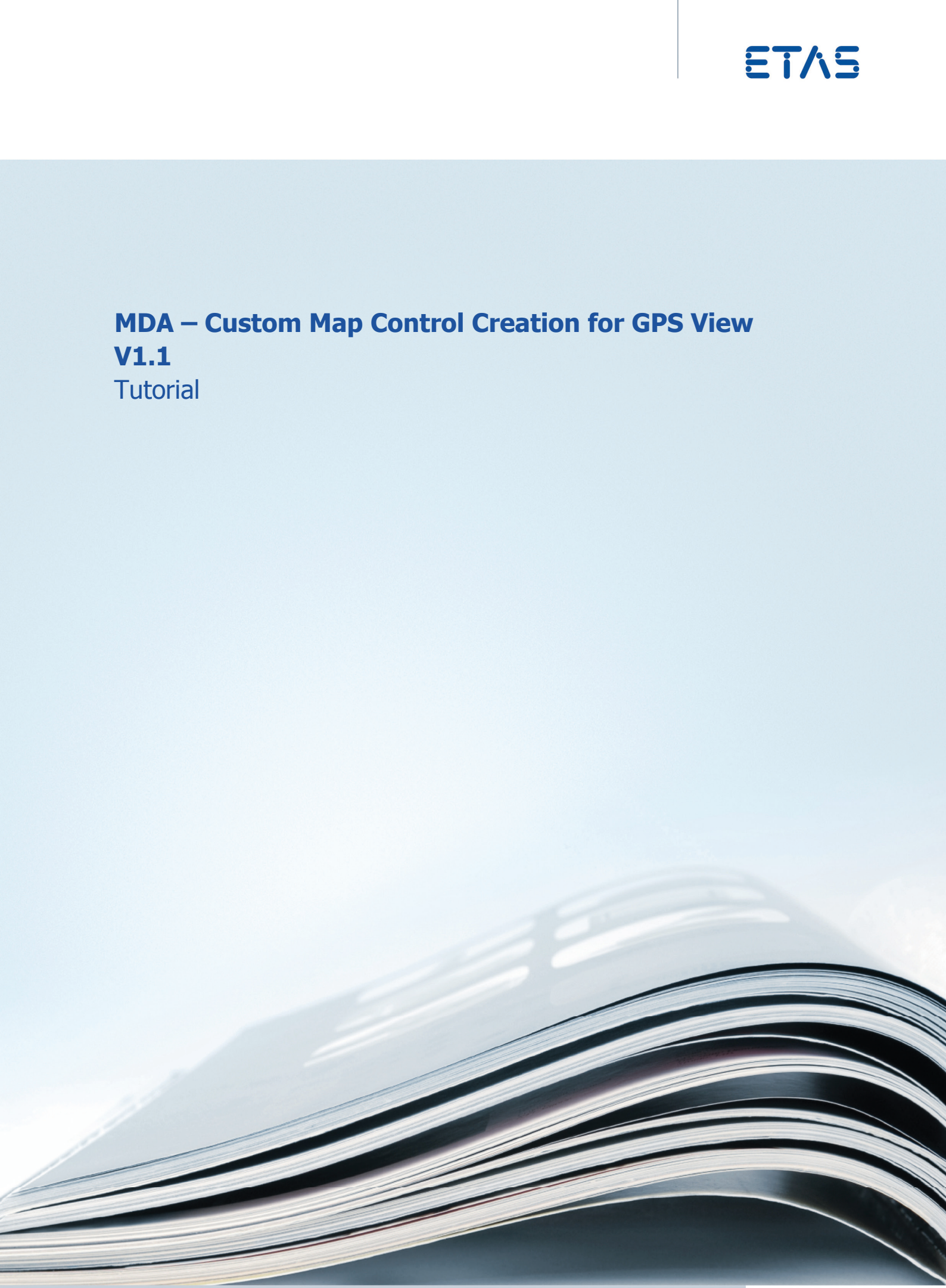
ETAS

MDA – Custom Map Control Creation for GPS View V1.1

Tutorial



Copyright

Document MDA-GPS-CustomMapControl_Tutorial V1.1 R03 EN – 03.2018

Contents

1 Introduction 4
1.1 Definitions and Abbreviations 4
1.2 Conventions 4
2 What you need to know 5
2.1 What you have learned after completing the tutorial 5
2.2 Prerequisites of the system 5
2.3 Prerequisites to use the tutorial successfully 5
2.4 Architecture of GPS View 6
3 Create a custom map control – "CustomMapControl" 8
3.1 Objectives 8
3.1.1 Create a new project 8
3.1.2 Add MapControlManager reference 10
3.1.3 Add Map Control provider reference 10
3.1.4 Data structures and enums in IMapControl 10
3.1.5 Implement IMapControl interface 11
3.1.5.1 CreateMapControl method 11
3.1.5.2 AddTrack method 11
3.1.5.3 UpdateTrack method 12
3.1.5.4 UpdateTrackProperties method 13
3.1.5.5 ZoomAndCenterTrack method 13
3.1.5.6 SetZoomLevel method 13
3.1.5.7 AddMarker method 14
3.1.5.8 UpdateMarker method 14
3.1.5.9 RemoveMarker method 14
3.1.5.10 RemoveTrack method 15
3.1.5.11 ZoomLevelChanged event 15
3.1.5.12 SetOfflineMapDataPath method 16
3.2 Use CustomMapControl in GPS View of MDA v7.2 16
4 Glossary 18
5 ETAS Contact Addresses 19

1 Introduction

MDA V7.1 offers the ability to show maps using a GPS view. By design MDA provides a framework for the integration of different map control providers, which actually realize the rendering and display of the maps. This enables customers to use their preferred map control, which may already be used in their organization. MDA provides OSM(via GMap.NET control) as a default map provider to render the map data in GPS View.

This tutorial shows the step by step procedure to implement custom map control and integrate with GPS view.

1.1 Definitions and Abbreviations

GPS

Global Positioning System

OSM

Open Street Map

1.2 Conventions

The following typographical conventions are used in this document:

Choose **File** → **Open**.	Menu commands are shown in boldface.
Click **OK**.	Buttons are shown in boldface.
Press ENTER.	Keyboard commands are represented in bold capitals
The **Open File** dialog box is displayed.	Names of programs, program windows, dialog boxes, fields, etc. are shown in bold font.
Select the file `setup.exe`	Code references like methods etc., path and file names are shown in Courier New font.
A *distribution* is always a one-dimensional table of sample points.	Note! and Warning! content as well as new or important terms are set in italics.

2 What you need to know

2.1 What you have learned after completing the tutorial

This tutorial will enable you:

- to create and implement custom map control
- to use the custom map control with the GPS view in MDAv7.1

2.2 Prerequisites of the system

- To implement `Custom Map Control` you need **Visual Studio 2010**
- The programming of Map Control has to be done in a .NET-language.
- Make sure your operating system supports the tools and standards listed above.
- Custom Map Control SDK that is of interest l

Note!

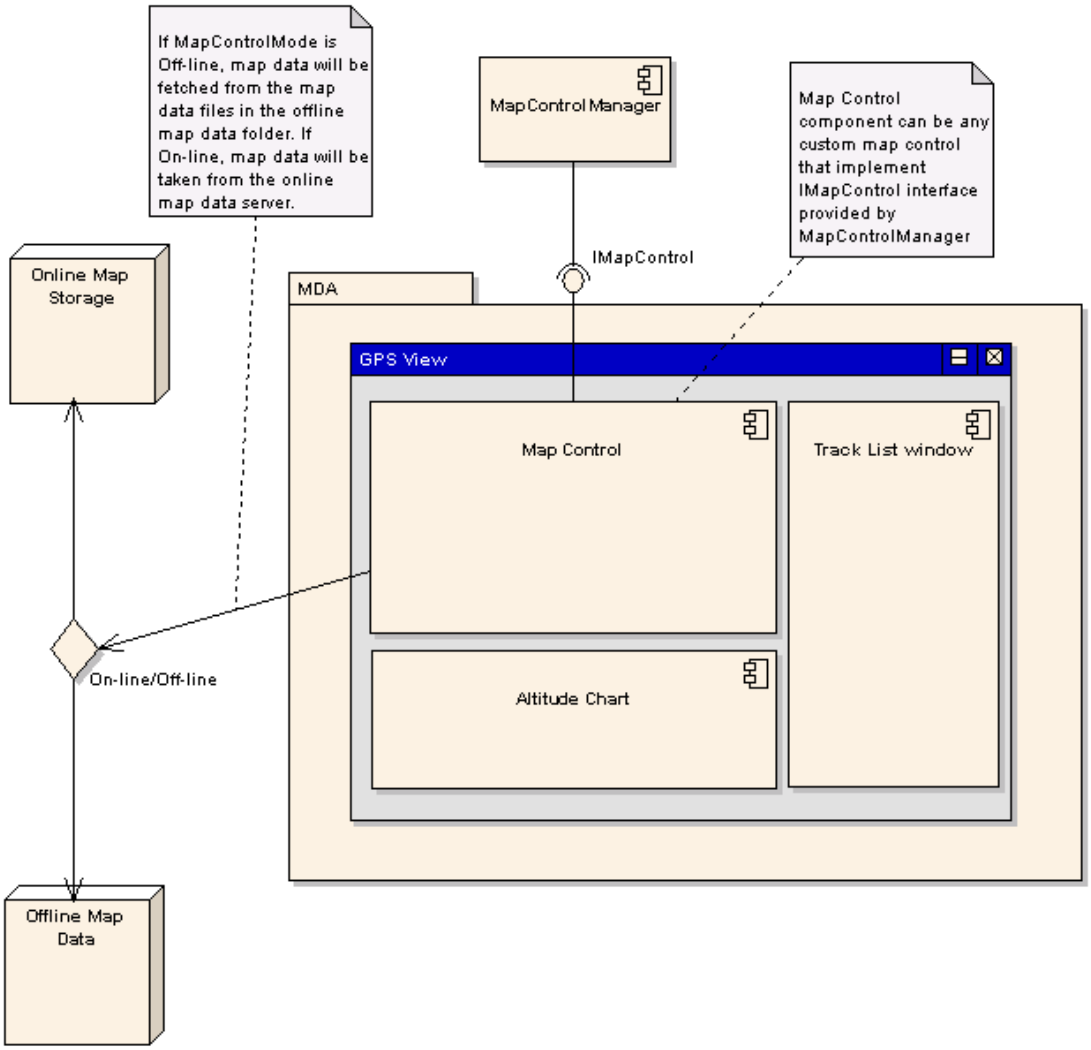
If the map control provider has certain license model, then your map control implementation is responsible for the doing the checks that the user has a valid license.

2.3 Prerequisites to use the tutorial successfully

This tutorial has been written for experienced software developers. Successful work with this tutorial requires a sound knowledge of:

- Knowledge of respective Map Control SDK
- Object Oriented Programming
- Basic knowledge of **Visual Studio 2010**
- The **.NET Framework 4.0**

2.4 Architecture of GPS View



The above diagram depicts the architecture of GPS View.

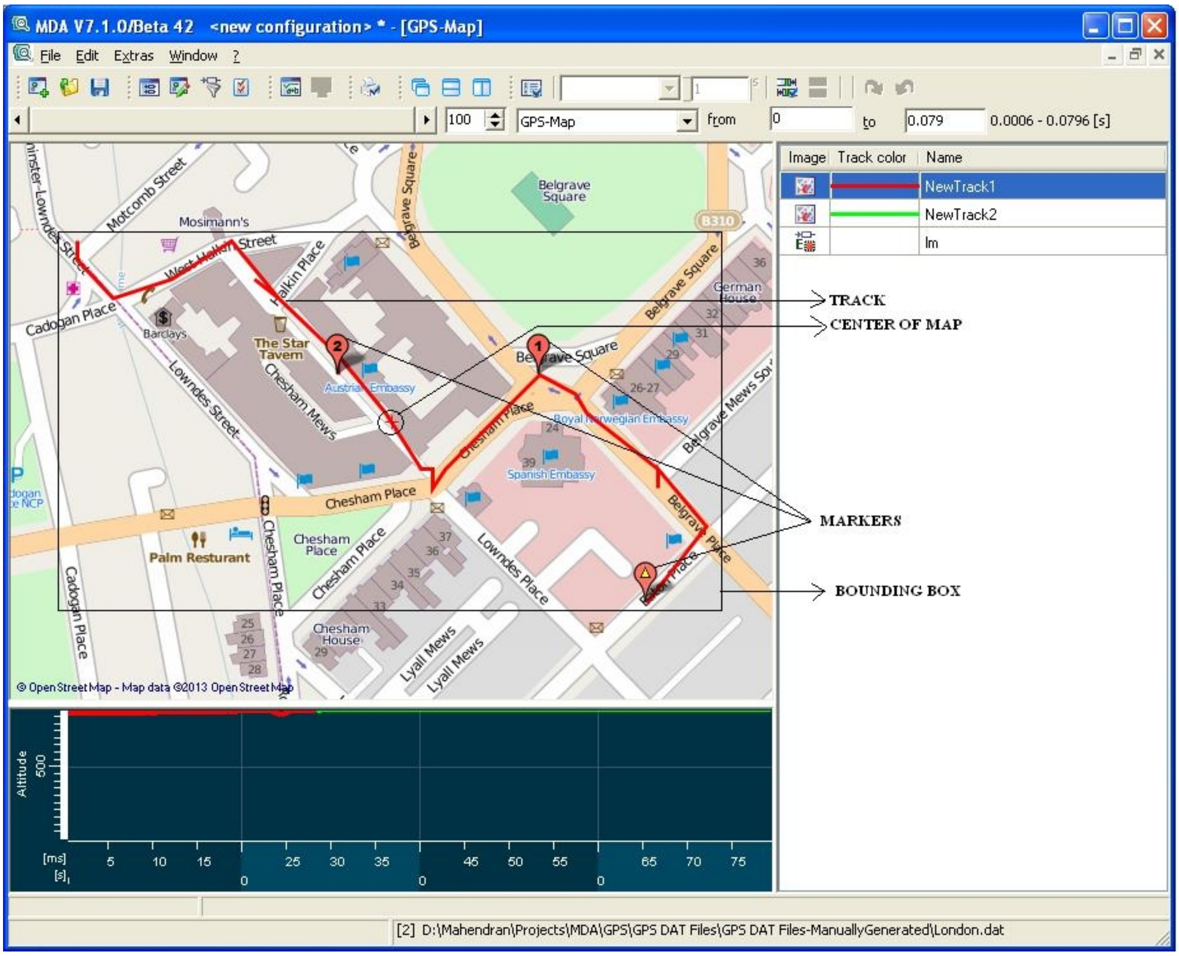
Responsibilities of MDA:

- Handling the assignments of tracks and markers
- Passing latitude and longitude points to the map control
- Managing the lifetime of the map control

Responsibilities of MapControl:

- Render the map data and display tracks and markers in the map
- Handling the zoom level changed events
- License checks specific to map provider/control
- Error handling related to map data rendering and license

Please refer to the image below to get familiarity with Map Control Terminology


MDA V7.1.0/Beta 42 <new configuration> * - [GPS-Map]
File Edit Extras Window ?
100
GPS-Map
from
0
to
0.079
0.0006 - 0.0796 [s]
Image
Track color
Name
NewTrack1
NewTrack2
lm
TRACK
CENTER OF MAP
MARKERS
BOUNDING BOX
© OpenStreetMap - Map data ©2013 OpenStreetMap
Altitude
500
[ms]
[s]
[2] D:\Mahendran\Projects\MDA\GPS\GPS DAT Files\GPS DAT Files-ManuallyGenerated\London.dat

3 Create a custom map control – "CustomMapControl"

3.1 Objectives

You will learn how to create a custom map control, here called "CustomMapControl".
You will learn how to use this custom map control in GPS View of MDA v7.1.

3.1.1 Create a new project

- Open **Visual Studio 2010**
- Select File **New Project**
- Select the project type as Class Library
- Change the project name to CustomMapControl
- Click **OK**

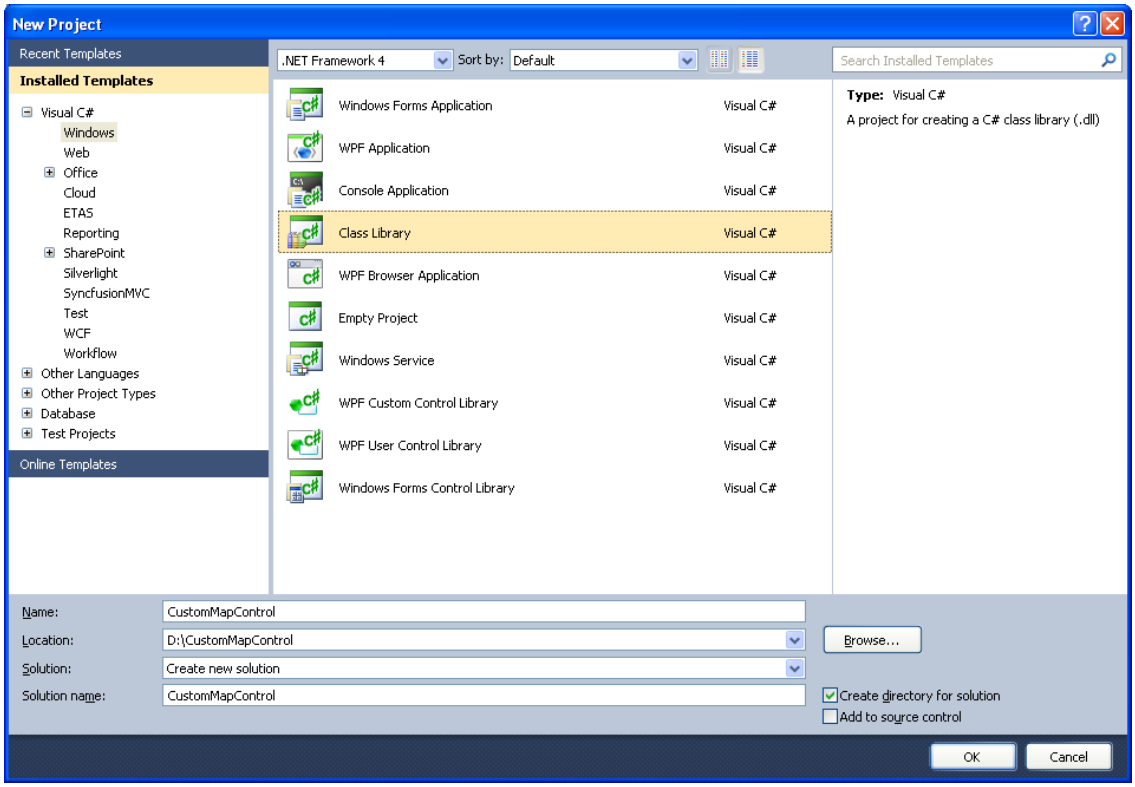
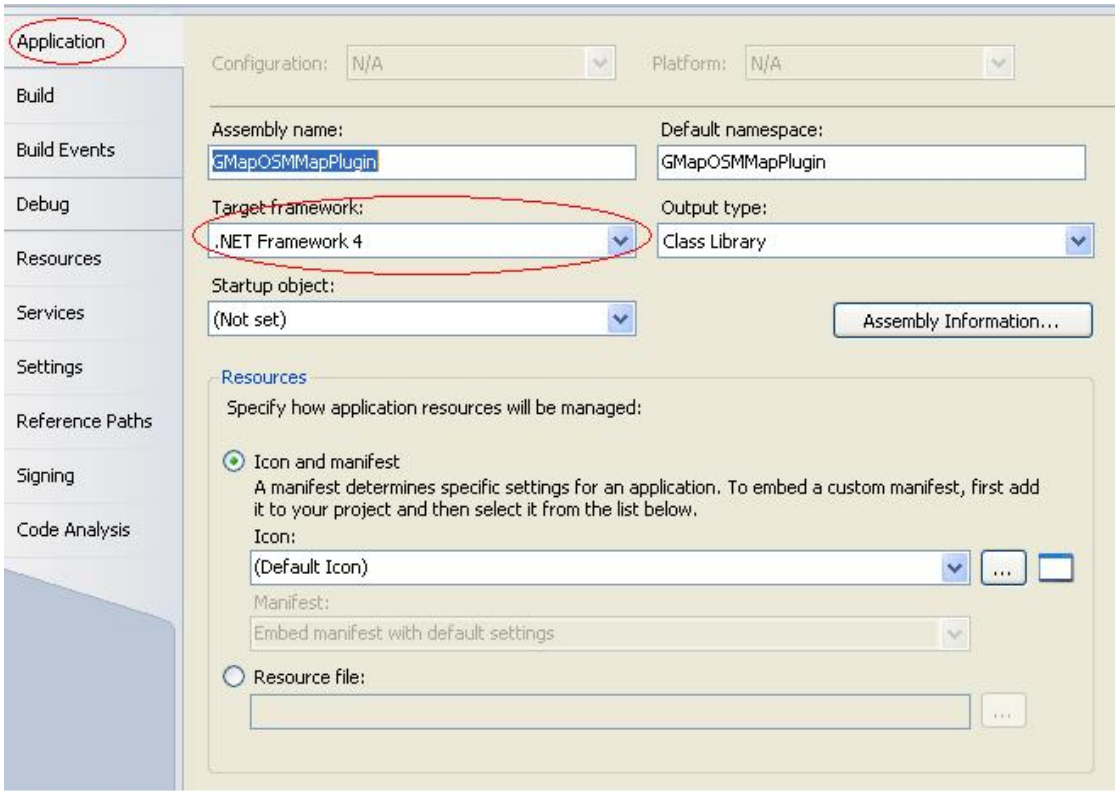


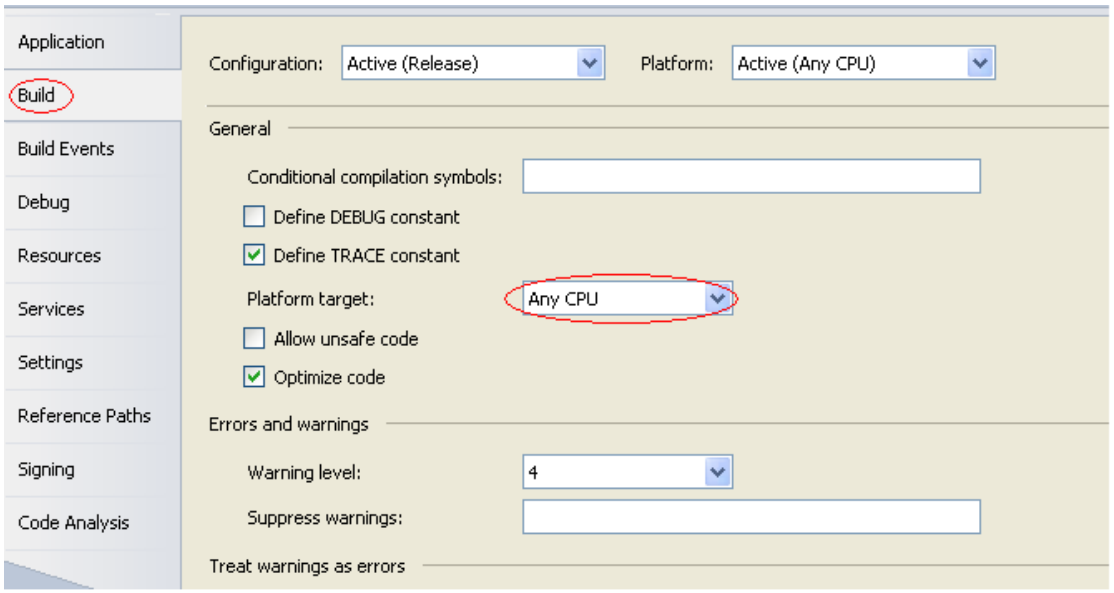
Fig 3 - 1 - Visual Studio Project Wizard

The **CustomMapControl project** is created.

- Open the project properties
- Check the Target Frame work version is **.NET Framework 4** as shown in the image below



- Check the Platform target is **Any CPU** as shown in the image below



3.1.2 Add MapControlManager reference

Add "MapControlManager.dll" assembly to the reference folder of the project

Note!

*The reference folder of your **Visual Studio project** contains a reference to the `MapControlManager.dll`.*

This is the DLL containing IMapControl interface classes for implementing custom map control.
Please find under<$MDAInstalledPath>|`MDA -> MapControlManager.dll`.

3.1.3 Add Map Control provider reference

Add the respective Map Control provider dependent assemblies as references of this project.

3.1.4 Data structures and enums in IMapControl

MapControlManager namespace contains the following data structures/enums which would be used by class implementing IMapControl

- TrackProperties: Represents the track properties like Color & Width of the track
- LatLong: Represents Latitude & Longitude of a track point. Latitude value ranges from -90 to 90 and longitude value ranges from -180 to 180 degrees.
- STATUSCODE: An enum represents the return value of actions like AddTrack, UpdateTrack etc which would be SUCCESS or FAILURE

```
public struct  TrackProperties
{
   public string Name;
   public int Width;
   public Color Color;
}

public struct LatLong
{
    public double Lat;
    public double Lng;
}


public enum STATUSCODE
{
    SUCCESS,
```

```
        FAILURE
    }

    public sealed class ZoomEventArgs : EventArgs
    {
        public ZoomEventArgs(double ZoomVal)
        {
            Zoom = ZoomVal;
        }

        public double Zoom
        {
            get;
            set;
        }
    }
```

3.1.5 Implement IMapControl interface

IMapControl is the generic interface that enables the use of any Custom Map Control in the GPSView of MDA. Following sections will describe the methods defined in the IMapControl interface that need to be implemented by the custom map control.

3.1.5.1 CreateMapControl method

Signature: UserControl CreateMapControl(string MapControlMode);

Description: Creates a new object of the System.Windows.Forms.UserControl which contains the Custom Map Control placed on the User Control. Initial default tile image is set in the map control. Ideal default image can be showing the entire globe in the map control.

Pre-condition: CustomMapControl is already loaded into the GPS View

Post-condition: User Control is successfully created. User control shows entire globe with the least number of tile images.

Parameters:

- MapControlMode
  - Type: string
  - Description: Input string stating should map data be read from internet connection or offline. Allowed input strings are On-line and Off-line.

Return Value:

- System.Windows.Forms.UserControl

  This will be placed into the Map region of GPS View, during the load time.

3.1.5.2 AddTrack method

Signature: STATUSCODE AddTrack(string TrackGUID, TrackProperties TrackProps, List<LatLong> LatLngPoints);

Description: Creates a new track of LatLon point in the LatLngPoints parameter. Sets the Color and Width of the Route. No changes required to the zoom level, center of the map and bounding box of the map control.

Pre-Condition: TrackGUID should be unique and LatLngPoints should have valid latitude and longitude values.

Post-Condition: Track is successfully created and shown in the Map Control.

Error-Condition: If the track with the TrackGUID already exists, then return value will be FAILURE which is of type enum STATUSCODE.

Parameters:

- TrackGUID
  - Type: string
  - Description: Contains the track ID of the new track to be created
- TrackProperties
  - Type: TrackProperties
  - Description: Contains the Color & Width of the track to be created
- List<LatLong> - Consists of the various Latitude & Longitude points of the track to be created.

Return Value:

- STATUSCODE:

3.1.5.3 UpdateTrack method

Signature: STATUSCODE UpdateTrack(string TrackGUID, List<LatLong> LatLngPoints);

Description: Replaces existing track with the new track in the map control with new set of latitude/longitude points and redraws it. No changes required to the zoom level, center of the map and bounding box of the map control.

Pre-condition: Track with the TrackGUID should already be added to the Map Control and LatLngPoints should have valid latitude and longitude values.

Post-condition: Track is shown in the Map Control with the update latitude and longitude values.

Error-condition: If the track with the TrackGUID doesn't exist, then no track will be updated. Return value will be FAILURE which is of type enum STATUSCODE.

Parameters:

- TrackGUID
  - Type: string
  - Description: Contains the name of the track to be updated.
- List<LatLong> - Consists of the various Latitude & Longitude points

Return Value:

- STATUSCODE

3.1.5.4 UpdateTrackProperties method

Signature: STATUSCODE UpdateTrackProperties(string TrackGUID, TrackProperties TrackProps);

Description: Redraws an already existing track with new TrackProperties – Color & Width.

Pre-condition: Track with the TrackGUID should already be there in the Map Control

Post-condition: Track color and width is applied to the track and shown in the Map Control

Error-condition: If the track with the TrackGUID doesn't exist, then no properties will be updated for the track. Return value will be FAILURE which is of type enum STATUSCODE.

Parameters:

- TrackGUID
  - Type: string
  - Description: Contains the track name whose properties needs to be updated

- TrackProperties
  - Type: TrackProperties
  - Description: Contains the Color & Width of the track to be updated

Return Value:

- STATUSCODE

3.1.5.5 ZoomAndCenterTrack method

Signature: STATUSCODE ZoomAndCenterTrack(string TrackGUID);

Pre-condition: Track with the TrackGUID should already be there in the Map Control

Post-condition: Track with the TrackGUID is fit to the visible area of the map control by calculating the track's left, top, and right and bottom position based on the min and max values of latitude and longitude points. Center of the map is changed and zoom level is changed to fit the track within the visible area of the map control. Other tracks may or may not be fully visible.

Error-condition: If the track with the TrackGUID doesn't exist, then no track will be aligned. Return value will be FAILURE which is of type enum STATUSCODE.

Description: Zooms and centers the track in context. So that the visible area of the map is set to the bounding box of a track.

Parameters:

- TrackGUID
  - Type: string
  - Description: Name of the track whose bounding box needs to be set.

Return Value:

- STATUSCODE

3.1.5.6 SetZoomLevel method

Signature: STATUSCODE SetZoomLevel(double ZoomValue);

Pre-condition: Map Control is loaded in the GPS View

Post-condition: Zoom level is applied to the Map Control. Bounding box and visibility of all the tracks in the map control will be changed according to the zoom level.

Error-condition: If the ZoomValue is not within the allowed range of map control's allowed valid range, and then return value will be FAILURE which is of type enum STATUSCODE.

Description: Sets the Zoom level of the Map control. Center of the map is not changed. Bounding box and zoom level of the map control is changed. If the value is outside the range then min or max value of the Map Control is used instead.

Parameters:

ZoomLevel – zoom level to be set for the map control.

Return Value:

- STATUSCODE

3.1.5.7 AddMarker method

Signature: STATUSCODE AddMarker(string MarkerID, LatLong LtLg, string ToolTip, Bitmap Bmp);

Pre-condition: MarkerID should be unique and LtLg should be a valid latitude and longitude value and Bmp should have the valid image.

Post-condition: Marker is added to the MapControl at the specified latitude and longitude point and the tool tip will be shown when the mouse pointer is over the marker.

Description: Creates a new Marker in the map control at the latitude/longitude point specified. Sets the tool tip of the marker as specified. If the bitmap is null, then default bitmap is used to draw the markers. No changes required to the zoom level, center of the map and bounding box of the map control. Also tags the marker with the MarkerID.

3.1.5.8 UpdateMarker method

Signature: STATUSCODE UpdateMarker(string MarkerID, LatLong LtLg);

Pre-condition: Marker with MarkerID should already be there in the Map Control

Post-condition: Marker is shown in the updated location of LtLg

Error-condition: If the marker with MarkerID doesn't exist, then it will not be updated and return value will be FAILURE which is of type enum STATUSCODE.

Description: Replaces and Redraws a marker with MarkerId at latitude/longitude point specified. No changes required to the zoom level, center of the map and bounding box of the map control.

Return Value:

- STATUSCODE

3.1.5.9 RemoveMarker method

Signature: STATUSCODE RemoveMarker(string MarkerID);

Pre-condition: Marker with the MarkerID should already be there in the Map Control

Post-condition: Marker is removed and disappeared from the Map Control

Error-condition: If the marker with MarkerID doesn't exist, then no operation and

return value will be FAILURE which is of type enum STATUSCODE.

Description: Removes a marker with tag MarkerID from the map control.

Return Value:

- STATUSCODE

3.1.5.10 RemoveTrack method

Signature: STATUSCODE RemoveTrack (string Name);

Pre-condition: Track with the Name should already be there in the Map Control

Post-condition: Track is removed and disappeared from the Map Control. Removing the track will not remove markers.

Error-condition: If the track with the Name doesn't exist, no track will be deleted and STATUSCODE returned will be FAILURE.

Description: Removes a track with the Name from the map control.

Return Value:

- STATUSCODE

3.1.5.11 ZoomLevelChanged event

Signature: event EventHandler<ZoomEventArgs> ZoomLevelChanged;

Description: Occurs every time the user modifies the Zoom Level of the map control.

MapControl needs to handle the MouseDoubleClick and MouseWheel events and trigger ZoomLevelChanged event by passing the updated zoom level value.

Sample code snippet is given below:

```
    private void gMapControl1_DoubleClick(object sender, EventArgs e)
        {
            if ((Control.ModifierKeys & Keys.Control) == Keys.Control)
            {
                this.gMapControl1.Zoom--; //Reduce the zoom level on CTRL + Mouse
double click
            }
            else
            {
                this.gMapControl1.Zoom++; //Increase the zoom level on Mouse
double click
            }

       //  Notify the GPS View by passing the updated zoom level
            TriggerZoomLevelChanged(this, new ZoomEventArgs(gMapControl1.Zoom));
        }

        private void gMapControl1_MouseWheel(object sender, MouseEventArgs e)
        {
       //  Notify the GPS View by passing the updated zoom level
            TriggerZoomLevelChanged(this, new ZoomEventArgs(gMapControl1.Zoom));
        }
```

3.1.5.12 SetOfflineMapDataPath method

Signature: STATUSCODE SetOfflineMapDataPath(string MapDataPath)

Pre-condition: MapDataPath should be a valid folder path

Post-condition: If the folder name with Map Data is not available, it will be created and Map data is taken from this path if MapControlMode is off-line. If the folder doesn't contain the map data files, then error/empty tile images will be shown in the map control. This behavior depends on the map control provider.

Error-condition: If folder is missing and can't be created, then return value will be FAILURE which is of type enum STATUSCODE.

Description: Method is used for setting the folder path of the Map Data stored in the hard drive. Open Street Map Control uses the Map Data stored in this in folder in case the mode is set to off-line.

Parameters:

- MapDataPath
  - Type: string
  - Description: Input string to specify the folder path of the map data

Return Value:

- STATUSCODE

Note:

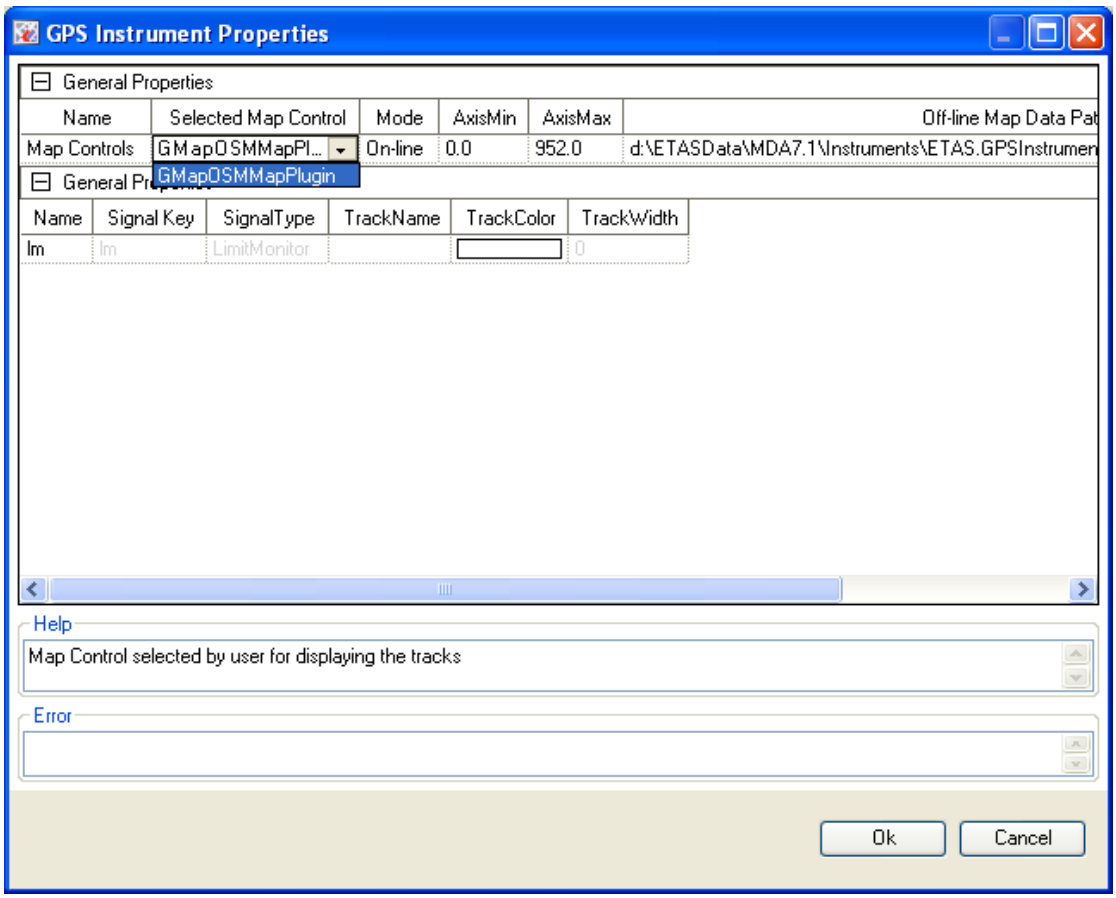
User might have to copy the off-line map data files to this folder manually. In case the folder doesn't contain required map data files, then error/empty tile images will be shown based on the map control provider.

3.2 Use CustomMapControl in GPS View of MDA v7.2

With the **Build** **Visual Studio,** generates the CustomMapControl.dll.

User needs to copy the CustomMapControl.dll & the required dependant assemblies to the folder <$ETASData>\MDA7.1\Instruments\ETAS.GPSInstrument\MapControlPlugins>.

MDA v7.1.0 is able to recognize the new Map Control. User is able to select the Map Control from the GPS-Map View Properties window. Properties window can be opened by right clicking on the GPS Instrument and clicking on the Properties context menu item. Please see the image below:


GPS Instrument Properties
General Properties
Name
Selected Map Control
Mode
AxisMin
AxisMax
Off-line Map Data Pat
Map Controls
GMapOSMMapPl...
On-line
0.0
952.0
d:\ETASData\MDA7.1\Instruments\ETAS.GPSInstrumen
GMapOSMMapPlugin
General Pr
Name
Signal Key
SignalType
TrackName
TrackColor
TrackWidth
lm
lm
LimitMonitor
0
Help
Map Control selected by user for displaying the tracks
Error
Ok
Cancel

4 Glossary

The chapter titled "Glossary" explains all technical terms used in the manual. The terms are listed in alphabetic order.

Marker	Icon/Image that shows the particular position in the track
Track	Path travelled by a vehicle shown as a route in the Map Control

5 ETAS Contact Addresses

ETAS HQ

ETAS GmbH
Borsigstraße 24 Phone: +49 711 3423-0
70469 Stuttgart Fax: +49 711 3423-2106
Germany WWW: www.etas.com

ETAS Subsidiaries and Technical Support

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

ETAS subsidiaries WWW: www.etas.com/en/contact.php
ETAS technical support WWW: www.etas.com/en/hotlines.php