



MapBasic

Version 9.5

SUPPLEMENT

Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor or its representatives. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, without the written permission of Pitney Bowes Software Inc., One Global View, Troy, New York 12180-8399.

© 2008 Pitney Bowes Software Inc. All rights reserved. MapInfo, the Pitney Bowes Software logo, and are trademarks of Pitney Bowes Software Inc. and/or its affiliates.

Corporate Headquarters:

Voice: (518) 285-6000

Fax: (518) 285-6070

Sales Info Hotline: (800) 327-8627

Government Sales Hotline: (800) 619-2333

Technical Support Hotline: (518) 285-7283

Technical Support Fax: (518) 285-6080

www.mapinfo.com

Contact information for all corporate offices is located at: <http://www.mapinfo.com/contactus>.

Adobe Acrobat® is a registered trademark of Adobe Systems Incorporated in the United States.

Products named herein may be trademarks of their respective manufacturers and are hereby recognized. Trademarked names are used editorially, to the benefit of the trademark owner, with no intent to infringe on the trademark.

May 2008

New and Enhanced MapBasic Statements and Functions

These are several new statements and functions added to the MapBasic API that assist you in working with grids and in developing with .Net to create your integrated mapping and other third-party solutions.

We have also added a new chapter called *Working with .Net* in the MapBasic *User Guide* to help you get started in integrating MapInfo Professional with your application. There are many helpful examples and code snippets that provide examples and demonstrate the functionality we have added.

Sections in this Chapter:

- ♦ **New Functions and Statements**2
- ♦ **Enhanced Functions and Statements**10

New Functions and Statements

We have added the following statement and functions in this version of MapBasic:

- **ControlPointInfo(.) function**
- **Declare Method statement**
- **GetGridCellValue() function**
- **GridTableInfo() function**
- **IsGridCellNull() function**
- **RasterTableInfo() function**

ControlPointInfo(.) function

Purpose

Returns raster and geographic control point coordinates for an image table. The geographic coordinates will be in the current MapBasic coordinate system.

Syntax

```
ControlPointInfo( table_id, attribute, controlpoint_num )
```

table_id is a string representing a table name, a positive integer table number, or 0 (zero). The table must be a raster, grid or WMS table.

attribute is an integer code indicating which aspect of the control point to return.

controlpoint_num is the integer number of which control point to return. Control point numbers start at 1. The maximum control point number can be found by calling

```
RasterTableInfo(table_id, RASTER_TAB_INFO_NUM_CONTROL_POINTS)
```

Return Value

The X or Y raster coordinate is returned as an Integer. The X or Y geographic coordinate is returned as a Float. The return type depends upon the attribute flag, for the control point specified by *controlpoint_num*.

The attribute parameter can be any value from the table below. Codes in the left column (for example, RASTER_CONTROL_POINT_X) are defined in MAPBASIC.DEF.

attribute code	ControlPointInfo() returns:
RASTER_CONTROL_POINT_X	Integer result, representing the X coordinate of the control point number specified by <i>controlpoint_num</i>
RASTER_CONTROL_POINT_Y	Integer result, representing the Y coordinate of the control point number specified by <i>controlpoint_num</i>

attribute code	ControlPointInfo() returns:
GEO_CONTROL_POINT_X	Float result, representing the X coordinate of the control point number specified by controlpoint_num
GEO_CONTROL_POINT_Y	Float result, representing the Y coordinate of the control point number specified by controlpoint_num

Declare Method statement

Purpose

Defines the name and argument list of a method/function in a .Net assembly, so that a MapBasic application can call the function.

Note We have added a chapter on integrating MapInfo Professional into your .Net programming environment called *Working with .Net* in the *MapBasic User Guide*.

Restrictions

This statement may not be issued from the MapBasic window.

Syntax

```
Declare Method fname Class "class_name" Lib "assembly_name"
    [ Alias function_alias ]
    ( [ [ ByVal ] parameter As var_type ]
      [, [ ByVal ] parameter As var_type... ] ) [ As return_type ]
```

fname is the name by which a function will be called; if the optional Alias clause is omitted, *fname* must be the same as the actual .Net method/function name. This option can not be longer than 31 characters.

class_name is the name of the .Net class that provides the function to be called, including the class's namespace (such as System.Windows.Forms.MessageBox)

assembly_name is the name of a .Net assembly file, such as filename.dll. If the assembly is to be loaded from the GAC, *assembly_name* must be a fully qualified assembly name.

function_alias is the original name of the .Net method/function (the name as defined in the .Net assembly). Note: Include the Alias clause only when you want to call the method by a name other than its original name.

parameter is the name of a parameter to the function.

var_type is a MapBasic data type, such as Integer

return_type is a standard MapBasic scalar variable type, such as Integer. If the As clause is omitted, the MapBasic program can call the method as a Sub (using the **Call statement**).

Description

The Declare Method statement allows a MapBasic program to call a function from a .Net assembly. The .Net assembly can be created using various languages, such as C# or VB.Net. For details on calling .Net from MapBasic, see the MapBasic *User Guide*.

MapBasic programs can only call .Net methods or functions that are declared as static. (VB.NET refers to such functions as “shared functions,” while C# refers to them as “static methods.”)

At run time, if the assembly_name specifies a fully-qualified assembly name, and if the assembly is registered in the Global Assembly Cache (GAC), MapInfo Professional will load the assembly from the GAC. Otherwise, the assembly will be loaded from the same directory as the .MBX file (in which case, assembly_name should be a filename such as "filename.dll"). Thus, you can have your assembly registered in the GAC, but you are not required to do so.

Examples

Here is a simple example of a C# class that provides a static method:

```
namespace MyProduct
{
    class MyWrapper
    {
        public static int ShowMessage(string s)
        {
            System.Windows.Forms.MessageBox.Show(s);
            return 0;
        }
    }
}
```

In VB.Net, the class definition might look like this.

```
Namespace MyProduct

    Public Class MyWrapper

        Public Shared Function ShowMessage(ByVal s As String) As Integer
            System.Windows.Forms.MessageBox.Show(s)
            Return 0
        End Function

    End Class

End Namespace
```

A MapBasic program could call the method with this syntax:

```
Declare Method ShowMessage
  Class "MyProduct.MyWrapper"
  Lib "MyAssembly.DLL" (ByVal str As String) As Integer

  . . .

Dim retval As Integer
retval = ShowMessage("Here I am")
```

The following example demonstrates how to declare methods in assemblies that are registered in the GAC. Note that when an assembly is loaded from the GAC, the Lib clause must specify a fully-qualified assembly name. Various utilities exist that can help you to identify an assembly's fully-qualified name, including the `gacutil` utility provided by Microsoft as part of Visual Studio.

```
' Declare a method from the System.Windows.Forms.dll assembly:
Declare Method Show
  Class "System.Windows.Forms.MessageBox"
  Lib "System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
  (ByVal str As String, ByVal caption As String)

' Declare a method from the mscorlib.dll assembly:
Declare Method Move
  Class "System.IO.File"
  Lib "mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
  (ByVal sourceFileName As String, ByVal destFileName As String)

' Display a .Net MessageBox dialog box with both a message and a caption:
Call Show("Table update is complete.", "Tool name")

' Call the .Net Move method to move a file
Call Move("C:\work\pending\entries.txt", "C:\work\finished\entries.txt")
```

GetGridCellValue() function

Purpose

Determines the value of a grid cell if the cell is non-null.

Syntax

GetGridCellValue(*table_id*, *x_pixel*, *y_pixel*)

table_id is a string representing a table name, a positive integer table number, or 0 (zero). The table must be a grid table.

x_pixel is the integer number of the X coordinate of the grid cell. Pixel numbers start at 0. The maximum pixel value is the (*pixel_width*-1), determined by calling:

GridTableInfo() function

```
RasterTableInfo(...RASTER_TAB_INFO_WIDTH)
```

y_pixel is the integer number of the Y coordinate of the grid cell. Pixel numbers start at 0. The maximum pixel value is the (pixel_height-1), determined by calling:

```
RasterTableInfo(...RASTER_TAB_INFO_HEIGHT)
```

Return Value

A Float is returned, representing the value of a specified cell in the table if the cell is non-null. The IsGridCellNull() function should be used before calling this function to determine if the cell is null or if it contains a value.

GridTableInfo() function

Purpose

Returns information about a grid table.

Syntax

```
GridTableInfo( table_id, attribute )
```

table_id is a string representing a table name, a positive integer table number, or 0 (zero). The table must be a grid table.

attribute is an integer code indicating which aspect of the grid table to return.

Return Value

String, SmallInt, Integer or Logical, depending on the attribute parameter specified.

The attribute parameter can be any value from the table below. Codes in the left column (for example, GRID_TAB_INFO_) are defined in MAPBASIC.DEF.

attribute code	GridTableInfo() returns:
GRID_TAB_INFO_MIN_VALUE	Float result, representing the minimum grid cell value in the file
GRID_TAB_INFO_MAX_VALUE	Float result, representing the maximum grid cell value in the file
GRID_TAB_INFO_HAS_HILLSHADE	Logical result, TRUE if the grid file has hillshade/relief shade information. This flag does not depend on whether the file is displayed using the hillshading.

IsGridCellNull() function

Purpose

Returns a Logical. Returns TRUE if the cell value location (x, y) is valid for the table, and is a null cell (a cell that does not have an assigned value). Returns FALSE if the cell contains a value that is non-null. The GetCellValue() function can be used to retrieve the value.

Syntax

```
IsGridCellNull( table_id, x_pixel, y_pixel )
```

table_id is a string representing a table name, a positive integer table number, or 0 (zero). The table must be a grid table.

x_pixel is the integer pixel number of the X coordinate of the grid cell. Pixel numbers start at 0. The maximum pixel value is the (pixel_width-1), determined by calling:

```
RasterTableInfo(...RASTER_TAB_INFO_WIDTH)
```

y_pixel is the integer pixel number of the Y coordinate of the grid cell. Pixel numbers start at 0. The maximum pixel value is the (pixel_height-1), determined by calling:

```
RasterTableInfo(...RASTER_TAB_INFO_HEIGHT)
```

Return Value

A Logical is returned, representing whether the specified cell in the table is null, or non-null. If the grid cell is non-null (IsGridCellNull() returns FALSE), then the GetGridCellValue() function can be called to retrieve the value for that grid pixel.

RasterTableInfo() function

Purpose

Returns information about a Raster or Grid Table. (WMS or Seamless Raster tables not supported).

Syntax

```
RasterTableInfo( table_id, attribute )
```

table_id is a string representing a table name, a positive integer table number, or 0 (zero). The table must be a raster, grid, or WMS table.

attribute is an integer code indicating which aspect of the raster table to return.

Return Value

String, SmallInt, Integer or Logical, depending on the attribute parameter specified.

The attribute parameter can be any value from the table below. Codes in the left column (for example, RASTER_TAB_INFO_IMAGE_NAME) are defined in MAPBASIC.DEF.

RasterTableInfo() function

attribute code	RasterTableInfo() returns
RASTER_TAB_INFO_IMAGE_NAME	String result, representing the image file name associated with this raster table.
RASTER_TAB_INFO_WIDTH	Integer result, representing the width of the image, in pixels
RASTER_TAB_INFO_HEIGHT	Integer result, representing the height of the image, in pixels
RASTER_TAB_INFO_IMAGE_TYPE	SmallInt result, representing the type of image: <ul style="list-style-type: none">• IMAGE_TYPE_RASTER - for raster images• IMAGE_TYPE_GRID - for grid images• IMAGE_TYPE_WMS - for WMS images
RASTER_TAB_INFO_BITS_PER_PIXEL	SmallInt result, representing the number of bits/pixel for the raster data
RASTER_TAB_INFO_IMAGE_CLASS	SmallInt result, representing the image class: <ul style="list-style-type: none">• IMAGE_CLASS_PALETTE (2) - for palette images• IMAGE_CLASS_GREYSCALE (1) - for greyscale images• IMAGE_CLASS_RGB (3)- for RGB images• IMAGE_CLASS_BILEVEL (0) - for 2 color bilevel images
RASTER_TAB_INFO_NUM_CONTROL_POINTS	SmallInt result, representing the number of control points. Use RasterControlPointInfo() and GeoControlPointInfo() to get specific control points.
RASTER_TAB_INFO_BRIGHTNESS	SmallInt result, representing the brightness as a percentage (0-100%)
RASTER_TAB_INFO_CONTRAST	SmallInt result, representing the contrast of the image as a percentage (0-100%)
RASTER_TAB_INFO_GREYSCALE	Logical result, representing if the image display should display as greyscale instead of the default image mode

attribute code	RasterTableInfo() returns
RASTER_TAB_INFO_DISPLAY_TRANSPARENT	Logical result, representing if the image should display with a transparent color. If TRUE, RASTER_TAB_INFO_TRANSPARENT_COLOR represents the color that will be made transparent.
RASTER_TAB_INFO_TRANSPARENT_COLOR	Integer result, represent the color of the transparent pixels, as BGR.
RASTER_TAB_INFO_ALPHA	SmallInt result, representing the alpha factor for the translucency of the image (0-255)

Enhanced Functions and Statements

We have enhanced the following functions and statements in this version of MapBasic:

- **Commit Table statement**
- **Create Text statement**
- **LayerInfo() function**
- **Register Table statement**
- **Save Window statement**
- **Server ConnectionNumber Create Map statement**
- **Server Create Map statement**
- **Set Layout statement**
- **Set Map statement**
- **Set Window statement**
- **TableInfo() function**
- **WindowInfo() function**

Commit Table statement

Purpose

Saves recent edits to disk, or saves a copy of a table. In the past, you were unable to save queries that contained indeterminate types, such as often occurred in ObjectInfo queries. We have added an Interactive parameter to allow you to specify indeterminate types in such a query. If you do not use the interactive parameter, the system uses a default type instead. You can issue this statement from the MapBasic Window in MapInfo Professional.

Syntax

Commit Table table

```
[ As filespec
  [ Type { NATIVE |
          DBF [ Charset char_set ] |
          Access Database database_filespec
          Version version
          Table tablename
          [ Password pwd ] [ Charset char_set ] |
          QUERY |
          ODBC Connection ConnectionNumber Table tablename
          [ ConvertDateTime {ON | OFF | INTERACTIVE}} ] } ]
  [ CoordSys... ]
  [ Version version ] ]
  [ Interactive ]
  [ { Interactive | Automatic commit_keyword } ]
  [ ConvertObjects {ON | OFF | INTERACTIVE}} ]
```

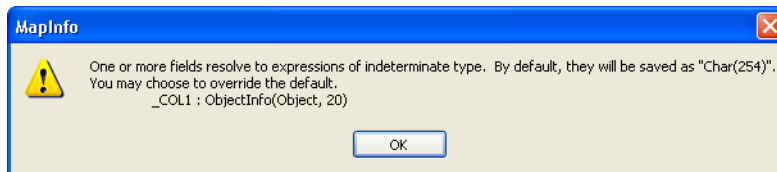
Interactive when invoked for Commit Table As, handles the case when a user is saving a query with one or more columns which are of indeterminate type. Using the Interactive parameter presents the user with a message indicating which column(s) contain the indeterminate type and allows the user to select new types and/or widths for these columns. If the Interactive parameter is not used, the system assigns a Char(254) type to the indeterminate type column(s) by default.

Example

Issue the following query in the SQL Select dialog box and click **OK** or type this query in the MapBasic window:

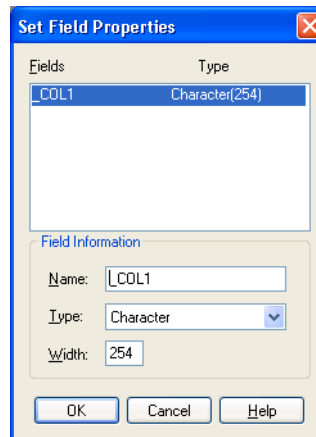
```
Select Highway, objectinfo(obj, 20) from US_HIWAY into Selection
```

When you select **File > Save Copy As**, select the current query, and click the Save As button, the following error message displays:



Typically this dialog box contains a list of all columns that contain indeterminate types. In this query, there is only one.

Click **OK** to display the Set Field Properties dialog box.



Use this dialog box to select the type information for this column. If there is more than one indeterminate type, you can set each of these types one at a time. If there are columns whose type is already defined, you will not be able to edit that information.

Click **OK** to save your query.

Create Text statement

Purpose

Creates a text object, such as a title, for a Map or Layout window. We have added a pen clause to allow you to persist the new label line styles in layouts. Changing layouts in this manner sets the version of the workspace to 9.5. Any MIF file that contains text objects with a Label line and a pen clause will be version 950 as a result. You can issue this statement from the MapBasic Window in MapInfo Professional.

Syntax

Create Text

```
[ Into { Window window_id | Variable var_name } ]
  text_string
  ( x1, y1 ) ( x2, y2 )
  [ Font... ]
  [ Label Line { Simple | Arrow } ( label_x, label_y ) Pen (pen_expr) ]
  [ Spacing { 1.0 | 1.5 | 2.0 } ]
  [ Justify { Left | Center | Right } ]
  [ Angle text_angle ]
```

Pen specifies the pen clause settings of callouts created in the Layout window.

Example

When the user creates a label line in a Layout window, the Create Text Label Line Pen clause is invoked and the workspace version is incremented to 950:

```
!Workspace
!Version 950
!Charset WindowsLatin1
Open Table "Data\Introductory_Data\World\WORLD" As WORLD Interactive
Map From WORLD
  Position (0.0520833,0.0520833) Units "in"
  Width 6.625 Units "in" Height 4.34375 Units "in"
Set Window FrontWindow() ScrollBars Off Autoscroll On
Set Map
  CoordSys Earth Projection 1, 104
  Center (35.204159,-25.3575215)
  Zoom 18063.92971 Units "mi"
  Preserve Zoom Display Zoom
  Distance Units "mi" Area Units "sq mi" XY Units "degree"
Set Map
  Layer 1
  Display Graphic
  Global Pen (1,2,0) Brush (2,16777215,16777215) Symbol (35,0,12)
Line (1,2,0) Font ("Arial",0,9,0)
  Label Line None Position Center Font ("Arial",0,9,0) Pen (1,2,0)
```

LayerInfo() function

Purpose

Returns information about Layer and Label translucency in the MapInfo Professional Maintenance Release. This is not operational for MapInfo Professional 9.5.

Attribute Code	LayerInfo() returns:
LAYER_INFO_LAYER_ALPHA (39)	<p>SmallInt value, representing the alpha factor for the specified layer. These values range from 0=fully transparent to 255=fully opaque.</p> <p>To turn set the translucency or alpha for a layer, use the Set Map Layer statement.</p>
LAYER_INFO_LAYER_TRANSLUCENCY (40)	<p>SmallInt value, representing the translucency percentage for the specified layer. These values range from 100=fully transparent to 0=fully opaque.</p> <p>To turn set the translucency or alpha for a layer, use the Set Map Layer statement.</p>
LAYER_INFO_LABEL_ALPHA (41)	<p>SmallInt value, representing the alpha factor for the labels of the specified layer. These values range from 0=fully transparent to 255=fully opaque.</p> <p>To turn set the translucency or alpha for labels in a layer, use the LayerAlpha token in the Set Map Layer LABELCLAUSE statement.</p>

Register Table statement

Supporting Opening Shapefiles with Projection Automatically

Purpose

Builds a MapInfo Professional table from a spreadsheet, database, text file, raster, or grid image. You can call this statement from the MapBasic Window in MapInfo Professional.

Syntax

```
Register Table source_file
Type "SHAPEFILE" [ Charset char_set ] CoordSys auto
```

where

auto - When the Shapefile dataset has a .PRJ file associated with it, use this option to display the Shapefile with the coordinate system specified in the .PRJ file. If the .PRJ file does not exist or the coordinate system is not converted to a MapInfo coordinate system, the command will fail and the application will post an error message.

Supporting Transaction Capabilities for WFS Layers

Syntax

```
Register Table source_file
{ Type "NATIVE" |
  Type "DBF" [ Charset char_set ] |
  Type "ASCII" [ Delimiter delim_char ][ Titles ][ CharSet char_set ] |
  Type "WKS" [ Titles ] [ Range range_name ] |
  Type "WMS" Coordsys...
  Type "WFS" [ Charset char_set ] Coordsys... [ Symbol... ]
    [ Linestyle Pen(...) ] [ Regionstyle Pen(...) Brush(...) ]
    [Editable]
```

where:

Editable reflects the Allow Edits choice.

Save Window statement

You can now use this statement in MapBasic to return translucency values for raster files. As of the MapInfo Professional 9.5 Maintenance Release, this will also work for vector files. The EMF+ and EMF+Dual are only available in the MapInfo Professional 9.5 Maintenance Release and later.

```
Save Window window_id
  As filespec
  Type filetype
  [ Width image_width [ Units paper_units ] ]
  [ Height image_height [ Units paper_units ] ]
  [ Resolution output_dpi ]
  [ Copyright notice [ Font... ] ]
```

where *filetype* can also be:

- "BMP" that specifies Bitmap format
- "WMF" that specifies Windows Metafile format
- "JPEG" that specifies JPEG format
- "JP2" that specifies JPEG 2000 format
- "PNG" that specifies Portable Network Graphics format
- "TIFF" that specifies TIFF format
- "TIFFCMYK" that specifies TIFF CMYK format
- "TIFFG4" that specifies TIFFG4 format
- "TIFFLZW" that specifies TIFFLZW format
- "GEOTIFF" that specifies georeferenced TIFF format

- "GIF" that specifies GIF format
- "PSD" that specifies Photoshop 3.0 format
- "EMF" that specifies Windows Enhanced Metafile format
- "EMF+" that specifies Windows EMF+ format
- "EMF+DUAL" that specifies a file format containing both EMF and EMF+ formats in a single file

Server ConnectionNumber Create Map statement

To support the changes to the Make Table Mappable dialog box, we use the Server <Connection Number> Create Map statement to create the actual Map_Catalog registration. To support annotation text, we need to introduce a new object type (Text) for linked_tables:

```
Type { MICODE columnname | XYINDEX columnname | SPATIALWARE }
CoordSys...
[ MapBounds { Data | Coordsys | Values ( x1, y1 )( x2, y2 ) } ]
[ ObjectType { Point | Line | Region | Text | ALL } ]
[ Symbol (...) ]
[ Linestyle Pen(...) ]
[ Regionstyle Pen(...) Brush(...) ]
[ Style Type style_number [ Column column_name ] ]
```

Text This option allows for the placement of Oracle Spatial annotation text into a text object. The type for this option is ORA_SP.

Note The *ALL* option does not include text.

The following is an example of the MapBasic Statement for the ANNOTEXT_TABLE:

```
Server 1 Create Map For ""MIPRO""."ANNOTEXT_TABLE"" Type ORA_SP
"TEXTOBJ" CoordSys Earth Projection 12, 62, "m", 0 Bounds (-34012036.7393,
-8625248.51472) (34012036.7393, 8625248.51472) mapbounds data ObjectType
Text
```

Server Create Map statement

Purpose

Identifies the spatial information for a server table. It does not alter the table to add the spatial columns. For this release, we have added the option to place Oracle 11g annotation text in MapInfo maps. You can issue this statement from the MapBasic Window in MapInfo Professional.

To support the changes to the Make Table Mappable dialog box, we use the Server <Connection Number> Create Map statement to register the metadata in the MAP CATALOG. To support ANNOTATION TEXT, we have introduced a Text object type. The statement is now:

Syntax

Server ConnectionNumber Create Map

```
For linked_table
Type { MICODE columnname | XYINDEX columnname | SPATIALWARE }
CoordSys...
[ MapBounds { Data | Coordsys | Values ( x1, y1 ) ( x2, y2 ) } ]
[ ObjectType { Point | Line | Region | Text | ALL } ]
[ Symbol (...) ]
[ Linestyle Pen(...) ]
[ Regionstyle Pen(...) Brush(...) ]
[ Style Type style_number [ Column column_name ] ]
```

Text supports the creation of the text object for annotation text. The ALL option does not include this text object.

Example

```
Type { MICODE columnname | XYINDEX columnname | SPATIALWARE | ORA_SP }
```

The type will be ORA_SP. The following is an example of the MapBasic Statement for the ANNOTEXT_TABLE:

```
Server 1 Create Map For ""MIPRO"."ANNOTEXT_TABLE"" Type ORA_SP
"TEXTOBJ" CoordSys Earth Projection 12, 62, "m", 0 Bounds (-34012036.7393,
-8625248.51472) (34012036.7393, 8625248.51472) mapbounds data ObjectType
Text
```

Set Layout statement

Purpose

Modifies an existing Layout window. You can issue this statement from the MapBasic Window in MapInfo Professional. The Objects Alpha and Objects Translucency options will not be operational until the MapInfo Professional 9.5 Maintenance Release.

Syntax

```
Set Layout
[ Window window_id ]
[ Center ( center_x, center_y ) ]
[ Extents { To Fit | ( pages_across, pages_down ) } ]
[ Pagebreaks { On | Off } ]
[ Frame Contents { Active | On | Off } ]
[ Ruler { On | Off } ]
[ Zoom { To Fit | zoom_percent } ]
[ {Objects Alpha alpha_value} | {Objects Translucency
translucency_percent} ]
```

where

alpha_value is an integer value representing the alpha channel value for translucency. Values range from 0-255 where 0 is completely transparent and 255 is completely opaque. Values between 0-255 make the objects in the layout display translucently.

translucency_percent is an integer value representing the percentage of translucency for the objects in a layout. Values range between 0-100. 0 is completely opaque. 100 is completely transparent.

Note Specify either Alpha or Translucency but not both, since they are different ways of specifying the same result. If you specify multiple keywords, the last value will be used.

Set Map statement

Translucency Changes

Purpose

There is new syntax for this statement to support vector translucency. This code will not be functional until MapInfo Professional 9.5 Maintenance Release or later.

Syntax

```
Set Map
  [ Window window_id ]
  .
  .
  .
  [ Layer layer_id
  [ Activate { Using launch_expr[ On { [ [ Labels ] | [ Objects ] ] }
  ] | [ Relative Path { On | Off } ] [ Enable { On | Off } ] },

  [ Using launch_expr[ On { [ [ Labels ] | [ Objects ] ] } ] | [ Relative
Path { On | Off } ] [ Enable { On | Off } ] ]
  [ Editable { On | Off } ]
  [ Selectable { On | Off } ]
  [ Zoom ( min_zoom, max_zoom ) [ Units dist_unit ] [{ On | Off } ] ]
  [ Arrows { On | Off } ]
  [ Centroids { On | Off } ]
  [ Default Zoom ]
  [ Nodes { On | Off } ]
  [ Inflect num_inflections [ by percent ] at
    color:value [, color:value ]
  [ Round rounding_factor ]
Contrast contrast_value ]
  [ Brightness brightness_value ]
  [ { Alpha alpha_value } | { Translucency translucency_percent } ]
  [ Transparency { Off | On } ]
  [ Color transparent_color_value ]
  [ GrayScale { On | Off } ]
  [ Relief { On | Off } ]
LABELCLAUSE
  [ Display { Off | Graphic | Global } ]
  [ Global Line... ]
  [ Global Pen... ]
  [ Global Brush... ]
  [ Global Symbol... ]
```

```
[ Global Font... ]
]
```

Note *Alpha* and *Translucency* are not new tokens for Set Map Layer. Previously they were only used for raster or grid layers. As of the Maintenance Release of MapInfo Professional 9.5, they can be used for vector layers too.

alpha_value is an integer value representing the alpha channel value for translucency. Values range from 0-255. 0 is completely transparent. 255 is completely opaque. Values between 0-255 make the layer display translucent.

translucency_percent is an integer value representing the percentage of translucency for a layer. Values range between 0-100. 0 is completely opaque. 100 is completely transparent.

Note Specify either Alpha or Translucency but not both, since they are different ways of specifying the same result. If you specify multiple keywords, the last value will be used.

Auto Retry, Percent Overhang, and Alpha Value for Label Layer Changes

There is new syntax for the *LABELCLAUSE*, which is shown in bold and described below:

```
[ Label [ Line { Simple | Arrow | None } ]
[ Position [ Center ] [ Above | Below ] [ Left | Right ] [ Auto Retry { On
| Off } ] ]
[ Font... ] [ Pen... ]
[ With label_expr ]
[ Parallel { On | Off } | Follow Path ] [ Percent Over percent]
[ Visibility { On | Off | Zoom( min_vis, max_vis ) [ Units dist_unit ] } ]
[ Auto [ { On | Off } ] ]
[ Overlap [ { On | Off } ] ]
[ PartialSegments { On | Off } ]
[ Duplicates [ { On | Off } ] ]
[ Max [ number_of_labels ] ]
[ Offset offset_amount ]
[ Default ]
[ Object ID
  [ Table alias ]
  [ Visibility { On | Off } ]
  [ Anchor ( anchor_x, anchor_y ) ]
  Text text_string
  [ Position [ Center ] [ Above | Below ] [ Left | Right ] ]
  [ Font... ] [ Pen... ]
  [ Line { Simple | Arrow | None } ]
  [ Angle text_angle | Follow Path ]
  [ Offset offset_amount ]
  [ Callout ( callout_x, callout_y ) ] ]
[ Object... ]
]
[ LabelAlpha alpha_value ]
```

Auto Retry

This parameter allows users to apply a placement algorithm that will try multiple label positions until a position is found that does not overlap any other label, or until all positions are exhausted.

- *When Writing Workspaces*, if the Auto Retry feature is On, we write Auto Retry On to the workspace after the Position clause (but the order isn't important), and increase the workspace version to 9.5. If the feature is Off, we do not write anything to the workspace and do not increase the version number. A version 9.5 workspace can have Auto Retry Off in it, but we do not explicitly write it out, to avoid increasing the version unnecessarily.
- *When Reading Workspaces* If Auto Retry On or Auto Retry Off is in the workspace, it must be a version 9.5 workspace, otherwise a syntax error occurs. If Auto Retry is On, different positions are tried to place the label. If Auto Retry is Off, no retry is attempted - this is the default behavior. Overlap must be Off to enable the Auto Retry feature. If Overlap is On and Auto Retry On/Off are in the same LABELCLAUSE, the Auto Retry mechanism is initialized but ignored, so overlapping labels are allowed.

Percent Over When curved labels are longer than the geometry they name, this is the amount (expressed as a percentage) of overhang permitted. For example, a sample entry might be:

```
Set Map Layer 1 Label Follow Path
Percent Over 40
```

Note This attribute only applies to curved labels.

alpha_value is a SmallInt that represents the alpha value of the labels in this layer. It is a value between 0-255 where 0 is completely transparent and 255 is completely opaque. Values in between display labels translucently.

Note This parameter exists in the MapBasic 9.5 version but are not operational until the MapInfo Professional 9.5 Maintenance Release.

Example of Alpha Parameter Use for a Label Layer

```
"Set Map Layer 1 Label LabelAlpha 123"
```

Set Window statement

Purpose

This statement has been updated to accommodate the anti-aliasing choices for vector, text, and image objects. The new code is in bold.

```
Set Window window_id
  [ Position ( x, y ) [ Units paper_units ] ]
  [ Width win_width [ Units paper_units ] ]
  [ Height win_height [ Units paper_units ] ]
  [ Font... ]
  .
  .
  .
  [ Enhanced { On | Off } ]
  [ Smooth [ Vector { None | Antialias } ] [ Text { None | Antialias } ]
    [ Image { None | Low | High } ] ]
```

Note For the MapInfo Professional 9.5 release, the *Enhanced* parameter is turned on by default when the Smooth Text **Antialias** option is turned on. The other parameters (*Smooth Vector*, *Smooth Text*, and *Smooth Image*) exist in the MapBasic 9.5 version but are not operational until the MapInfo Professional 9.5 Maintenance Release.

Enhanced

Sets the version of the rendering technology used to display and print graphics.

For MapInfo Professional 9.5: The *On* option enables the newest rendering technology and is enabled when the user selects the Smooth Text **Anti-alias** option in the MapInfo Professional GUI. The *Off* option enables the older rendering technology and is enabled when the user selects the Smooth Text **None** option in MapInfo Professional.

For MapInfo Professional 9.5 Maintenance Release: The *On* parameter enables the new rendering technology and is set when the user selects the **Enable Enhanced Rendering** check box in the MapInfo Professional. The *Off* option enables the older rendering technology and is set when the user does not select the **Enable Enhanced Rendering** check box in MapInfo Professional.

Smooth

Sets the new rendering technology enhancements for anti-aliasing vector, text and labels, and images.

Note The Smooth options (*Vector*, *Text*, and *Image*) require that the *Enhanced* parameter be set to *On*. MapBasic will throw an error if you turn *Enhanced Off* and set any of the Smooth options to an option other than *None*.

Vector - sets the vector smoothing options for vectors. This feature is operational in the MapInfo Professional 9.5 Maintenance Release only.

None indicates that smoothing is turned off and the vector line and border objects are drawn without anti-aliasing.

Antialias indicates that smoothing is turned on and the vector objects. This option requires that the *Enhanced* parameter be set to *On*.

Text - sets the text smoothing options for non-curved labels (in 9.5) and the non-curved labels and text objects (in the 9.5 Maintenance Release).

For MapInfo Professional 9.5:

The *None* parameter indicates that smoothing is turned off for rotated and horizontal labels. The *Antialias* parameter indicates that the smoothing is turned on for rotated and horizontal labels. This option requires that the *Enhanced* parameter be set to *On*.

For MapInfo Professional 9.5 Maintenance Release:

The *None* parameter indicates that smoothing is turned off for rotated and horizontal labels and text objects.

The *Antialias* parameter indicates that smoothing is turned on for rotated and horizontal labels and text objects. This option requires that the *Enhanced* parameter be set to *On*.

Image - sets the raster image smoothing options. This feature is operational in the MapInfo Professional 9.5 Maintenance Release only.

None indicates that the smoothing is turned off for raster images.

Low indicates that the smoothing is turned on for raster images using a bilinear interpolation method. Using this method, the application displays better quality raster images than *None* but not as good as *High*. Using the *Low* option, the application displays raster images slower than when *None* is used but faster than when *High* is used. This option requires that the *Enhanced* parameter be set to *On*.

High indicates that the smoothing is turned on for raster images using a bicubic interpolation method. Using this method, the application displays better quality raster images than *Low* but results in slower display performance. This option requires that the *Enhanced* parameter be set to *On*.

TableInfo() function

Purpose

Returns information about an open table. Has a define for FME (Universal Data) tables. You can call this function from the MapBasic window in MapInfo Professional.

Syntax

```
TableInfo(table_id, attribute)
```

table_id A string representing a table name, a positive integer table number, or 0 (zero).

attribute An integer code indicating which aspect of the table to return.

Return Value

String - The table CoordSys string without bounds.

MapBasic.Def Attribute/(Code)	TableInfo() returns:
TAB_INFO_COORDSYS_CLAUSE_WITH_OUT_BOUNDS (37)	String result, representing the table's CoordSys clause without bounds.

Examples

```
TableInfo(table_id, TAB_INFO_COORDSYS_CLAUSE)
TableInfo(table_id, 29) - Returns the coordsys clause with bounds
```

WindowInfo() function

Purpose

The additions to this function return information about the enhanced and smooth settings for a window. The new features are fully operational in the MapInfo Professional 9.5 Maintenance Release. In MapBasic and MapInfo Professional 9.5, the *Enhanced* and *Smooth Text* options are operational. The Smooth Image and Smooth Vector options are not fully operational until the MapInfo Professional Maintenance Release.

Syntax

```
WindowInfo( window_spec, attribute )
```

window_spec is a number or a code that specifies which window you want to query.

attribute is an integer code indicating which information about the window to return.

Description

We have added 4 new attribute codes to return information about enhanced rendering and smoothing:

attribute code	WindowInfo(attribute) returns:
WIN_INFO_ENHANCED_RENDERING (45)	Logical value: TRUE if enhanced rendering is on for this window. To turn enhanced rendering on or off, use the Set Window statement.
WIN_INFO_SMOOTH_TEXT (46)	String value: The string representation of the current smooth text mode for the window. To change the smooth mode, use the Set Window statement.
WIN_INFO_SMOOTH_IMAGE (47)	String value: The string representation of the current smooth image mode for the window. To change the smooth mode, use the Set Window statement.
WIN_INFO_SMOOTH_VECTOR (48)	String value: The string representation of the current smooth vector mode for the window. To change the smooth mode, use the Set Window statement.
