MapPro80 SDK, Reference Manual Addendum – Rev: 3.19

Documents modifications and enhancements since the original printing of the User's Reference Manual and includes all previous addenda.

Contents

A. Modifications to Landmark searching

The following modification was made to the ExecLandMark, dialog:

If only a specific general category of Landmarks are of interest, then the appropriate category string is appended at the end of the search string, following the ">" symbol. The category string to be used, depends on which landmark database is selected, with the .LandmarkSource property. See section on LandmarkSource. For example, if LandmarkSource=0, then one of the two letter TANA categories can be used (e.g., "rockland,MA>AL"), if LandmarkSource=1, then one of the GNIS category strings can be used (e.g. "rockland,MA>School").

The same applies

- ***(i) when using the ExecSearch dialog with the Option to search for Landmarks.***
- ***(ii) when using the FindLandmarkFirst() or FindLandmarkList()functions.***
  Note that when using the TANA Landmarks database with the TANA09Q4 dataset (Based on Dynamap data) and FindLandMarkFirst or FindLandMarkList(), the original method of specifying the CFCC category as the<u>Filter</u> string may also be used. When using the TANA11Q2 data and later (based on Multinet) the 4-digit category can be used, instead of the CFCC. Look at the documentation of the FindLandMarkFirst in this document for more details.

C# Example

```
private void menuItem171_Click(object sender, EventArgs e)
 {
  // --- Using the TANA09Q4, Dynamap Dataset ---
  listBox1.Items.Clear();
  axMapPro1.MapProperties.LandmarkSource = MapPro80.TxLandmark.tmGNIS;
  // This call would work for LandmarkSource = tmGNIS, all Landmarks
  // with a "School" it its name, and All CFCCs, since Option=""
  string s = axMapPro1.FindLandmarkFirst("Boston,MA>School", -71.05, 42.34,
20, "");
  while (s!=null)
    {
      listBox1.Items.Add(s);
      s = axMapPro1.FindLandmarkNext();
    }
  axMapPro1.FindLandmarkClose();
  listBox3.Items.Clear();
  axMapPro1.MapProperties.LandmarkSource = MapPro80.TxLandmark.tmTANA;
  // This call would work for LandmarkSource = tmTANA, all Landmarks
  // with a "School" it its name, and those with CFCC=D43
  s = axMapPro1.FindLandmarkFirst("Boston,MA", -71.05, 42.34, 5, "D43");
  while (s != null)
    {
      listBox3.Items.Add(s);
      s = axMapPro1.FindLandmarkNext();
    }
  axMapPro1.FindLandmarkClose();
 }
```

B. Modifications to the Magnitude property array, new entries # 18, 19 and 20

Magnitude:TxMagnitude — Property

This property controls the scale of the map in the viewport. When the magnitude property is set, the value of the internal "Scale" property changes as well.

Magnitude is an enumerated variable that can take the following values (Note that as you can see below, the scale of Magnitude is approximate. The Miles property can be used for more accurate scaling):

Index	Enumerated	Miles value	
0	M1000_00	(965.58 miles property setting)	
1	M0500_00	(482.79 miles property setting)	
2	M0300_00	(289.67 miles property setting)	
3	M0200_00	(193.12 miles property setting)	
4	M0100_00	(96.56 miles property setting)	
5	M0050_00	(48.28 miles property setting)	
6	M0030_00	(28.97 miles property setting)	
7	M0020_00	(19.31 miles property setting)	
8	M0010_00	(9.66 miles property setting)	
9	M0005_00	(4.83 miles property setting)	
10	M0003_00	(2.9 miles property setting)	
11	M0002_00	(1.93 miles property setting)	
12	M0001_00	(0.97 miles property setting)	
13	M0000_50	(0.48 miles property setting)	
14	M0000_30	(0.29 miles property setting)	
15	M0000_20	(0.19 miles property setting)	
16	M0000_10	(0.1 miles property setting)	
17	M0000_05	(0.05 miles property setting)	
18	***M0000_03***	***(0.03 miles property setting)***	***- New***
19	***M0000_02***	***(0.02 miles property setting)***	***- New***
20	***M0000_01***	***(0.01 miles property setting)***	***- New***

VB.Net Example

```
Private Sub Button125_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button125.Click
        ' Routine to list the Built-in, preset scale magnitudes
        Dim i As Integer
        Dim smag As String
        Dim MyEnumVal As MapPro80.TxMagnitude
        For i = 0 To 20
            AxMapPro1.Magnitude = CType(i, MapPro80.TxMagnitude)
            AxMapPro1.Redraw()
            smag = [Enum].GetName(GetType(MapPro80.TxMagnitude), i)
            ListBox1.Items.Add(i.ToString + ":" + smag + " (" + _
            AxMapPro1.Miles.ToString + " miles property setting)")
            System.Threading.Thread.Sleep(1000)
        Next
End Sub
```

C. Enhancements to Search Dialog Functions

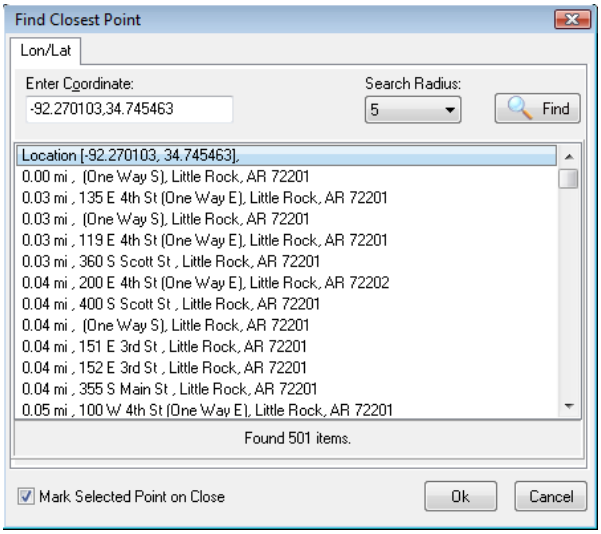
A string parameter was added to the Search dialogs to return the item the user had selected, prior to closing the dialog.

ExecClosest(Result:String) — Procedure

Opens a dialog that permits the user to search and display up to 500 street segments closest to the current view port center point (its coordinates displayed in the dialog). Double clicking on one of the 500 listed street segments names, or highlighting it and clicking OK, will reposition the viewport around that point, at the Zoom scale specified by the user. Note that the first entry in the list corresponds to the Lon/Lat coordinates used as the center of search.

Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.

Note that the companion functions *FindClosestStreetFirst*, *FindClosestStreetNext* can also be used to find the closest street info and then manage that info programmatically.



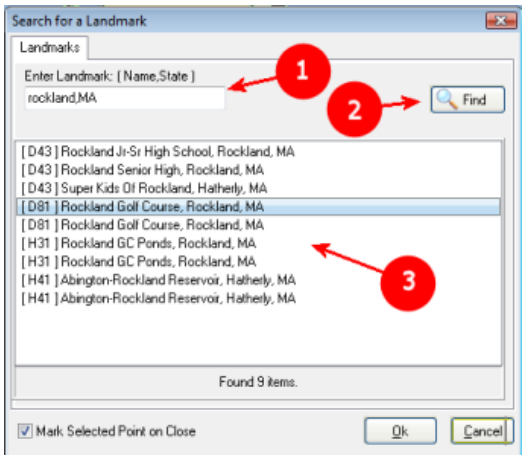
VB.Net Example

```
Private Sub Button65_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button65.Click
Dim sRes As String
        ' Test the ExecClosest Function
        AxMapPro1.ExecClosest(ref sRes)
End Sub
```

ExecLandMark(Result:String); Procedure

Opens up the Search dialog and enables the user to search for a landmark point. Note that within the context of MapPro, Landmark points are different than POIs. They originate from different databases and are treated different.

Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.



To search for a Landmark, enter part of the Landmark name followed by a comma and a two-letter space abbreviation **(1)** and press Find **(2)**. A sub-string search is performed based on the portion of the name and within the state you specified. For example, if you specified “rockland,MA” all landmarks that contain “rockland” in their name and are located in “MA” will be returned **(3)**.

If, on the other hand you spacified “rock, MA”, then a lot more landmarks would be returned, i.e., all landmarks in MA, with “rock” in their name.

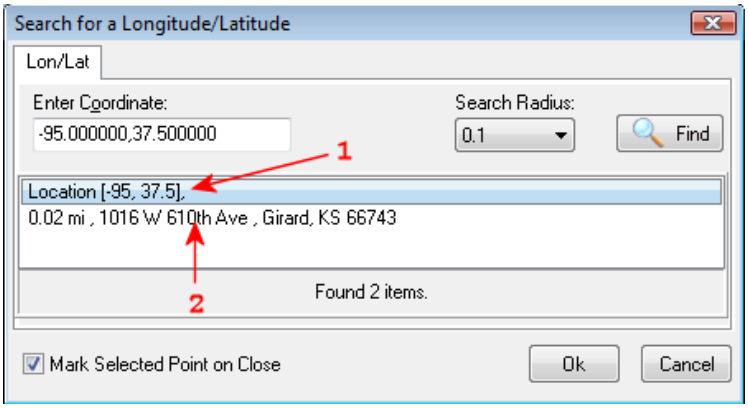
It is is also possible to search the entire database (all states) for a landmark, by specifying “*” for the state name. That, however, is not recommended, because such a search would take a significant length of time.

VB.Net Example

```
Private Sub Button66_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button66.Click
Dim sRes As String
        'Test ExecLandMark
        AxMapPro1.ExecLandmark(ref sRes)
End Sub
```

ExecLonLat(Result:string) **Procedure**

Presents user with a dialog that permits the input of Latitude and longitude coordinates, and upon confirmation, places the viewport around the point specified by the user. Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.



The search returns two hits. (1) the location of the coordinates that the user entered, and (2) the closest address (road segment) to the coordinates the user entered. Double-clicking on either of the two hits, or highlighting one and clicking O.K., centers the Viewport around that location.

VB.Net Example

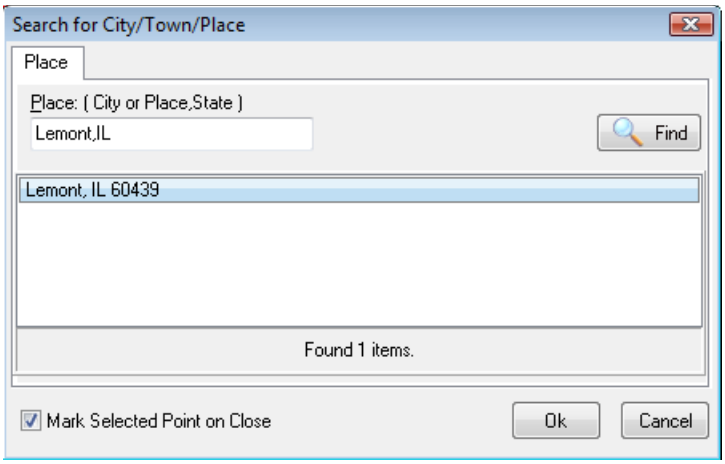
```
Private Sub Button68_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button68.Click
Dim sRes As String
        ' Test ExecLonLat
        AxMapPro1.ExecLonLat(ref sRes)

End Sub
```

ExecPlace(Result:String) **Procedure**

Presents user with the standard place search dialog. Once a Place is specified and the search is completed, the user is presented with a listbox containing the retrieved matches. Selecting one of the listbox choices presented, by double clicking, will center the viewport at the Lon/Lat of the selected Place centroid.

Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.



Entering a place name followed by an asterisk (wildcard) will find all places that contain the specificed string, otherwise an exact match search is performed. For example, specifying "Lemont, IL" will return just an exact match of Lemont, IL 60439. Entering "Lem*,IL" will return two hits, "Lemont, IL 60439" and "Lemmon, IL 62590"

VB.NET Example

```
Private Sub Button72_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button72.Click
Dim sRes As String
        ' Test ExecPlace
        AxMapPro1.ExecPlace(ref sRes)
End Sub
```

ExecPOI(Result:String) — Procedure

This is the same as calling the **ExecSearch()** dialog with Option=4. The difference is that calling it this way opens it as a stand-alone dialog, rather than as part of a tabbed dialog. Refer to the explanation of the ExecSearch dialog for details.

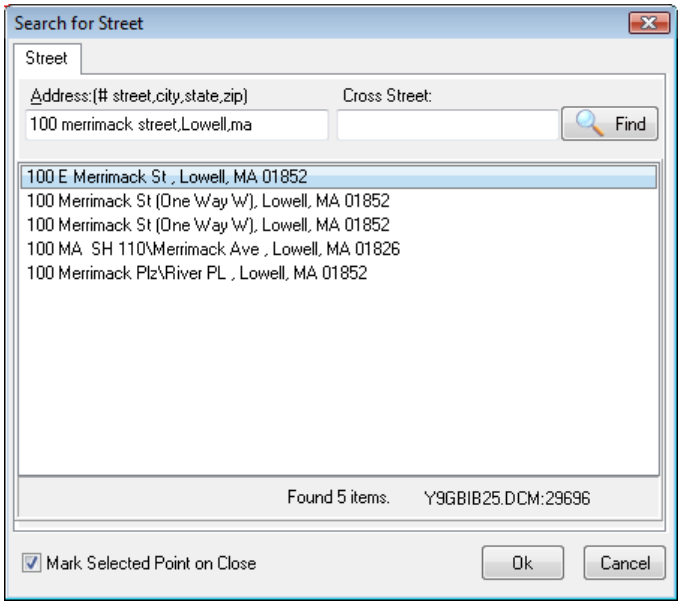
Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.

ExecStreet(Result:String) — Procedure

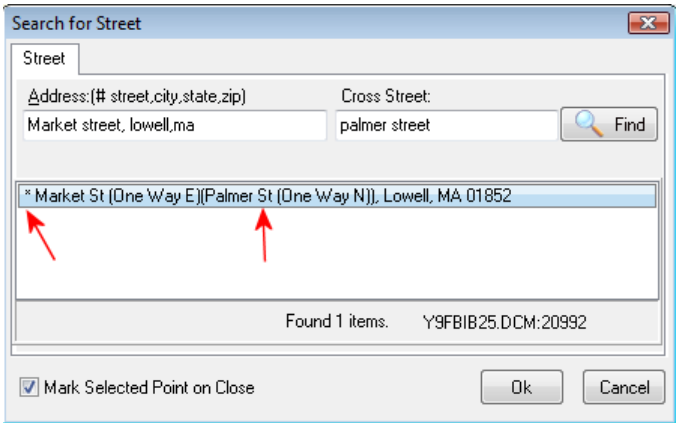
Presents user with a street search dialog shown below. Note that this dialog and its behavior are the same as the dialog invoked when ExecSearch is called with Option 0. However, when opened through this call, the dialog is stand-alone and not part of a tabbed dialog. It also does not provide

the user the opportunity to specify their own caption for the dialog. Once a street is specified and the search is completed, the user is presented with a listbox containing the retrieved matches. Selecting one of the listbox choices presented, by double clicking, or highlighting it and clicking OK will center the viewport at the Lon/Lat of the selected street segment.

Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.



Note that a cross Street may also be specified in the search. When a cross street search is requested, the result is a single hit, and it is identified by an asterisk in the first character position.



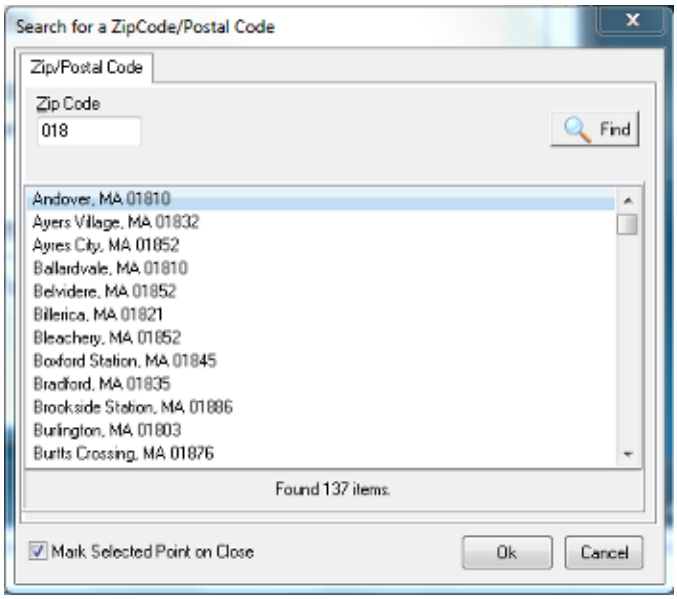
VB.Net Example

```
Private Sub Button76_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button76.Click
Dim sRes As String
        ' Test ExecStreet
        AxMapPro1.ExecStreet(ref sRes)
End Sub
```

ExecZipcode(Result:String) **Procedure**

Presents user with the standard Zip Code search dialog. Once a ZipCode is entered and the search is completed, the user is presented with a listbox containing the retrieved matches. Selecting one of the listbox choices presented, by double clicking, will center the viewport at the Lon/Lat of the selected ZipCode centroid. The substring search that is used allows the user to search for multiple ZipCodes in an area. For example, the screen shown below presents the results for all ZipCodes in the "018" area. The search was done by typing "018" as the criterion.

Result, contains the internal, unparsed string returned by the search engine, corresponding to the item the user selected from the list of returned results, prior to closing the dialog.



C# Example

```
private void button36_Click_1(object sender, EventArgs e)
  {
    string mResult = "";
    axMapPro1.ExecZipcode(ref mResult);
    axMapPro1.ExecZipcode();
  }
```

D. New Additions to the *MapProperties* Interface

RoadInnerColor(index:integer):Integer; — Property

Sets the color of roads, for the road type identified by index. Note that in some development environments, indexed properties cannot be accessed directly and the helper functions ***_getRoadInnerColor(i)*** and ***_setRoadInnerColor(i)*** need to be used, instead.

Where

Index =	0	Interstate Highways
	1	US Highways
	2	Secondary Highways (State highways, County Road, etc.)
	3	Local Roads
	4	Jeep Trails
	5	Access Ramps
	6	Ferries

Note that for categories 3,4,5,6 when the scale is below 0.01 miles, the user specified colors are not used, but the built-in default colors are used, instead.

C# Example

```
        private void menuItem163_Click_1(object sender, EventArgs e)
        {
            int RoadIndex;
            // Get index of road to change.  If none, then change road
            // index=o, i.e. Interstate
            if (textBox1.Text == "")
            {
                RoadIndex = 0;
            }
            else
            {
                RoadIndex = Convert.ToInt32(textBox1.Text);
            }

            // Get and display Interstate Inner Color
            listBox1.Items.Add("Inner Old:
"+axMapPro1.MapProperties.get_RoadInnerColor(RoadIndex).ToString());
            // Set and Interstate Color
            axMapPro1.MapProperties.set_RoadInnerColor(RoadIndex,
ColorTranslator.ToOle(Color.LightCoral));
            listBox1.Items.Add("Inner New: " +
axMapPro1.MapProperties.get_RoadInnerColor(RoadIndex).ToString());
            //--
            // Get and display Interstate Outer Color
            listBox1.Items.Add("Outer Old: " +
axMapPro1.MapProperties.get_RoadOuterColor(RoadIndex).ToString());
```

```
            // Set and Interstate Color
            axMapPro1.MapProperties.set_RoadOuterColor(RoadIndex,
ColorTranslator.ToOle(Color.Yellow));
            listBox1.Items.Add("Outer New: " +
axMapPro1.MapProperties.get_RoadOuterColor(RoadIndex).ToString());
            axMapPro1.Redraw();
        }
```

RoadOuterColor(index:integer):Integer; — Property

Sets the Outer color of *double-lined* roads, for the road type identified by index. Note that in some development environments, indexed properties cannot be accessed directly and the helper functions ***_getRoadOuterColor(i)*** and ***_setRoadOuterColor(i)*** need to be used, instead.

Where

Index =	0	Interstate Highways
	1	US Highways
	2	Secondary Highways (State highways, County Road, etc.)
	3	Local Roads
	4	Jeep Trails
	5	Access Ramps
	6	Ferries

LandmarkSource:TxLandmark; — Property

Specifies whether the TANA (TxLandmark.tmTANA or 1) or the GNIS (TxLandmark tmGNIS or 0) landmarks database will be used.

C# Example

```
        private void menuItem166_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            // Select TANA LandMarks
            axMapPro1.MapProperties.LandmarkSource =
                  MapPro80.TxLandmark.tmTANA;
            // Specify a Glyphs bitmap file to use
            axMapPro1.MapProperties.LandmarkGlyphFilename =
            "c://mappro80//landSym.bmp";
            // Set Both Glyphs and Text Scale to 1.0 mile
            axMapPro1.MapProperties.LandmarkGlyphScale = 1.0;
            axMapPro1.MapProperties.LandmarkTextScale = 1.0;
            axMapPro1.Redraw();
        }
```

LandmarkFilter:widestring; **Property**

Specifies which landmark type should be rendered on the map. The format of the specified filter depends on the setting of the LandmarkSource property, but in all cases,

If LandmarkFilter = "*"
ALL Landmarks in the specified database will be rendered

If LandmarkFilter = "" *(blank)*
NO Landmarks in the specified database will be rendered

If LanmarkSource = 0 (and you are using the TANA09Q4 Dynamap Data)
The **two letter codes** of the TANA general categories below need to be specified, separated by a comma. (see table in *LandmarksGlyphFile* section for description of these Landmark categories).

AL (includes CFCC: D10, D31, D37, D43, D61, D62, D64, D67, D81, D82) **
AP (includes CFCC: D58, D59)
IN (includes CFCC: D31, D43, D44, D65, D82)
RA (includes CFCC: D81, D92)
TT (includes CFCC: D52, D53, D54)
RC (includes CFCC: D61)
PK (includes CFCC: D83, D85, D89)
WP (includes CFCC: H11, H21, H31, H41)

** There may be some overlap in the CFCCs, because these two groups are ***area*** landmarks and the same points may also be in one of the other groups as a point landmark. For example, using "RC" one would expect "Shopping Center" Landmarks to be visible, but they may not be, if the Shopping Centers in that area are defined as ***area*** Landmarks. In order to render those, the "AL" group should also be selected.

If LanmarkSource = 0 (and you are using the TANA11Q2 and later Multinet Data)
The **two letter codes** of the TANA general categories below need to be specified, separated by a comma. (see table in *LandmarksGlyphFile* section for description of these Landmark categories).

AL: includes 7110, 7180, 9353, 9715, 9733, 9744, 9748, 9768, 9771, 9780, 9788, 9789, 9790, 9791 ******

AP : includes 9732, 9776

RC: includes 7314,7317, 7318, 7320, 7321, 7339, 7352, 7356, 7366, 7367, 7374, 7375, 7376, 7380, 7383, 7395, 9357, 9362, 9364, 9377, 9382, 9902, 9911, 9913, 9920, 9935 **

PK: includes 7170

WP: includes 4310

** There may be some overlap in the Landmarks in these groups because some of them are defined both as points and as polygons/areas. In general, the 7… series are point landmarks and the 9… series are area landmarks.

If LanmarkSource = 1

The expanded GNIS category codes, see below, need to be specified, separated by a comma. (Note that the bolded, italics below are the general categories. Only the entries under these categories can be used as filters – not the categories themselves.

Dams
- Dam

Green Space
- Forest
- Woods
- Reserve

Mineral
- Mine
- Oilfield

Manmade
- Bldg
- Bridge
- Cemetery
- Church
- Hospital
- Levee
- Mil
- Milh
- School
- Tower
- Trail
- Tunnel
- Well

Terrain
- Area
- Arroyo
- Bench
- Cape
- Cliff
- Crater
- Flat
- Island
- Isthmus
- Lava
- Pillar
- Plain
- Range
- Ridge
- Slope
- Valley

Mountain/Hills
- Summit

Water Related
- Bar
- Basin
- Bay
- Beach
- Bend
- Canal
- Channel
- Falls
- Gap
- Geyser
- Glacier
- Gut
- Harbor
- Lake
- Rapids
- Resv
- Sea
- Spring
- Stream
- Swamp

C# Example

```
private void menuItem170_Click(object sender, EventArgs e)
        {
            // Set TANA Landmark Filter(s) – Dynamap Data
            axMapPro1.MapProperties.LandmarkFilter = "IN,RC";
            // --  Or
            // Set GNIS Landmark Filter(s)
            // axMapPro1.MapProperties.LandmarkFilter = "School";
            axMapPro1.Redraw();
        }
```

LandmarkGlyphScale:Double; **Property**

Specifies the scale (miles) below which the glyphs associated with the landmarks selected to be rendered, will be visible. The default value is 0.25 mi. Note, however, that there is a built-in upper value for visibility for the Landmarks layer of 1.0 miles, so no landmarks info is visible above 1.0 mile. Also, it should be noted that for the text to be visible, the glyph of the Landmark has to be visible, as well, so the user must make sure that,

LandmarkGlyphScale >= LandmarkTextScale

C# Example

```
private void menuItem366_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    // Select TANA LandMarks
    axMapPro1.MapProperties.LandmarkSource =
          MapPro80.TxLandmark.tmTANA;
    // Specify a Glyphs bitmap file to use
    axMapPro1.MapProperties.LandmarkGlyphFilename =
    "c://mappro80//landSym.bmp";
    // Set Both Glyphs and Text Scale to 1.0 mile
    axMapPro1.MapProperties.LandmarkGlyphScale = 1.0;
    axMapPro1.MapProperties.LandmarkTextScale = 1.0;
    axMapPro1.Redraw();
}
```

LandmarkTextColor:OleColor; **Property**

Specifies the font color to be used for displaying the string (name) associated with each landmark, when the tmGNIS option is selected and the GNIS landmarks database is used. This has no effect when using the tmTANA landmarks database, because those landmarks have different, hardwired font colors, depending on the general category they belong to.

Note that this is visible only when the map is at scales smaller than what is specified by LandMarkTextScale.

C# Example

```
private void menuItem167_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    // Select TANA LandMarks
    axMapPro1.MapProperties.LandmarkSource =
                MapPro80.TxLandmark.tmGNIS;
    // Set Landmark text color
    axMapPro1.MapProperties.LandmarkTextColor =
          ColorTranslator.ToOle(Color.LightGreen);
    // Echo some of the settings
    listBox1.Items.Add("GlyphScale = " +
```

```
            axMapPro1.MapProperties.LandmarkGlyphScale.ToString());
        listBox1.Items.Add("TextScale = " +
            axMapPro1.MapProperties.LandmarkTextScale.ToString());
        axMapPro1.Redraw();
    }
```

LandmarkTextScale:Double; — Property

Specifies the scale (miles) below which the labels associated with the landmarks selected to be rendered, will be visible. The default value is 0.25 mi. Note, however, that there is a built-in upper value for visibility for the Landmarks layer of 1.0 miles, so no landmarks info is visible above 1.0 mile. Also, it should be noted that for the text to be visible, the glyph of the Landmark has to be visible, as well, so the user must make sure that,

LandmarkTextScale =< LandmarkGlyphScale

C# Example

```
    private void menuItem366_Click(object sender, EventArgs e)
    {
        listBox1.Items.Clear();
        // Select TANA LandMarks
        axMapPro1.MapProperties.LandmarkSource =
            MapPro80.TxLandmark.tmTANA;
        // Specify a Glyphs bitmap file to use
        axMapPro1.MapProperties.LandmarkGlyphFilename =
        "c://mappro80//landSym.bmp";
        // Set Both Glyphs and Text Scale to 1.0 mile
        axMapPro1.MapProperties.LandmarkGlyphScale = 1.0;
        axMapPro1.MapProperties.LandmarkTextScale = 1.0;
        axMapPro1.Redraw();
    }
```

LandmarkGlyphFile:String; — Property

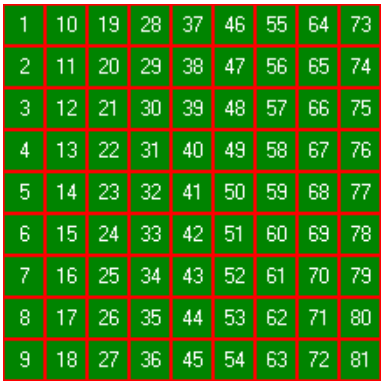
Specifies the bitmap file that contains the glyphs to be used for identifying Landmark locations on the map. If no file is specified, then the built-in, default glyphs (which are simply small rectangles of different colors) are used. A full file spec may be used. If ONLY a file name is specified, then the file is assumed to be in the local folder (where the application was executed from).

The specified glyphs file may be of BMP or GIF format and it is a single bitmap with a grid of 9x9 individual bitmaps for the Landmark categories embedded in it. The built-in bitmap uses simple representations for each Landmark glyph, i.e., a green rectangle with a number in it, that denotes its position in the glyphs matrix (see glyphs bitmap below).

The size of each individual glyph automatically calculated based on the size of the overall bitmap (the program divides the width/height of the glyphs bitmap image by 9 to get the size of each

individual glyph), so the user may actually define and use glyphs of any size they want. The only stipulation is that all bitmaps are the same size and they are rectangular.

The lower left pixel color is, of each individual glyph, is used as the transparent color. Here is the build-in bitmap containing all the default glyphs:



Depending on which Landmarks are being used (TANA or GNIS) here are the Col/row position of each Glyph in the Glyph file that would be used. For example, when using the TANA Landmarks, campgrounds, CFCC = D28, would be displayed using the glyph in col:2, row:2 of the Glyphs bitmap, i.e., when using the default bitmap above, a green rectangle with the #11.

Along the same lines, when using the GNIS database, Hospitals would be displayed using the glyph in col:3, row:5 of the Glyphs bitmap, i.e., when using the default bitmap above, a green rectangle with the #23..

Because the mapping of the Landmark categories is very different in the TANA and the GNIS databases, it is recommended that the user creates a couple of different Glyph bitmaps, one for each database and then select each in their code, depending on which database is used.

A sample Glyphs bitmap for use with the GNIS Landmarks database is provided as guidance, but the user is urged to create their own glyphs.

(I) Location of Glyphs in bitmap for the TANA09Q4, Dynamap Dataset

(Note that not ALL CFCC categories are available in the Landmarks file. The CFCC for those in the database, is bolded in the table below and the string that needs to be passed to the property MapProperties.LandmarkFilter to select that landmark category, is displayed next to it.)

Col	Row	CFCC	#	Description
1	1	D00	1	Major and minor categories unknown Military Installation Base, yard, or depot used by the U.S. Army, Navy, Air Force, Marines, he Coast Guard, or the National Guard. With the exception of the Coast Guard which is administered by the Department of Transportation, and the National Guard which is administered by states, these areas are administered by the U.S. Department of Defense.
	2	**D10 (AL)**	2	Military installation or reservation; major category used alone
	3	D20	3	Multi-household or transient quarters; major category used alone when the minor category could not be determined
	4	D21	4	Apartment building or complex
	5	D22	5	Rooming or boarding house
	6	D23	6	Trailer court or mobile home park
	7	D24	7	Marina
	8	D25	8	Crew-of-vessel area
	9	D26	9	Housing facility for workers
2	1	D27	10	Hotel, motel, resort, spa, hostel, YMCA, or YWCA
	2	D28	11	Campground
	3	D29	12	Shelter or mission
	4	D30	13	Custodial facility; major category used alone when the minor category could not be determined
	5	**D31 (IN,AL)**	14	Hospital
	6	D32	15	Halfway house
	7	D33	16	Nursing home, retirement home, or home for the aged
	8	D34	17	County home or poor farm
	9	D35	18	Orphanage
3	1	D36	19	Jail or detention center
	2	**D37 (AL)**	20	Federal penitentiary, state prison, or prison farm
	3	D40	21	Educational or religious institution; major category used alone when the minor category could not be determined
	4	D41	22	Sorority or fraternity
	5	D42	23	Convent or monastery
	6	**D43 (IN,AL)**	24	Educational institution, including academy, school, college, and university

	7	**D44 (IN)**	25	Religious institution, including church, synagogue, seminary, temple, and mosque
	8	D50	26	Transportation terminal; major category used alone when the minor category could not be determined
	9	D51	27	Airport or airfield
4	1	**D52 (TT)**	28	Train station
	2	**D53 (TT)**	29	Bus terminal
	3	**D54 (TT)**	30	Marine terminal
	4	D55	31	Seaplane anchorage
	5	D56	32	SRI Subway or metro station *** NEW FROM ESRI
	6	D57	33	Airport – Statistical Representation used as part of urban area delineation where major airports are contiguous with urban areas
	7	**D58 (AP)**	34	(TANA/GDT FCC) Airport property Boundary **NEW
	8	**D59 (AP)**	35	(TANA/GDT FCC) airport runway ** NEW
	9	D60	36	Employment center; major category used alone when the minor category could not be determined
5	1	**D61 (RC,AL)**	37	Shopping center or major retail center
	2	**D62 (AL)**	38	Industrial building or industrial park
	3	D63	39	Office building or office park
	4	**D64 (AL)**	40	Amusement center
	5	**D65 (IN)**	41	Government center
	6	D66	42	Other employment center
	7	**D67 (AL)**	43	Stadium (TANA) **NEW
	8	D70	44	Tower; major category used alone when minor category could not be determined
	9	D71	45	Lookout tower
6	1	D80	46	Open space; major category used alone when the minor category could not be determined
	2	**D81 (RA,AL)**	47	Golf course
	3	**D82 (IN,AL)**	48	Cemetery
	4	**D83 (PK)**	49	National Park Service land
	5	D84	50	National forest or other Federal land
	6	**D85 (PK)**	51	State or local park or forest
	7	**D89 (PK)**	52	Local Park
	8	D90	53	Special purpose landmark; major category used alone when the minor category could not be determined
	9	D91	54	Post office
7	1	**D92 (RA)**	55	Point of Interest (*** THIS WAS CHANGED)
	2	D93	56	Fire Department
	3	D94	57	Police Station
	4	D95	58	Library
	5	D96	59	City/Town Hall

	6	Reserved	60	
	7	Reserved	61	
	8	Reserved	62	
	9	Reserved	63	
8	1	**H10 (WP)**	64	Stream or river
	2	**H20 (WP)**	65	Canal, ditch, or aqueduct
	3	**H30 (WP)**	66	Lake or pond; major category used when the minor category could not be determined
	4	**H40 (WP)**	67	Reservoir
	5	H50	68	Bay, estuary, gulf, sound, sea, or ocean
	6	H60	69	Gravel pit or quarry filled with water
	7	H81	70	Glacier
	8	Reserved	71	
	9	Reserved	72	
9	1	Reserved	73	
	2	Reserved	74	
	3	Reserved	75	
	4	Reserved	76	
	5	Reserved	77	
	6	Reserved	78	
	7	Reserved	79	
	8	Reserved	80	
	9	Reserved	81	

(II) Location of Landmark Glyphs in bitmap for newer TANA Multinet datasets

(Note that not ALL Landmark Categories are available in the Landmarks file. The ones present in the data are presented in the table below.

Col	Row	CFCC	#	Description
1	1	Reserved	1	
	2	**(9388, 9789**	2	Military installation or reservation; major category used alone
	3	Reserved	3	
	4	**9381**	4	Apartment building or complex
	5	Reserved	5	
	6	Reserved	6	
	7	Reserved	7	
	8	Reserved	8	
	9	Reserved	9	
2	1	**7314, 9749**	10	Hotel, motel, resort, spa, hostel, YMCA, or YWCA
	2	**7360,9735**	11	Campground
	3	Reserved	12	
	4	Reserved	13	
	5	**9748, 7321, 7391**	14	Hospital
	6	Reserved	15	
	7	Reserved	16	
	8	Reserved	17	
	9	Reserved	18	
3	1	Reserved	19	
	2	**9919,9761**	20	Federal penitentiary, state prison, or prison farm
	3	Reserved	21	
	4	Reserved	22	
	5	Reserved	23	
	6	**7377,7386,7372, 9791, 9771**	24	Educational institution, including academy, school, college, and university
	7	**9901, 9731, 9906, 9739, 9916, 9753, 7339**	25	Religious institution, including church, synagogue, seminary, temple, and mosque
	8	Reserved	26	
	9	Reserved	27	
4	1	**7380, 9762**	28	Train station
	2	**7384**	29	Bus terminal

	3	**7352**	30	Marine terminal
	4	Reserved	31	
	5	Reserved	32	
	6	Reserved	33	
	7	**7383,9732**	34	(TANA/GDT FCC) Airport property Boundary **NEW
	8	Reserved	35	
	9	Reserved	36	
5	1	**9790**	37	Shopping center or major retail center
	2	**9715,9383**	38	Industrial building or industrial park
	3	Reserved	39	
	4	**9902, 9733**	40	Amusement center
	5	**9745,7367**	41	Government center
	6	Reserved	42	
	7	**7474, 9768**	43	Stadium (TANA) **NEW
	8	Reserved	44	
	9	Reserved	45	
6	1	Reserved	46	
	2	**9911,9744**	47	Golf course
	3	**9030, 9788, 9915, 9752, 9918**	48	Cemetery
	4	Reserved	49	
	5	Reserved	50	
	6	**7120,9387**	51	State or local park or forest
	7	Reserved	52	
	8	Reserved	53	
	9	Reserved	54	
7	1	Reserved	55	
	2	Reserved	56	
	3	Reserved	57	
	4	Reserved	58	
	5	Reserved	59	
	6	Reserved	60	
	7	Reserved	61	
	8	Reserved	62	
	9	**4770**	63	Water Feature
8	1	Reserved	64	
	2	Reserved	65	
	3	Reserved	66	
	4	Reserved	67	
	5	Reserved	68	

	6	Reserved	69	
	7	Reserved	70	
	8	Reserved	71	
	9	Reserved	72	
9	1	Reserved	73	
	2	Reserved	74	
	3	Reserved	75	
	4	Reserved	76	
	5	Reserved	77	
	6	Reserved	78	
	7	Reserved	79	
	8	Reserved	80	
	9	Reserved	81	

(III) Location of Glyphs for the GNIS Landmarks database

Col	Row	#	Type
1	1	1	Area
	2	2	Bar
	3	3	Basin
	4	4	Bay
	5	5	Beach
	6	6	Bench
	7	7	Bend
	8	8	Bldg
	9	9	Bridge
2	1	10	Canal
	2	11	Cape
	3	12	Cemetery
	4	13	Channel
	5	14	Church
	6	15	Cliff
	7	16	Dam
	8	17	Falls
	9	18	Flat
3	1	19	Forest
	2	20	Gap
	3	21	Gut
	4	22	Harbor
	5	23	Hospital
	6	24	Island
	7	25	Lake
	8	26	Levee
	9	27	Mil
4	1	28	Milh
	2	29	Mine
	3	30	Oilfield
	4	31	Pillar
	5	32	Plain
	6	33	Range
	7	34	Rapids

	8	35	Reserve
	9	36	Resv
5	1	37	Ridge
	2	38	School
	3	39	Sea
	4	40	Spring
	5	41	Stream
	6	42	Summit
	7	43	Swamp
	8	44	Tower
	9	45	Trail
6	1	46	Tunnel
	2	47	Valley
	3	48	Well
	4	49	*Not Currently Used*
	5	50	*Not Currently Used*
	6	51	*Not Currently Used*
	7	52	*Not Currently Used*
	8	53	*Not Currently Used*
	9	54	*Not Currently Used*
7	1	55	*Not Currently Used*
	2	56	*Not Currently Used*
	3	57	*Not Currently Used*
	4	58	*Not Currently Used*
	5	59	*Not Currently Used*
	6	60	*Not Currently Used*
	7	61	*Not Currently Used*
	8	62	*Not Currently Used*
	9	63	*Not Currently Used*
8	1	64	*Not Currently Used*
	2	65	*Not Currently Used*
	3	66	*Not Currently Used*
	4	67	*Not Currently Used*
	5	68	*Not Currently Used*
	6	69	*Not Currently Used*
	7	70	*Not Currently Used*
	8	71	*Not Currently Used*
	9	72	*Not Currently Used*

9	1	73	*Not Currently Used*
	2	74	*Not Currently Used*
	3	75	*Not Currently Used*
	4	76	*Not Currently Used*
	5	77	*Not Currently Used*
	6	78	*Not Currently Used*
	7	79	*Not Currently Used*
	8	80	*Not Currently Used*
	9	81	*Not Currently Used*

E. Changes to Results Returned by Certain Functions

FindStateAtPoint(X,Y:Double):String — Function

Returns the full name of the state, followed by the two-letter state name abbreviation, followed by the state FIPS code at the specified Lon/Lat coordinates (separated by a tab character). If no State Information is available, then the string "None Found" is returned.

VB.Net Example

```
Private Sub Button296_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button296.Click
        Dim X, Y As Double
        Dim s As String

        s = AxMapPro1.FindCountyAtPoint(X, Y)
        If s <> "" Then
            ListBox1.Items.Add("County = " + s)
        Else
            ListBox1.Items.Add("NO County Retrieved at location: " + _
            X.ToString("N6") + ", " + X.ToString("N6"))
        End If
End Sub
```

FindCountyAtPoint(X,Y:Double):String — Function

Returns the County name followed by the County FIPS code at the specified Lon/Lat location.

VB.Net Example

```
Private Sub Button296_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button296.Click
        Dim X, Y As Double
        Dim s As String
        s = AxMapPro1.FindCountyAtPoint(X, Y)
        If s <> "" Then
            ListBox1.Items.Add("County Name and FIPS Code = " + s)
        Else
            ListBox1.Items.Add("NO County Retrieved at location: " + _
            X.ToString("N6") + ", " + X.ToString("N6"))
        End If

End Sub
```

F. Changes/Enhancements to Other Functions or New Functions Added

FindClosestCityFirst(X,Y,Rad,Pop,Num,Opt):string — Function

A new Option (Opt=2) was added to this function, so the description for "Opt", the last parameter specified in the function call, is now as follows:

Opt: Option specifying the type of search,
-1 Find closest place with population less than the one specified
1 Find closest place with population higher than the one specified
2 Find closest place based on layer visibility at current scale
0 Find closest place regardless of population

Every other aspect of the function behavior remains the same.

User.ZoomExtents() — Function

Zooms in or out, as needed, so that all the UserObjects currently defined in the 0-th (first) user Layer are visible in the viewport. Note that there is also a UserMgr.ZoomExtents() function off each of the user Layers. See example below.

C# Example

```
private void button15_Click(object sender, EventArgs e)
        {
           //- User Items Layers
           //--------------------------
           MapPro80.IUserObj mObject1, mObject2, mObject3, mObject4, mObject5;
           MapPro80.IUser   mLayer1, mLayer2, mLayer3;
           int i, j, LayerCount, LayerObjectCount,n;
           Double a;
           //- Create three layers
            mLayer1=axMapPro1.UserMgr.add();
            mLayer2=axMapPro1.UserMgr.add();
            mLayer3=axMapPro1.UserMgr.add();
           //- Set some User Layer visibility parameters
           mLayer1.Name = "My First Layer";
           mLayer1.Upper = 2000;
           mLayer1.Lower = 0.1;
           mLayer1.Visible = true;
           //--
           mLayer2.Name = "My Second Layer";
           mLayer2.Upper = 2000;
           mLayer2.Lower = 0.1;
           mLayer2.Visible = true;
           //--
           mLayer3.Name = "My Third Layer";
           mLayer3.Upper = 2000;
```

```
mLayer3.Lower = 0.1;
mLayer3.Visible = true;
//-----------------------------------------------------------------
// Add a new object to mLayer1, and give it an ID
mObject1 = mLayer1.NewItem(101);
mLayer1.Font.Color = ColorTranslator.ToOle(Color.Red);
// Change some attributes
mLayer1.Font.Color = ColorTranslator.ToOle(Color.Red);
mLayer1.Font.angle = 25;
mLayer1.Font.Style = 1; // 1- Bold, 2 - Italic, 3 - Both
mLayer1.Font.Mode = 2;  // 1- Transparent, 2 - Opaque
//Set some object Attributes
mObject1.x = -76;
mObject1.y = 34;
mObject1.LoadImage("c:\\develop\\undertow\\mappro80\\rv1.bmp");
mObject1.Caption = "(1) Bold,25d All other default";
//-----------------------------------------------------------------
// Add a new object to mLayer2 and give it an ID
// But first change some attributes
mLayer2.Font.Color = ColorTranslator.ToOle(Color.Yellow);
mLayer2.Font.angle = -20;
mLayer2.Font.BackColor = ColorTranslator.ToOle(Color.Green);
mLayer2.Font.Style = 3;  // 1- Bold, 2 - Italic, 3 - Both
mLayer2.Font.Mode = 2;  //1- Transparent, 2 - Opaque
mLayer2.Font.Name = "Times New Roman";
mLayer2.Font.Align = 7; // Center Aligned
mLayer2.Font.Size = 16;
// Set some object Attributes
mObject2 = mLayer2.NewItem(201);
mObject2.x = -80;
mObject2.y = 40;
a = mObject2.Font.angle;
mObject2.LoadImage("c:\\develop\\undertow\\mappro80\\rv2.bmp");
mObject2.Caption = "(2) B/I,-20d,16p,ctr";
//-----------------------------------------------------------------
// Add two objects to mLayer3 and give them IDs - First Set some attributes
mLayer3.Font.Color = ColorTranslator.ToOle(Color.Blue);
// -- If a TrueType font name is not specified, the internal
//    font used by default gets pixelated very quickly
mLayer3.Font.Name = "Arial";
mLayer3.Font.Size = 18;
mLayer3.Font.Style = 1;
mLayer3.Font.Mode = 1;
// Set Layer Text angle to -15 deg
mLayer3.Font.angle = -15;
mObject3 = mLayer3.NewItem(301);
//Setsome object Attributes
mObject3.x = -100;
mObject3.y = 46;
mObject3.LoadImage("c:\\develop\\undertow\\mappro80\\rv3.bmp");
mObject3.Caption = "(3),18p,Tr,0d,B";
//--
mObject4 = mLayer3.NewItem(302);
mObject4.x = -105;
mObject4.y = 41;
mObject4.LoadImage("c:\\develop\\undertow\\mappro80\\rv3.bmp");
// Change the text angle for User object #4, sreciically
mObject4.Font.angle = 15;
```

```
        // Change Font size and type for user object #4, only
        mObject4.Font.Size = 12;
        mObject4.Font.Style = 3;
        mObject4.Caption = "(4),12p,+15d,B/I";
        // Set the transparency for the 4th last user object for the red color
        mObject4.TransColor = ColorTranslator.ToOle(Color.Red);
        mObject4.Transparent = true;
        //--
        // Set a fifth user object, on layer 3 to make sure defaults
        // settings are used.  Should be the same as #3, w/title offset
        mObject5 = mLayer3.NewItem(303);
        mObject5.x = -110;
        mObject5.y = 36;
        mObject5.LoadImage("c:\\develop\\undertow\\mappro80\\rv3.bmp");
        mObject5.Caption = "(5), 18p,Tr,-15d,B off 5/5";
        // Offset in pixels
        mObject5.xofs = 0;
        mObject5.yofs = mObject5.ImageHeight + 5;
        axMapPro1.Refresh();

       //---  List Layers and Objects ---
        listBox1.Items.Add("# of Layers: " + axMapPro1.UserMgr.Count.ToString());
        LayerCount = axMapPro1.UserMgr.Count;
        // The 0-th layer is the old SetItem which is reserved
        for (int ii=0; ii<=LayerCount - 1;ii++)
           {
               listBox1.Items.Add(" ");
            LayerObjectCount = axMapPro1.UserMgr.Layer[ii].Count;
            listBox1.Items.Add("Layer [ " + ii.ToString()+" ]:
                 "+axMapPro1.UserMgr.Layer[ii].Name+"
                 ("+LayerObjectCount.ToString()+" Objects)");
              // The first Object should be zero (0-based array)
              for (int jj=0;jj<=LayerObjectCount-1;jj++)
                {
                    listBox1.Items.Add("  ID [ " + jj.ToString() + " ]: " +
                    axMapPro1.UserMgr.Layer[ii].Items[jj].id.ToString() +
                    ",  at Coords: " +
                    axMapPro1.UserMgr.Layer[ii].Items[jj].x.ToString() + ", " +
                    axMapPro1.UserMgr.Layer[ii].Items[jj].y.ToString());
                }
           }

        // The 0-th layer is the old SetItems layer, so it's managed
        // by User.ZoomExtents()
        for (int k = 1; k < axMapPro1.UserMgr.Count; k++)
        {
            axMapPro1.UserMgr.Layer[k].ZoomExtents();
            listBox4.Items.Add("Going to extents of Layer # " + k.ToString());
            listBox4.Items.Add("Viewport Center at:" + sCenter);
            Application.DoEvents();
            // Delay to see zoom action
            axMapPro1.Cad.Clear();
            Thread.Sleep(5000);
        }

        // This will Zoom ALL the user Layers extents
        axMapPro1.UserMgr.ZoomLayers();
    }
```

UserMgr.ZoomLayers() Function

Zooms in or out, as needed, so that all the UserObjects currently defined in all user Layers are visible in the viewport. Note that there is also a .ZoomExtents() function off each of the user Layers. See code example for User.ZoomExtents();

CAD.FindItemLonLat(x,y:Double):ICadObj Function

Returns the CAD object at the specified Lon,Lat coordinates. This is determined by a point-in-polygon calculation, if the object is a polygon, or a proximity test otherwise.

C# Example

```
private void axMapPro1_MouseUpEvent(object sender,
AxMapPro80.IMapProEvents_MouseUpEvent e)
     {
         MapPro80.ICadObj myCAD;
         if (PiPMode == true)
         {
             myCAD = axMapPro1.Cad.FindItemLonLat(axMapPro1.Xcord, axMapPro1.Ycord);
             string s = myCAD.Caption;
             if (s != null)
             {
                 listBox1.Items.Add("CAD Item Located:" + s +", Type: "+
                 myCAD.ObjectType.ToString());
             }
             else listBox1.Items.Add("CAD Item Located, No Caption, CAD Item Type: " +
             myCAD.ObjectType.ToString());
         }

     }
```

CAD.FindItemScreen(x,y:Integer):ICadObj Function

Returns the CAD object at the specified screen (pixel) coordinates. This is determined by a point-in-polygon calculation, if the object is a polygon, or a proximity test otherwise.

C# Example

```
private void button19_Click(object sender, EventArgs e)
     {
         MapPro80.ICadObj myCAD;
         int CenterX=0;
         int CenterY=0;
         axMapPro1.LL2Screen(axMapPro1.LonCenter, axMapPro1.LatCenter,
          ref CenterX, ref CenterY);
         myCAD = axMapPro1.Cad.FindItemScreen(CenterX, CenterY);
```

```
            string s = myCAD.Caption;
              if (s != null)
                {
                    listBox1.Items.Add("CAD Item Located:" + s + ", Type: " +
                    myCAD.ObjectType.ToString());
                }
                else listBox1.Items.Add("CAD Item Located, No Caption, CAD Item Type: " +
                myCAD.ObjectType.ToString());
        }
```

CAD.FindItemClose() Function

Clears the list object created by the CAD.FindItemFirst() function and returns any resources it was using back to the system. Also see *CAD.FindItemFirst().*

CAD.FindItemFirst(x,y:Double, Option:Integer):ICadObj Function

Locates all CAD items at the specified Coordinates, places them in a list object and returns the first element of the list object, as a CAD object interface. The results are determined using a Point-in-polygon test (if the CAD object is a polygon), or a proximity test otherwise.

X,Y: Coordinates to use for the Search
Option: 0 = Use Lon/Lat coordinates, 1 = Use screen coordinates (X,Y will be truncated)

C# Example

```
   private void axMapPro1_MouseUpEvent(object sender,
AxMapPro80.IMapProEvents_MouseUpEvent e)
        {
            MapPro80.ICadObj myCAD;
            if (PiPMode == true)
            {
                listBox1.Items.Add("* Start Searching for CAD Objects...");
                int i = 1;
                myCAD = axMapPro1.Cad.FindItemFirst(axMapPro1.Xcord, axMapPro1.Ycord, 0);
                while (myCAD != null)
                {
                    string s = myCAD.Caption;
                    if (s != null)
                      {
                         listBox1.Items.Add("# "+i.ToString()+", CAD Item Located:" + s
                         +", Type: "+ myCAD.ObjectType.ToString());
                      }
                    else listBox1.Items.Add("# " + i.ToString() + ", CAD Item Located, No
                    Caption, CAD Item Type: " + myCAD.ObjectType.ToString());
```

```
                myCAD = axMapPro1.Cad.FindItemNext();
                i++;
            }
            axMapPro1.Cad.FindItemClose();
        }
    }
```

CAD.FindItemNext():ICadObj — Function

Returns the next CAD object from the list object created by the CAD.FindItemFirst(). Returns null if the last object has already been returned.

.AutoQueryStr:String — Property

Returns the hint string that appears when the cursor rests on a place, street, etc. is AutoQuery is set to True.

C# Example

```
private void axMapPro1_MouseMoveEvent(object sender,
            AxMapPro80.IMapProEvents_MouseMoveEvent e)
    {
        // Display the QutoQuery hint string in a list box as the mouse moves around
        string s = axMapPro1.AutoQueryStr;
        if (s != null) { listBox1.Items.Add(s); }
    }
```

Modifications regarding Expired Evaluation Licenses

A modification was made to let the developer now when an evaluation copy has exceeded its evaluation period, in non-visual instances of the control. While in visual implementations, when the Map would redraw, the expired state of the evaluation license was detected and an appropriate dialog was displayed, in non-visual instances, that would not take place and the OCX would simply not allow access to Street Level data, or return null strings as the result of search operations. At any point , the developer could use the OCX function ***.DaysLeft*** to determine if the evaluation period was expired, but that was not immediately evident from the behavior of the OCX.

An appropriate expiration message is now displayed when non-visual instances are used to render maps (DirectDraw, DirectView) and the results of any programmatically initiated searches now return each field with the string “Expired!”.

FindStreetFirst(Block, Street1, Street2, City,State:String, X,Y,Radius:Double):String

Initiates a substring search for the specified street and optional cross street, optional address, center of search (LonLlat) and search radius (miles) and optional state specifier. The complete list of parameters passed to the function are:

Block	-	Street #
Street1	-	Street name
Street2	-	Optional cross street name
City	-	City Name
State	-	Two letter state name abbreviation
x,y	-	Center of search (Lon/Lat) **
Radius	-	Search radius (miles) **

**** If these values are zero, then a City/State search is done first, internally, and the results are used as the center of search (similar to what is done when the built-in search dialog is used).**

The search performed is a sub-string search, so if “Pri” is specified as the street name, for example, then streets with hames like **Pri**nceton, **Pri**me, Entr**pri**se, etc. will match the criterion.

Note: Each street segment that is found matching the criteria, is placed in a list Object that is created and can be used by the user to retrieve all matches found. The first match in the list object is returned by the function as a tab delimited string which contains the following information:

Block#|Street|Place|PostalPlace|MCD|County|State|ZipCode|Country|Lon|Lat

See the ParseTabField() for more details

If a cross street search is specified, a single match should be returned (if found) and is returned by the function call. The rest of the matches of the search are retrieved by calling the FindStreetNext() function of the OCX.

VB.Net Example

```
Private Sub Button164_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button164.Click
        '  ** Find StreetFirst/Next - Looking for a single Address **
        Dim mBlock, mStr1, mStr2, mCity, mState, mResult As String
        Dim mLon, mLat, mRadius, x, y As Double
        Dim i As Integer
        ' Set some initial parameters
        mBlock = "245"
```

```
        mStr1 = "Summer Street"
        mStr2 = ""
        mCity = "Boston"
        mState = "MA"
        mLon = -71.1
        mLat = 42.4
        mRadius = 10
        i = 0
        ' Zoom in and Mark the center
        AxMapPro1.GotoPoint(mLon, mLat)
        AxMapPro1.Miles = 2
        AxMapPro1.Cad.Marker(mLon, mLat, AxMapPro1.Cad.GetMarker(14))
        AxMapPro1.Cad.Text(mLon, mLat, "Search Center")
        ' Set the color for the next CAD text Objects
        AxMapPro1.Cad.Font.Color = RGB(0, 0, 255)
        AxMapPro1.Refresh()
        mResult = AxMapPro1.FindStreetFirst(mBlock, mStr1, mStr2, mCity, _
        mState, mLon, mLat, mRadius)
        While mResult <> ""
            i = i + 1
            ' Mark each hit returned by the search
            x = Val(AxMapPro1.ParseTabField(mResult, 10))
            y = Val(AxMapPro1.ParseTabField(mResult, 11))
            AxMapPro1.Cad.Text(x, y, "S" + i.ToString)
            ListBox1.Items.Add(x.ToString + ", " + y.ToString + "  ***  " + _
            mResult)
            mResult = AxMapPro1.FindStreetNext
        End While
        AxMapPro1.FindStreetClose()
        AxMapPro1.Refresh()
End Sub


Private Sub Button164_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button164.Click
        '  ** Find StreetFirst/Next - Looking for Intersection **
        Dim mBlock, mStr1, mStr2, mCity, mState, mResult As String
        Dim mLon, mLat, mRadius, x, y As Double
        Dim i As Integer
        i = 0
        mResult = AxMapPro1.FindStreetFirst("", "Summer St", _
        "S Boston Bypass","Boston", "MA", -71.1, 42.4, 10)
        While mResult <> ""
            i = i + 1
            ' Mark each hit returned by the search
            x = Val(AxMapPro1.ParseTabField(mResult, 10))
            y = Val(AxMapPro1.ParseTabField(mResult, 11))
            ListBox1.Items.Add(mResult)
            mResult = AxMapPro1.FindStreetNext
        End While
        AxMapPro1.FindStreetClose()
End Sub
```

GeoFindFirst(s:String):String

The behavior of the GeoFindFirst function was incorrectly documents in the original User Manual. It does not take a search radius, after the search string, using the piping character. However, the piping character *is* used to specify a cross street search. The specification is actually,

GeoFindFirst(Street# Street Name*[|CrossStreetName]*, City State,..)

Where the [|CrossStreetName is optional.

Example:

```
s = Map.GeoFindFirst("Main Street, North Andover, MA");  will return
```
|MA SH 28\N MAIN ST |ANDOVER||ANDOVER|ESSEX:25009|MA|01810|USA|-71.152816|42.678966

```
Whereas, s = Map.GeoFindFirst("Main Street|Sutton St, North Andover, MA");  will return
```

*|MAIN ST AND SUTTON ST |NORTH ANDOVER||NORTH ANDOVER|ESSEX:25009|MA|01845|USA|-71.133333|42.706460

PhoneRegInfo:String **Property**

This will allow you to modify the text in the Register Over the Phone dialog. This is useful when you would like a different phone number so users can call your company directly, instead of contacting UnderTow Software. The first character of the string determines the Registration options in the dialog.

If the first character is:

- 1, The option to **Register Over the Internet** is disabled
- 2 , The option to **Register by Phone** is disabled
- 4, The option to **Register Later** is disabled

The above options are based on the bit position, so they may also be uniquely combined. So, if the first character of the string is 3, the Option to register over the Internet (value 1, bit position one) and the Option to register by Phone (value 2, bit position two) are disabled. The string specified by PhoneRegInfo can also contain the information that is displayed as part of Step #2, in the Register by Phone part of the dialog, using the piping character "|" followed by the desired string (see the two sample code segments below). The portion of the string preceding the piping character is also displayed on the surface of the map if the 15-day evaluation period has expired.

C# Examples

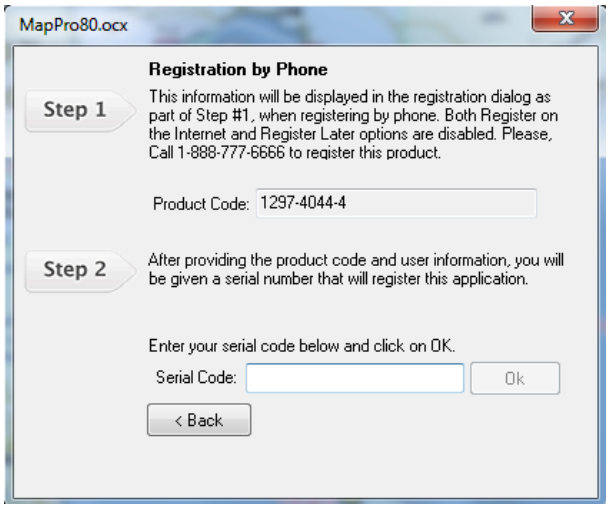
```
private void button23_Click(object sender, EventArgs e)
        {

          // Set the string to be displayed and disable the Internet and Later Options
          axMapPro1.PhoneRegInfo = "5This information will be displayed in the "
                 + "registration dialog as part of Step #1, when registering by "
                 + "phone. Both Register on the Internet and Register Later options are "
```

```
            + "disabled. Please, Call 1-888-777-6666 to register this product.";
        // Open the Registration dialog to test the strings
        axMapPro1.ExecRegister(-1);

    }
```

And this is what the Register by Phone dialog should look like,




```
private void button23_Click(object sender, EventArgs e)
    {
      // Set the string to be displayed and disable the Internet and Later Options
      axMapPro1.PhoneRegInfo = "1This information will be displayed in the "
       + "registration dialog as part of Step #1, when registering by "
       + "phone. Only the Internet registration option is disabled"
       + "Please, Call 1-888-777-6666 to register this product.|And this would "
       + "be the info displayed in Step #2, e.g., you will be provided a “
       + "Serial Registration Code...";
          // Open the Registration dialog to test the strings
          axMapPro1.ExecRegister(-1);
       }
```

And this is what the Register by Phone dialog should look like,

MapProperties.RoadLabelSpacing:Integer — Property

The screen is divided into grids and one unique label is allowed per grid. Roads are typically ordered from left-to-right (Bottom-to-top) and the first segment encountered, if unique, is the segment that gets labeled. Spacing is a function of the Screen Dpi setting which is graphic card dependant. (on standard monitors about 96dpi). So, a factor of 2, means that grids are about 2 inches square. For example , if the screen was 6 inches square, there would be a total of 9 grids (3x3). If a street appeared in every single grid, one would get a maximum of 9 labeling opportunities (of course label collision detection might reduce this number of street labels placed).

The default value is 2.

MapProperties.GrabPanMode:Boolean — Property

A new zoom behavior was introduced to MapPro80. When this property is TRUE, if the user left-clicks on the map and holds down the mouse button for N milliseconds, then the cursor changes to a small hand and the user may pan the map in the desired direction. Releasing the mouse button, repaints the map at the new location. If the user clicks and moves the mouse within the N millisecond delay, then the zoom rectangle appears that allows the user to zoom into a specified area, as before. The delay time is 300 milliseconds and may be customized by the control's property, MapProperties.GrabPanDelay.

The default value of this property is FALSE.

MapProperties.GrabPanDelay:Integer — Property

Sets the delay time in milliseconds before switching from the standard Zoom window mode to the grab-and-pan mode, when the user clicks the left mouse button on the map (also see the property MapProprties.GrabPanMode). If a very small value is specified, then the control switches into the grab-and-pan mode almost immediately. There is however some finite delay in the CPU's processing of events, so even if this property is set to one (1) millisecond, it's possible that the user can very quickly click and drag the mouse pointer and still zoom the map using the default, zoom rectangle mode. Setting the value to zero has the same effect as disabling the mode.

The default value of this property is set to 300 milliseconds.

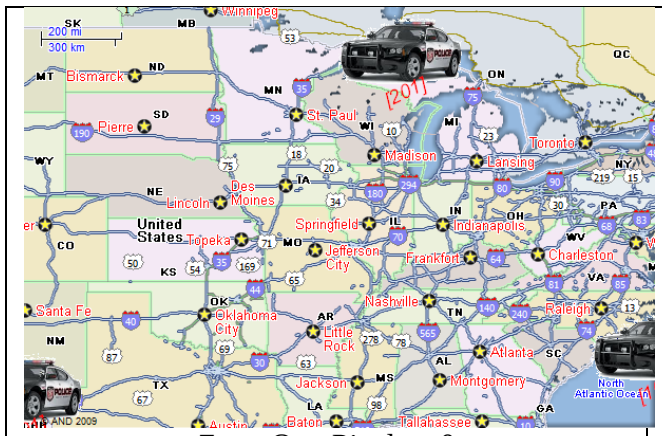
MapProperties.MapRotation:Double — Property

Controls the map rotation (clockwise) in degrees. Note that the map rotation may also be set using the MapPro80.RotateMap() function, off the OCX main interface, whereas this property allows the user to both set and query the rotation angle.

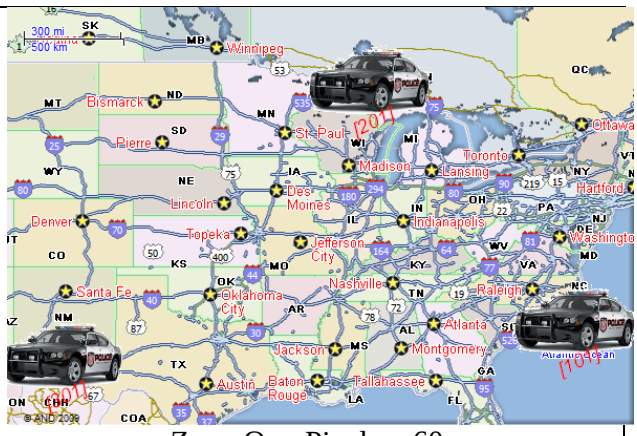
MapProperties.ZoomOverPixels:Integer — Property

Specifies the number of pixels that amy ZoomExtents() operation should overshoot the calculated extents by, so that the user can avoid missing portions of information that might be clipped by the viewport. The number of specified pixels is added to both sides of the controlling (larger) dimension of the viewport and a proportional number of pixels is added to both sides of the second dimension, before the map is redrawn.

Here is an example of a map following a MapPro80.UserMgr.Layer[1].ZoomExtents() with ZoomOverPixels = 0 (default) and ZoomOverPixels = 60, to avoid clipping the user objects in that layer.



ZoomOverPixels = 0



ZoomOverPixels = 60

.UserMgr.ExcludeNonvisFromExtents:boolean — Property

When set to True, it excludes any non-visible user objects (whose visible property is set to false) from the calculations trying to determine the extents of ANY User object layer. The default setting of this property is False.

.UserMgr.Layer[n].ExcludeNonvisFromExtents:boolean — Property

When set to True, it excludes any non-visible user objects (whose visible property is set to false) from the calculations trying to determine the extents of the specific User object layer, n. Note that a setting of the global property, **.UserMgr.ExcludeNonvisFromExtents = true**, overrides the setting of this individual layer property, so if the developer wants to control the behavior of each layer individually, then the global property should be set to false. The default setting of this property is False.