Project no. 035003

u-2010

Ubiquitous IP-centric Government & Enterprise Next Generation Networks Vision 2010

Instrument: Integrated Project

Thematic Priority 2

D3.2.2 Prototype of the Presence Management Solution

Due date of deliverable: 31st October 2008

Submission date: 20th November 2008

Start date of project: May 1st 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: Lancaster University

Revision: V1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract

This deliverable presents the prototype of the Presence Management Solution tailored but not limited to the Mountain Rescue Domain. This document details the hardware and software requirements of the developed system and serves as a how-to for installation, configuration and usage. Finally, this document presents a troubleshooting chapter and outlines future work that could be undertaken to improve and extend the Presence Management Solution to aid the overall aim of U-2010.

Keywords:

Mountain Rescue, Presence, Location, Management, GPS, Windows Mobile, Prototype, Implementation.

History of Change

Issue	Status	Date	Details	Responsible
v0.1	Draft	13/11/08	General document skeleton and ToC.	Martin Dunmore
V0.2	Draft	17/11/08	Revised TOC, Added Introduction	Panagiotis Georgopoulos
V0.3	Draft	18/11/08	Updated Introduction, Added Chapter 2, Started Chapter 3	Panagiotis Georgopoulos
V0.4	Draft	19/11/08	Updated Chapter3, Added Chapter 4, Added Chapter 5	Panagiotis Georgopoulos
V0.5	Draft	20/11/08	Updated Chapter 5, Added Chapter 6, Added Chapter 7	Panagiotis Georgopoulos
V0.6	Draft	20/11/08	Revised document	Panagiotis Georgopoulos
V0.7	Draft	20/11/08	Revised document, corrected grammar, final edit	Martin Dunmore
V1.0	Final	20/11/08	Revision, minor corrections to orthography, final conversion to PDF	EPT

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable presents the prototype of the Presence Management Solution (PMS) developed at Lancaster University to facilitate the concept of location awareness in emergency and crisis scenarios. The developed system is tailored for the Mountain Rescue scenario but can be adapted to other search and rescue scenarios in U-2010.

The PMS is composed of a client application, a server application and a connectivity framework that has been designed to allow the applications to communicate. The applications have been implemented using Visual C# and the .NET Framework v3.5 and .NET Compact Framework v2.0, for the server and the client application respectively. In addition, some third party libraries have been used at both ends to extend and improve the functionality of the applications.

Both client and server applications are quite resource intensive and have high hardware requirements. The server application does complex processing and rearrangement of out-of sequence incoming packets, in addition to performing heavy graphical tasks when plotting rescuers on top of Ordinance Survey maps. The multi-threaded client application also performs various tasks such as obtaining GPS coordinates, finding the most suited and available connectivity option and transmitting the GPS coordinates to the server application. Furthermore, the client application confronts the total lack of connectivity by storing and sending packets when connectivity is regained, a function that is also quite resource intensive. However, despite the high hardware and software requirements of the applications, the PMS can run in publicly available medium cost equipment.

This detailed how-to deliverable, describes as well the steps to install, configure and use the PMS with a short troubleshooting guide. These steps actually reveal that the deployment and configuration of the PMS (excluding the intermediate network equipment) can be achieved in a couple of hours. Finally, a few hours training to the users of the PMS are found to be more than enough to adequately learn and use the custom-made PMS.

1. Introduction

This deliverable presents the prototype of the Presence Management Solution developed at Lancaster University for the emergency and crisis scenarios of U-2010.

The following chapters do not intend to describe the Presence Management Solution, referred onwards as PMS, as this is done in detail in D3.2.1 [1]. The reader should refer to D3.2.1, and will be referred to specific chapters of D3.2.1 throughout this document, to examine the details of the architecture of the PMS and also the underlying reasoning of many components, features and decisions taken.

This deliverable is written as a how-to use the PMS and is focused on practical tasks. Therefore, Chapter 2 and Chapter 3 describe the hardware and software requirements of the PMS, respectively. Chapter 4 presents installation and configuration steps that should be followed in order to setup the PMS. Chapter 5 describes in detail the actions that should be taken to use the PMS. Chapter 6 provides a troubleshooting of the system and finally, Chapter 7 examines further work that can be done to improve the system and extend its functionality.

1.1. Presence Management Solution Overview

The PMS developed at Lancaster University identifies and monitors the presence or location of emergency workers and vehicles in emergency scenarios, following the presence management definition given in Section 1.1, D3.2.1. Using the PMS, the mission coordinator of a rescue team can keep track of the rescue workers during a mission, and also keep a broad picture of the search and rescue operation. This helps the coordinator to take informed decisions and increases the efficiency of the team. The PMS can be used before, during and after an emergency operation according to the needs of the emergency organisation using it.

Keeping our promise to implement systems that can be used to real-life emergencies, we are collaborating with the Cockermouth Mountain Rescue Team [2] operating in the valley areas of Lake District (Cumbria, UK) and have been tailoring the PMS for their requirements. However, the system is implemented in a component-based approach that can be used in any emergency crisis, including the U-2010 Fire in the Tunnel scenario [3]. This deliverable describes the PMS focused, as an example, on the Mountain Rescue Domain, which actually provides the most difficult set of requirements for a presence management solution.

Figure 1 illustrates the architecture of the PMS. It shows that it is composed of three distinctive parts; the server application running on a server box at the Headquarters of the Rescue Team, the client application running on mobile devices carried by the rescuers at the Mountain Rescue Domain and the connectivity options that are available and used to interconnect the aforementioned domains (Headquarters and Rescue Domain).

Briefly, the client application is sending GPS live updates to the server application according to the devised Communication Framework of the PMS (see Section 3.2, D3.2.1). In the Mountain Rescue Domain, the available connectivity options that are used for the transmission of the GPS coordinates are WIFI, GPRS and SMS. The server application receives the coordinates sent from the clients (regardless of the connectivity option used for their transmission), and plots them onto a map in order to provide to the mission coordinator a holistic and informative view of the mission.

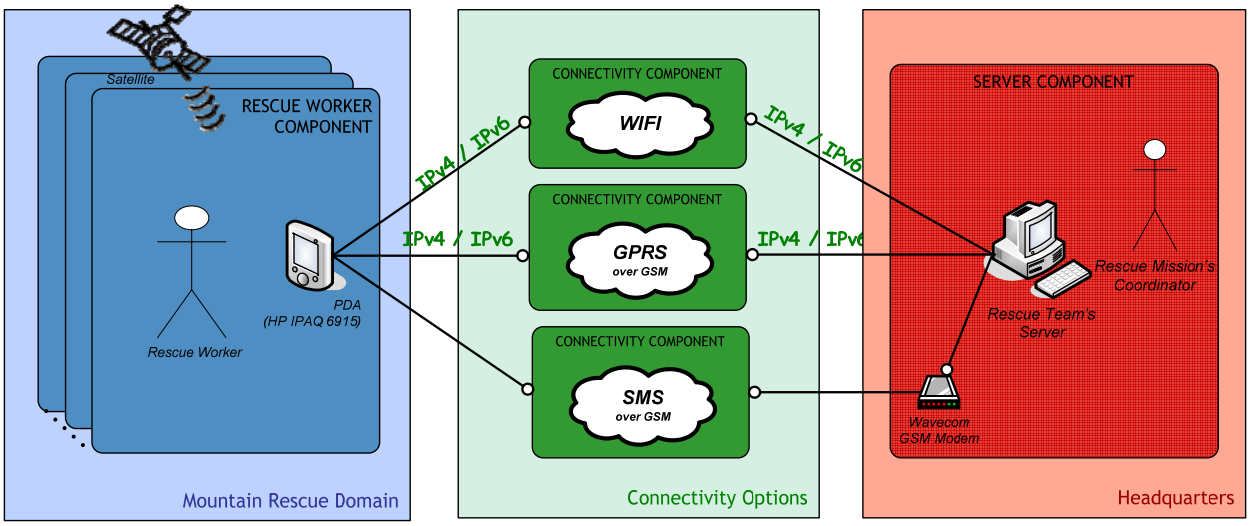


Figure 1 Architecture of the PMS

As the architecture of the PMS is component based, the server application can be located in any Headquarters of any team, the client application being carried from a rescuer in almost any domain and the modularized connectivity options be replaced with almost any available connectivity option interconnecting the two domains. Therefore, the PMS can be adjusted to any emergency scenario.

2. Hardware Requirements

This section describes the hardware requirements of both the server and client's side that should be met in order to satisfy the overall requirements of the PMS (see Section 2, D.3.2.1) and have it run smoothly. For a more detailed reasoning behind the hardware decisions taken, the reader is referred to Sections 4.6 and 5.5 in D3.2.1.

2.1. Server Side

The server application is resource intensive and should run on a fairly high-spec machine at the Headquarters of the respective Rescue Team.

The most resource intensive part of the server application is the real-time process and display of the Ordinance Survey map tiles, especially when the user is exploring the area and dragging/scrolling the map. Although the map tiles are provided in two different resolutions (occupying approximately 750MB) and are loaded dynamically on demand according to a specifically designed algorithm, they are still CPU, RAM and GPU intensive.

The server application additionally requires dual monitor support, with monitors and a graphics card that can support at least a 1600 by 1024 resolution. A lower resolution can be supported but it is not recommended, as the GUI forms of the server application will need resizing.

The server application should be able to listen to packets coming from clients over WIFI, GPRS or SMS. The network infrastructure of the PMS (see Section 1.3.1 in D3.2.1 and Section 4.5.3 in D4.2.1 [4]) ensures that packets sent over WIFI or GPRS from clients are routed over the Internet and thus the PC running the server application must be Internet enabled. Furthermore, Internet connectivity is required because an alternative implementation of the map functionality is provided using Google Maps API [5], which requires the server PC to be online. The server PC should also get a public IPv4 address on its Internet interface and a global IPv6 address (either from an IPv6 provider or via an IPv6 broker) so that it will be reachable from the clients over the Internet.

The server should also be able to receive SMS messages sent from clients and thus should have a GSM modem with an activated SIM card. The GSM modem should be able to reply to AT comments over a COM connection. We suggest using one of the following GSM modems that we have successfully tried:

- GSM 100 Maestro modem
- Teltonika T-ModemUSB/G10 USB GSM Modem

A summary of the server's hardware requirements is provided in Table 1.

Table 1 Hardware requirements of the server application

H/W	Minimum	Recommended
CPU	Core 2 Duo 3GHz	Core 2 Quad 2.4GHz
RAM	2GB	4GB
GPU	NVIDIA Quadro FX 1700, 512MB, dual monitor or similar card	NVIDIA GeForce GTX280, 1GB, dual monitor or similar card
MONITOR	Two monitors with at least 1600 by 1024 resolution	
HD	Approximately 1GB free space for the application	
NETWORKING	Broadband Internet connection with a public IPv4 address and a global IPv6 address	

2.2. Client Side

The client application is designed to run in small, lightweight and mobile devices that are easily carried from the rescuers at the Rescue Domain[1]. Candidate devices for such usage are PDAs, UMPCs, TabletPCs and mini-laptops.

As a proof of concept, we have decided to use PDA devices that are compact and easily carried. Therefore, the client application has been designed and implemented to run on any Windows Mobile 5.0 or 6.0 device being WIFI, GPRS and GSM enabled, as such are the connectivity options that are used for the transmission of the GPS coordinates.

In terms of the networking requirements of the client application, the hardware of the mobile device should be able to support both IPv4 and IPv6 for both the WIFI and GPRS connectivity options. In theory, most of the mobile devices that run Windows Mobile 5.0 or 6.0 support IPv4 and IPv6. However, it was found in practice that in some mobile devices the IPv6 stack has been stripped off. Therefore, IPv4 is considered as a sufficient and minimum requirement of the client application whereas IPv6 is preferred and recommended.

Ideally, the GPRS module of the client device should be of class A so that the client application will be able to use GPRS and send SMS at the same time. However, PDA devices usually have a GPRS module of class B that is found to be sufficient as a minimum hardware requirement but not preferable.

Regarding the acquisition of the GPS coordinates, the client device should either have an internal GPS module or be able to obtain coordinates from an external GPS module over Bluetooth.

Regarding the storage requirements of the client's device, the binary file of the client application with the third-party libraries occupy approximately 2.5MB. However, at least 10 more megabytes are needed for the extensive log file that the client application writes during an emergency and therefore the client device should have this amount of space free.

A summary of the client's hardware requirements is provided in Table 2.

Table 2 Hardware Requirements of the Client Application

H/W	Minimum	Recommended
GENERIC	mobile, small, lightweight device with a touch screen and battery runtime of at least 4 hours	Mobile, small, lightweight, waterproof, wearable device with a touch screen and battery runtime of at 6 hours
OS	Windows Mobile 5.0 or 6.0 device / other Windows based OS can be supported (see Section 7)	
RAM/SDRAM	2.5MB for the application and at least 10MB for the log	
INTERFACES	WIFI, GPRS (preferably of class A, sufficiently of class B), GSM enabled device	
IP	IPv4	IPv4 and IPv6
GPS	Either a device with an Internal GPS module or a device with Bluetooth that will be used with an external GPS device)	
DISPLAY	A device with a touch screen that supports at least a 240 by 240 resolution	

[1] For a more extensive list of clients' requirements tied with the requirements of the PMS, see Section 2 in D.3.2.1

At this point, it worth mentioning that the client application has been successfully used on the following PDA devices :

- HP IPAQ 6915 (Intel PXA270 416Mhz CPU, 128 MB ROM and 64 MB SDRAM, WM 5.0, WIFI, GPRS, GSM, BLUETOOTH)
- HP IPAQ 914c (Marvell XScale PXA270 520MHz CPU, 128 MB ROM and 128 MB RAM, WM6.1, WIFI, GPRS, GSM, BLUETOOTH, GPS)

The client application was running successfully in both the aforementioned devices, observing a better performance on the IPAQ 914c, which is a most recent device running Windows Mobile 6.0 with generally better specification. In addition, our hands-on experience proved that the battery on the HP IPAQ 6915 was drained faster as it was obtaining GPS coordinates from an external GPS device over Bluetooth, whereas the HP IPAQ 914c had a longer battery lifetime using a built-in GPS module. In terms of the GPS functionality of the client devices, our experience shows that the internal GPS module of the HP IPAQ 914c although needed more time to be initialized, it was a bit more accurate and less "jumpy" compared with the external GPS devices used with the HP IPAQ 6915.

Concluding, the client application is resource intensive and has quite a few threads and concurrent functions to run. Therefore, a powerful mobile device is recommended. The aforementioned PDAs were successfully used for the proof of concept, however we are in the process of finding a more powerful device that can meet more of the PMS requirements described in Section 2 of D.3.2.1 and does not pose a lot of hardware and software restrictions such as the ones described in Section 4.8 of D3.2.1.

3. Software Requirements

This section describes the software requirements of both the server and client's side of the PMS. For a more detailed reasoning behind the software decisions taken the reader is referred to Sections 4.7 and 5.6 in D3.2.1.

3.1. Server Side

The server application has been developed in Visual C# using initially the .NET v2.0 [6] and then the .NET v3.5 [7] programming framework in Visual Studio 2005 and then 2008.

A standard PC, covering the hardware requirements described in Section 2.1, with the following software requirements will be sufficient for the execution of the server application:

- A Windows based operating system. The server application has been tested extensively on Windows XP Pro SP2 and less extensively on WIN XP Vista Business and was found to run smoothly.
- The .NET 3.5 Framework
- Some additional libraries (DLL files) that contain useful functionalities that have been developed either by us or are third-party libraries. Table 3 summarizes the DLL files that are needed for the execution of the server application. These files do not need any installation but should be in the folder where the server's executable resides.

Table 3 DLL used for the server application

Library	DLL filenames	Scope	Developer
GSMCom	GSMCommunication.dll PDUConverted.dll	For the communication of the server application with the GSM modem	Third-party library (see [8])
GPSLib	GPSLib.dll	For processing the GPS coordinates of the incoming messages	Computing Department, Lancaster University
VectorMapLib	VectorMapLib.dll	For displaying the map tiles and plotting coordinates on the map	Computing Department, Lancaster University

Although the server application has extensive functionalities on both networking (using SOAP and XML schemas) and graphics (processing and drawing map tiles as well as plotting data on top of them), there is no extra software requirement as all these depend on libraries provided by the .NET 3.5 framework. Finally, the server application does need its configuration and image files to run.

3.2. Client Side

The client application has been developed in Visual C# using the .NET Compact Framework [9] in Visual Studio 2005 and then 2008. It has been designed to run in any mobile device that meets the following software requirements:

- Windows Mobile 5.0 or 6.0 operating system
- The .NET Compact Framework [9]
- Some additional libraries (DLL files) that have been developed by Microsoft or OpenNET [10]. Table 4 describes the DLL files that are needed for the execution of the client application. These files do not need any installation but should be in the folder where the client's executable resides.
- Has the Minimo web-browser installed [11]. Minimo web-browser is used to provide a "Map-me" feature to the rescuer showing him his exact location using Google Maps. However, having Minimo web-browser installed is not an essential software requirement for the client application but can be considered as a useful add-on, especially taking into account that this feature is resource intensive and slows down the execution of the more useful functions of the client application.

Table 4 DLL used for the client application

Library	DLL filenames	Scope	Developer
Microsoft. WindowsMobile	Microsoft.WindowsMobile.dll Microsoft.WindowsMobile.PacketOutlook.dll Microsoft.WindowsMobile.Status.dll	creating SMS messages, acquiring battery life	Microsoft
OpenNETCF	OpenNETCF.dll OpenNETCF.Configuration.dll OpenNETCF.Drawing.dll OpenNETCF.Net.dll OpenNETCF.Windows.Forms.dll OpenNETCF.WindowsCE.Messaging.dll	handling network sockets and interfaces, automatically configuring the device, processing SMS messages, efficient drawing/displaying at screen	OpenNET [10]

Finally, the client application needs its `setup.txt` configuration file to run.

4. Installation and Configuration

This section describes the installation and configuration steps that have to be taken before running the PMS, taking as granted that the hardware and software requirements described in the two previous sections are met.

4.1. Server side

The server application does not need any installation but rather a copy of the server project to the server PC that meets the hardware and software requirements that have been described in Sections 2.1 and 3.1.

The server project contains two main folders; the "`Bin`" folder that contains the executable of the server application with the DLL files described in Table 3, and the "`Resources`" folder that contains the following :

- Two configuration files (`network_settings.txt, rescuers_data.txt`) that are read when the server application is initialized (more details bellow)
- A log file (`log.txt`) that the server application writes to
- Two folders with the Ordinance Survey image map tiles covering the area of Lake District, UK. The first folder contains high-resolution images of the map and the other one low-resolution images of the map.
- An `index.html` file that contains the Javascript implementation of Google Maps
- Playback files of previous missions

When the server application is executed, it reads the `network_settings.txt` and `rescuers_data.txt` configuration files and thus these have to contain the correct configuration settings in advance. The `network_settings.txt` file includes an IP address and Port pair according to the following pattern:

PATTERN	EXAMPLE
`IP_address[SPACE]Port`	`148.88.225.150 10010`

The server application is using this IP-Port pair to listen and receive packets sent from clients. To be specific, this IP-Port pair is used for incoming TCP connections from the clients and for receiving IP packets from them and thus it is important to use a port number that is not blocked in any intermediate firewalls in the network.

The server application also loads information for the rescuers from the `rescuers_data.txt` file. This information is used to contact the rescuers and inform them of the emergency, using the AlarmTilt alerting system from M-Plify (see Section 5.7, D.3.2.1 and future deliverable D.3.3.1). Thus, the `rescuers_data.txt` file should be configured with the personal information of the rescuers that the mission coordinator wants to alert for the emergency. The `rescuers_data.txt` file has a new separate line for each rescuer according to the following format:

PATTERN OF EACH LINE	EXAMPLE
Name[SPACE]Surname[SPACE]Call Sign[SPACE]Mobile phone for voice calls[SPACE] Mobile phone for SMS[SPACE]Email[SPACE]Availability	Panagiotis Georgopoulos Panos +447927180000 +447927180000 pg@comp.lancs.ac.uk 1

We believe that the only non self-explanatory field of the above pattern of each rescuer is the `Availability` field. This field is represented with a Boolean value, being zero when the rescuer is not available, for example, when the rescuer is on vacation, and one when he is available. It is important to edit the `rescuers_data.txt` file correctly, as there are GUI items that are loaded dynamically when the server application starts based on the values found in this file. Furthermore, it has to be stated that this file is read only when the server application starts and thus should be edited before executing the application.

When the above steps are done, the server application is ready for execution.

4.2. Client side

The client application is advised to be deployed at the client devices using Active-Sync and Visual Studio as it will automatically deploy the client project with the .NET Compact Framework and all the third party libraries (described in Table 4) that are required. In addition, it has been noted that Active-Sync will also automatically update core Windows Mobile libraries if required, and thus this method of "installing" the client application is preferred. The alternative method of installation is to manually copy the client's project files with the required DLLs to the mobile device, and manually install the .NET Compact Framework at the client device.

The client project is composed of one folder containing the following files :

- The executable file of the client application and the required DLL libraries (described in Table 4)
- A log file (`log.txt`) that the client application writes to
- An `index.html` file that contains the Javascript implementation of Google Maps for the "Map me" function of the client application
- A `setup.txt` configuration file that the client application reads when is initialized

The `setup.txt` configuration file, as its name implies, contains set up values for the client application and it should be correctly configured before the application is initialized according to the following format (each field in a separate line):

PATTERN	EXAMPLE
NodeID	101
Security Code	1234
Com port for internal GPS module	COM7
Com port for external GPS module	COM6
SSID for the WIFI network	lancaster

Telephone number of the server's SIM card	00447511535947
Server's Public IP Address	148.88.225.150
Port	10010

The NodeID and Security Code fields are device dependant and should be set according to the actual device and the rescuer carrying it. The third and fourth lines of the setup.txt depend on the hardware of the mobile device used and the version of its operating system. Therefore, the "Com port for the internal GPS module" (third line) should be set according to the COM port that Windows Mobile is using to communicate with the internal GPS module, if such a module exist[2]. If an external GPS device is going to be used for the acquisition of the GPS coordinates, then the fourth line should be set according to the COM port that Windows Mobile is using to pair the Bluetooth external GPS device[3]. The SSID field of the setup.txt file specifies the SSID of the WIFI network that the application will use for the transmission of the GPS coordinates. The sixth line specifies the telephone number of the SIM card that the server is using, that is the SIM card in the GSM modem. This telephone number will be used from the client application to send SMS messages when it does not have any other IP based connectivity option available. Finally, the seventh and eighth line specify the IP address and the Port number that the client application should send the packets to, and thus they should be the same as the IP address and Port number that are defined in the network_settings.txt file that the server application is listening to.

It is important to mention at this point that most of these settings can be altered when the client application has been executed, from GUI Tabs that are provided. However, it is advised to set them correctly in advance in the setup.txt file so that the client application will run neatly without many run-time modifications from the rescuer.

When the above steps are done, the client application is ready for execution.

[2] In Windows Mobile 5.0 devices, defining the COM port for the internal GPS module is a trial and error procedure whereas Windows Mobile 6.0 explicitly mentions the COM port used for the internal GPS module in its settings.

[3] In Windows Mobile 5.0 devices, setting up the COM port to use an external GPS receiver, is a trial and error procedure. The correct COM port would wake up the Bluetooth module of the device and then trigger a method to scan for enabled Bluetooth devices populating an appropriate list. In a WM6 device you would have to pair the external GPS device in advance, and this procedure would report the COM port that should be used at the fourth line of the setup.txt.

5. Using the PMS

This section describes the sequential series of steps that should be taken to successfully use the PMS in the case of an emergency. This section assumes that both the client and server applications have been deployed, installed and configured as described in Section 4.

5.1. Initializing the server application

Having correctly set the two configuration files of the server application (`network_settings.txt` and `rescuers_data.txt`) the only action left to initialize the server is to execute the `myServer.exe` file found in the `Bin` folder of the server project. The server application will then read the configuration files, process them and load the required data to its forms and functions. In addition, the server application will load its two GUI forms, and load the Ordinance Survey image tiles to its map grid, a process that needs a few seconds.

The two GUI forms that are presented to the user are the `Control Form` (Figure 2) and the `Map Form` (Figure 3) and each takes up a monitor to present the required information to the mission coordinator. The former form is used at the initial phase of an emergency to alert the rescuers, and during the emergency to present statistical and logging information, whereas the latter form displays the Ordinance Survey maps and overlays rescuers' data on top it.

The Control Form (Figure 2) presents the following Tab pages:

- **Alert** tab page used to trigger an alert using M-PLIFY's servers (Figure 4)
- **Map** tab page that contains a Google Maps implementation that gives a "terrain feel" of the region that the rescuers roam (Figure 2). This tab page is also used as an alternative to the Ordinance Survey maps for regions or countries that Ordinance Survey map tiles do not exist.
- **Log** tab page that contains extensive logging information of the execution of the server application and the occurred events, in order to help troubleshooting and debugging the server

At the right side of the `Control Form`, three differently coloured panels are presented with useful information (Server Status, Connections Status and Last Packet Received). Finally, the `Control Form` presents some buttons to enable the functionality of the server and actually start the mission.

The `Map Form` (Figure 3) presents a large window with the actual map and has many controls in order to drag and move the map. On the left side of the `Map Form` a list of the available rescuers is presented so that the mission coordinator can chose whose track to see overlaid on the map during a mission. On the top right side of the `Map Form` there is a panel presenting information for the rescuer that has been clicked on the map. This information plays an important role as it aids the mission coordinator to organize the team more effectively during the mission by knowing where everyone is. Finally, on the bottom right side of the `Map Form` there are recording and playback controls that enable the recording of a mission and its playback afterwards.

When the server application has finished its initialization process the Log tab page should report that the configuration files and all the GUI components have been successfully loaded and that the application is ready to start a mission.

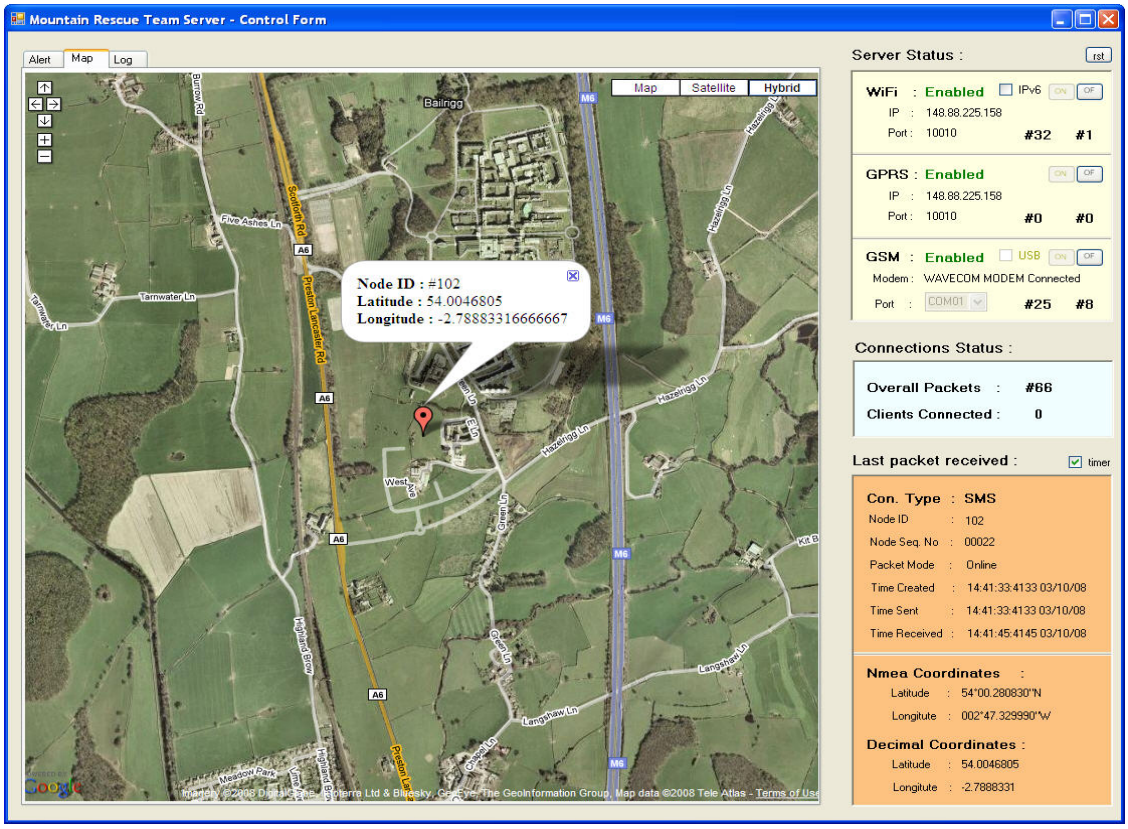


Figure 2 The Control Form of the server application

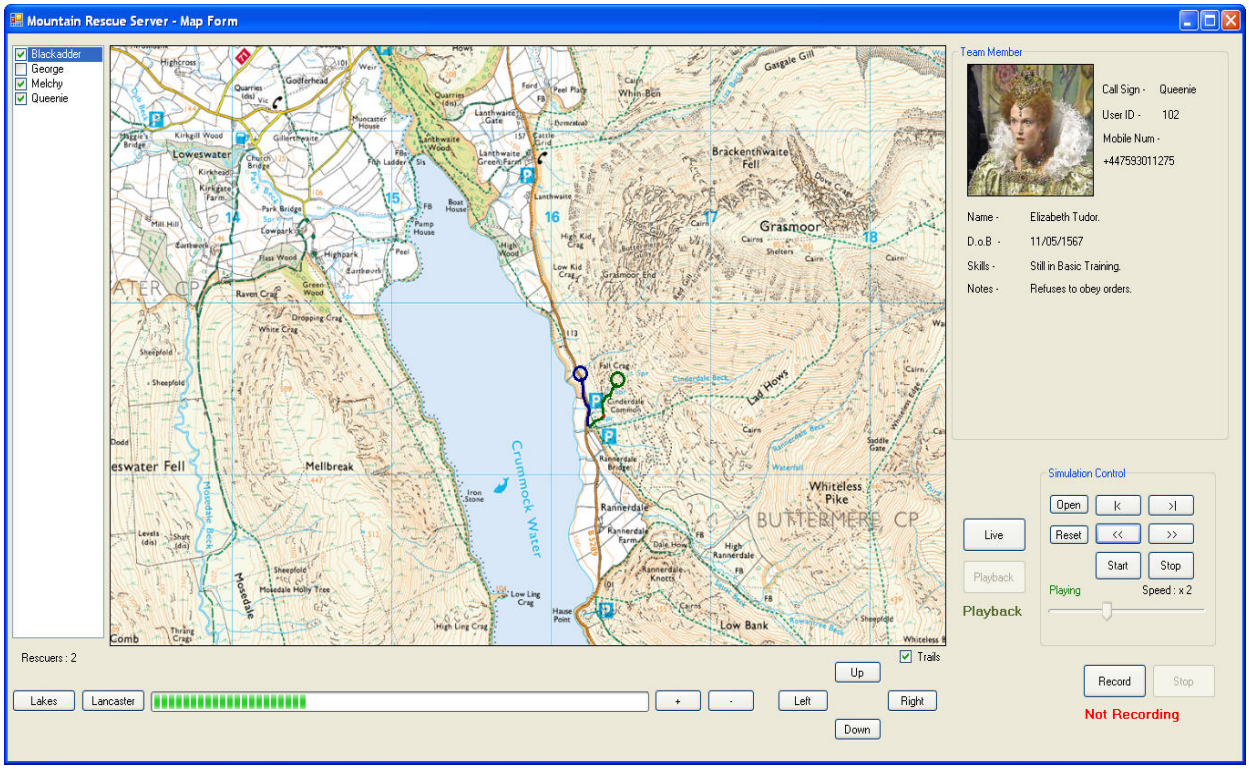


Figure 3 The Map Form of the server application

5.2. Starting a mission from the server application and alerting the rescuers

In the case of an emergency incident, for example a hiker being reported missing, a call reaches the Headquarters of the Rescue Team and the mission coordinator alerts the rescuers to report to the Headquarters and a mission begins.

Taking for granted that the server application has been initialized correctly, the mission coordinator should firstly use the Alert tab page of the `Control Form` to alert the rescuers for the emergency and then the buttons on the top right hand side of the `Control Form` to activate the server to start monitoring and processing incoming packets.

Alerting the rescuers and monitoring their responses can be done by choosing the Alert tab page in the `Control Form` and by filling in the Alert Title and Alert Message textboxes with information about the alert (Figure 4). The title of the alert is only informative, but is used in the communication of the server application with the M-Plify servers that actually alert the rescuers. The text in the message textbox is the one that will be sent via voice call, sms and/or email to the rescuers. The third box, titled as “End Message”, is for informing the rescuers with an end message when a mission is over. Filling in the “End Message” textbox is optional and can be done either before or after launching an alert.

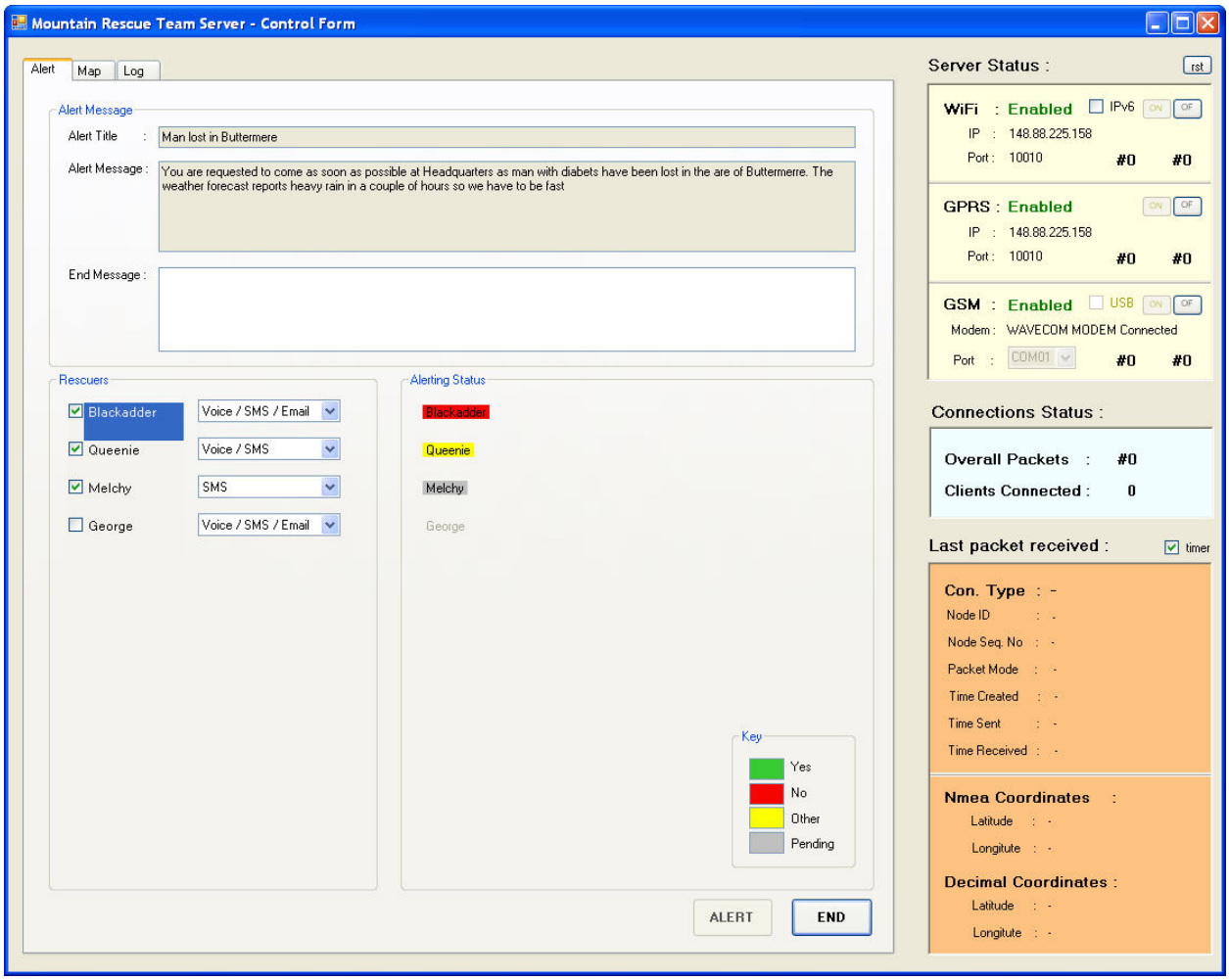


Figure 4 The Alert tab page of the Control Form of the server application

After filling in the text boxes, the next action is to choose from the combo boxes on the left bottom corner the rescuers that the mission coordinator would like to inform by ticking the box next to his/her name. Next to each one of the rescuers there is also a choice of the means that will be used by the AlarmTilt server to inform the rescuers (any combination of Phone, SMS, email)[4]. Finally, the actual alert goes through when the "ALERT" button is chosen at the bottom of the Alert tab page and an informative messages shows whether the alert has been launched successfully or not. From that point on, the server application automatically queries M-PLIFY's servers over SOAP every 15 seconds and collects the responses from the rescuers' replies. These responses are displayed colour-coded in the Alerting Status panel of the `Control Form` of the server application so that the mission coordinator can be aware of who is available and can respond to the emergency call or not. To be precise, the background colour of the alerted rescuer is changed to green, red, yellow or grey when he\she has replied positively, negatively, has replied something else, or has not replied at all, respectively.

At this point, the mission coordinator has to enable the monitoring functionality of the server application and this can be done by clicking the two "ON" buttons on the right side of the `Control Form`; one next to the WIFI label and the other one next to the GSM label. This will trigger the server to start listening for incoming messages over the Internet and the GSM network. If there is any particular problem with the IP-PORT pair, it will be reported here by either a pop-up error message or by reporting a failure in the Log tab of the `Control Form`. If the mission coordinator wants to record a mission for offline analysis and later playback, he can do so by clicking the Record button at the `Map Form` of the application.

At this point, the rescuers have been alerted for the emergency and the server application is ready to start monitoring/tracking the rescuers. This is the time that the client devices should be turned on and the client applications initialized.

5.3. Initializing the client application

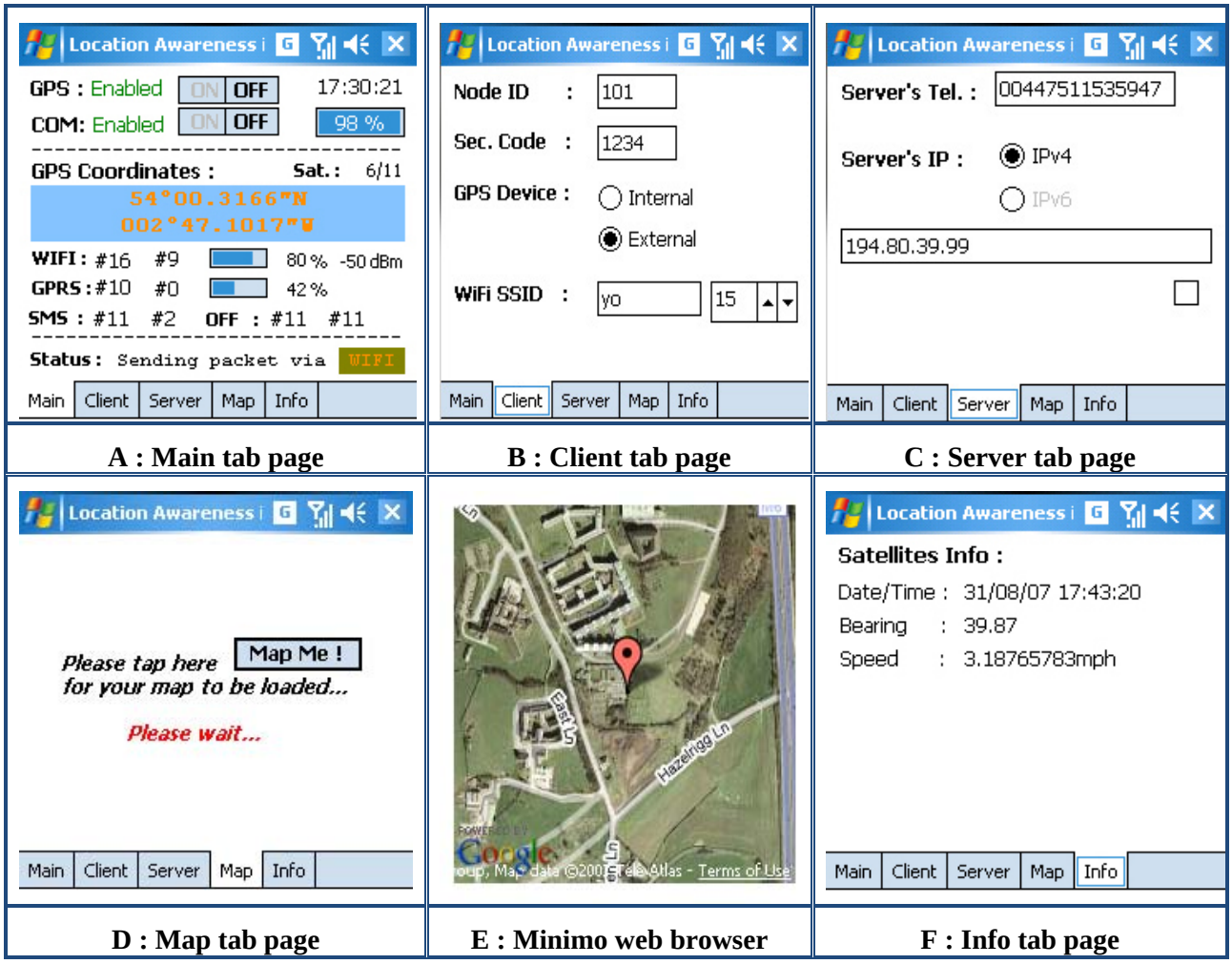
In theory, the rescuers that have been alerted do have a client device with a properly configured `setup.txt` file in advance. Therefore, if the rescuer is able to respond to the emergency and has replied positively to the alerting voice/sms/email, he should execute the `myClient.exe` file and initialize the client application.

When the client application is initialized the rescuer observes the following five tab pages:

- Main tab page (Figure 5A); the main screen of the client application that the rescuer sees
- Client tab page (Figure 5B); settings regarding the client device/application
- Server tab page (Figure 5C); settings regarding the server device/application
- Map tab page (Figure 5D); displaying a map to the rescuer showing his location
- Info tab page (Figure 5F); information obtained from the GPS satellites.

If the client application has been initialized successfully, then the rescuer can immediately join a mission by acquiring GPS coordinates and start transmitting them to the server (see next Section). However, if the rescuer needs to change the configuration of the application, then at this point he can easily achieve that by choosing the Client and Server tab pages to modify the settings according to his needs.

[4] The combo and the list boxes on the Alert tab are built dynamically from the `rescuers_data.txt` configuration file and therefore, for example, if a rescuer is known to be unavailable would not appear on the combo box.



A : Main tab page	B : Client tab page	C : Server tab page
D : Map tab page	E : Minimo web browser	F : Info tab page

Figure 5 Screenshots of the client application

5.4. Joining a mission with the client application

When the `setup.txt` file is correctly set and the client application has been initialized successfully, the rescuer is able to join a mission via his client device and be monitored by the server application.

A client application can join a mission when the following two phases have been successfully completed. Phase One includes the initialization of the GPS module (internal or external) and the acquisition of the GPS coordinates, whereas Phase Two includes the initialization of the transmission of the coordinates. Both phases are turned on and off from the appropriate screen buttons in the Main tab page of the client application (Figure 5A). These phases/functions can work separately (i.e. having only one to work) but be advised that if only the transmission of the coordinates is turned on without having actual GPS coordinates, the client will transmit zeros to the server application and will not be monitored. On the other side, if only the acquisition of the GPS coordinates is turned on (Phase One) then the client will see valid coordinates on screen but will not be monitored by the server as it will not transmit any coordinates.

5.4.1. Phase One: Obtaining GPS coordinates

Phase One is activated when the rescuer taps the ON button next to the GPS label at the Main tab page of the client application (Figure 5A). Rescuer's next action depends on whether his device uses an internal or external GPS module.

If the rescuer is using a mobile device that has a built-in GPS module, this must be indicated during the application's initialisation procedure by choosing the "internal" option in the Client tab page and correctly setting the COM port in the `setup.txt` configuration file. If these settings have been chosen correctly, the rescuer should enable the GPS functionality of the application by tapping the ON button at the Main tab next to the GPS label. At this point it is important to mention that the internal GPS module can obtain coordinates if the rescuer is outdoors for at least a couple of minutes. If there is any time-out error message displayed at this step, the rescuer should just click ok and retry. There is no reason to enable the Bluetooth module of the device when using the internal GPS module.

If the rescuer is using a mobile device without a built-in GPS module, then he has to use an external GPS device that communicates with the client device over Bluetooth. Therefore, the rescuer should turn on the external GPS device and enable the Bluetooth module of the mobile device. When this step is done, the rescuer should tap the ON screen button to enable the acquisition of the GPS coordinates. At this point, in Windows Mobile 5.0 devices, a pop-up list will be populated with the Bluetooth devices that the client application sees and the rescuer should choose the external GPS device from them. In Windows Mobile 6.0 devices, the external GPS module should have been paired in advance and its COM port should have been correctly set in the fourth line of the `setup.txt` file before starting the client application.

By doing one of the above, the rescuer should have completed Phase One and should have started receiving GPS coordinates.

5.4.2. Phase Two: Start transmitting coordinates to the server application

The transmission of the GPS coordinates from the client application to the server application can be enabled from the ON screen button next to the COM label at the Main tab of the client application. From this point on the client application will start transmitting coordinates over the best-suited connectivity option available according to each option's signal strength. In addition, the client application will seamlessly swap from transmitting to storing, and then back to transmitting coordinates when required.

When communication is started, an informative message will pop-up, mentioning that a connection with the server has been established. This assures that every setting at the client's side is correctly defined and that there is not any intermediate firewall blocking the connections of the client to the server. If the connection cannot be established, the function will timeout and inform the rescuer of the error.

By successfully finishing both phases (One and Two) the rescuer has effectively joined a mission and will be tracked from the PMS, and in particular, the server application. There is no need for the rescuer to press any other button or tab page during the mission, but if he wants to, he can check the Info tab page for satellites' information and also see his position on the map if the client has Internet connectivity.

5.5. Monitoring/tracking rescuers at the server application

At this point, the server application should be up and running, mentioning "ENABLED" in green in both the WIFI and GSM functionality in the top right side of the `Control Form`. In addition, the rescuers' responses should have already been collected from the launch of the alert and the mission coordinator should be able to monitor the members of the rescue team who take part in the search and rescue operation. If the mission coordinator wants to record this mission for offline analysis, the record button on the `Map Form` should be clicked now.

At this phase of the mission, the main form that the mission coordinator watches is the `Map Form` as it displays the region that the rescuers are roaming and their exact location. The mission coordinator is also able to see the exact trail of each of the rescuers and identify where everyone is at any given point. In addition, the mission coordinator has the ability to dynamically choose which rescuers to see on the map and decide whether trails should be displayed or not. Furthermore, he has the ability to explore the map by using the zoom in/out and drag and drop functionalities of the map. Moreover, the mission coordinator can right click on a rescuer shown on the map, identify him and see his information on the right panel of the form. This aids him to organize the mission more effectively and "lay out" certain people according to the needs each time (for example, find where the closest rescuer with medical training is). It has to be noted, that the "browsing the map" functionality is a quite resource intensive mechanism that processes image files in two scales, occupying approximately 750MB. Therefore, although the server application has a very elegant mechanism to keep up with the demands of the user when browsing the map, some delay might be experienced when the user is dragging the map very quickly.

Complementing the Ordinance Survey maps of the `Map Form`, the PMS includes on the Map tab page of the `Control Form` a Google Maps implementation for two basic reasons. Firstly, Google maps give a very good geographical terrain-feel of the region that the rescuers are to the mission coordinator. Secondly, this Google Maps implementation can be used for regions that detailed Ordinance Survey maps do not exist. Currently, Google Maps implementation signifies only the last pair of coordinates received on the map, as is used as a secondary mapping function. Further work should be done to extend this functionality and add features to this Google Maps implementation (see Section 7).

5.6. Ending a mission

When a casualty is found, or at any given point, the mission coordinator decides to, a mission should end and all the rescuers should be informed and return at the Headquarters of the Rescue Team.

A mission can end using the server application and using the AlarmTilt functionality to transmit end messages to the rescuers holding the client devices. This can be achieved by choosing the Alert tab page at the `Control Form` and optionally filling in the "End Message" text box with an end message that will be sent to the rescuers. Finally, the actual end of the mission is done when the END button at the Alert tab page is clicked. This is the point where the server application will inform M-PLIFY's servers to contact the rescuers by the means that have been used when launching an alert, and inform them about the end of the mission. If the "End Message" text box is not filled in, then the rescuers will be informed for the end of the mission with the default "Stop Emergency" message.

If the mission coordinator has chosen to record the mission, then at this point he has to stop the recording by choosing the "Stop button" at the `Map Form` recording controls and then save the playback file on the hard drive.

5.7. Shutting down the PMS

By this moment, the mission should have ended and all the rescuers should have been informed about it. Both the server and the client application can be closed as any other Windows or Windows Mobile applications. Both applications include a graceful shutting down implementation that closes one by one each function and gracefully close the log file.

If the user wants to manually close the server application, then he firstly has to click the OFF buttons next to the WIFI and GSM panels of the `Control Form` and secondly, click the X button in either the `Control` or the `Map Form` to exit the application. The user should be aware that the server application needs approximately two seconds to shut down gracefully as it writes data to the log and releases

resources, and thus a small delay will be noticed when pressing the close button of either form of the server application.

If the rescuer wants to manually close the client application, then he needs to turn off the communication and the acquisition of the GPS coordinates by tapping the OFF buttons next to the COM and GPS labels (Figure 5 Screenshots of the client application A) . Finally, since pressing the OK button at the top right corner of the client application will just minimize it, the user needs to close it by choosing the application on the task manager of Windows Mobile.

5.8. Replaying a mission

Replaying a mission is a very useful feature that enables the team to analyze previous missions, and improves the way its members approach the search and rescue domain.

Viewing a playback on the server application is an easy task that can be done without any extra setup. If the user has done a fresh start up of the server application, then he has to click the playback button at the `Map Form` and then the open button to choose and load the playback file of the previous mission (usually being saved in the “Resources” folder). If the server application has been used, then the user has to firstly click the OFF buttons at the WIFI and GSM panels at the `Control Form` and then load the playback file. The playback controls offer extensive functionality when viewing the playback, for example pause it at certain points or speed it up, without losing the ability to interact with the map (zoom in/out, scroll, dynamically choose to view or not rescuers on the map and see details about them).

6. Troubleshooting

The PMS developed at Lancaster University is, code-wise, mainly composed of a server and a client application with tens of thousands lines of code. As any other large and complex system when used, there is a possibility of behaving erroneously.

Here we provide two layers of troubleshooting tips, when the applications do not behave as expected or any other problem arises. The first, high-level troubleshooting layer, includes short and informative pop-up messages that the end user sees when something goes wrong, which usually include a tip of how to fix the error. Frequently, this high-level troubleshooting layer includes errors caused by wrong configuration files, for example "The Port number provided in the `network_settings.txt` file is invalid", or run-time user negligence "The Alert Message box cannot be empty". In addition, this high-level troubleshooting layer includes error messages due to networking problems, for example "The connection with the server could not be established. Please try again".

The second, low-level troubleshooting layer is provided by extensive logging functions that both the client and the server applications have, varying from the payload of received and sent packets to timestamps of user actions. This low-level troubleshooting layer might not at first hand interest the end user, but is available to him either as information or for aiding him to correct a more complex problem/bug. Furthermore, this extensive logging functionality enables the offline troubleshooting and analysis of the applications of the PMS in order to correct bugs and enhance their functionality.

In our experience, Table 5 Troubleshooting contains the most common erroneous or seemingly erroneous behaviour that can occur while using the PMS and provides an explanation and possible solution.

Table 5 Troubleshooting

ERRORNEOUS BEHAVIOUR	EXPLANATION	SOLUTION
Some DLL file is reported missing at either the client or server application.	The application has not been deployed correctly at the target device or PC.	The user is advised to follow precisely the steps described in Section 4. Reinstallation of the .NET full or compact framework might solve the problem.
When the COM ON button is tapped at the client device the client application seems to hang for several seconds	The client application tries to establish a TCP connection with the server application and this might happen over a variety of connectivity options (WIFI, GPRS, Satellite Link) that can introduce a great delay or packet loss. Therefore, the client application waits for the connection to be established until it times out and thus the client application seems to be hanged.	The user is advised to be patient and wait for the application to report if the connection was established or not.
The client application continuously reports that the TCP connection with the	There might be an intermediate firewall blocking the TCP connection of the client application with the server.	The user is advised to change the port that the server application listens for TCP connections and set that new

server cannot be established although it is proven that the client can “reach” the server.		port number in the `setup.txt` configuration file of the client application.
When using internal GPS module of a device and the GPS ON button is tapped, the application reports a “Time Out” error	Windows Mobile wakes up the internal GPS module when the appropriate COM port is probed from the client application. However, when the user is indoors or has not been outdoors for at least a couple of minutes, the internal GPS module cannot get a satellite fix and thus times out.	The user is advised to stay outside for a couple of minutes and then retry.

7. Future Work / To Do

The prototype of the Presence Management Solution at Lancaster University has been implemented to provide location awareness of the rescuers to the mission coordinator and enable him to take more informed decisions and organize more efficiently the missions of the rescue team.

In our design and implementation phases, many hardware and software restrictions and difficulties have been found, especially in regards to the mobile devices that the rescuers carry (for more information see Section 4.8 in D3.2.1). Therefore, we are currently in the process of finding more suitable devices that meet more of the PMS requirements and have fewer hardware and software restrictions. For example, client devices with higher resources, full IPv6 support, wearable, waterproof and with longer battery life are still needed. To this effect, we are considering of porting the client application to other Windows based devices (e.g. Windows Tablet or Windows XP) so that tablet PCs or mini-laptops will be used as client devices.

Our future work also includes the improvement of our server's Google Maps implementation in order to concurrently display many rescuers and overlay their trail. Although the PMS does support this functionality using the Ordinance Survey maps, these maps cover only the UK. Therefore, our intention is to extend Google Map's implementation and be able to preserve our mapping functionality for other countries as well. Another relevant future work item is to compose a library of how the Ordinance Survey maps are processed and displayed and be able to use it with any other map images for another area. It has to be noted that this a very difficult task, as correctly adjusting and aligning map images is a very time consuming procedure that requires many carefully chosen compensation factors.

An additional future step is to fully integrate to the PMS the VoIP solution that has been developed so that the rescuers will be able to freely talk to each other and communicate with the Headquarters when connectivity exists. Finally, our future plans include the full integration of our search theory component with the PMS.

References

[1] U-2010 Deliverable 3.2.1, “Report on the Presence Management Solution”, November 2008.

[2] Cockermouth Mountain Rescue Team website : http://www.cockermouthmrt.org.uk/

[3] U-2010 Deliverable 4.3.1, “Concept of Fire Services Solution”, July 2008.

[4] U-2010 Deliverable 4.2.1, “Report on the Mountain Rescue Service Concept”, July 2008.

[5] Google Corporation. “Google Maps API Concepts”. Available at: http://www.google.com/apis/maps/documentation/index.html

[6] Microsoft .NET Framework 2.0. Available at : http://www.microsoft.com/downloads/details.aspx?familyid=0856eacb-4362-4b0d-8edd-aab15c5e04f5&displaylang=en

[7] Microsoft .NET Framework 3.5. Available at : http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en

[8] Stephan Mayr. “GSMCom Library for the .NET programming framework”. Available at : http://www.scampers.org/steve/sms/

[9] Microsoft .NET Compact Framework. Available at: http://msdn2.microsoft.com/en-gb/netframework/Aa497273.aspx

[10] OpenNETC Consulting, LLC. “Compact Framework”. Available at http://www.opennetcf.com/CompactFramework/tabid/85/Default.aspx

[11] Mozila Foundation. “Minimo Web Browser for Windows Mobile”. Available at : http://www-archive.mozilla.org/projects/minimo/