

## MANUAL AND DOCUMENTATION



*Breaking down barriers between carriers*



## Table of Contents

<b>User Manual</b>	<b>- 1 -</b>
<i>Getting Started</i>	- 1 -
What you need	- 1 -
Creating your mEYEtrak Account	- 1 -
Setting up your mEYEtrak Account on Device	- 2 -
<i>Windows Mobile 6 Professional Client</i>	- 3 -
Using Buddy Info	- 4 -
Using Buddy Request Alerts	- 5 -
Using Buddy in Range Alerts	- 6 -
User Settings	- 7 -
Buddy Map (coming soon)	- 8 -
<i>Web Interface</i>	- 9 -
Buddy List	- 10 -
Settings	- 11 -
<b>Server Documentation</b>	<b>- 12 -</b>
<i>Communicating with the Server</i>	- 12 -
<i>Developing Trakets</i>	- 12 -
Basic requirements for designing a traket	- 12 -
<i>Database</i>	- 13 -
<i>Errors</i>	- 13 -
<i>Security</i>	- 13 -
<b>Website Documentation</b>	<b>- 14 -</b>
<b>Trakets Documentation</b>	<b>- 15 -</b>
<i>General Notes</i>	- 15 -
HTTPS	- 15 -
Timestamp	- 15 -
Scenarios	- 15 -
<i>Standard Form</i>	- 15 -
Request	- 15 -
Response	- 16 -
<i>Credentials</i>	- 16 -
Elements	- 16 -
Scenarios	- 16 -
<i>Account Info</i>	- 17 -



Attributes	- 17 -
Elements	- 17 -
Scenarios	- 17 -
<i>Buddy</i>	- 18 -
Elements	- 18 -
Scenarios	- 19 -
<i>Add Buddy</i>	- 19 -
Elements	- 19 -
Scenarios	- 20 -
<i>Remove Buddy</i>	- 20 -
Elements	- 20 -
Scenarios	- 20 -
<i>Avatars</i>	- 20 -
Elements	- 20 -
Scenarios	- 21 -
<i>Errors</i>	- 21 -
Error Codes	- 22 -
<i>Request Buddy Location</i>	- 22 -
Elements	- 22 -
Scenarios	- 22 -
<i>Send Location</i>	- 23 -
Elements	- 23 -
Scenarios	- 23 -
<i>Update Location</i>	- 23 -
Elements	- 23 -
Scenarios	- 23 -
<i>Update Status</i>	- 24 -
Elements	- 24 -
Scenarios	- 24 -
<b>Additional Features</b>	<b>- 25 -</b>
<i>Stolen PhoneTrak</i>	- 25 -
<i>Today Screen Menu Item</i>	- 25 -
<i>Phone Notifications Outside the Application</i>	- 25 -
<i>Add Buddies from Phone Contacts</i>	- 25 -
<i>Location-based Advertisements</i>	- 25 -



## User Manual

### Getting Started

#### What you need

To use mEYEtrak, you need:

- An Internet browser that is SSL compliant
- A Mobile Phone running Windows Mobile 6.1 Professional
- An active Internet Connection<sup>i</sup>

#### Creating your mEYEtrak Account

Before you can use any mEYEtrak and any of its services, you must first create and validate an account.

A mEYEtrak account may be created at <http://meyetrak.fit.edu/?id=register>

In order to successfully register you must provide:

- E-mail address<sup>ii</sup>
- Strong Password
- Cell phone carrier<sup>iii</sup>
- Phone number<sup>iv</sup>
- First Name
- Last Name

After creating an account, 2 unique verification codes will be sent to you; one via SMS<sup>v</sup> and the other via E-Mail. Upon providing these codes, along with your 10 digit phone number access will be granted to your mEYEtrak account.

A screenshot of a registration form with a black background and white text. It contains three input fields labeled "Phone Number:", "TXT Validation Code:", and "Email Validation Code:". Below these fields is a "Submit" button. The entire form is enclosed in a red rectangular border.

Congratulations, you are now a validated mEYEtrak user; you can now begin connecting with your friends in ways not previously possible!

### Setting up your mEYEtrak Account on Device

Please point your Device's browser to <http://meyetrak.fit.edu/?id=download> . The provided file will install mEYEtrak on your device.

After the installation completes mEYEtrak will be located in the "Installed Programs" section of your device. Anytime you wish to view mEYEtrak updates you may go there and click on the icon.

The application will then launch, if you are running mEYEtrak for the first time on this device you will be prompted for your username and password.



If you are unable to provide valid credentials you will be provided a link that will direct you to the password recovery page: [https://meyetrak.fit.edu/?id=recover\\_password](https://meyetrak.fit.edu/?id=recover_password), from which we will help you revalidate your ownership of this account by sending another combination of E-Mail and SMS<sup>v</sup> validation codes.

## Windows Mobile 6 Professional Client

After successfully logging into the mEYEtrak application you will be taken to your Buddy list page.



Here you have access to a list of all of your friends, you are also able to see their status, distance from you<sup>vi</sup> as well as the last time they updated their information.

**Update Status:** You may also update your status, by inputting text into the status box and clicking 'update'.

**Add Buddy:** Additional users may be added to your buddy list by clicking on the '+'. You will be directed to the "Add Buddy" page where you can input either the telephone number or email address of your desired buddy. A add buddy alert will be sent to this buddy, to which they can either accept or reject.

**Buddy Info:** More specific information and options for each buddy can be displayed by simply clicking on any individual buddy.

**Menu:** From here you can logout of the service or exit the application.

The information about each buddy will be updated at a frequency you determine. In this way you will only get updates about your buddies at intervals which are convenient for you.

Please be aware that by clicking "X" at the top of the application, mEYEtrak will continue to run in the background. To terminate the application you must explicitly click Menu >> Exit.

## Using Buddy Info

If you click on any specific buddy you will be taken to “Buddy Info” here there is a whole host of ways you may interact with this buddy.



**Buddy Info:** Your buddies First and Last Name, Status and Avatar are all displayed here at the top

**E-Mail/SMS/Call:** You are provided with the email address, and phone number your buddy has used to validate their account. Clicking on any one of these you will start an alternate communication session: E-Mail, Phone Call or SMS<sup>v</sup> respectively.

**Alert Radius:** Here you can set the range in which you wish to be notified that your buddy is ‘in range’. If you set this to 0 you will never be alerted about this buddy.

**Privacy:** You may set the level of privacy you wish to have with this particular buddy. You may set this to:

- **Private:** Buddy will not see any location information about you
- **Semi-Private:** Buddy will see an approximation of your location.
- **Public:** Buddy will see your exact location.

**Delete:** You may choose to no longer be associated with any particular buddy; in this case you may click on ‘Delete’. As further verification, a confirmation box will pop up to ensure that you really want to delete this buddy. They will also be forced to delete you as a buddy.

## Using Buddy Request Alerts

In the event that something happens mEYEtrak will notify you of this occurrence.

When this happens the “Alert” tab will show the number of alerts you currently have, after selecting this tab you will have the following options.



**Friend Request:** In the event that another user requests to add you as a buddy (see [Buddy Requests](#)), you have the option to either accept them or not.

**Accept Buddy:** This user will be added to your buddy list. You will also be added to their buddy list. Now you may set privacy and alert relationships with this user.

**Reject Buddy:** This will ignore the user’s “Buddy Request”. They will **not** be notified of this.



## Using Buddy in Range Alerts

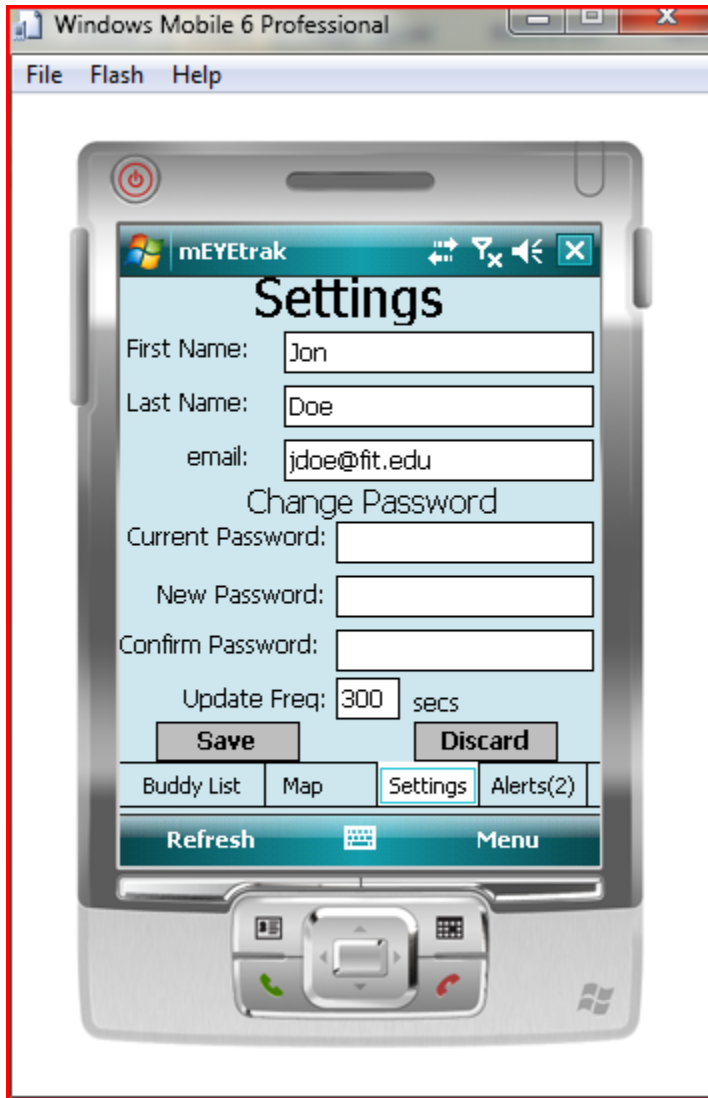
This is a list of all of the buddies that fall within the “Alert Radius” which you have previously specified for them.



**Dismiss:** This will remove this alert from the “Buddies in Range” tab. If this buddy goes out of range, and then comes back in range, you will be notified again.

## User Settings

This Screen will allow you to change your contact information, along with the frequency you wish to update your buddy information from the server (default 5 minutes).



**Save:** Will Save the changes you have submitted. If you are attempting to change your password, the 'new' and 'confirm' password fields **must match**.

**Discard:** All of the changes you have made will be reset to the previously saved settings.

### Buddy Map (coming soon)

The buddy map is where you may come to get a visual representation of the location of your buddies<sup>vi</sup>.

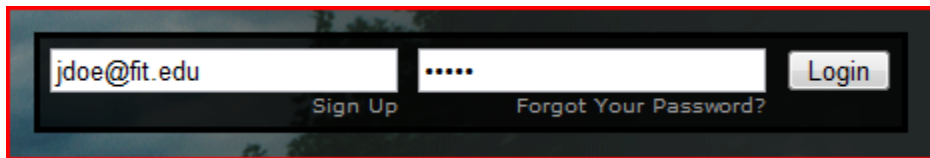


## Web Interface

You may also log into your account utilizing our Web Interface. From this you can interact with your buddies and account in much the same way you have done on the mobile mEYEtrak application.

Point your browser to <http://meyetrak.fit.edu>

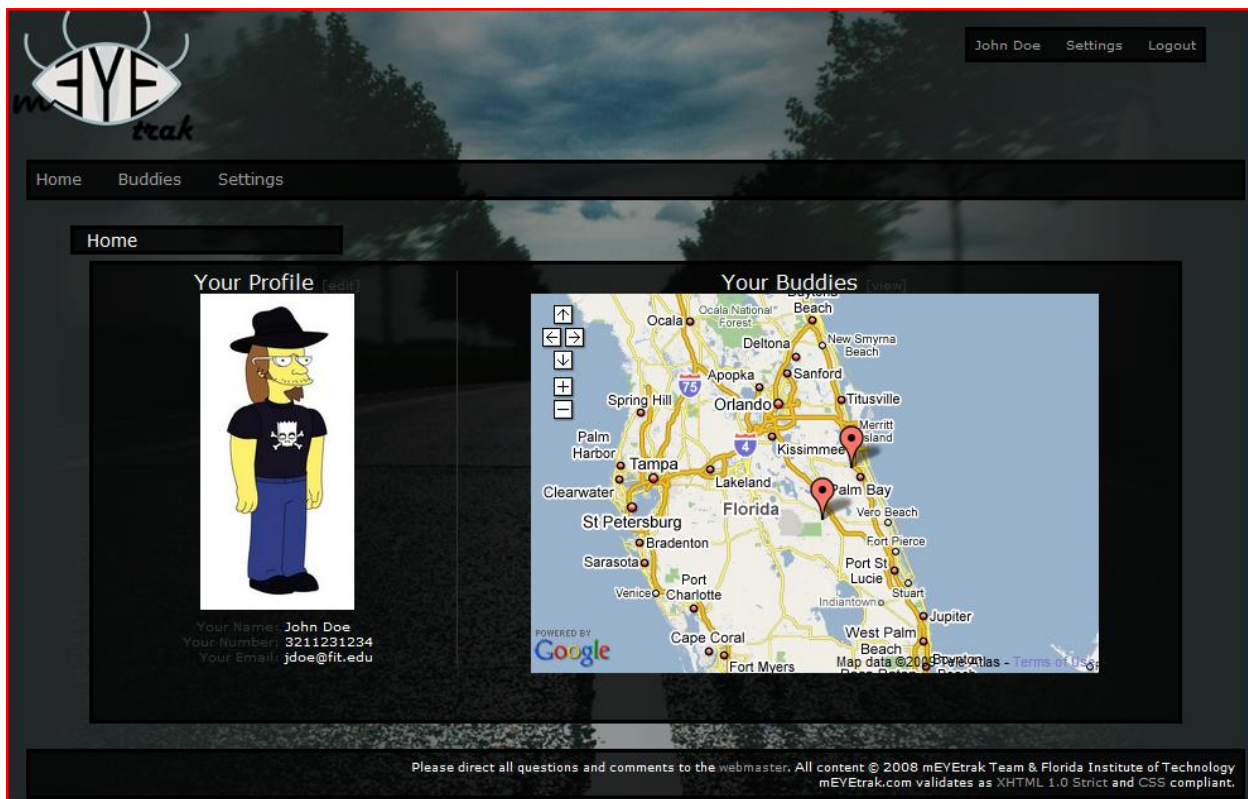
Login at the top right-hand corner, for account creation please see [Creating your mEYEtrak Account](#).



A screenshot of the login form on the mEYEtrak web interface. It features a dark background with a red border. There are two input fields: the first contains the email address 'jdoe@fit.edu' and the second contains a masked password '.....'. Below the email field is a 'Sign Up' link, and below the password field is a 'Forgot Your Password?' link. To the right of the password field is a 'Login' button.

Here you enter the email address and password you have previously registered with mEYEtrak.

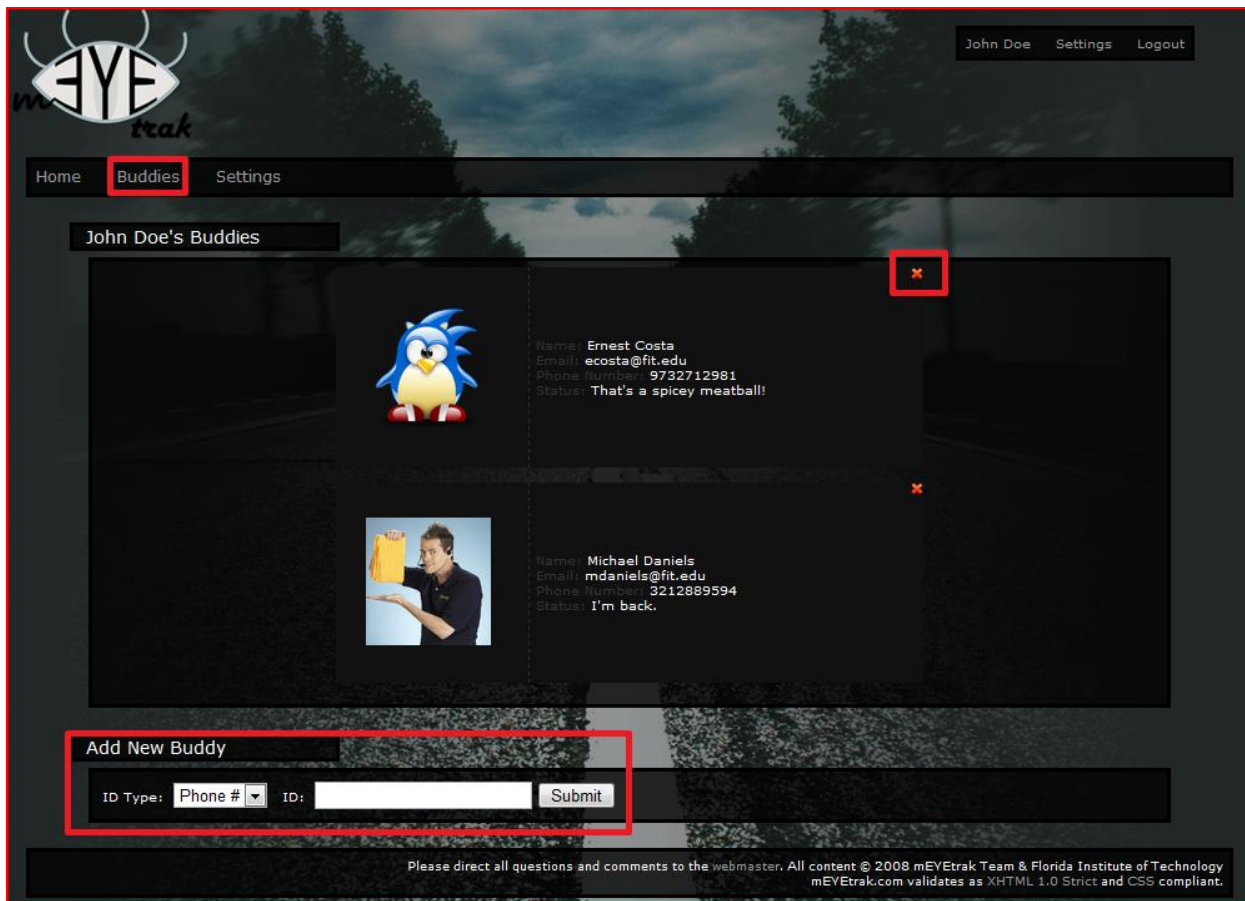
If your credentials are accepted you will be directed to your profile. Here you will be able to see your personal information as well as a map of where your buddies actually are<sup>vi</sup>.



A screenshot of the mEYEtrak web interface showing a user's profile and a map of their buddies. The interface has a dark theme with a red border. At the top left is the mEYEtrak logo. At the top right are links for 'John Doe', 'Settings', and 'Logout'. Below the logo is a navigation bar with 'Home', 'Buddies', and 'Settings'. The main content area is divided into two sections: 'Your Profile' on the left and 'Your Buddies' on the right. The 'Your Profile' section shows a cartoon avatar of a man with a hat and a skull on his shirt, and text indicating 'Your Name: John Doe', 'Your Number: 3211231234', and 'Your Email: jdoe@fit.edu'. The 'Your Buddies' section shows a map of Florida with several red location pins. The map is powered by Google. At the bottom of the page is a footer with copyright information: 'Please direct all questions and comments to the webmaster. All content © 2008 mEYEtrak Team & Florida Institute of Technology. mEYEtrak.com validates as XHTML 1.0 Strict and CSS compliant.'

## Buddy List

Selecting the Buddies Link will direct you to a view of all your buddies where you can add/remove buddies.





The screenshot shows the mEYEtrak web application interface. At the top right, there is a user profile section for "John Doe" with links for "Settings" and "Logout". Below this is a navigation bar with "Home", "Buddies" (highlighted with a red box), and "Settings". The main content area is titled "John Doe's Buddies" and contains a list of buddies. Each buddy entry includes a profile picture, a red 'X' icon for removal, and a list of details: Name, Email, Phone Number, and Status. Two buddies are listed: Ernest Costa (with a Sonic the Hedgehog avatar) and Michael Daniels (with a photo of a man). At the bottom, there is an "Add New Buddy" section with a dropdown menu for "ID Type" (set to "Phone #"), an "ID:" input field, and a "Submit" button. The footer contains a copyright notice for 2008 mEYEtrak Team & Florida Institute of Technology.



John Doe   Settings   Logout

Home   **Buddies**   Settings

John Doe's Buddies

Name: Ernest Costa  
Email: ecosta@fit.edu  
Phone Number: 9732712981  
Status: That's a spicy meatball!

Name: Michael Daniels  
Email: mdaniels@fit.edu  
Phone Number: 3212889594  
Status: I'm back.

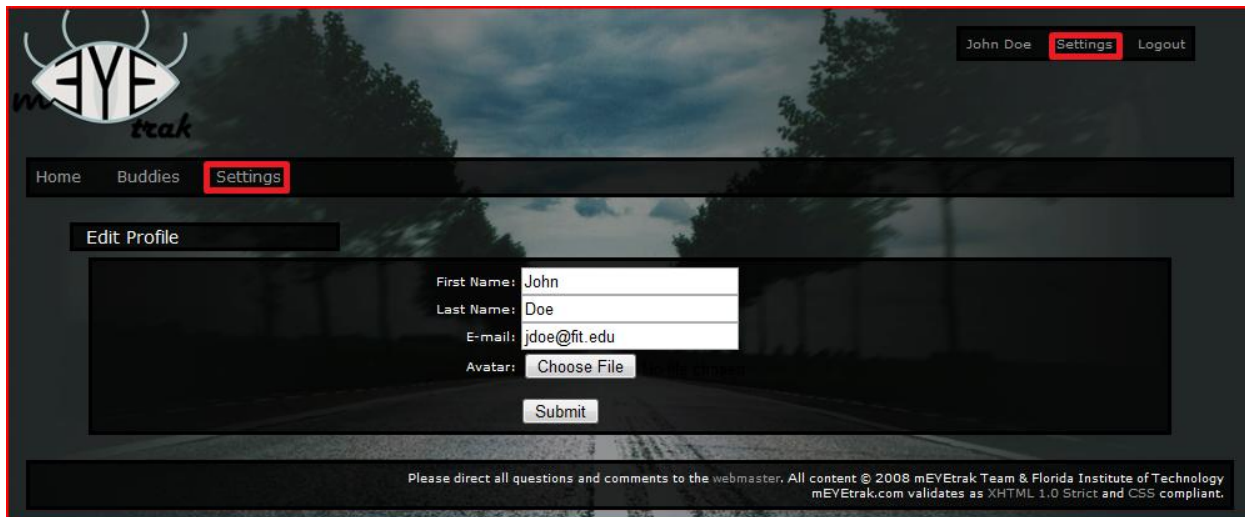
Add New Buddy

ID Type: Phone #   ID:   Submit

Please direct all questions and comments to the webmaster. All content © 2008 mEYEtrak Team & Florida Institute of Technology  
mEYEtrak.com validates as XHTML 1.0 Strict and CSS compliant.

## Settings

Selecting the 'Settings' link will direct you to a view of your settings where you can update your contact information, including avatars.



The screenshot shows the mEYEtrak web application interface. At the top right, the user is logged in as 'John Doe' with links for 'Settings' (highlighted with a red box) and 'Logout'. Below the header is a navigation bar with 'Home', 'Buddies', and 'Settings' (also highlighted with a red box). The main content area is titled 'Edit Profile' and contains a form with the following fields: 'First Name' (John), 'Last Name' (Doe), 'E-mail' (jdoe@fit.edu), and 'Avatar' (with a 'Choose File' button). A 'Submit' button is at the bottom of the form. The background of the page is a dark, atmospheric image of a road. At the very bottom, a footer contains copyright information: 'Please direct all questions and comments to the webmaster. All content © 2008 mEYEtrak Team & Florida Institute of Technology. mEYEtrak.com validates as XHTML 1.0 Strict and CSS compliant.'

- 
- i Unlimited Data plan Recommended.
  - ii An e-mail address and phone number is required to validate an account.
  - iii Will be used to determine the domain needed to communicate with device via SMS.
  - iv If your account is not validated within 24 hours your credentials will be purged from our system.
  - v Standard rates apply.
  - vi User location will be provided according to the privacy level your buddy establishes with you.

## Server Documentation

### Communicating with the Server

- Invoking an action on the server is similar to calling a function in the client side code, except all data regarding the call is passed via XML
  - The action type is specified by the child's node name
  - The variables associated with the action are specified via the node's children
    - The child's name specifies the variable name
    - The child's content specifies the data assigned to that variable
- All action requests must be specified as children of the 'requests' node
- If the client expects a response from the trak, an 'id' parameter in the XML must be specified for the trak.
- Authentication of the client occurs with every trak. The user's credentials must be passed via the 'credentials' node
- The server provides a generic interface to the server-side traks allowing clients of any type (web, phone, pc) to communicate with it via XML over HTTPS

### Developing Traks

mEYetrak was designed with the possibility of expandability in mind; to that end all server-side functionality related to the client is contained within the "trak" modules. These traks are designed to be developed independently of the rest of the system, with only a few basic requirements to allow them to communicate with the client. A current list and descriptions of traks can be found in [Traks Documentation](#).

#### Basic requirements for designing a trak

- All traks currently available to the client must be declared in '/utils/Trak.php'
- All traks that modify (dirty) any data in the backend-database must explicitly declare those functions in their constructor in the '\$this->dirtyList' array.
- All responses to the client must be added to the '\$this->response' XML object. All headers for the child are handled by the Application; all that is required from the trak is the sub-children.
- All actions accessible by the client must bear the same name as the action its specified in '/utils/Trak.php'
- All actions accessible by the client must take in a variable called '\$children', which is a pointer to the XML children passed to it by the client.
- All actions which communicate with the database should utilize the global '\$this->database' object which handles database connections/queries for the Application
- There are several helper functions / variables available
- A global unction to sanitize inputs for individual databases is accessible via '\$this->database->sanitize()'
- A global function for pulling data directly from the database and passing it to the client called '\$this->constructXMLFromSQL()', it is documented in '/utils/Trak.php.'



- Information on the current user is available via \$\_SESSION
  - user\_id
  - email
  - name\_last
  - name\_first
  - phone\_number
  - The id of the current user
  - The email address of the current user
  - The last name of the current user
  - The first name of the current user
  - The phone number of the current user

## Database

1. A database object is globally available in the Application.
2. Database objects should be created via the DatabaseFactory, which will take the credentials from and the required database type and create the connection and pass it back.
3. All queries to the database should use the Database class' 'query()' function.

## Errors

1. All errors passed to the client should be specified in '/vars/errors.inc.php'
2. A current list of errors is defined in [Error Codes](#).

## Security

Security is handled several functions provided by /utils/SecurityHeader.php

authenticate()	authenticates a user's credentials
isAuthenticated()	determines if the current user has been authenticated
logout()	logs the user out and destroys the associated session



## Website Documentation

- All pages accessible via the website must be placed in the '/pages/' directory.
- All pages are accessed via the 'id' get variable which maps directly to the page's filename in the pages directory.
- All pages accessible via the website must end with the extension of either '.ajax.php' or '.ajax.html'
- All pages have two global arrays associated with them to allow for site templates to be implemented.
- \$page: this array contains variable associated with the global site template.

• Auth	• the authentication box
• Menu	• the site menu
• Date	• the current date
• Title	• the page title
• Content	• the content
• Javascript	• javascript
• Onload	• the javascript function to be executed on page load
• Onunload	• the javascript function to be executed on page unload

- \$pagvars: this array contains variable which are specified by the developer and will be parsed into \$page['content']

## Trakets Documentation

### General Notes

#### HTTPS

All communication with the server occurs over HTTPS. This is why text in the XML is not encrypted. HTTPS is HTTP tunneled over SSL, which is inherently encrypted. More information can be found by visiting <http://en.wikipedia.org/wiki/Https>.

#### Timestamp

Timestamps are a part of every XML element. They are in UNIX timestamp form. There is no specific purpose for them, but in some cases can be used for monitoring when certain actions occur. When generating a request from the client, timestamps should be automatically generated using the current time, such as `DateTime.Now` (in C#).

#### Scenarios

There are two possible scenarios: Client to Server and Server to Client. Not all elements are two-way. If only one scenario is given for an element, then it can be assumed that the other direction is invalid and should not be used.

### Standard Form

#### Request

The standard form of an XML request is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<xml>
  <credentials>
    <email/>
    <password/>
  </credentials>
  <requests>
    <buddy id="buddy_list" action="buddy_list"/>
    <mobile id="mobile" action="everything"/>
    <user id="account_info" action="account_info"/>
    <avatars id="avatars" action="verify"/>
  </requests>
</xml>
```

#### Notes

1. The "id" attributes are used to define what the response of the request is returned as. For example, `<buddy id="buddy_list" action="buddy_list"/>` is used to request the user's buddylist. The response will be in an element with the name "buddy\_list". If the "id" of the element was instead "bl", then the response element would have the name "bl".
2. All added elements should be appended as children in the `<requests>` node.

3. As a general rule, the four (4) children in requests should always be present when sending a request from a client such as a phone. This will allow the buddylist, account information, alerts, and avatars to be up-to-date as of the most recent request.

## Response

The standard form of an XML response is as follows:

```
<?xml version="1.0"?>
<xml>
  <buddy_list></buddy_list>
  <mobile>
    <alert_queue></alert_queue>
    <buddy_requests></buddy_requests>
  </mobile>
  <account_info></account_info>
  <avatars></avatars>
</xml>
```

## Notes

1. This is the basic form of the response given the above request.
2. `<buddy_list>` will have [buddy](#) children.
3. `<alert_queue>` will have children that are alerts such as [location requests](#).
4. `<buddy_requests>` will have children that are [add buddies](#).
5. `<account_info>` will be in this [user](#) form.
6. `<avatars>` will have children that are [avatars](#).

## Credentials

```
<credentials>
  <timestamp>-62135578800</timestamp>
  <email>default value</email>
  <password>default value</password>
</credentials>
```

## Elements

### *email*

The email address of the current user.

### *password*

The user's password.

## Scenarios

### *Client to Server*

A credential element should always be present in the XML request to the server. If it is not, the server will not be able to authenticate the user.

## Account Info

```
<user action="default value">
  <timestamp>-62135578800</timestamp>
  <email>e@mail.com</email>
  <id>0</id>
  <last_name>default value</last_name>
  <first_name>default value</first_name>
  <status>default value</status>
  <phone_number>default value</phone_number>
</user>
```

## Attributes

### *action*

The action specifies the purpose of the element.

## Elements

### *email*

The user's email address.

### *id*

The user's id on the server as assigned by the server.

### *last\_name*

The user's last name.

### *first\_name*

The user's first name.

### *status*

The user's status.

## Scenarios

### *Client to Server*

This can be sent to the server to update certain user information, such as the first name.

### *Server to Client*

This can be sent from the server in the event that the user's information is requested by the client. Generally this is always requested by the client to insure that the information is up to date in the event that the information is updated via the web interface.

## Buddy

```
<buddy>
  <timestamp>1240369082</timestamp>
  <email>e@mail.com</email>
  <id>0</id>
  <last_name>default value</last_name>
  <first_name>default value</first_name>
  <distance>-1</distance>
  <notify_distance>0</notify_distance>
  <status>default value</status>
  <phone_number>default value</phone_number>
  <privacy>2</privacy>
  <latitude>-1</latitude>
  <longitude>-1</longitude>
  <location_time>-2208970800</location_time>
</buddy>
```

## Elements

### *email*

The buddy's email address.

### *id*

The buddy's server-assigned id.

### *last\_name*

The buddy's last name.

### *first\_name*

The buddy's first name.

### *distance*

The distance (calculated by the server) of the buddy from the current user. If the distance is not known, then the value is "-1".

### *notify\_distance*

The user's setting for the buddy. This defines the upper bound of the distance for which the user will be notified.

### *status*

The buddy's status.

### *phone\_number*

The buddy's phone number. This is defined in the format "#####". The client should handle conversion into whichever format is preferred.



### *privacy*

This defines the user's privacy setting for the buddy. Possible values are:

0

Private - the buddy will not be able to see any location information about the user.

1

Semi-private - the buddy will only be able to see the distance of the user.

2

Public – the buddy will be able to see the distance as well as the exact coordinates of the user.

### *latitude*

The buddy's latitude. The value is "-1" if the value is unknown.

### *longitude*

The buddy's longitude. The value is "-1" if the value is unknown.

### *location\_time*

The UNIX time of the buddy's location.

## **Scenarios**

### *Client to Server*

If the client sends this element to the server, an attribute "action" should be added to the element with the value "update". This can be used to update settings such as the privacy or notify\_distance.

### *Server to Client*

Buddy elements are sent to the client when the buddy list is requested.

## **Add Buddy**

```
<buddy action="add">
  <timestamp>1240369082</timestamp>
  <email>e@mail.com</email>
  <id>0</id>
  <phone_number>default value</phone_number>
</buddy>
```

## **Elements**

### *email*

The buddy's email address.

### *id*

The buddy's server-assigned id.

### *phone\_number*

The buddy's phone number.



## Scenarios

### *Client to Server*

This element can be sent to the server to add a buddy. In this scenario, only the email or the phone number is needed (not both).

### *Server to Client*

This element can be sent from the server in the event that someone has added the user as a buddy. It is used to notify the user that there is a pending buddy request.

## Remove Buddy

```
<buddy action="remove">
  <timestamp>1240369078</timestamp>
  <email>e@mail.com</email>
  <id>0</id>
</buddy>
```

## Elements

### *email*

The buddy's email address.

### *id*

The buddy's server-assigned id.

## Scenarios

### *Client to Server*

This can be sent to the server to either remove a buddy, or to reject a buddy request.

## Avatars

```
<avatar>
  <timestamp>-62135578800</timestamp>
  <id>0</id>
  <hash>default value</hash>
  <file>default value</file>
  <type>default value</type>
</avatar>
```

## Elements

### *id*

The server-assigned id of the user pertaining to the avatar.

### *hash*

The hash of the avatar. This is generated by the server.

### *file*

A hex representation of the avatar file.

### *type*

The mime type of the file.

## **Scenarios**

### *Client to Server*

Avatar elements can be sent to the server to verify that the current avatars on the client are up-to-date. In this event, only the id and hash are needed.

### *Server to Client*

Avatar elements are sent to the client when a verify action is received from the client. A buddy's avatar is returned in two cases:

#### Case 1

The hash sent by the client does not match the hash of the avatar on the server (i.e. the client has an outdated avatar).

#### Case 2

The client did not send a hash for a buddy (i.e. the client doesn't have any avatar for the buddy).

## **Errors**

`<error>000</error>`

The value in the error element is an error code that correlates to the following table:



## Error Codes

Code	Error
<b>000</b>	unknown_error
<b>001</b>	connection_notSecure
<b>002</b>	database_error
<b>101</b>	credentials_incorrectFormat
<b>102</b>	credentials_invalid
<b>103</b>	credentials_inactive
<b>201</b>	traket_invalid
<b>202</b>	traket_invalidParameters
<b>203</b>	traket_buddy
<b>301</b>	xml_error
<b>302</b>	xml_notSet
<b>303</b>	xml_noAction
<b>304</b>	xml_badParams
<b>401</b>	buddy_exists
<b>402</b>	buddy_requestNotFound
<b>403</b>	buddy_userNotBuddy
<b>404</b>	buddy_pending
<b>501</b>	user_DNE
<b>502</b>	user_alreadyExists
<b>601</b>	alert_DNE
<b>602</b>	alert_alreadyExists

## Request Buddy Location

```
<request_location>
  <timestamp>1240369081</timestamp>
  <email>e@mail.com</email>
</request_location>
```

## Elements

### *email*

An email address. Interpretation varies depending on the direction of the request.

## Scenarios

### *Client to Server*

The user does not currently know the location of the buddy (i.e. the buddy's privacy settings prevent it), and the user wants to request the location. In this situation, the email address is that of the buddy whose location is being requested.

### *Server to Client*

A buddy has requested the user's location. In this situation, the email address is that of the requesting buddy.



## Send Location

```
<send_location>  
  <timestamp>1240369080</timestamp>  
  <email>e@mail.com</email>  
  <seconds_valid>0</seconds_valid>  
</send_location>
```

### Elements

#### *email*

The recipient's email address for the user's location.

#### *seconds\_valid*

The number of seconds the buddy will be able to see the user's location. If the value is "0", then the buddy will be able to view the user's location until the user manually changes the privacy setting.

### Scenarios

#### *Client to Server*

The client wishes to allow a buddy, who cannot currently see the user's location, to see their location. The request changes the privacy setting for the buddy (either temporarily or permanently depending on the value of 'seconds\_valid').

## Update Location

```
<gps action="update">  
  <timestamp>1240369081</timestamp>  
  <longitude>0</longitude>  
  <latitude>0</latitude>  
</gps>
```

### Elements

#### *longitude*

The longitude value returned by the GPS.

#### *latitude*

The latitude value returned by the GPS.

### Scenarios

#### *Client to Server*

The client is updating the current location of the user. This should be provided in every request so long as the GPS can provide an accurate location.

## Update Status

```
<user action="status_set">  
  <timestamp>-62135578800</timestamp>  
  <message>default value</message>  
</user>
```

## Elements

### *message*

The new status message.

## Scenarios

### *Client to Server*

The user wishes to change the current status.



## **Additional Features**

There is an endless list of features that could be added to mEYEtrak. Unfortunately, time constraints limited the amount of features that were implemented during the span of the project. Below is just a few of the features that would be desirable in the future.

### **Stolen PhoneTrak**

The idea behind Stolen PhoneTrak is to be able to track your phone while maintaining security if the phone is lost or stolen. A few requirements would need to be met to succeed in this task:

1. Make sure the application starts when the phone is turned on.
2. Prevent the application from being terminated.
3. Prevent any actions from within the application (including viewing alerts, buddies, etc.)

### **Today Screen Menu Item**

Windows Mobile implements items that are viewed from the phone's home screen (Today Screen Menu Items). The application could implement such a menu item that would show the current status of the mEYEtrak (alerts, connectivity status, etc.)

### **Phone Notifications Outside the Application**

When a new alert is received, the user would be notified similarly to the notification received when a text message is received. This should be an option that can be changed in the application settings. There should be an auditory notification as well as a text-based notification.

### **Add Buddies from Phone Contacts**

The user should be able to add buddies by selecting contacts from their contact list.

### **Location-based Advertisements**

Revenues could be generated by collecting user interests and preferences, combining them with the user's location, and then providing suggestions for different venues (shopping, eating, etc.) in the nearby area.