PandoraFMS Agent for Windows Phone 7

Track your phone and maintain inventory information of your Windows mobile equipments using PandoraFMS Enterprise.

By: Patricio Martínez (patricio.martinez@frameworks.com.ec)

This article will explain some available features of the Windows Phone 7 (WP7) equipments like background tasks, LINQ-To-SQL isolated storage, GPS tracking and Bing Maps. But also will introduce you some sockets programming using the Tentacle protocol which is used to transfer information to the PandoraFMS Enterprise Server, where you will be able to manage your corporate Windows mobile phones. An intermediate knowledge of Visual Studio 2010, C# language and Windows Phone Emulator is desirable. The source code for this project and the "PandoraFMS.xap" file to be installed in a real phone WP7 are available for download.

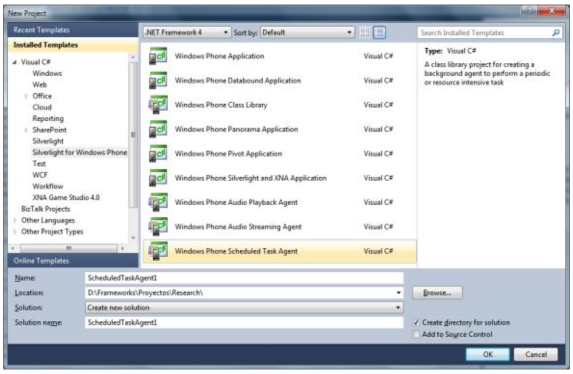
Objective

The main objective of this development is to provide a mechanism to maintain an inventory of corporate phones using PandoraFMS Enterprise to track those using GPS and maps. This functionality should continue working even when the phone has been reset.

Background Tasks onWP7

The WP7 Mango has some new features not included in previous versions, like this: you can execute background tasks registered from your application and the operating system will schedule them no matter what application is running on the foreground, even when the phone is locked. These tasks can be: Periodic Tasks, to be executed every 30 minutes (there is no configuration option) to execute small and simple tasks in a small period of time, with a maximum 25 seconds; and Resource Intensive Tasks for longer actions with a maximum duration of 10 minutes.

For this project we will use the Periodic Tasks using Microsoft Visual Studio 2010: First go to File -> New -> Project and then select in the "Silverlight for Windows Phone" templates, the one called "Windows Phone Scheduled Task Agent":


New Project
Recent Templates
Installed Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Silverlight for Windows Phone
Test
WCF
Workflow
XNA Game Studio 4.0
BizTalk Projects
Other Languages
Other Project Types
Online Templates
.NET Framework 4
Sort by: Default
Search Installed Templates
Windows Phone Application Visual C#
Windows Phone Databound Application Visual C#
Windows Phone Class Library Visual C#
Windows Phone Panorama Application Visual C#
Windows Phone Pivot Application Visual C#
Windows Phone Silverlight and XNA Application Visual C#
Windows Phone Audio Playback Agent Visual C#
Windows Phone Audio Streaming Agent Visual C#
Windows Phone Scheduled Task Agent Visual C#
Browse...
OK
Cancel

Type: Visual C#
A class library project for creating a background agent to perform a periodic or resource intensive task

Name: ScheduledTaskAgent1
Location: D:\Frameworks\Proyectos\Research\
Solution: Create new solution
Solution name: ScheduledTaskAgent1
☑ Create directory for solution
☐ Add to Source Control

Type your Project Name, select the path location of it and Visual Studio will create for you a new class called "ScheduledAgent" which inherits from ScheduledTaskAgent. Inside this new class you will see the constructor, the exception handler and the "OnInvoke" method to be executed when a periodic task is invoked; here is where we will start tracking our GPS positions by using the "GeoCoordinateWatcher" class:

```
protected override void OnInvoke (ScheduledTask task)
{
    GeoCoordinateWatcher gcw = new GeoCoordinateWatcher (GeoPositionAccuracy.High)
    {
        MovementThreshold = 100
    };
    gcw.PositionChanged += new System.EventHandler<GeoPositionChangedEventArgs
        <GeoCoordinate>>(gcw_PositionChanged);
    gcw.Start(false);
}
```

The “gcw” variable is instantiated using High Accurracy, this means for a Assisted-GPS device the use of the satellites for positioning (but not use the cellular antennas or the wi-fi link). The threshold for movement recognition is 100 meters. The next instruction configures the “gcw_PositionChanged” function to be called when there is a position change detected. Finally the geo coordinate watcher is started.

The “gcw_PositionChanged” function is the one to execute the real job of this agent: It reads the agent name from the database, then acquires the actual position from “gcw” and stops the GPS. The next instructions are for building the XML message to be sent to the PandoraFMS Server, creates the socket and starts the asynchronous connection using sck.ConnectAsync(). The TCP port used for the Tentacle Protocol is 41121.

```
void gcw_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
{
     AgentName = clsConfiguration.fnReadName();
     if (AgentName.Length <= 0)
        ErrorMessage("PandoraFMS",
           "The agent name is incorrect. Please reinstall the PandoraFMS Agent");
     GeoCoordinateWatcher gcw = (GeoCoordinateWatcher)sender;
     GeoCoordinate position = gcw.Position.Location;
     gcw.Stop();
     if (position != null)
     {
        string strPlatform = Environment.OSVersion.Platform.ToString();
        string strOsVersion = Environment.OSVersion.Version.ToString();
        strMessage = "<?xml version='1.0' encoding='utf-8' ?>\n" +
                     "<agent_data …>"
                     … // Here are the modules definitions
                     "</agent_data>";
        IPEndPoint dirServer = new
             IPEndPoint(IPAddress.Parse(clsConfiguration.fnReadAddress()), 41121);
        SocketAsyncEventArgs args = new SocketAsyncEventArgs();
        args.RemoteEndPoint = dirServer;
        args.Completed += SocketAsyncEventArgs_Completed;
        Socket sck = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                    ProtocolType.Tcp);
        sck.ConnectAsync(args);
     }
     else
     {
         NotifyComplete();
     }
}
```

The “strMessage” variable will store the information as a XML string using the <agent_data> node. We will explain its structure below on the Tentacle Protocol section.

The Tentacle Protocol

The Tentacle protocol is explained here (http://www.openideas.info/wiki/index.php?title=Tentacle:Protocol). We start connecting to the server “dirServer” using port 41121 with asynchronous sockets. The event function for the sockets is “SocketAsyncEventArgs_Completed”, it will be called after any completion event: Connect, Send and Receive, then it calls the handler functions required:

When the Agent is connected, it will send to the server the SEND command using the next format:

```
SEND < agent.#.data > SIZE size \n
```

Notice the “agent.#.data” is the file name surrounded by the characters ‘<’ and ‘>’ to be sent to the server using the Tentacle protocol. The ‘#’ in the file name should be a sequential unique number, but for this application it worked fine with number “1” (one). The “agent” is the equipment name we discussed previously. The “size” is the number of bytes of the XML message to be explained next.

The “HandleSend()” function is executed when some data is received from the socket, this function just check if we have sent a “QUIT” command, in which case is called the “CleaUp()” function, otherwise it calls the “ReceiveAsync()” function in order to receive some data.

When data is received from the socket connection, then is executed the “HandleReceive()” function which converts the data message “strMessage” in a byte array to be sent asynchronously through the socket: The message is formatted using the next XML format:

```
<?xml version='1.0' encoding='utf-8' ?>
<agent_data … attributes>
       …
       Module definitions
</agent_data>
```

Modules are each individual data item to be sent by the agent to the server. The module definitions is built in “fnModule()” function and uses the next format:

```
<module>
    <name><![CDATA[_name_]]></name>
    <description><![CDATA["_description_"]]></description>
    <type><![CDATA["_type_"]]></type>
    <data><![CDATA["_data_"]]></data>
</module>
```

The attributes in <agent_data> nodes are used to identify the client sending the information, the available attributes are:

Name	**Description**	**Example**
description	Long description of the agent	‘’
group	Group to which is assigned the agent	‘’
os_name	Operating System Name	‘windows’
os_version	Operating System Version	‘7.1’
interval	Interval in seconds between messages sent by the agent	‘1800’
version	Agent’s version number	‘1.0’
agent_name	Agent Name	‘MiEquipment’
latitude	Latitude in degrees received from the GPS	‘-0.17877’
longitude	Longitude in degrees received from the GPS	‘-78.4767’
altitude	Altitude in meters received from the GPS	‘2800’

The list of modules or individual items of information sent by the Agent is described next:

Name	**Description**	**Data Type**	**Source**
`AplicationCurrentMemoryUsage`	`Current Application Memory Usage`	`generic_data`	`DeviceStatus.ApplicationCurrentMemoryUsage`
`AplicationPeakMemoryUsage`	`Peak Memory Usage`	`generic_data`	`DeviceStatus.ApplicationPeakMemoryUsage`
`AplicationMemoryUsageLimit`	`Memory Usage Limit per Aplication`	`generic_data`	`DeviceStatus.ApplicationMemoryUsageLimit`
`DeviceManufacturer`	`Device Manufacturer`	`generic_data_string`	`DeviceStatus.DeviceManufacturer`
`DeviceName`	`Device Name`	`generic_data_string`	`DeviceStatus.DeviceName`
`DeviceFirmwareVersion`	`Device Firmware Version`	`generic_data_string`	`DeviceStatus.DeviceFirmwareVersion`
`DeviceHardwareVersion`	`Device Hardware Version`	`generic_data_string`	`DeviceStatus.DeviceHardwareVersion`
`DeviceTotalMemory`	`Device Total Memory`	`generic_data`	`DeviceStatus.DeviceTotalMemory`
`IsKeyBoardPresent`	`Is Keyboard Present in Device`	`generic_proc`	`DeviceStatus.IsKeyboardPresent?"1":"0"`

PowerSource	Power Source	generic_data_string	DeviceStatus.PowerSource
DeviceUniqueId	Unique Id of the device	generic_data_string	clsConfiguration. fnDeviceUniqueId()
OsPlatform	Operating System Platform	generic_data_string	Environment. OSVersion.Platform
Osversion	Operating System Version	generic_data_string	Environment. OSVersion.Version

Isolated Database using Linq-To-SQL

The information shared between the background task and the Agent's user interface is stored in a LINQ-To-SQL database defined in the "clsData.cs" file. Everything you need to manage the data base is located in this file. First of all is defined a descendant of the "DataContext" class, here is the Connection String to the database and the table definition called "Configuration".

```
public class clsData : DataContext
{
    static string connectionString = "DataSource=isostore:/data.sdf";
    public clsData() : base(connectionString)
    {
    }

    public Table<configuration> Configuration
    {
        get { return this.GetTable<configuration>(); }
    }
 }
```

Next is defined the "configuration" class as a table with two columns: co_name and co_address to store the device name and the IP address of the PandoraFMS Server. Both of them are strings or NVarChars(30).

```
public class configuration
{
    [Column(Name = "CO_NAME", DbType = "NVarChar(30)", IsPrimaryKey = true)]
    public String co_name { get; set; }

    [Column(Name = "CO_ADDRESS", DbType = "NVarChar(30)", IsPrimaryKey = true)]
    public String co_address { get; set; }
}
```

Finally we find the "clsConfiguration" class which includes some helper functions to read and write the information from the LINQ-To-SQL database: fnReadName(), fnReadAddress(), fnWriteData() and fnDeviceUniqueId(), this last function retrieves the Device Unique Id from the "DeviceExtendedProperties" of the "Microsoft.Phone.Info" namespace. We will just show here the fnReadname() function:

```
public static string fnReadName()
{
    string strName = string.Empty;
    using (var db = new clsData())
    {
        if (!db.DatabaseExists())
            strName = "";
        else
        {
            configuration cConfig = (from p in db.Configuration
                                     select p).Single();
            strName = cConfig.co_name;
        }
    }
    return strName;
}
```

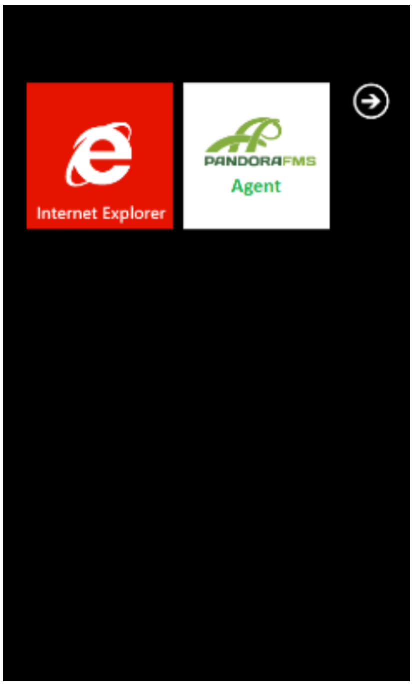
Please observe the LINQ-To-SQL instruction to retrieve the data: "from object in table select object", the instruction format is: FROM … WHERE … SELECT, just remember the SELECT is at the end.

The Agent's User Interface

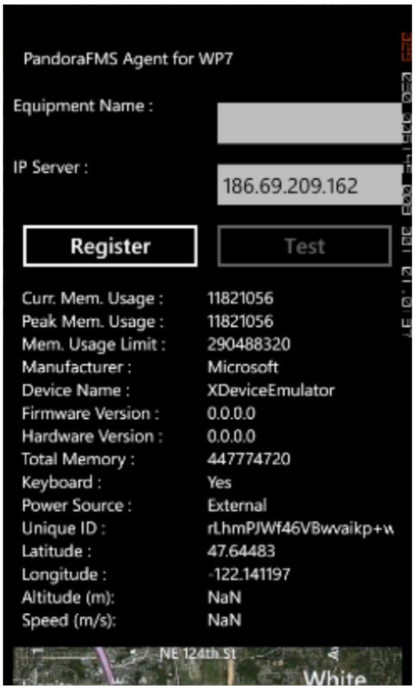
This Visual Studio Solution includes two projects, the GPSAgent Project described in previous sections and the PandoraFMS project to display the user interface of the application. This second project references the former and in the WMAppManifest.xml file is identified the extended task created in the GPSAgent Project, to be executed as a background task, here we present just the <Tasks> node:

```
<Tasks>
      <DefaultTask Name="_default" NavigationPage="MainPage.xaml" />
      <ExtendedTask Name="BackgroundTask">
        <BackgroundServiceAgent Specifier="ScheduledTaskAgent" Name="GPSAgent"
Source="GPSAgent" Type="Frameworks.GPSAgent.ScheduledAgent" />
      </ExtendedTask>
</Tasks>
```

The Visual Interface of the Agent uses the "Windows Metro Design" and is presented below:



When you execute for the first time the PandoraFMS Agent, you will see a screen page like this:



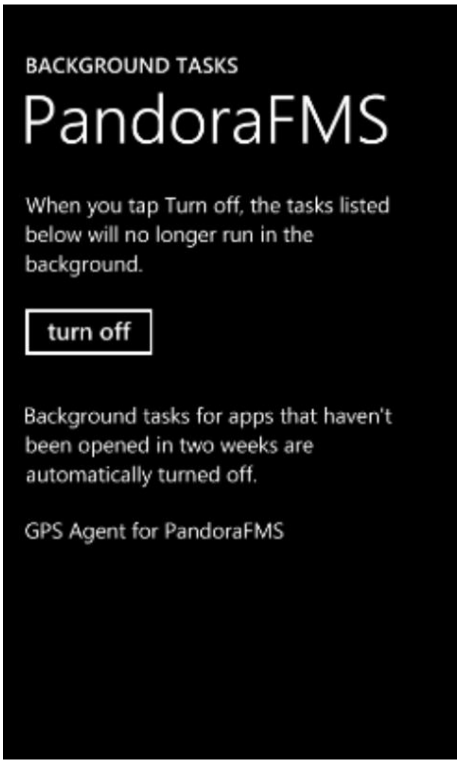
PandoraFMS Agent for WP7

Equipment Name :

IP Server : 186.69.209.162

Curr. Mem. Usage : 11821056
Peak Mem. Usage : 11821056
Mem. Usage Limit : 290488320
Manufacturer : Microsoft
Device Name : XDeviceEmulator
Firmware Version : 0.0.0.0
Hardware Version : 0.0.0.0
Total Memory : 447774720
Keyboard : Yes
Power Source : External
Unique ID : rLhmPJWf46VBwvaikp+w
Latitude : 47.64483
Longitude : -122.141197
Altitude (m): NaN
Speed (m/s): NaN

You must type the Equipment Name and the IP Server address, the press the Register button and you are ready, the agent will be registered as a background task and you can confirm this in Settings -> Applications -> Background Tasks:



BACKGROUND TASKS

PandoraFMS

When you tap Turn off, the tasks listed below will no longer run in the background.

Background tasks for apps that haven't been opened in two weeks are automatically turned off.

GPS Agent for PandoraFMS

Notice the warning in this screen: If you don't open the PandoraFMS Agent in two weeks, the agent will be turned off. Our advice for this would be to develop this functionality in your enterprise applications used commonly by your customers, then the agent will not be turned off by the OS.

Returning to our application, the Equipment Name, IP Server fields and the "Register" button are disabled when you make the registration. The next step would be to make a test, to execute this, please press the "Test" button, dismiss the message box and quit the application. The first message will be sent in approximately 30 seconds to the PandoraFMS server. The new agent will be created automatically using the name you registered. Please be careful to register distinct equipment names.

The Agent screen also displays the information to be sent to the server and a Bing Maps instance with your actual geographical position. The definition of the map is in the file "MainPage.xaml", you should register in Bing Maps and generate a new key to be replaced in the XYZ … of the next definition:

```
<my:Map x:Name="map" Grid.Row="2" Height="600" ZoomLevel="12" Margin="0,10,0,10"
     CopyrightVisibility="Collapsed" ScaleVisibility="Visible"
     LogoVisibility="Collapsed"
     CredentialsProvider="XYZ your Bing Maps Key goes here">
   <my:Pushpin x:Name="position" Foreground="White" Background="Orange"
               Content="You are here" />
</my:Map>
```

The registration is executed by function `btnRegister_MouseLeave()`:

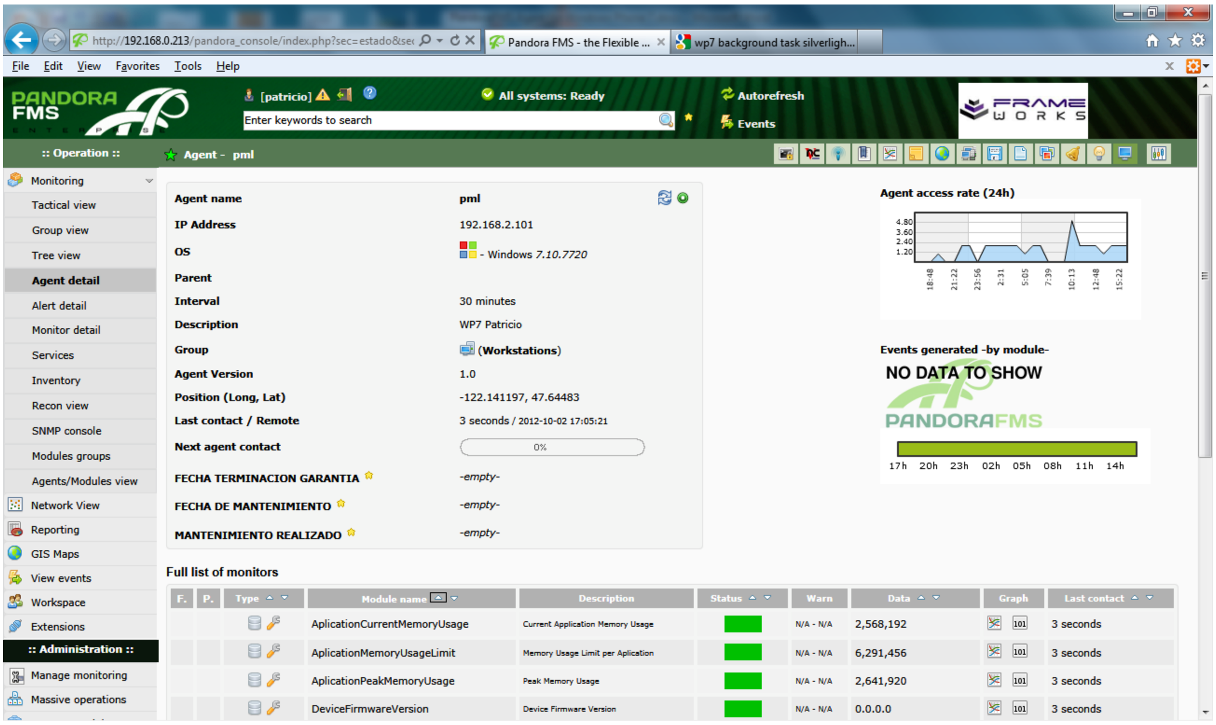
```
private void btnRegister_MouseLeave(object sender, MouseEventArgs e)
{
   if ((tbEquipment.Text.Length > 0) && (tbServer.Text.Length > 0))
   {
       Frameworks.GPSAgent.clsConfiguration.fnWriteData(tbEquipment.Text, tbServer.Text);
       PeriodicTask exist = ScheduledActionService.Find(taskName) as PeriodicTask;
       if (exist != null)
          ScheduledActionService.Remove(taskName);
       PeriodicTask task = new PeriodicTask(taskName)
       {
         Description = "GPS Agent for PandoraFMS"
       };
       ScheduledActionService.Add(task);
       tbEquipment.IsEnabled = false;
       tbServer.IsEnabled = false;
       btnRegister.IsEnabled = false;
       btnTest.IsEnabled = true;

       MessageBox.Show("You are now registered and your device will send your position to
server every 30 minutes");
   }
   else
   {
       MessageBox.Show("Please enter Device Name and Server IP Address");
   }
}
```

This function validates if the “tbEquipment” and “tbServer” fields have values, then writes the data to the Data Base using fnWriteData(). Then it confirms if the Agent has been previously registered, if it is, then removes the background task using “ScheduledActionService.Remove(taskName)”. The next instruction is for task creation and the task is registered using ScheduledActionService.add(task). Finally it disables the input fields and the register button and displays a MessageBox with the confirmation.

Let’s see the information in PandoraFMS Enterprise

PandoraFMS will receive the information through the Tentacle server. Please make sure the tentacle_serverd daemon is running and using port 41121, also make sure this port is open in your firewall. As soon as PandoraFMS receives the information, it will create a new Agent. Navigate to Monitoring -> Agent Detail and select the new Agent created with the name you registered in the phone. The information presented will be something like this:

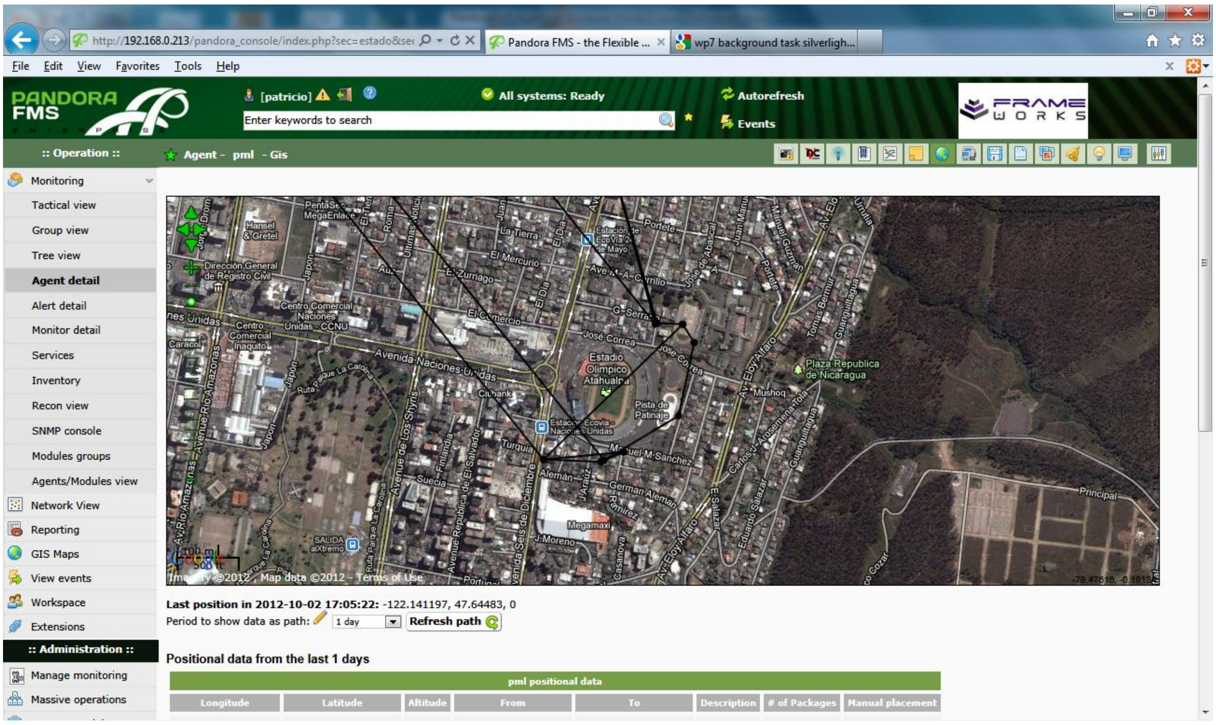


Agent name	**pml**
IP Address	192.168.2.101
OS	- Windows *7.10.7720*
Parent	
Interval	30 minutes
Description	WP7 Patricio
Group	**(Workstations)**
Agent Version	1.0
Position (Long, Lat)	-122.141197, 47.64483
Last contact / Remote	3 seconds / 2012-10-02 17:05:21
Next agent contact	0%
FECHA TERMINACION GARANTIA	*-empty-*
FECHA DE MANTENIMIENTO	*-empty-*
MANTENIMIENTO REALIZADO	*-empty-*

Full list of monitors

F.	P.	Type	Module name	Description	Status	Warn	Data	Graph	Last contact
			AplicationCurrentMemoryUsage	Current Application Memory Usage	🟩	N/A - N/A	2,568,192	101	3 seconds
			AplicationMemoryUsageLimit	Memory Usage Limit per Aplication	🟩	N/A - N/A	6,291,456	101	3 seconds
			AplicationPeakMemoryUsage	Peak Memory Usage	🟩	N/A - N/A	2,641,920	101	3 seconds
			DeviceFirmwareVersion	Device Firmware Version	🟩	N/A - N/A	0.0.0.0	101	3 seconds

Also if you select the “GIS data” button 🌐, located in the upper right side of the PandoraFMS console, then you will be tracking your mobile device every 30 minutes all around the earth planet. PandoraFMS uses Google Maps to display the Aerial Mode shown in the next screen:



Last position in 2012-10-02 17:05:22: -122.141197, 47.64483, 0

Period to show data as path: 1 day **Refresh path**

Positional data from the last 1 days

pml positional data							
Longitude	Latitude	Altitude	From	To	Description	# of Packages	Manual placement