name of user.

`LastName`, define the last name of user.

`GenderId`, define the gender identifier of user. Gender identifier values are detailed in the following table.

Gender ID	Description
0	Male
1	Female

`Address`, define the user address. This parameter can be filled by street name, building name, etc.

`CityId`, define the identifier of city. The list of city and its identifier can be obtained by requesting to server. See how to request the list of city around the world or request the list by specified country.

`PostalCode`, define the postal code of user address.

`BirthPlace`, define user place of birth.

`BirthData`, define user date of birth.

`MobilePhoneNumber`, define user mobile phone number.

`Email`, define user email. This parameter will be used by system to verify user in registration process. This parameter is mandatory.

`AlternateEmail`, define user secondary email.

`Website`, define user website URL.

`Notes`, define the notes that important to be marked.

`Picture`, define the user profile picture. Server will receive picture in base46 encode only.

Response

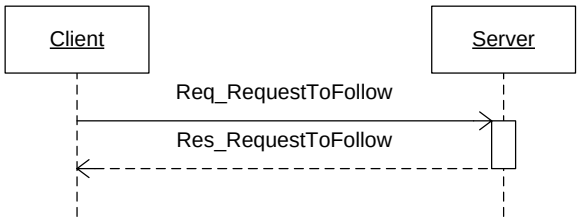
Server will send response to client follow this format below:

```
class ResponseUserProfileUpdate
{
    int ErrorCode;
}
```

`Errorcode`, returned by server follow as table below:

Code	Label	Description
0	ErrorSuccess	Returned if server processed the request successfully.
-1	ErrorEmailIsMandatory	Error code returned if user does not define the email parameter.
-2	ErrorInvalidEmail	Error code returned if user does not define the email parameter or user gave email in invalid format.
-98	ErrorUserMustBeAuthorizedFirst	Error code returned if user tried to request in logout state.
-99	ErrorUnknown	Error code for unknown error.

Request to Follow



`Req_RequestToFollow` message should be performed by client to apply the request for following the other user.

Request ID

Request ID for `Req_RequestToFollow` is (in integer) `100009`.

Pre Conditions

User must login first.

Input Parameters

Input data to perform this request follow the data structure as below:

```
class RequestRequestToFollow
{
    string[13] Key;
    string[100] Identifier;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. This parameter is mandatory. See how to perform the login process.

`Identifier`, define the username or email address of the user that will be followed.

Response

Server will send response to client follow the format below:

```
class ResponseRequestToFollow
{
    int ErrorCode;
    Int64 RequestId;
    bool IsRestricted;
}
```

`ErrorCode`, returned by server follow as table below.

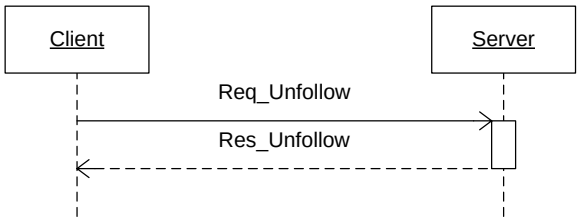
Code	Label	Description
0	ErrorSuccess	Returned if server processed the request successfully.
-1	ErrorIdentifierIsMandatory	Error code returned if no identifier parameter was defined.

-2	ErrorIdentifierNotFound	Error code returned if given identifier cannot be found within the system.
-98	ErrorUserMustBeAuthorizedFirst	Error code returned if user tried to request in logout state.
-99	ErrorUnknown	Error code for unknown error.

`RequestId`, if request process succeeded, server will fill this field with identifier of the following request. Otherwise, this field will be empty.

`IsRestricted`, if request process succeeded and the followed id is an identifier of a restricted user, this parameter will be set to true. Otherwise, it will be set to false.

Stop Following Someone



Client should send `Req_Unfollow` message if user of client in question wants to stop following someone.

Request ID

Request ID for `Req_Unfollow` is (in integer) `100012`.

Pre Conditions

User must login first.

Input Parameters

Input data to perform this request must follow this format bellow.

```
class RequestUnfollow
{
    string[13] Key;
    string[100] Idenfifier;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. This parameter is mandatory. See how to perform the login process.

`Identifier`, define the username or email address of the user that will stop to be followed.

Response

Server will send response to client follow the format below:

```
class ResponseUnfollow
{
    int ErrorCode;
}
```

`ErrorCode`, returned by server follow as table below.

Code	Label	Description
0	ErrorSuccess	Returned if server processed the request successfully.
-1	ErrorIdentifierIsMandatory	Error code returned if no identifier parameter was defined.
-2	ErrorIdentifierNotFound	Error code returned if given identifier cannot be found within the system.

-3	ErrorFollowedIdIsRequested	Error code returned if user tried to make a request to follow someone who has been in the pending list.
-4	ErrorAlreadyFollowed	Error code returned if user tried to follow someone who has followed.
-98	ErrorUserMustBeAuthorizedFirst	Error code returned if user tried to request in logout state.
-99	ErrorUnknown	Error code for unknown error.

Get User's Following List

The `Req_GetListFollowing` request is sent to server when client needs to get the list of anyone who is followed by the user in question.

Request ID

Request ID of `Req_GetListFollowing` is (in integer) `100010`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request follows format below:

```
class RequestGetListFollowing
{
    string[13] Key;
    string[100] Identifier;
    int PageNo;
    int PageRowCount;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. This parameter is mandatory. See how to perform the login process.

`Identifier`, define the username or email address of the user which his followed list will be obtained. This parameter is mandatory.

`PageNo`, define the page number (in zero base) of following list which its rows will be obtained. If this parameter is not defined or defined but filled with negative integer, the server will assume the value of this parameter with `0`.

`PageRowCount`, define the size of following list which is counted starting from the value specified by `PageNo` parameter. If this parameter is not specified or specified but filled with negative integer, the server will assume the value of this parameter with `20`.

Response

Server will send the response to client following the format as below:

```
class ResponseGetListFollowing
{
    int ErrorCode;
    int FollowedCount;
    int ListCount;
    List<EntryFollowed> Followeds;
```

```
}
```

`ErrorCode`, returned by server follow as table below.

Code	Label	Description
0	`ErrorSuccess`	Returned if server processed the request successfully.
-1	`ErrorIdentifierIsMandatory`	Error code returned if no identifier parameter was defined.
-2	`ErrorIdentifierNotFound`	Error code returned if given identifier cannot be found within the system.
-98	`ErrorUserMustBeAuthorizedFirst`	Error code returned if user tried to request in logout state.
-99	`ErrorUnknown`	Error code for unknown error.

`FollowedCount`, server will fill this field with the total number of people followed by the user in question.

`ListCount`, server will fill this field with the size of `Followeds` field.

`Followeds`, if server succeeded to execute the request, server will send the followed people entries of user in question. A followed-people entry follows the structure as format below.

```
class EntryFollowing
{
    Int64 UserId;
    Int64 FollowedId;
    string[25] Username;
    int UserTypeId;
    string UserTypeName;
    Int64 ServiceStationId;
    string[100] ServiceStationName;
    double Latitude;
    double Longitude;
}
```

`UserId`, this field will contain the identifier of user who follows another user specified by the value of `FollowedId` field.

`FollowedId`, server will fill this field with identifier of user who is followed by user specified by the value of `UserId` field.

`Username`, this field will contain the username of user specified by the value of `FollowedId` field.

`UserTypeId`, contain the code of user type.

`UserTypeName`, contain the label of user type specified by the value of `UserTypeId` field.

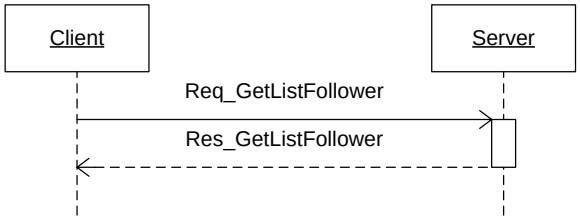
`ServiceStationId`, server will fill this field with identifier of service station that associated with the value of `FollowedId` field.

`ServiceStationName`, contain the label/caption of service station specified by the value of `ServiceStationId` field.

`Latitude`, contain the latitude of service station specified by the value of `ServiceStationId` field.

`Longitude`, contain the longitude of service station specified by the value of `ServiceStationId` field.

Get User's Follower List



`Req_GetListFollower` request is sent to server when client needs to obtain the follower list of user in question.

Request ID

Request ID of `Req_GetListFollower` is (in integer) `100011`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request follows format as below:

```
class RequestGetListFollower
{
    string[13] Key;
    string[100] Identifier;
    int PageNo;
    int PageRowCount;
}
```

`Key`, define the authorization key. Client obtains the authorization key through the login process. This parameter is mandatory. See how to perform the login process.

`Identifier`, define the username or email address of the user which his follower list will be obtained. This parameter is mandatory.

`PageNo`, define the page number (in zero base) of follower list which its rows will be obtained. If this parameter is not defined or defined but filled with negative integer, the server will assume the value of this parameter with `0`.

`PageRowCount`, define the size of follower list which is counted starting from the value specified by `PageNo` parameter. If this parameter is not specified or specified but filled with negative integer, the server will assume the value of this parameter with `20`.

Response

Server will send the response to client following the format as below:

```
class ResponseGetListFollower
{
    int ErrorCode;
    int FollowerCount;
    int ListCount;
    List<EntryFollower> Followers;
```

```
}
```

`ErrorCode`, returned by server follow as table below.

Code	Label	Description
0	`ErrorSuccess`	Returned if server processed the request successfully.
-1	`ErrorIdentifierIsMandatory`	Error code returned if no identifier parameter was defined.
-2	`ErrorIdentifierNotFound`	Error code returned if given identifier cannot be found within the system.
-98	`ErrorUserMustBeAuthorizedFirst`	Error code returned if user tried to request in logout state.
-99	`ErrorUnknown`	Error code for unknown error.

`FollowerCount`, server will fill this field with the total number of follower.

`ListCount`, server will fill this field with the size of `Followers` field.

`Followers`, if server succeeded to execute the request, server will send the follower entries of user in question. A follower entry follows the structure as format below.

```
class EntryFollower
{
    Int64 UserId;
    Int64 FollowerId;
    string[25] Username;
    int UserTypeId;
    string UserTypeName;
    Int64 ServiceStationId;
    string[100] ServiceStationName;
    double Latitude;
    double Longitude;
    bool IsBlocked;
}
```

`UserId`, this field will contain the identifier of user who is followed by another user specified by the value of `FollowerId` field.

`FollowerId`, server will fill this field with identifier of user who follows the user specified by the value of `UserId` field.

`Username`, this field will contain the username of user specified by the value of `FollowerId` field.

`UserTypeId`, contain the code of user type.

`UserTypeName`, contain the label of user type specified by the value of `UserTypeId` field.

`ServiceStationId`, server will fill this field with identifier of service station that associated with the value of `FollowerId` field.

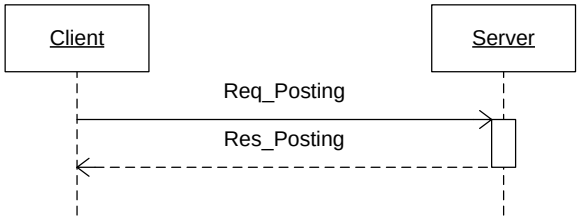
`ServiceStationName`, contain the label/caption of service station specified by the value of `ServiceStationId` field.

`Latitude`, contain the latitude of service station specified by the value of `ServiceStationId` field.

`Longitude`, contain the longitude of service station specified by the value of `ServiceStationId` field.

`IsBlocked`, if this field contains true, user specified by the `FollowerId` parameter is currently blocked by user in question.

Posting



To drop a posting on the post land, client should send `Req_Posting` message to server. The posting can contain text or media type file (image and video) or both of them.

Request ID

Request ID for `Req_Posting` (in integer) is `100012`.

Pre Conditions

User must login first.

Input Parameters

Input data for this request follows format as below:

class RequestPosting

{

    string[13] Key;

    string[500] Message;

}

Response

User Types

User types recognized by system are as listed in the following table.

Code	User Type
0	Regular
1	Service station
2	Gas station (not available yet)
99	Unknown user type