



LatitudeZERO User Manual

Introduction

With LatitudeZERO you can bring the world of latitude and longitude coordinates into FileMaker.

- Calculate the distance between two ZIP / postal codes
- Calculate the distance between two coordinates
- Return results in miles, kilometers or nautical miles
- Determine whether coordinates are within or outside a specified radius
- Convert distances from one unit of measure to another
- Allocate Sales Reps / Agents that service the area within a certain radius
- Sort records by distance
- ... and more

LatitudeZERO is a set of Custom Functions. The Custom Functions are extremely easy to incorporate into your solutions ... you can literally be up and running within minutes.

The Custom Functions work with a standard copy of FileMaker Pro, however FileMaker Pro Advanced is required to copy and paste the Custom Functions into your solutions. Once the Custom Functions are in place, they will be available to all users.

Importing the Custom Functions

In order to install the custom functions, you will need FileMaker Pro Advanced.

Open your solution within **FileMaker Pro Advanced**.

Go to: File > Manage > Custom Functions and click on the 'Import' button.

Locate the LatitudeZERO example file **Example.fmp12** and import all the Custom Functions, then click OK.

That's it; you have installed the custom functions!

Custom Functions Summary

LatitudeZERO is a set of custom functions. The custom functions will be described in more detail on the following pages, but here is a summary of what they all do:

LatitudeZERO Distance

Calculates the distance between two coordinates.

LatitudeZERO WithinRadius

Determines whether two coordinates are within a specified radius.

LatitudeZERO OutsideRadius

Determines whether two coordinates are outside a specified radius.

LatitudeZERO UnitConversion

Converts a distance from one unit of measure to another.

LatitudeZERO Bearing

Determines the bearing between two coordinates.

LatitudeZERO_Distance

Description

Calculates the distance between two coordinates.

Syntax

LatitudeZERO_Distance (Lat1, Lon1, Lat2, Lon2, Unit, Rounding)

Parameters

Lat1, Lon1 Latitude and Longitude of the first coordinate as a decimal value.

Lat2, Lon2 Latitude and Longitude of the second coordinate as a decimal value.

Unit Unit of measure for the returned distance.
(*M=Miles, K=Kilometers, N=Nautical Miles*)

Rounding The number of decimal places for the returned distance.

Result

The returned value is the distance between *Lat1/Lon1* and *Lat2/Lon2*, and the distance is based on the unit of measure specified.

NOTE:

If you would like the calculation to be dynamically recalculated, you will need to turn on the “**Do not store calculation**” feature within the calculation. If on the other hand you need the calculation to be “stored”, then leave this option unchecked.

LatitudeZERO_WithinRadius

Description

Determines whether two coordinates are within a specified radius.

Syntax

LatitudeZERO_WithinRadius (Lat1, Lon1, Lat2, Lon2, Unit, Radius)

Parameters

Lat1, Lon1 Latitude and Longitude of the first coordinate as a decimal value.

Lat2, Lon2 Latitude and Longitude of the second coordinate as a decimal value.

Unit Unit of measure for the returned distance.
(*M=Miles, K=Kilometers, N=Nautical Miles*)

Radius The distance to compare whether the two coordinates are within the radius.

Result

The returned value is either 1 (true) or 0 (false). A true value means the two coordinates are within the specified radius, a false value means the two coordinates are outside the specified radius.

NOTE:

If you would like the calculation to be dynamically recalculated, you will need to turn on the “**Do not store calculation**” feature within the calculation. If on the other hand you need the calculation to be “stored”, then leave this option unchecked.

Function: LatitudeZERO_OutsideRadius

Description

Determines whether two coordinates are outside a specified radius.

Syntax

LatitudeZERO_OutsideRadius (Lat1, Lon1, Lat2, Lon2, Unit, Radius)

Parameters

Lat1, Lon1 Latitude and Longitude of the first coordinate as a decimal value.

Lat2, Lon2 Latitude and Longitude of the second coordinate as a decimal value.

Unit Unit of measure for the returned distance.
(*M=Miles, K=Kilometers, N=Nautical Miles*)

Radius The distance to compare whether the two coordinates are outside the radius.

Result

The returned value is either 1 (true) or 0 (false). A true value means the two coordinates are outside the specified radius, a false value means the two coordinates are within the specified radius.

NOTE:

If you would like the calculation to be dynamically recalculated, you will need to turn on the “**Do not store calculation**” feature within the calculation. If on the other hand you need the calculation to be “stored”, then leave this option unchecked.

LatitudeZERO_UnitConversion

Description

Converts a distance from one unit of measure to another.

Syntax

LatitudeZERO_UnitConversion (Distance, fromUnit, toUnit, Rounding)

Parameters

Distance The distance that will be converted to another unit of measure.

fromUnit The originating unit of measure for the distance specified.
(*M=Miles, K=Kilometers, N=Nautical Miles*)

toUnit The unit of measure for the converted distance.
(*M=Miles, K=Kilometers, N=Nautical Miles*)

Rounding The number of decimal places for the converted distance.

Result

Returns the converted distance.

NOTE:

If you would like the calculation to be dynamically recalculated, you will need to turn on the “**Do not store calculation**” feature within the calculation. If on the other hand you need the calculation to be “stored”, then leave this option unchecked.

LatitudeZERO_Bearing

Description

Determines the bearing between two coordinates.

Syntax

LatitudeZERO_Bearing (Lat1, Lon1, Lat2, Lon2)

Parameters

Lat1, Lon1 Latitude and Longitude of the first coordinate as a decimal value.

Lat2, Lon2 Latitude and Longitude of the second coordinate as a decimal value.

Result

Returns the bearing from Point A to Point B.

NOTE:

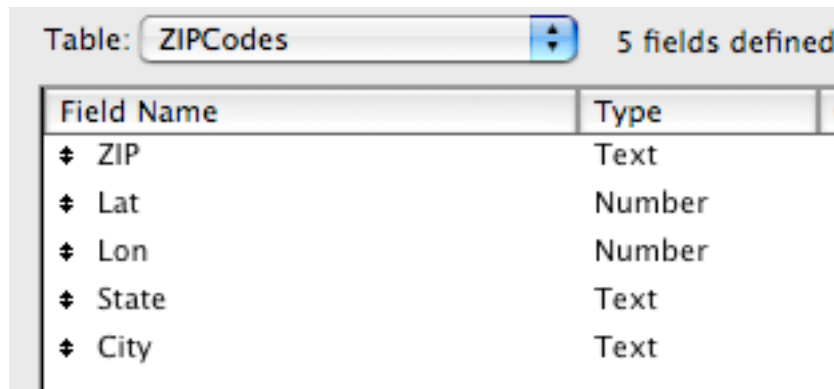
If you would like the calculation to be dynamically recalculated, you will need to turn on the “**Do not store calculation**” feature within the calculation. If on the other hand you need the calculation to be “stored”, then leave this option unchecked.

Working with ZIP / postal codes

LatitudeZERO does not come with any ZIP / postal geocoded data. You will need to source your own data provider. Contact your local postal service provider for more information.

If you refer to the Example.fmp12, you will gain a better understanding of how it all works.

First, you will need to create the ZIPCodes table. You may copy the table from the Example.fmp12 file and paste it into your solution, or you may create the table manually.

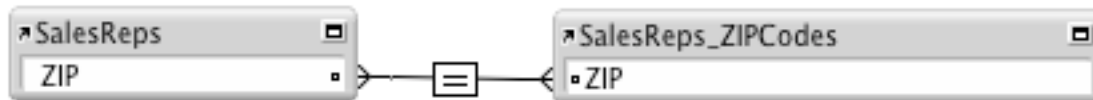


The screenshot shows a database table definition window for a table named 'ZIPCodes'. It indicates that 5 fields are defined. The fields are listed in a table with two columns: 'Field Name' and 'Type'.

Field Name	Type
ZIP	Text
Lat	Number
Lon	Number
State	Text
City	Text

You will need 5 fields as shown above.

Next, you will need to create a relationship between your main table i.e. SalesReps and ZIPCodes so that you can reference the appropriate coordinates. The relationship is simply ZIP to ZIP as shown below:



You now have a link within your main table (i.e. SalesReps) to ZipCodes. Please note that SalesReps_ZIPCodes shown above is a table occurrence referencing the ZIPCodes base table.

Now that your relationship is set, you need to create Latitude and Longitude fields (i.e. Lat & Lon) within your main table (i.e. SalesReps). As shown bellow:

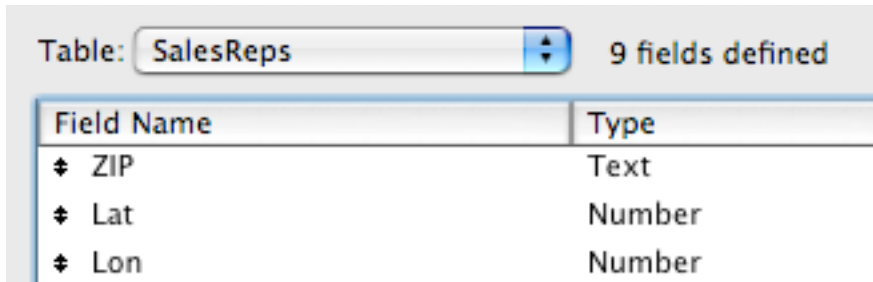
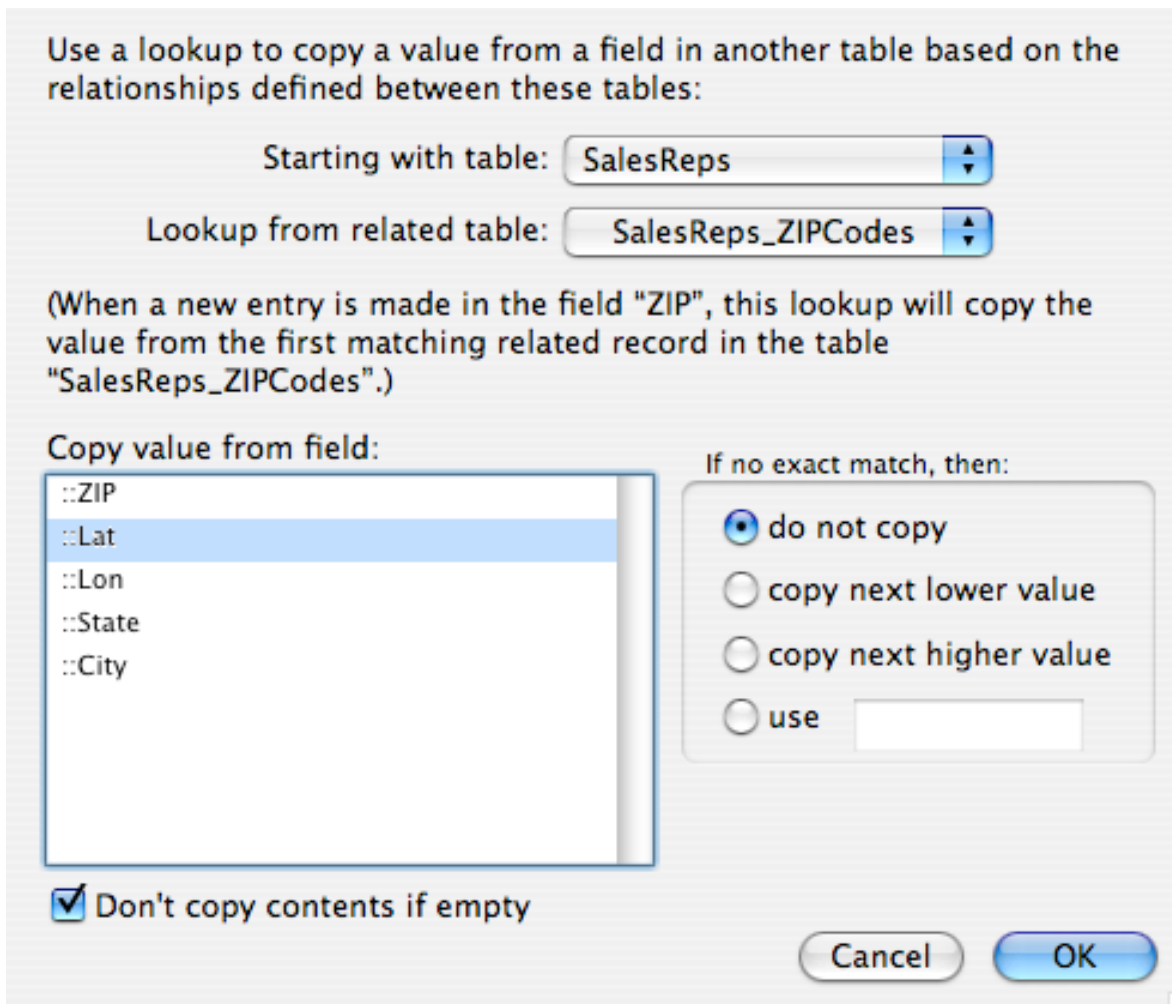


Table: SalesReps	9 fields defined
Field Name	Type
ZIP	Text
Lat	Number
Lon	Number

Note: The Latitude and Longitude fields will be used to look up the coordinates so that they are stored locally within the main table, allowing faster computation later on.

Next, you need to set up a lookup for both latitude and longitude fields. To set up the lookup, double click the Latitude (i.e. Lat) field and specify the lookup as show bellow:



Use a lookup to copy a value from a field in another table based on the relationships defined between these tables:

Starting with table: SalesReps

Lookup from related table: SalesReps_ZIPCodes

(When a new entry is made in the field "ZIP", this lookup will copy the value from the first matching related record in the table "SalesReps_ZIPCodes".)

Copy value from field:

- ZIP
- Lat
- Lon
- State
- City

If no exact match, then:

- ☒ do not copy
- ☐ copy next lower value
- ☐ copy next higher value
- ☐ use

☒ Don't copy contents if empty

Cancel OK

Now repeat the process for the longitude field (i.e. Lon) and set the lookup as follows:

Use a lookup to copy a value from a field in another table based on the relationships defined between these tables:

Starting with table:

Lookup from related table:

(When a new entry is made in the field "ZIP", this lookup will copy the value from the first matching related record in the table "SalesReps_ZIPCodes".)

Copy value from field:

- ::ZIP
- ::Lat
- ::Lon**
- ::State
- ::City

If no exact match, then:

- ☒ do not copy
- ☐ copy next lower value
- ☐ copy next higher value
- ☐ use

☒ Don't copy contents if empty

If you're installing LatitudeZERO within an existing solution which contains data, you will need to bring the coordinates data over to your main table (i.e. SalesReps) so that each record will contain the appropriate coordinates.

First you will need to select "Show All Records" from the Records menu while you're on a layout assigned to your main table. This will ensure all your records will be updated.

Next, click inside the ZIP code field and select "Relookup Field Contents" from the Record menu. Your main table now contains the appropriate coordinates for each ZIP code.

Before you can do anything meaningful with LatitudeZERO, you will need to create two global fields within your main table (i.e. SalesReps) as shown below.

Field Name	Type	Options
✦ gZIP	Text	Global
✦ gRadius	Number	Global

Next, you will also need a calculation field called cRadiusStatus as shown bellow:

Field Name	Type
✦ cRadiusStatus	Calculation

The calculation formula might look something like this:

```
cRadiusStatus =  
Case(cDistance > gRadius;  
  TextColor("Outside Radius"; RGB(170;0;0));  
  TextColor("Within Radius"; RGB(0;102;0))  
)
```

You may modify the calculation to suit your needs

And finally you will need to decide where to create an interface for all these components. If you refer to the Example.fmp12 file, you will find that we have a table called AllocateSalesRepExample that relates to the SalesReps table via a Cross-Join (i.e. matching all records).

The Cross-Join [X] relationship allows us to display all of the sales reps and sort by distance, i.e. showing the closes sales reps first.

The portal should be sorted by the cDistance calculation field, as shown bellow:

The screenshot shows the sorting configuration interface. On the left, a list of fields includes ZIP, Lat, Lon, RepName, cDistance, cKey, gZIP, gRadius, and cRadiusStatus. In the center, there are 'Clear All' and 'Clear' buttons. On the right, the 'Sort Order' window shows 'cDistance' selected with an ascending sort icon. Below these, there are three radio button options: 'Ascending order' (selected), 'Descending order', and 'Custom order based on value list'. A dropdown menu shows '<unknown>'. Further down, there are checkboxes for 'Reorder based on summary field' (with a 'Specify...' button) and 'Override field's language for sort' (with a dropdown showing 'English'). At the bottom right are 'Cancel' and 'OK' buttons.

When all strung together, the end result may look something like this:

Example

ZIP	10021						
Radius	30	Miles					
Rep	City	State	ZIP	Distance	Radius Status		
Lisa Joe	Manhattan	NY	10021	0	Within Radius		
Terry Lane	Bronx	NY	10463	8	Within Radius		
Zak Groves	Staten Island	NY	10302	13	Within Radius		
Michale Peterson	Ardsley	NY	10502	18	Within Radius		
Kathy Hearts	Ardsley	NY	10502	18	Within Radius		
Pete Bird	Hawthorne	NY	10532	25	Within Radius		
Chris Martin	Bedford Hills	NY	10507	35	Outside Radius		

Please refer to the *Example.fmp12* file to get a better understanding of how it works.