Linear Referencing and Dynamic Segmentation in ArcGIS™ 8.1

An ESRI® Technical Paper • May 2001

ESRI 380 New York St., Redlands, CA 92373-8100, USA • TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

Linear Referencing and Dynamic Segmentation in ArcGIS 8.1

An ESRI Technical Paper

Contents **Page**

J-8673

Contents **Page**

Linear Referencing and Dynamic Segmentation in ArcGIS 8.1

Introduction

This document is a supplement to many other publications and the online help. It is intended to illustrate ArcGIS™ 8.1 software's linear referencing and dynamic segmentation capabilities. Both the out-of-the-box functionality and application programming interface (API) level support are discussed. Where necessary, sample code is provided and discussed. A good understanding of ArcGIS is recommended before trying to apply the concepts contained in this document.

The Need for Linear Referencing

Highways, city streets, railroads, and rivers, as well as distribution networks such as telephone lines and water and sewer networks, are all examples of linear features. Our ability to model them effectively requires that we understand, maintain, and analyze these features.

Geographic information systems often represent spatial information with a two-dimensional x,y coordinate system. This is good for representing the locations of boundaries, water bodies, and road networks. Other measuring systems such as river kilometer and route kilometer can also record information along linear features. Instead of using x,y measurements, these systems simplify recording the data in the field by using a single relative position. Location is given in terms of a known feature and a position or measure on it. For example, route I-10, kilometer 23, uniquely identifies a position in geographic space without having to express it in x,y coordinates or latitude–longitude terms. This is linear referencing.

What Is Dynamic Segmentation?

Dynamic segmentation allows multiple sets of attributes to be associated with any portion of a linear feature. These attributes can be stored, displayed, queried, and analyzed without affecting the underlying linear data's x,y coordinates.

Dynamic segmentation models linear features using routes and route events. A route represents a linear feature such as a city street, highway, or river. Routes contain measures, which describe the distance along them. The measures are used to locate data, which describes parts of the route. Data along routes is modeled using route events.

Any application involving linear features can benefit from using the functionality provided by dynamic segmentation. Some examples include

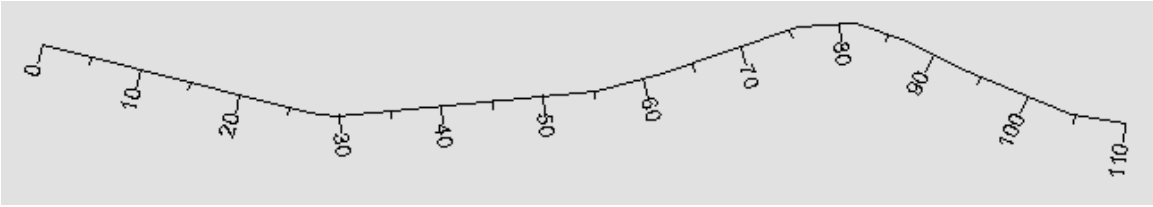
- Collecting data along such linear features as roads, rivers, and railways
- Managing pavement quality

- Managing urban networks and inventories such as for street signs, traffic lights, pedestrian crossings, bicycle paths, curbs, and sidewalks
- Managing railroad track quality
- Managing rivers and streams
- Modeling shorelines
- Modeling maritime navigation routes
- Analyzing oil and gas exploration
- Modeling communication and distribution networks such as electric grids, telephone lines, water and sewer, and television cable networks

Routes and Measures

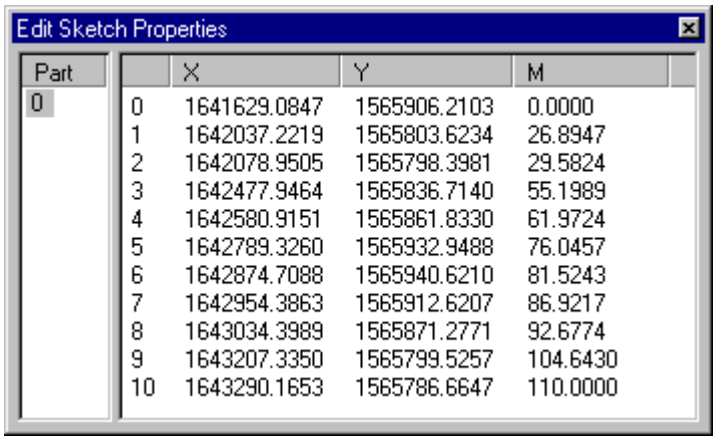
A polyline is an ordered collection of paths that can be connected or disjointed. All linear features are stored in a feature class whose geometry type is polyline (for more information on polylines, refer to Chapter 6, "The Shape of Features," of *Modeling Our World*). A route is simply one or more linear features on which attributes can be defined. Attributes can be assigned to a route because each route (a) has an identifier stored in a field and (b) has an associated measurement system. This measurement system defines discrete locations along a linear feature.

A route's measurements are stored with its geometry. Route's geometry differs from other linear geometry in that instead of being a collection of x,y coordinates, a route's geometry is a collection of x,y,m values. In the following example, there is a route that is 110 miles long in the real world. Displaying such a route on a map might look something like this.



A route's geometry is a polyline with measures.

The vertices that make up this polyline might look something like this.



Each vertex in a route has an x,y,m value. Measure values are only stored at the vertices. Measure values between vertices are interpolated.

It is important to note that although many applications use measures to represent increasing linear distances along a line, measure values can arbitrarily increase, remain constant, or decrease. In special cases, certain locations along a linear feature may not have any known measure value at all. In this case, a "not a number" (NaN) value is used instead of a measure. When any form of analysis is used to determine the measure between (i) a measure and a NaN value or (ii) two NaN measures, a NaN will be returned.

A measure value is independent of a geometry's coordinate system. That is, measures do not have to be of the same units as the x,y coordinates. For example, a route stored in UTM meters might have its measure values stored in feet or miles.

Note: Routes stored in coverage route systems are turned into polylines with measure values "on the fly" when read by the ArcGIS Desktop applications (refer to Appendix A—Coverage Route Systems in ArcMap).

A collection of routes with a common system of measurement can be stored in a single feature class—for example, a set of all highway routes in a county. In the context of a geodatabase, many feature classes containing routes can be stored in a feature data set. For example, school bus, truck, and ambulance feature classes could exist in a feature data set of a city.

A feature class can store routes when (i) its geometry type is polyline and (ii) it can store measures. The following Visual Basic® code demonstrates one way to determine whether a feature class can store routes. Here, the feature class in question is stored in a personal geodatabase. Routes, however, can also be stored in a coverage route system, a PolylineM shapefile, or a Polyline feature class in an ArcSDE™ geodatabase.

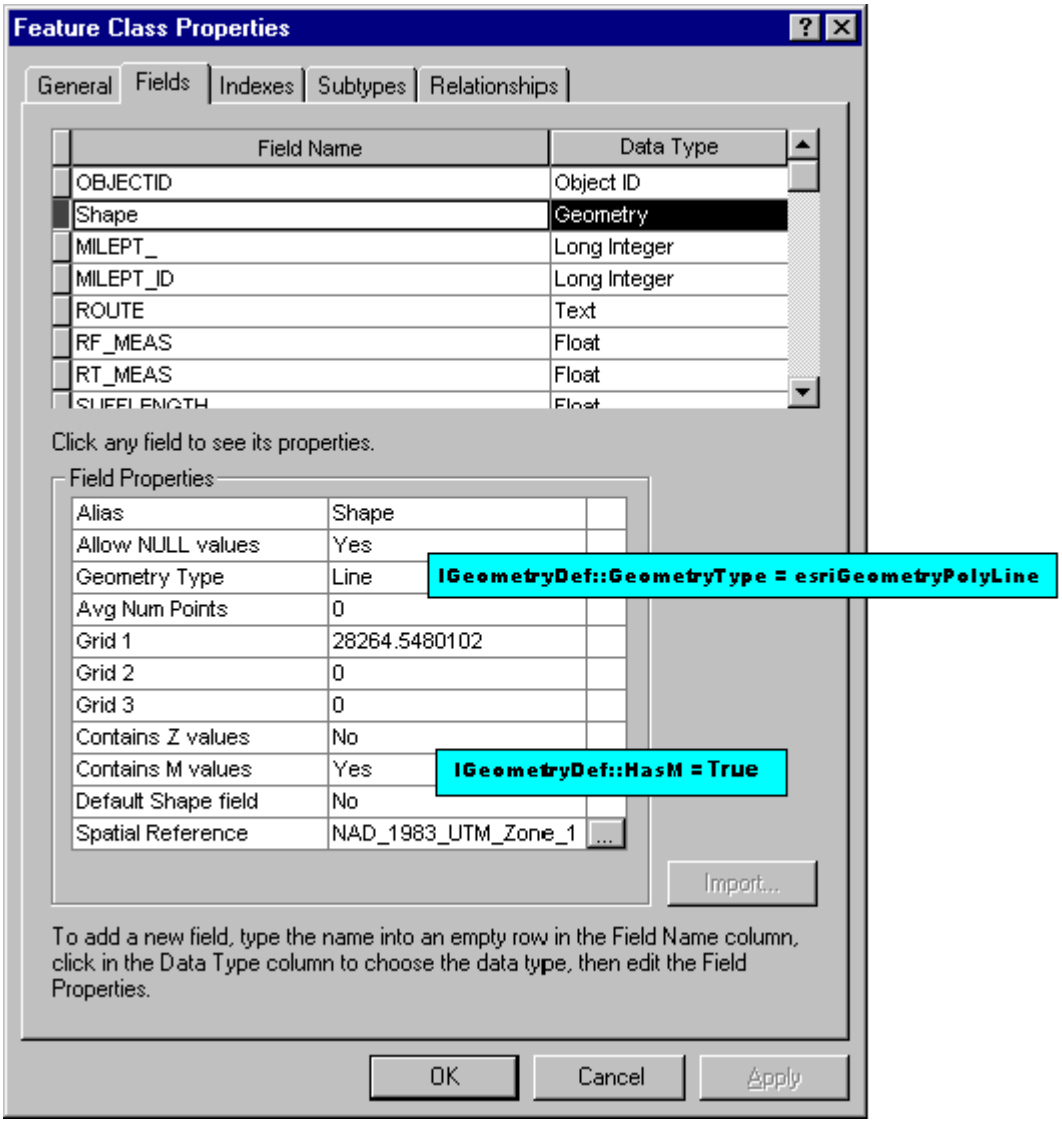
```
Public Sub RouteTest()
  Dim pFact As IWorkspaceFactory
```

```
  Dim pFeatWS As IFeatureWorkspace
  Dim pFC As IFeatureClass
  Set pFact = New AccessWorkspaceFactory
  Set pFeatWS = pFact.OpenFromFile("d:\data\nysdot\nysdot.mdb", 0)
  Set pFC = pFeatWS.OpenFeatureClass("route_milept")

  Dim pFlds As IFields
  Dim sShpFld As String
  Dim pShpFld As IField
  Dim lIdx As Long
  Dim pGDef As IGeometryDef
  Set pFlds = pFC.Fields
  lIdx = pFlds.FindField(pFC.ShapeFieldName)
  Set pGDef = pFlds.Field(lIdx).GeometryDef
  If pGDef.hasM And pGDef.GeometryType = esriGeometryPolyline Then
    MsgBox "This feature class can contain routes!!", vbExclamation, "Route Test"
  Else
    MsgBox "This feature class can NOT contain routes!!", vbExclamation, "Route Test"
  End If
End Sub
```

Determining whether a feature class can store routes is exposed on the Fields tab in the ArcCatalog™ Feature Class Properties dialog.

J-8673



Right-clicking a geodatabase feature class in ArcCatalog accesses the Feature Class Properties dialog.

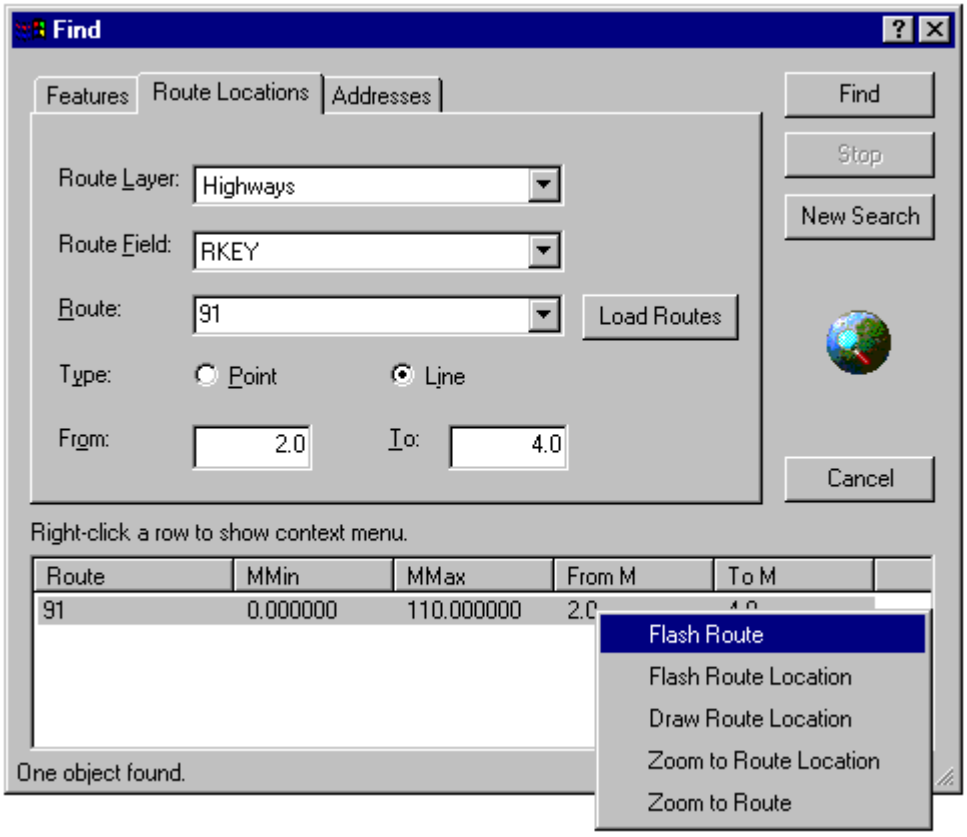
Route Locations and Route Event Tables

Route Locations

A route location describes a portion of a route (line) or a discrete location along a route (point). A linear route location uses both a from- and a to-measure value to describe a portion of a route. "Mile 2 to mile 4 on the I-91" is an example of a linear route location. A point route location uses only a single measure value to describe a discrete location along a route. An example of a point route location would be "mile 3.2 on the I-91."

J-8673

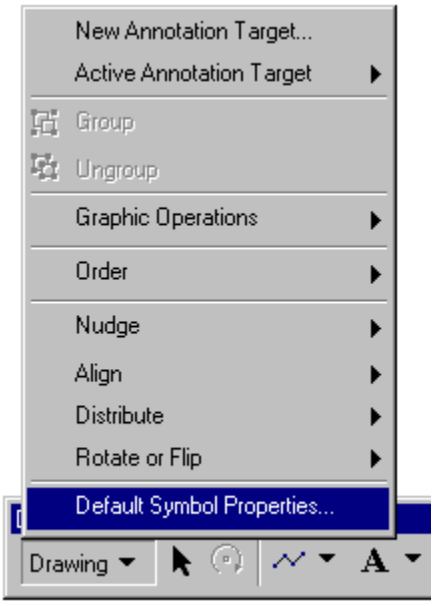
In ArcMap™ software, the Find tool can be used to find either point or line route locations. For more information on finding route locations in ArcMap, refer to the ArcGIS Desktop Help.



The ArcMap Find tool finds route locations.

Tip: The Find tool's Draw Route Location menu option uses ArcMap default symbols. To change these symbols, use the Default Symbol Properties option on the Draw toolbar.

J-8673



The Default Symbol Properties Option on the ArcMap Draw Toolbar

An explanation of the objects necessary to find a route location programmatically can be found in Chapter 8, "Accessing the Geodatabase," of *Exploring ArcObjects*. For a complete code sample, refer to the dynamic segmentation samples in ArcObjects™ Developer Help.

Route Events

When route locations (and their associated attributes) are stored in a table, they are known as route events or simply events. Events are organized into tables based on a common theme. For example, five event tables containing information on speed limits, year of resurfacing, present condition, signs, and accidents can reference highway routes.

A route event table, at a minimum, consists of two fields: an event key and measure location(s). An event key field is a numeric or character value used to identify the route an event belongs to. This field is used in establishing the relationship between event records and the routes they belong to. A measure location is either one or two values describing the position(s) on the route where the event occurs. These values can be defined as any numeric item.

Because there are two types of route locations, there are two types of route event tables: point and line.

An event table can be any type of table that ArcGIS supports. These include INFO™, dBASE®, and text files; geodatabase tables; and OLE DB tables.

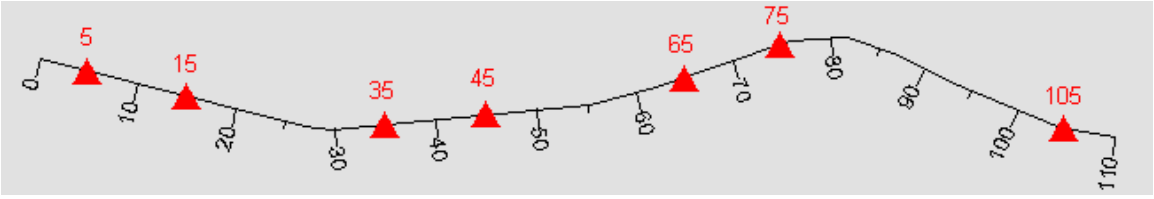
Point Events

Point events occur at a precise point location along a route. For example, a point event table might record the location of accidents along a highway defined by their distance from the start of the route, rather than their x,y coordinate location.

J-8673

RKEY	MILE	DAY_	ALCHOL	HITRUN
91	5	7	0	1
91	15	1	0	0
91	35	3	0	0
91	45	5	0	0
91	65	6	0	0
91	75	3	1	0
91	105	1	0	1

This is an event table of traffic accidents. Each event is recorded in terms of a linear measure along the route.



Point Events Represented with a Point Marker Symbol

Line Events

Linear events describe portions of routes. For example, a line event table might record the pavement quality along different sections of a highway, where the sections are defined by distance along the highway.

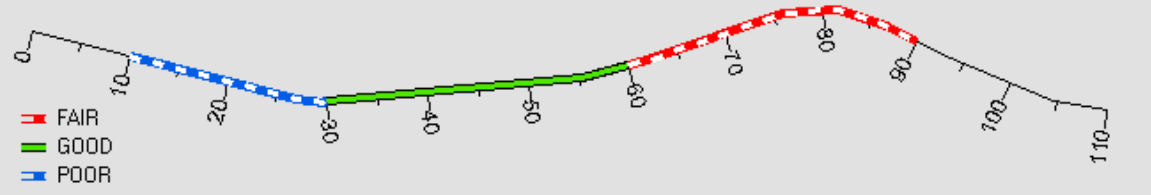
Linear events differentiate from point events in that they use two measure values to describe the measure location of the event.

In the highway application shown, pavement quality has been labeled according to the measures, poor from 10 to 30, good from 30 to 60, and fair from 60 to 90.

RKEY	FMP	TMP	COND
91	10	30	POOR
91	30	60	GOOD
91	60	90	FAIR

This is pavement data contained in an event table. The pavement events are defined in terms of a linear measure along the route.

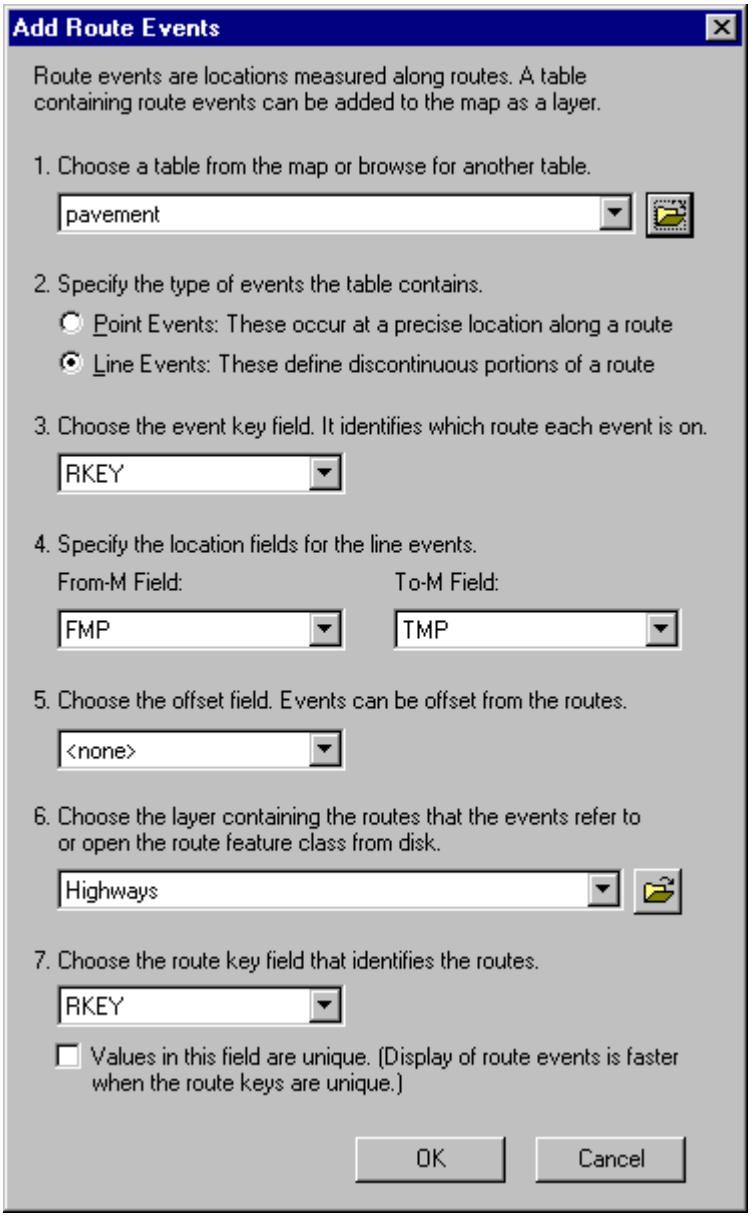
J-8673



Line events are represented with a line symbol.

Route events are added to a map document in ArcMap via the Add Route Events dialog, which is accessed via the Tools/Add Route Events command. Right-clicking an event table in the ArcMap Table of Contents and selecting Display Route Events can also access this dialog. In this case, the event table to be displayed has already been chosen. Hence, the event table drop-down menu will be disabled when the dialog is displayed.

J-8673



The ArcMap Add Route Events Dialog

Tip: Route keys may or may not be unique within a feature class containing routes. For routes stored in an Access or ArcSDE geodatabase, locating route events will be faster if the route keys are unique. However, if the data does not have unique route keys, this option should not be checked. This is because when it is indicated that the route keys are unique, then the underlying segmentation algorithm will only look for one route with the corresponding route key. The event being processed may or may not locate itself on the route feature that was found. This setting has no effect for routes stored in coverage route systems and shapefiles and is disabled.

Tip: Indexes make dynamic segmentation faster. An attribute index for the route key field in both the route feature class and the event table is recommended. It is also recommended that a spatial index for the route feature class be present. Refer to ArcGIS Desktop Help for more information on creating indexes.

When event data has been added to an ArcMap document (e.g., by using the Add Route Events dialog), there will be a feature layer. A feature layer is based on a feature class. In the case of a feature layer created via the dynamic segmentation process, the feature class will be a dynamic one called a RouteEventSource. In a RouteEventSource, the shape of the features (the events) will be calculated on the fly every time they are asked for. For example, when panning or zooming, the dynamic segmentation process is applied and the shape of the events is recalculated.

For the most part, a "dynamic" feature layer behaves just like any other feature layer. It is possible to decide whether or not to display it; the scale at which it should be visible; what features or subset of features to display; how to draw the features, store it as a layer file (.lyr), and export it; and so forth.

Dynamic feature classes can also be edited in ArcMap. It is important to note, however, that it is possible to only edit the attributes. The shapes cannot be edited because the dynamic segmentation process is generating them. Editing this type of feature class actually edits the underlying event table itself.

Note: There may be some editing limitations imposed by the event table. For example, it is not possible to directly edit the attributes of a RouteEventSource created from a delimited text file table since ArcMap does not allow text files to be edited directly.

One interesting side effect of editing a dynamic feature class is that editing an event record's measure values will automatically cause the event to move to its new location. Similarly, editing an event's key value will cause the event to automatically move to its new route. Taking this one step further, editing the geometry (this includes the measures) that route events are located along will cause the events to automatically move to their new location along the edited geometry.

An explanation of the objects necessary to add route events to a map programmatically can be found in Chapter 8, "Accessing the Geodatabase," of *Exploring ArcObjects*. For complete code samples, refer to the dynamic segmentation samples in ArcObjects Developer Help.

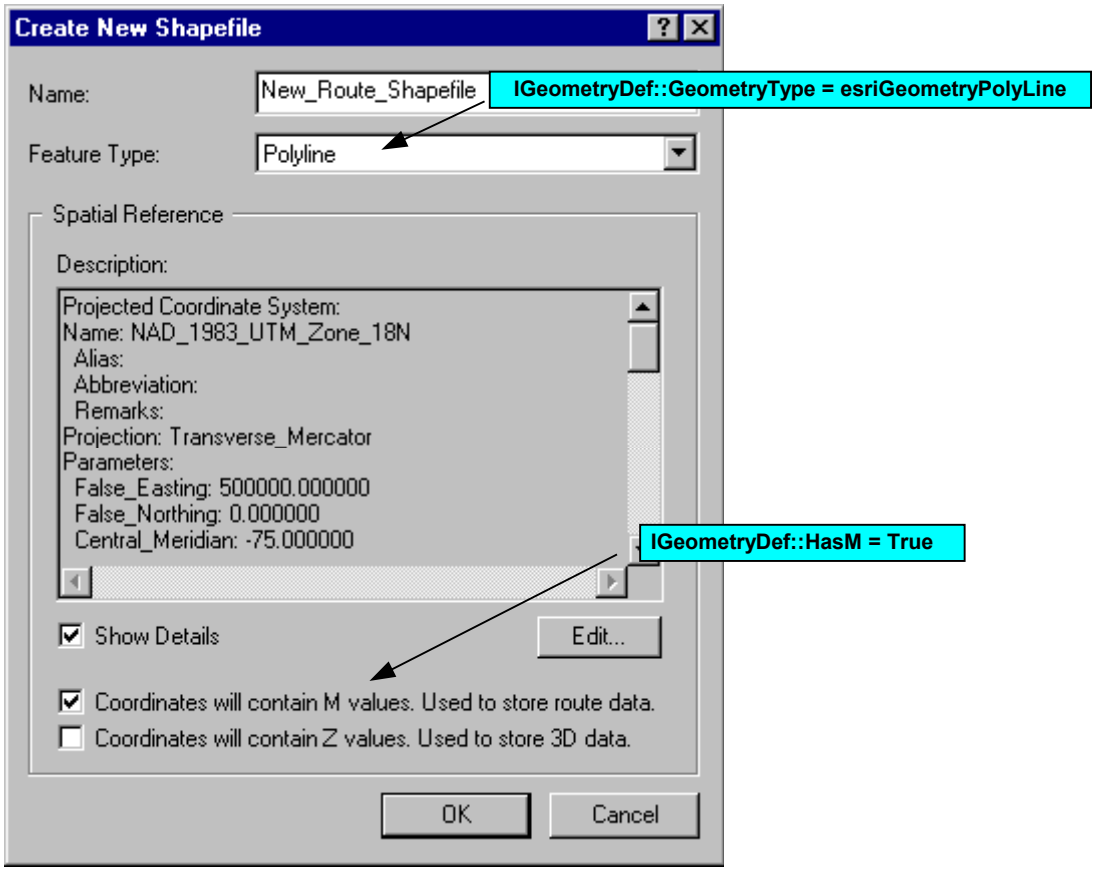
J-8673

Creating and Converting Route Data

Creating New Route Data

ArcCatalog can be used to create new (empty) route feature classes. When creating a shapefile or geodatabase feature class, the geometry field's properties, such as the geometry type and whether measure values can be stored, must be defined. When creating a coverage route system, these considerations need not be made, as the data model is different. Refer to ArcGIS Desktop Help for more information on creating all types of feature classes.

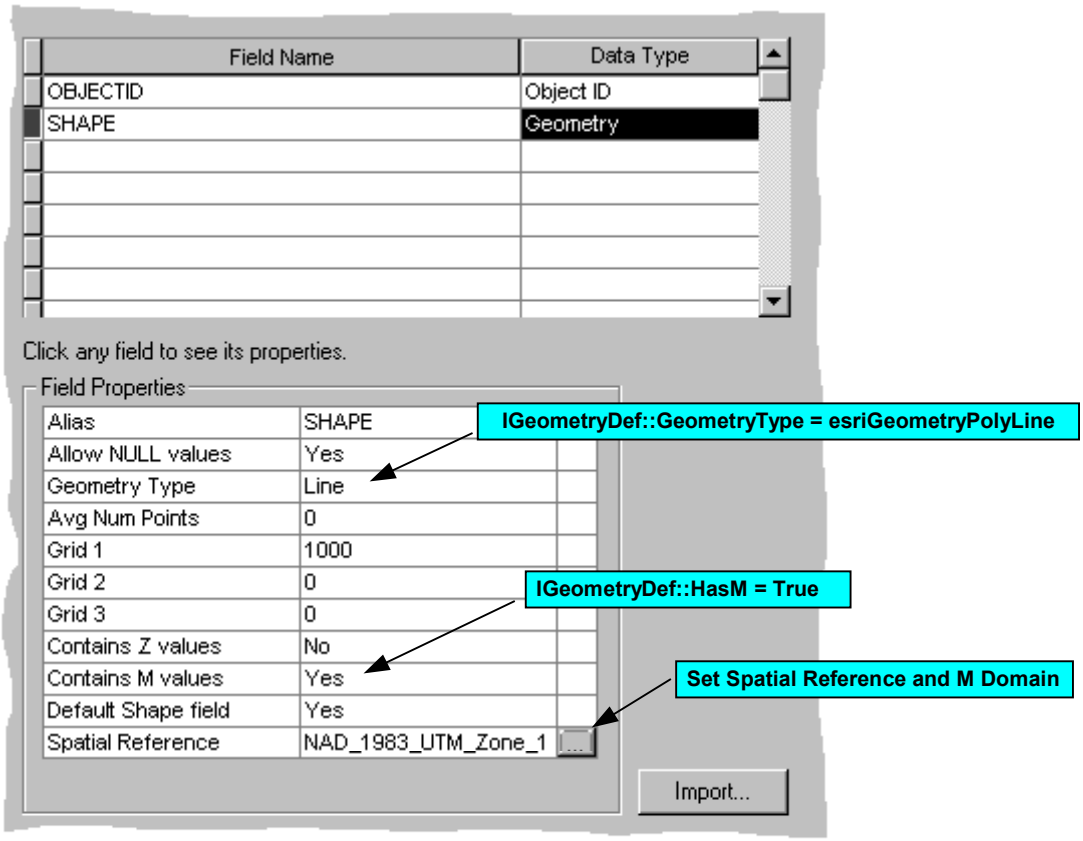
When creating a shapefile that will store routes, the type of features must be polyline (IGeometryDef::GeometryType = esriGeometryPolyline), and the features must be able to store measures (IGeometryDef::HasM = True). These properties cannot be modified after the shapefile has been created. The shapefile's coordinate system can be specified at creation time, or it can be defined later. If the coordinate system is not defined, it will be classified as "Unknown."



Creating a new shapefile requires that the geometry field's properties be set.

J-8673

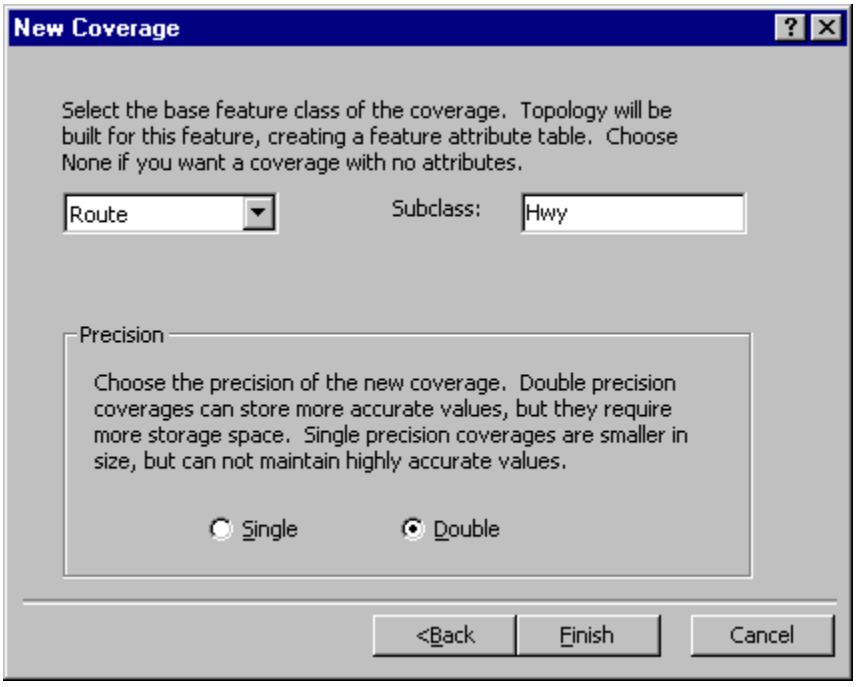
Creating a geodatabase feature class to store routes also requires defining the geometry field's properties. Note that when creating a stand-alone feature class, the spatial reference must be defined. Adding a new feature class to an existing feature data set will use the data set's spatial reference. This was defined when the data set was created. The exception is m domains; feature classes in the same feature data set can have different m domains (see Choosing an Appropriate M Domain).



Creating a geodatabase feature class requires that the geometry field's properties be set (and the spatial reference in the case of stand-alone feature classes).

Creating a new coverage with a route system in ArcCatalog does not require that the geometry field's properties be set. Note, however, that it is only possible to create coverages in ArcInfo™ and ArcEditor™ software. This functionality is not available in ArcView® software.

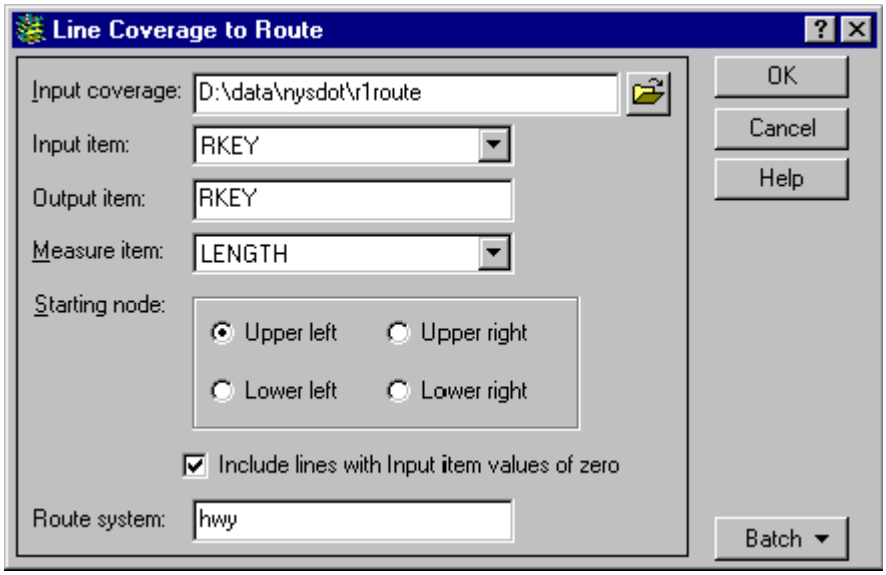
J-8673



When creating a coverage in ArcCatalog, specifying that the new feature class will be of type Route is all that is required.

ArcInfo Workstation also provides a complete set of commands for creating (and appending to) route systems within existing line coverages. Although a discussion of these commands is beyond the scope of this document, be aware of commands such as ARCROUTE, ARCSECTION, and MEASUREROUTE (see ArcDoc™ software for more details). Note that ArcToolbox™ software exposes the ARCROUTE command through its Line Coverage to Route tool.

J-8673



The ArcToolbox Line Coverage to Route tool creates a route system from lines or appends lines to the specified, existing route system.

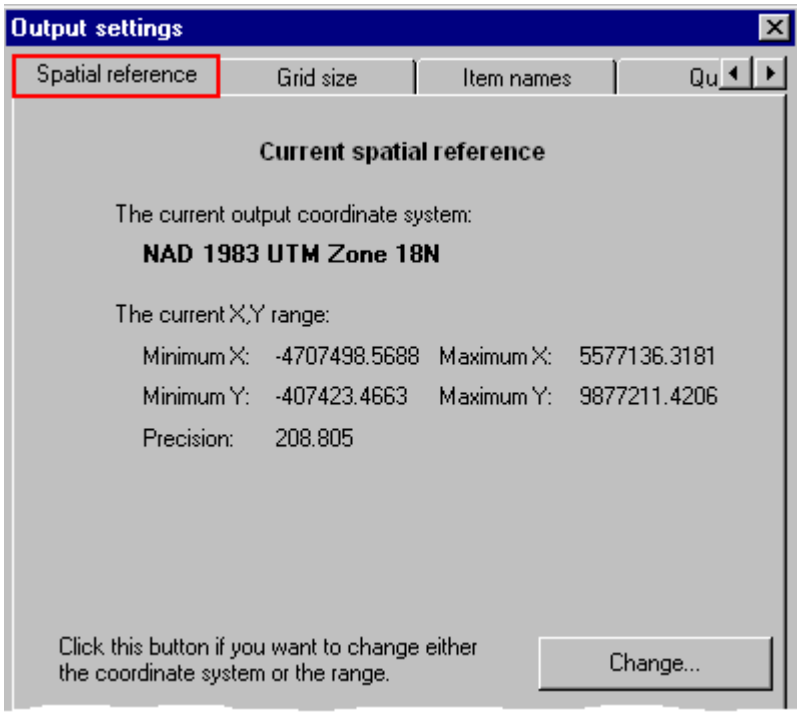
Converting Existing Route Data

The ArcToolbox/ArcCatalog data conversion tools can be used to convert existing route data between storage formats. These conversions automatically preserve the m values. Note, however, that some conversion tools are not available in ArcView.

From/To	Coverage	Shapefile	Geodatabase
Coverage	**N/A**	**Yes**	**Yes**
Shapefile	**Yes**	**N/A**	**Yes**
Geodatabase	**Yes**	**Yes**	**N/A**

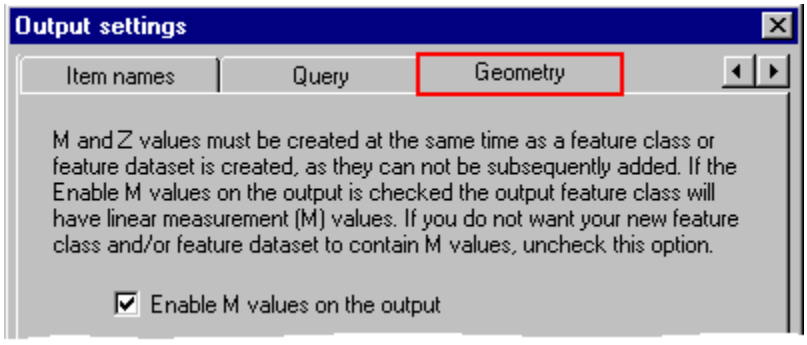
Converting between different storage formats preserves m values.

When converting an existing route feature class to a geodatabase, a default m domain will be chosen. An m domain, however, should always be explicitly set. This is because there is no way for the tools to know what units the route measures are or what precision is needed (see Choosing an Appropriate M Domain).



The Change button from the Spatial reference tab allows the m domain to be set.

Once a feature class is created, it is not possible to go back and tell it that it will start having the ability to store m values. If the need arises, the data will need to be re-created with the appropriate properties set. Fortunately, ArcToolbox data conversion tools expose the ability to "turn on" the m values when creating geodatabase feature classes. Note that if m values on the output are enabled, the m values for each feature's geometry will not be set. They will be NaN. They can be set at some point in the future.



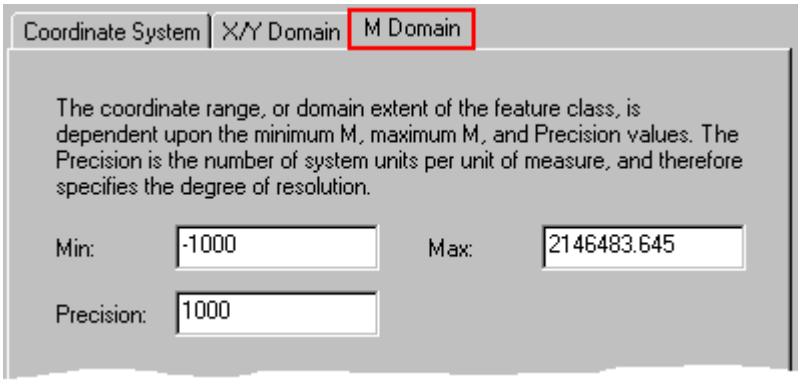
The Geometry tab can be used to enable m values on the output.

Note: If m values are enabled on the output, it is important to not forget to go back to the Spatial reference tab and use the Change button to set an appropriate m domain (see above).

J-8673

Choosing an Appropriate M Domain

Whether creating new route feature classes or converting existing nongeodatabase route feature classes to a geodatabase, one important goal is preserving measure accuracy. When data is stored in a geodatabase, all numeric values are converted to integers. Because of this, converting data to and from integer space can sacrifice measure accuracy if an inappropriate m domain precision value is chosen.

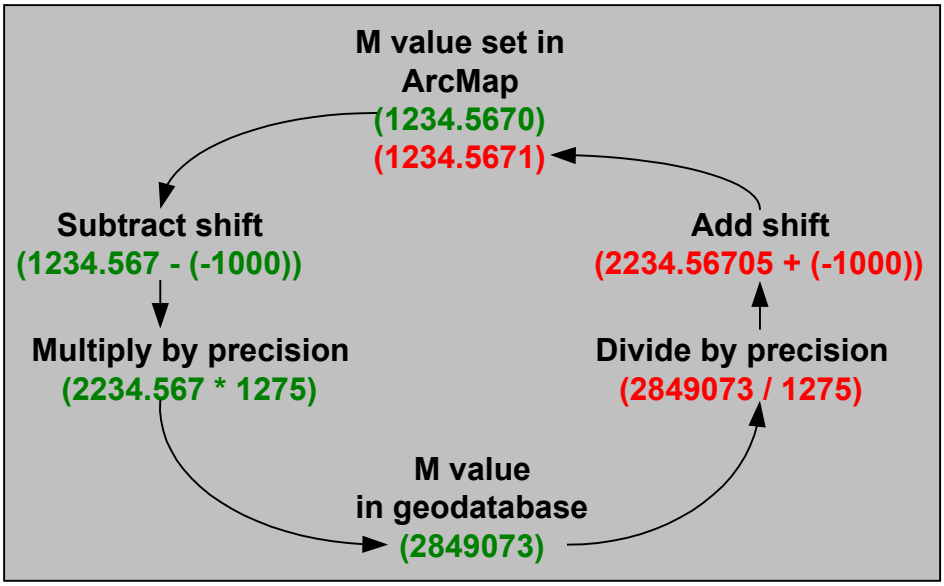


Always set an appropriate m domain when creating or converting route data.

When setting an m domain, only the minimum value and the precision need to be set. Setting these two values will automatically determine the maximum m value.

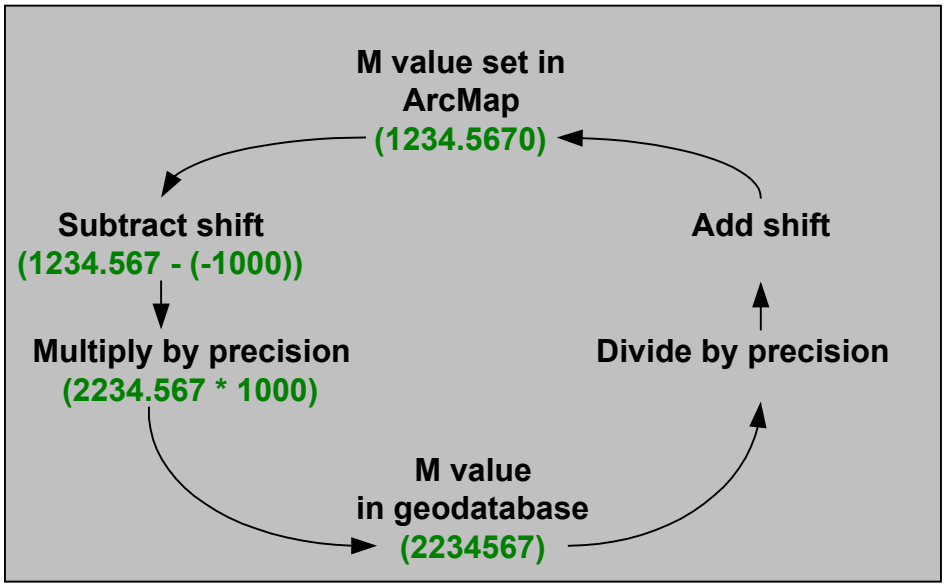
A good rule of thumb is to always choose a precision value that is a multiple of 10. For example, suppose ArcCatalog was used to create a feature class where the m domain had a minimum m value of -1,000 (some arbitrary offset) and a precision of 1,275 (i.e., not a multiple of 10). Then, in ArcMap, a new route feature was digitized, and its m values were set to range between 0 and 1234.5670 (these represent exact measurements taken in the field). The following diagram illustrates what would happen to the maximum m value when it is stored in a geodatabase and then subsequently retrieved back. Note that the same conversion would happen to every m value on the route during the round-trip.

J-8673



An inappropriate m precision can cause loss of accuracy during a round-trip to the geodatabase.

Conversely, if m the precision was set to 1,000, no accuracy would be lost during the round-trip to the geodatabase and back.



An appropriate m precision avoids loss of accuracy during a round-trip to the geodatabase.

Choosing the appropriate storage units (which will help determine the precision value to use) is another important consideration when setting the m domain. The storage units are based on the actual accuracy of the data. One alternative is to use an order of magnitude

greater than the accuracy of the data to allow for the future storage of more accurate data. Be advised, however, that there is a trade-off between accuracy and the range of measure in which values can be stored.

Routes measure units may be feet, meters, miles, and so forth. This has nothing to do with the accuracy at which the data was collected and stored. For example, data may be accurate to the foot, meter, or event centimeter level.

Precision is the multiplier that scales measure units into storage units. Precision should be the route measure units divided by the storage units. For example, if the route measure units are meters and there is centimeter accuracy, a precision value of 100 should be used. Further, if the route measure units are miles and there is accuracy to the foot, the formula indicates a precision of 5,280.

Plugging a precision of 5,280 into the diagrams above demonstrates that what is put in (1234.5670) is not exactly what is retrieved. Events will still be located at the correct spot (i.e., positional accuracy is maintained). Things such as calculating the maximum m on such a route, however, will give results (e.g., 1234.567045…) that might not be expected. For this reason, considering a strategy of using a precision value that is both a multiple of 10 and a power of 10 (10, 100, 1,000, etc.) might be in order. For this particular case, a precision value of 10,000 (the first power of 10 larger than 5,280) would be appropriate.

Editing Routes in ArcMap—What Happens to M Values

Because routes are simply features that exist in feature classes, they can be edited in ArcMap. Editing in ArcMap is fully documented in the *Editing in ArcMap* book. The purpose here is to explain what happens to measure values on polylines when various editing operations are performed.

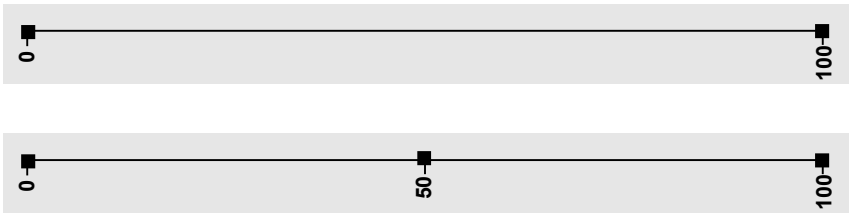
Note: Editing coverage route systems is a complex subject. Certain editing operations are not possible for specific reasons. A full explanation of editing coverages in ArcMap can be found in the *Guide to Editing Coverages in ArcMap* technical paper, which is available at http://arconline.esri.com/arconline/technicalpapers. cfm?PID=1.

ArcMap follows four basic rules when dealing with the m values on polylines:

1. Never set the m values on newly created features (they will be NaN).

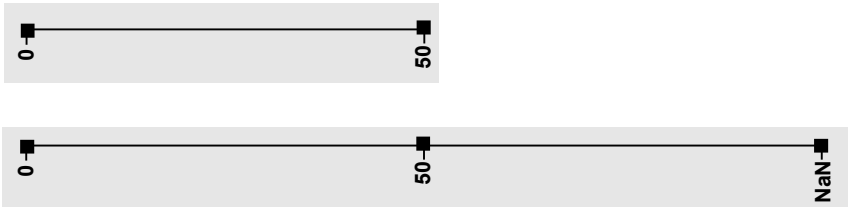
2. Edit operations that insert a new vertex between two existing vertices with known m values will interpolate the m value at the new vertex. The interpolated value is based on the difference between m values at the vertices on either side of the new vertex, divided by the length, multiplied by the distance to the previous vertex. Interpolation can happen on edit operations such as splitting, dividing, insert vertex, trimming, and reshaping. For example, if a new vertex was inserted halfway along the following two-point polyline, which is 20 meters long, the measure value at the new vertex would be $m = ((100 - 0) / 20) * 10$, which is 50.

J-8673



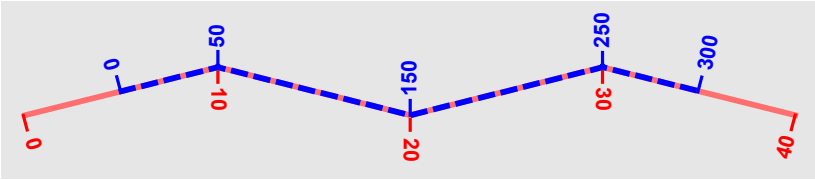
New vertices between existing ones have their m value interpolated.

3. Edit operations that create a new vertex outside of the existing vertices will not have the m value at the new vertex extrapolated. Creating new vertices outside of existing ones can happen when extending or modifying a polyline. For example, if the following two-point polyline was extended, the new vertex will have an NaN m value.

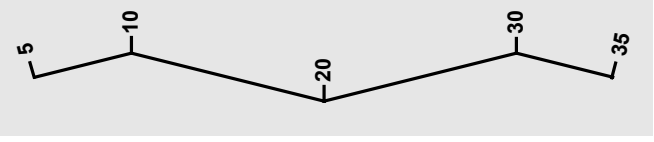


New vertices outside of existing ones do not have their m values extrapolated.

4. When performing topological operations, the m values from the primary geometry take precedence (see Appendix C—Topological Operators and Polylines with M Values for more information). The primary geometry is determined to be the one that is selected first. For example, there are two polylines that overlap one another (solid red and blue dots, respectively). The red polyline's measures range from 0 to 40. The blue polyline's measures range from 0 to 300. Intersecting these two polylines creates two possible outcomes.

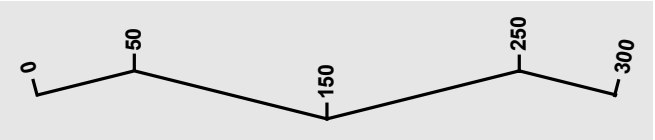


The Two-Input Polylines



The Result of Intersecting Red and Blue Polylines if Red Was Selected First

J-8673



The Result of Intersecting Red and Blue Polylines if Blue Was Selected First

M Values and the Geometry API

The Geometry Object Model provides many methods and properties for dealing with the measure values on polylines (for a more detailed discussion on how to work with polylines, see Chapter 9, "Shaping Features with Geometry," of *Exploring ArcObjects*).

Before getting started on the interfaces that manipulate m values, one important point to note is that when working with polylines, it is important to ensure that it is topologically correct before it is used in geometric operations. A topologically correct polyline is known as "simple." For polylines with m values, there is something called "network simple." To ensure a polyline is network simple, the IPolyline::SimplifyNetwork method is used. SimplifyNetwork will do the following:

- Remove empty or zero length *Segments* and empty *Paths.*
- Ensure *Segment* orientation is consistent with order (the *ToPoint* of a segment is equal to the *FromPoint* of the succeeding *Segment*).
- Create a new path for discontiguous *Segments* or *Segments* with different attributes, ensuring each *Path* is contiguous.
- Merge parts where exactly two *Paths* share an endpoint.
- For two contiguous *Segments* in a *Path* with a common endpoint *(FromPoint* or *ToPoint*), if one endpoint has non-NaN z-, m-, or ID attributes and the other has NaN attributes, these values will be copied to the attributes of the other endpoint. If the endpoints have unequal non-NaN values, the path will be split into two paths at this point.

There are four interfaces that are of particular interest for working with and manipulating the measures on polylines: IMAware, IMCollection, IMSegmentation, and IMSegmentation2.

The IMAware interface determines whether or not a polyline is aware that it has measures. All route features should be based on polylines that are MAware. To make a polyline MAware, simply set the MAware property to True. By default, the m values of a shape will be NaN. The MSimple property returns True if a polyline contains no NaN m values. To set all m values of a polyline to NaN, call DropMs.

The IMCollection interface is used to change existing measure values on a polyline. The MultiplyMs and OffsetMs methods will update all the measure attributes for a polyline and should only be used if every measure has been set (i.e., there are no NaNs). Check

the IMAware::MSimple property or use the IMSegmentation methods to fill in any missing measure values.

The IMSegmentation interface provides methods designed to work with the dynamic segmentation functionality in ArcObjects. The CalculateNonSimpleMs method will interpolate or extrapolate missing (NaN) measure values based on the existing values. Use the ExtrapolateMs or InterpolateMsBetween methods to set m attributes for only the specified vertices of a polyline. InsertMAtDistance will update a single vertex's m value, or if the specified distance does not fall on a vertex, a new vertex will be inserted into the polyline at that location.

IMSegmentation also provides methods to query a measured polyline based on its m values. For example, GetSubCurveBetweenMs and GetPointsAtM can be used to find geometries along a measured polyline. Note that if the m attributes do not monotonically increase along the line (check the MMonotonic property), the result of GetSubcurveBetweenMs may have more than one part or the result of GetPointsAtM will have more than one point.

The CalculateNonSimpleMs, SetAndInterpolateMsBetween, and SetMsAsDistance methods all ensure a Polyline is MSimple.

The IMSegmentation2 interface (which inherits from IMSegmentation) provides access to members that provide additional support for linear referencing operations on polylines. They offer extended ways to interpolate and update the m values on a polyline by cumulative distance and also by existing m values.

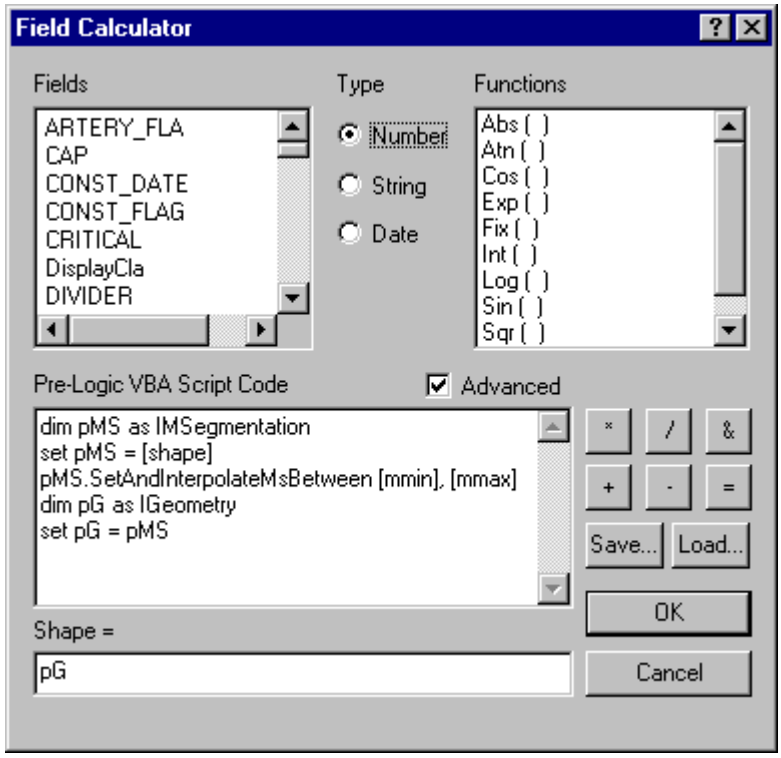
Refer to Appendix B—Manipulating M Values via the API for more detailed information on working with measure values. Refer also to the dynamic segmentation samples in ArcObjects Developer Help for complete code samples that outline how to manipulate measure values with code.

Working with M Values in ArcMap

ArcMap allows tables to be edited in many ways. The functionality includes the ability to add new records and delete existing ones, modify a record's values, or copy and paste between records. Additionally, the field calculator can be used to perform mathematical calculations to set a field value for any or all records.

One very useful capability of the field calculator is that it can be used to do things such as manipulate the measure values on routes. This is because the field calculator facilitates advanced calculations using Visual Basic for Applications statements that process the data before calculations are made on the selected field. Building on the contents of the M Values and the Geometry API section above, taking advantage of this functionality is straightforward.

J-8673



The ArcMap field calculator can be used to manipulate the measure values on a route. Here, the measures will range between the values stored in a field called MMIN and another called MMAX. All measure values between these two values will be interpolated.

Refer to "fields in tables, calculating values" in ArcGIS Desktop Help for more information on calculating field values.

Analyzing Events

Spatial Analysis of Events

In ArcMap, geographic data is represented on a map as a layer. A layer based on a feature class (coverage, shapefile, or geodatabase) is called a feature layer. The Add Route Events dialog is used to add a feature layer to an ArcMap document based on a route event table. Once added, a feature layer based on a route event table behaves just like any other feature layer. (This is because a dynamic feature class called a RouteEventSource is created behind the scenes.) It can be queried according to feature locations and attributes to solve problems. Further, new spatial relationships can be discovered by asking questions such as Where is …? Where is the closest? and What intersects? Refer to ArcGIS Desktop Help to find out how ArcMap can be used to find answers to these types of questions.

Event Overlay

Overlaying events is another method to analyze events. This process combines two or more event tables to produce a single output event table.

The output event table can contain either the intersection or union of the input events. The union of the input events splits all linear events at their intersections and writes them

to the new event table. The intersection of the input event tables writes only overlapping events to the output event table.

Line-on-line, point-on-line, and point-on-point overlays can be performed at ArcGIS 8.1 using the dynamic segmentation objects outlined on the geodatabase Object Model (Supplemental) diagram. Refer to Appendix D for how the various overlays work. Refer to ArcObjects Developer Help for complete code samples.

Line-on-Line Overlay

A line-on-line overlay involves the overlay of two or more linear event tables to produce a single linear event table.

The following example demonstrates how to find the union of a linear event table, which describes pavement cracking, and another linear event table, which contains resurfacing dates. The results can be used to find the characteristics of the oldest paved sections.

RKEY	FMP	TMP	Perc_Crack
101	23.5	44.2	50
101	44.2	84.7	30
101	84.7	167.4	80
101	167.4	182.8	95
101	182.8	209.5	45

An Event Table Showing the Amount of Pavement Cracking on Route 101

RKEY	FMP	TMP	Resurfaced
101	3.2	21.1	2/5/85
101	21.1	95.5	9/3/87
101	95.5	190	4/28/61
101	190	209.5	1/21/74

An Event Table Containing Resurfacing Dates for Portions of Route 101

RKEY*	FMP	TMP	Perc_Crack	Resurfaced
101	3.2	21.1	0	2/5/85
101	23.5	44.2	50	9/3/87
101	44.2	84.7	30	9/3/87
101	84.7	95.5	80	9/3/87
101	21.1	23.5	0	9/3/87
101	95.5	167.4	80	4/28/61
101	167.4	182.8	95	4/28/61
101	182.8	190	45	4/28/61
101	190	209.5	45	1/21/74

The Union of the Cracking and Resurfacing Event Tables

In this case, overlaying the cracking and resurfacing event tables shows that the stretch of Route 101 from measure 167.4 to 182.8 is the most cracked and has the oldest surface.

Point-on-Line Overlay

A point-on-line overlay involves the overlay of one or more point event tables with a linear event table to produce a single point event table.

The following example demonstrates how to find the intersection of a point event table called ACCIDENTS, which contains highway accident data, and a linear event table called PAV_CRACK, which describes pavement cracking. The results can be used to analyze the pavement characteristics of accident locations.

RKEY	Location	Injured	Alcohol
101	25.9	2	0
101	95.6	1	1
101	172.3	1	0
101	180.3	3	1

An Event Table Containing Information on Accident Locations, Number of People Injured, and Alcohol Involvement

RKEY	FMP	TMP	Perc_Crack
101	23.5	44.2	50
101	44.2	84.7	30
101	84.7	167.4	80
101	167.4	182.8	95
101	182.8	209.5	45

An Event Table with Data on the Amount of Pavement Cracking Present on Route 101

RKEY*	Location	Injured	Alcohol	Perc_Crack
101	25.9	2	0	50
101	95.6	1	1	80
101	172.3	1	0	95
101	180.3	3	1	95

The Intersection of Accident Locations and Pavement Cracking

Event Errors

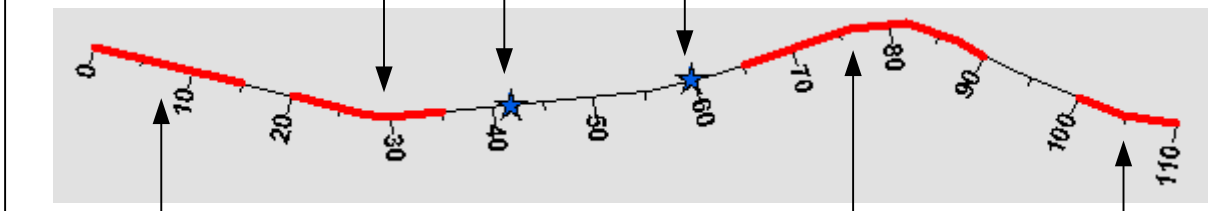
As noted, when the ArcMap Add Route Events dialog is used to create a feature layer based on route events, a dynamic feature class called a RouteEventSource is created behind the scenes. In a RouteEventSource, there is one feature for every row of the original event table. In some cases, however, the features have empty shapes. This is because there was some reason the event could not be properly located. Other times, an event can only be partially located (this happens for line events only).

The IEventSourceErrors interface on the RouteEventSource object exposes some methods that determine the locating errors of events. The following illustration shows a hypothetical route feature class containing a single route whose measures range from 0 to

J-8673

110. Both the point and line event tables that are shown have a certain type of locating errors. See the dynamic segmentation samples in ArcObjects Developer Help for a complete demonstration of how to add locating error codes to event tables.

RKEY	MILE	ErrCode
101	42	LOCATING_OK
101	60	LOCATING_OK
101	111	LOCATING_E_CANT_FIND_LOCATION
102	50	LOCATING_E_CANT_FIND_ROUTE



RKEY	FMP	TMP	ErrCode
101	-5	15	LOCATING_E_FROM_PARTIAL_MATCH
101	20	35	LOCATING_OK
101	65	90	LOCATING_OK
101	100	115	LOCATING_E_TO_PARTIAL_MATCH
101	120	135	LOCATING_E_CANT_FIND_EXTENT
102	10	50	LOCATING_E_CANT_FIND_ROUTE

Hypothetically Locating Errors for a Route Feature Class with Only One Route

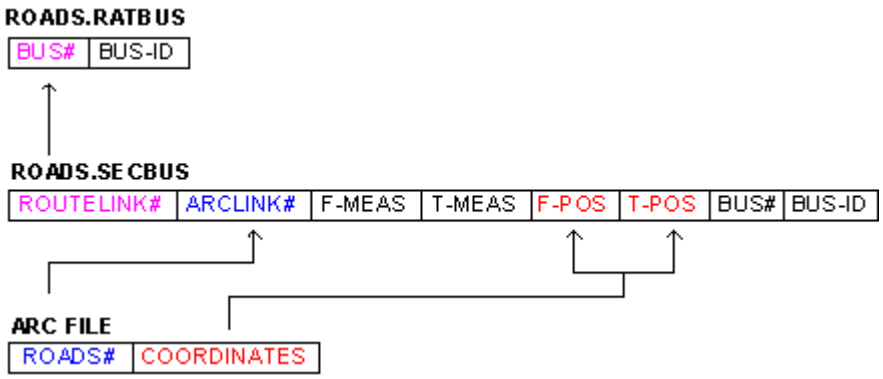
J-8673

Appendix A—Coverage Route Systems in ArcMap

The Routes and Measures section of this document outlined that a route is simply a linear feature on which attributes can be defined. Attributes can be assigned to a route because each route (a) has an identifier stored in a field and (b) has an associated measurement system.

In a coverage, however, routes are actually stored in two INFO data files. These tables are the route attribute table (.RAT), which defines a route feature class, and the section table (.SEC), which defines a section feature class. Together, these feature classes define what is called a route system.

A coverage route system must always have a name, and a single coverage can have many route systems. Each route system is dependent on the existence of an arc feature class. For example, a coverage called ROADS may have a route system called BUS containing school bus routes. The corresponding INFO data files are named ROADS.RATBUS and ROADS.SECBUS. Similarly, a route system called HAZARD has INFO files called ROADS.RATHAZARD and ROADS.SECHAZARD.



How the RAT, SEC, and ARC Files in a ROADS Coverage Fit Together to Implement a Route System Named BUS

When a route system is loaded into ArcMap, each route (one record from the .RAT file) is transformed into a calibrated polyline shape with m (measures) values. The records from the .SEC file and the arcs that make up the coverage define the geometry of the polyline and its measures.

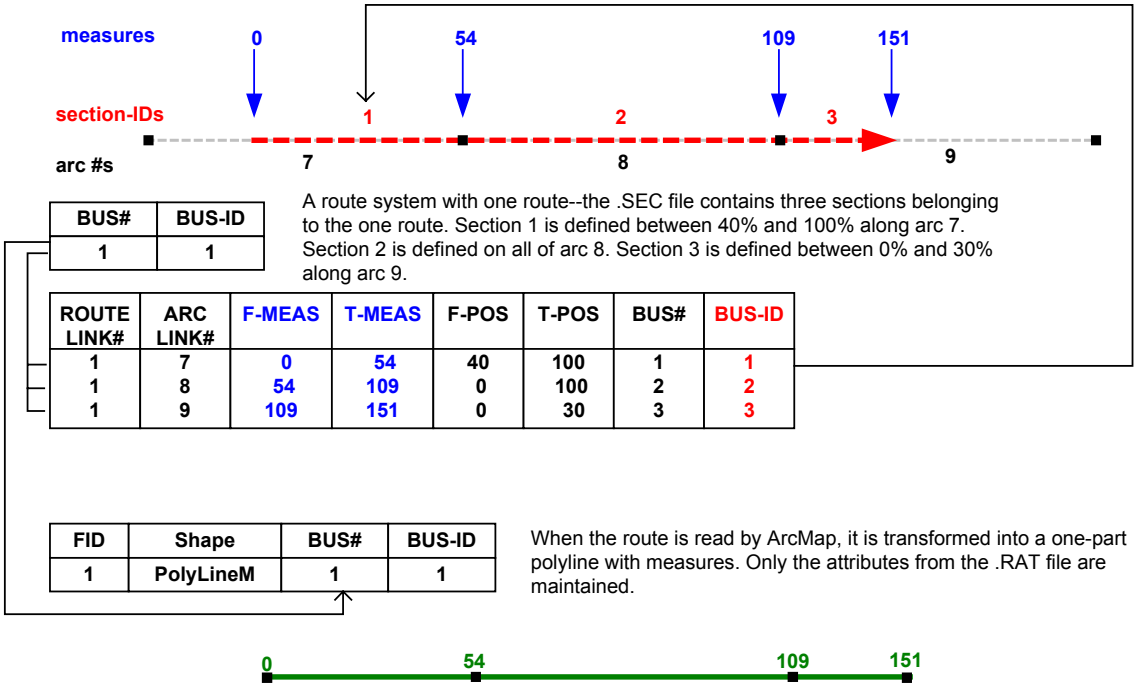
To explain how a coverage route is transformed into a polyline with measures, let us use the bus route system illustrated above. For each BUS# in the .RAT file, all of the sections from the .SEC file with the corresponding ROUTELINK# are queried. For each

J-8673

of the selected .SEC records, the ARCLINK# is used to find the geometry of the arc on which the section is based. Then, the F-POS and T-POS values (the from- and to-position, measured in percent) are used to construct a polyline of appropriate length. Last, F-MEAS and T-MEAS (from- and to-measure) are queried to assign the m values to the first and last vertex of the polyline. Once these values are assigned, interpolation is used to assign the m values to the intermediate vertices.

As each new part of the polyline is constructed (e.g., each section's geometry), a check is done to see if the x,y,m coordinates of the first vertex of the current section have the same x,y,m values of the last vertex of the previous section. If they are, then the two parts are merged into one. As a result, a route can be transformed into a one-part polyline with m values, or it can be transformed into a multipart polyline with m values.

One last point to note is that only the .RAT attributes are maintained when ArcMap reads a route system. The .SEC attributes are used only in the construction of the polyline.



Coverage routes are read in ArcMap as polylines with measures.

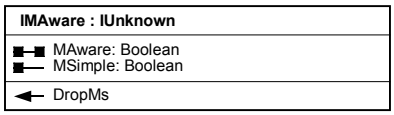
J-8673

Appendix B—Manipulating M Values via the API

This section will demonstrate how some of the interfaces in the Geometry Object Model can be used to manipulate the m values on polylines. This section assumes a good working knowledge of geometry and the Geometry Object Model. Refer to Chapter 6, "The Shape of Features," in *Modeling Our World* and Chapter 9, "Shaping Features with Geometry," in *Exploring ArcObjects* for more information about geometry objects.

IMAware

The **IMAware** interface controls whether or not the geometry object recognizes that it should use the m values when performing operations.



The IMAware Interface

MAware returns or sets the m awareness state of a geometry object. If **MAware** is TRUE, then the object will recognize that it has m attributes and perform operations on them as necessary. If **MAware** is FALSE, the object will ignore m values while performing operations. Some geometry methods using m values require that an object is aware of m values before they can be called. Others function regardless of m awareness.

```
Public Sub MakeMAware(pPl As IPolyline)
  Dim pMA As IMAware
  Set pMA = pPl
  If Not pMA.MAware Then pMA.MAware = True
End Sub
```

DropMs resets all of the current m values held by the geometry object back to NaN. **MAware** does not change.

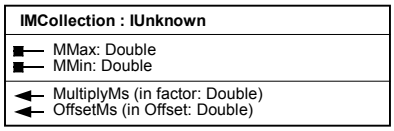
```
Public Sub DropMs(pPl As IPolyline)
  Dim pMA As IMAware
  Set pMA = pPl
  pMA.DropMs
End Sub
```

MSimple is TRUE if none of the m values held by a geometry object are NaN. If any of the m values within the object are NaN, **MSimple** is FALSE.

```
Public Function IsMSimple(pPl As IPolyline) As Boolean
  Dim pMA As IMAware
  Set pMA = pPl
  If pMA.MAware Then
    IsMSimple = pMA.MSimple
  Else
    Err.Raise vbObjectError + 11282000, "IsMSimple", "Not m Aware"
  End If
End Function
```

IMCollection

IMCollection supports operations performed on geometry objects that have m values. For **IMCollection** operations to be performed, the object must be **MAware** (and should have non-NaN m values).



The IMCollection Interface

MMin/MMax returns the minimum/maximum m value contained within an **MAware** geometry object. In the case of a polyline, this may or may not correspond to one of the end vertices. This is because there is no requirement that m values monotonically increase or decrease along a polyline (check **IMSegmentation::MMonotonic** property). Note that calling **MMin/MMax** on a geometry object that is **MAware** but has no m values set will return NaN.

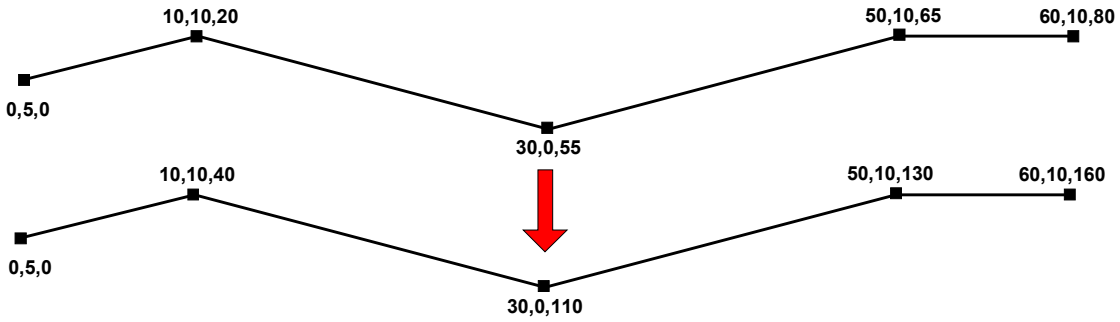
```
Public Function GetMMin(pPl As IPolyline) As Double
  Dim pMA As IMAware
  Dim pMColl As IMCollection
  Set pMA = pPl
  If pMA.MAware Then
    Set pMColl = pPl
    GetMMin = pMColl.MMin
  Else
    Err.Raise vbObjectError + 11282000, "GetMMin", "Not m Aware"
  End If
End Function
```

J-8673

MMin/MMax

The **MultiplyMs** method will update all the measure values for a polyline and should only be used if every measure has been set (i.e., there are no NaNs). Check the **IMAware::MSimple** property or use the **IMSegmentation** methods to fill in any missing measure values. **MultiplyMs** is analogous to scaling spatial coordinates and is useful for things such as m unit conversion (e.g., from miles to meters).

```
Public Sub MultiplyMs(pPl As IPolyline, dFactor As Double)
  Dim pMA As IMAware
  Dim pMColl As IMCollection
  Set pMA = pPl
  Set pMColl = pMA
  If pMA.MAware Then
    Set pMColl = pPl
    pMColl.MultiplyMs dFactor
  Else
    Err.Raise vbObjectError + 11282000, "MultiplyMs", "Not m Aware"
  End If
End Sub
```



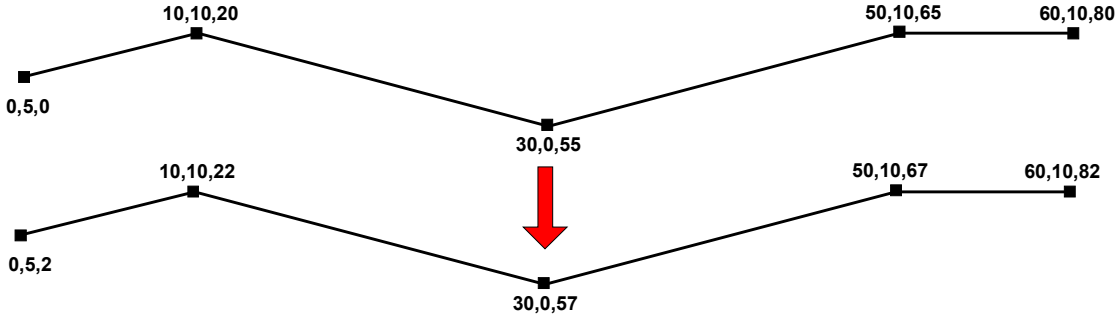
MultiplyMs(2)

The **OffsetMs** method will update all the measure attributes for a polyline and should only be used if every measure has been set (i.e., there are no NaNs). Check the **IMAware::MSimple** property or use the **IMSegmentation** methods to fill in any missing measure values. **OffsetMs** is analogous to moving spatial coordinates.

```
Public Sub OffsetMs(pPl As IPolyline, dFactor As Double)
  Dim pMA As IMAware
```

J-8673

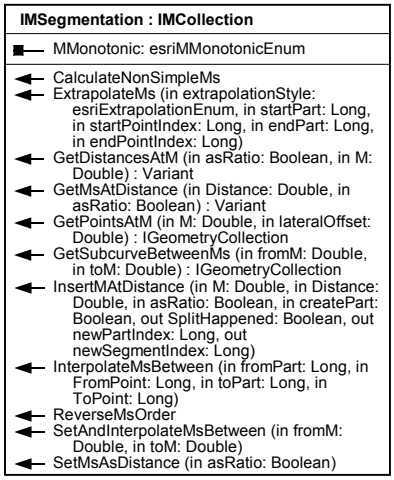
```
  Dim pMColl As IMCollection
  Set pMA = pPl
  Set pMColl = pMA
  If pMA.MAware Then
    Set pMColl = pPl
    pMColl.OffsetMs dFactor
  Else
    Err.Raise vbObjectError + 11282000, "OffsetMs", "Not m Aware"
  End If
End Sub
```



OffsetMs(2)

IMSegmentation

The **IMSegmentation** interface provides methods and properties designed to set an m-based linear reference system on a polyline. In other words, **IMSegmentation** was designed to transform a polyline into a route.

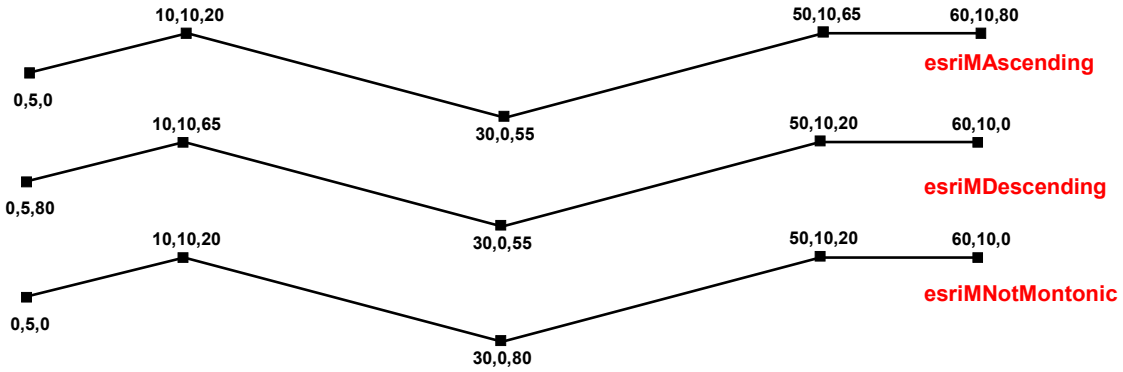


The IMSegmentation Interface

The **MMonotonic** property returns an **esriMMonotonicEnum** indicating whether or not the m values are monotonic as well as whether they are ascending or descending. The m values are esriMAscending if the m values are ordered such that M1 < M2 < M3 < . . .

J-8673

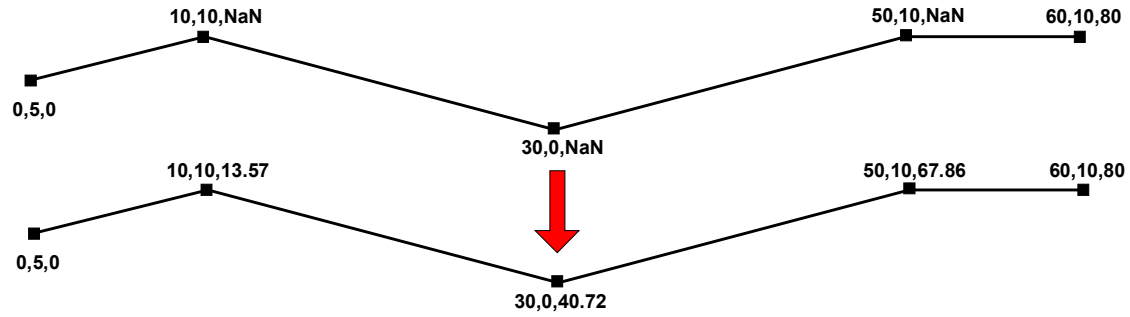
< Mn. The m values are esriMDescending if the m values are ordered such that M1 > M2 > M3 > . . . > Mn.



MMonotonic

The **CalculateNonSimpleMs, SetAndInterpolateMsBetween,** and **SetMAsDistance** methods all ensure a Polyline is **MSimple.**

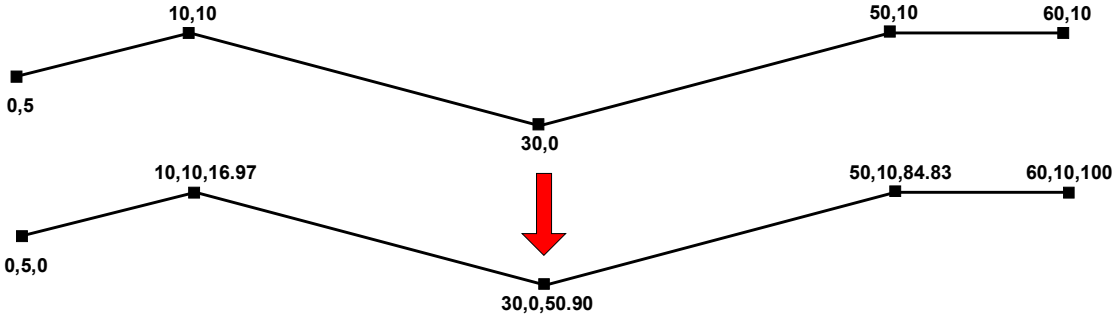
CalculateNonSimpleMs redefines the nonsimple m values to be values obtained from interpolation of surrounding defined m values or extrapolation of m values.



CalculateNonSimpleMs

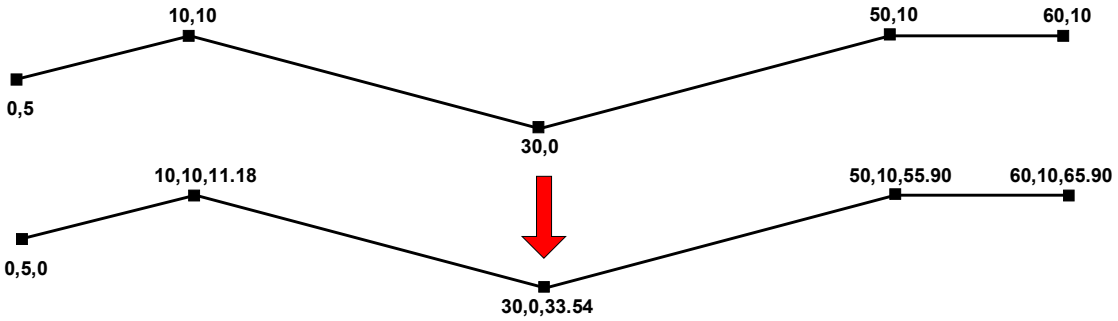
SetAndInterpolateMsBetween sets the m values at the beginning and the end of the geometry and interpolates the m values between these values.

J-8673



SetAndInterpolateMsBetween 0,100

SetMAsDistance sets the m values to the cumulative length from the origin of the geometry. That is, the m values will increase in the digitized direction of the polyline.



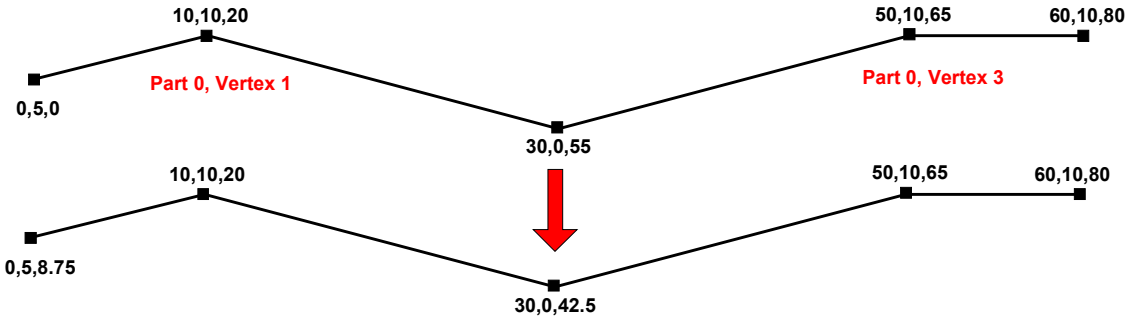
SetMAsDistance

Use the **InterpolateMsBetween** and **ExtrapolateMs** methods to set m values for specific vertices of a polyline.

InterpolateMsBetween generates m values by linear interpolation of segment distances for all vertices in the specified range (e.g., start+1, end-1).

```
Dim pMSeg As IMSegmentation
Set pMSeg = pPl   '+++ pointer to IPolyline interface
pMSeg.InterpolateMsBetween 0, 1, 0, 3
```

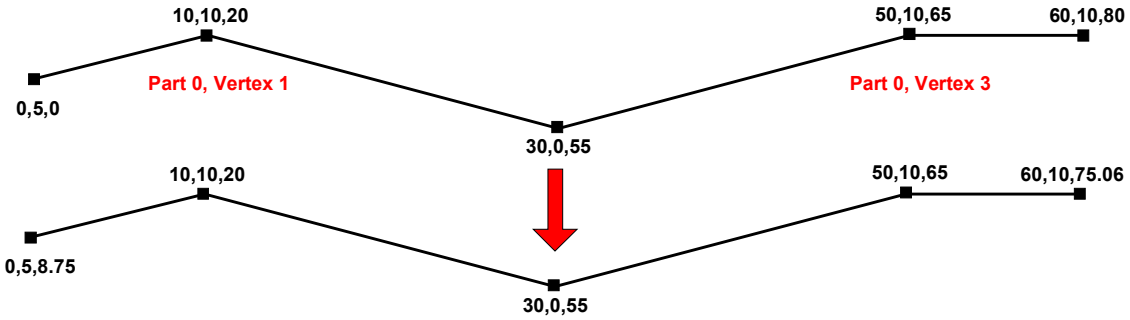
J-8673



InterpolateMsBetween

ExtrapolateMs recalculates the m values at one or both ends of a polyline based on the m interval between the from-index and the to-index.

```
Dim pMSeg As IMSegmentation
Set pMSeg = pPl   '+++ pointer to IPolyline interface
pMSeg.ExtrapolateMs esriExtrapolateBoth, 0, 1, 0, 3
```

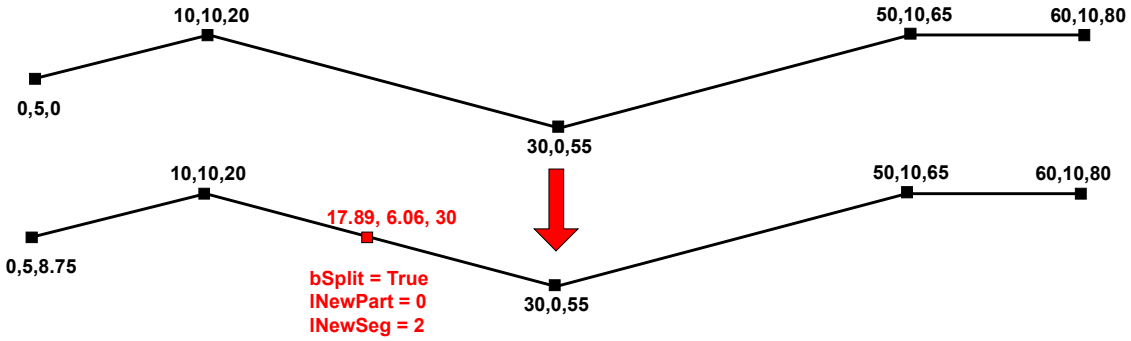


ExtrapolateMs

InsertMAtDistance will update a single vertex's m value. If the specified distance does not fall on a vertex, a new vertex will be inserted into the polyline at that location. **InsertMAtDistance** also sets some variables to (a) indicate whether a split happened and (b) identify the new part and segment indices.

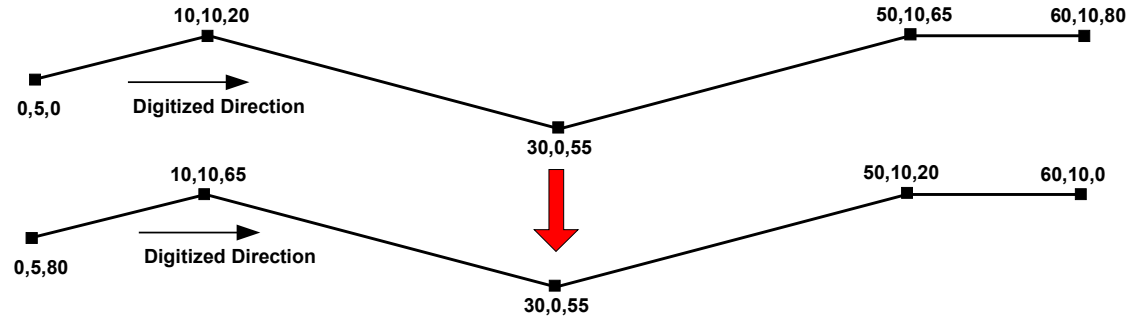
```
Dim pMSeg As IMSegmentation
Set pMSeg = pPl   '+++ pointer to IPolyline interface
Dim bSplit As Boolean, lNewPart As Long, lNewSeg As Long
pMSeg.InsertMAtDistance 30, 20, False, False, bSplit, lNewPart, lNewSeg
```

J-8673



InsertMAtDistance

ReverseMsOrder simply reverses the order of the m values along a polyline. Note that the m values are simply reversed. Any length/measure ratios that were present before **ReverseMsOrder** is called are not preserved. This method differs from **ICurve::ReverseOrientation,** which reverses the parameterization of the curve ("from" point becomes "to" point, first segment becomes last segment, etc.). That is, **ReverseMsOrder** differs from **ReverseOrientation** in that **ReverseMsOrder** does not change the digitized direction of a polyline.

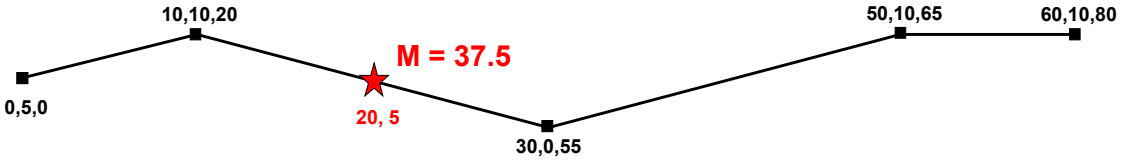


ReverseMsOrder

GetMsAtDistance returns m values at the distance along the geometry. An array of one or two m values is returned. Two m values can be returned if the given distance is exactly at the beginning or ending of a part. The following code uses GetMsAtDistance to determine the m value(s) of a point (20, 5) along a route. In this case, the m value is 37.5.

```
Public Function GetMsAtPoint(pPl As IPolyline, pPt As IPoint) As Variant
  Dim pMA As IMAware
  Dim pCurve As ICurve
  Dim pTmpPt As IPoint, dAlong As Double, dFrom As Double, bRight As Boolean
  Dim pMSeg As IMSegmentation
  Dim Ms
  Set pMA = pPl
  If pMA.MAware Then
    Set pCurve = pPl
```

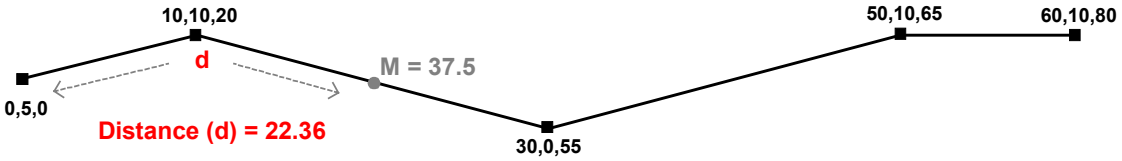
```
    pCurve.QueryPointAndDistance esriNoExtension, pPt, False, pTmpPt, dAlong,
dFrom, bRight
    Set pMSeg = pCurve
    Ms = pMSeg.GetMsAtDistance(dAlong, False)
    GetMsAtPoint = Ms
  Else
    Err.Raise vbObjectError + 11282000, "GetMsAtPoint", "Not m Aware"
  End If
End Function
```



GetMsAtDistance

GetDistancesAtM returns an array of distances along the polyline at which the specified m is located. If the polyline's m values are monotonic (check the **MMonotonic** property), then the array will only have one element. When applied to the illustrated route, the following code demonstrates that at measure 37.5 the distance is 22.36.

```
Public Function GetDistancesAtM(pPl As IPolyline, m As Double) As Variant
  Dim pMA As IMAware
  Dim pMSeg As IMSegmentation
  Set pMA = pPl
  If pMA.MAware Then
    Set pMSeg = pPl
    GetDistancesAtM = pMSeg.GetDistancesAtM(False, m)
  Else
    Err.Raise vbObjectError + 11282000, "GetDistancesAtPoint", "Not m Aware"
  End If
End Function
```

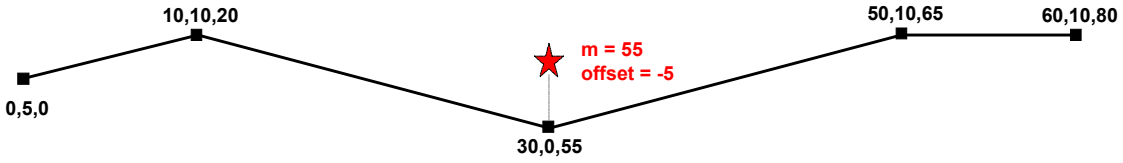


GetDistancesAtM

GetPointsAtM returns a MultiPoint geometry corresponding to the locations along the geometry where the specified m occurs. Note that if the m attributes do not monotonically increase along the line (check the **MMonotonic** property), the result of GetPointsAtM can have more than one point. The following code demonstrates that GetPointsAtM (measure 55, offset –5) will return a MultiPoint that contains one point.

J-8673

```
Public Function GetPointLocation(pPl As IPolyline, dMeas As Double, dOffset As
Double) As IMultipoint
  Dim pMA As IMAware
  Dim pMSeg As IMSegmentation
  Set pMA = pPl
  If pMA.MAware Then
    Set pMSeg = pPl
    Set GetPointLocation = pMSeg.GetPointsAtM(dMeas, dOffset)
  Else
    Err.Raise vbObjectError + 11282000, "GetPointLocation", "Not m Aware"
  End If
End Function
```

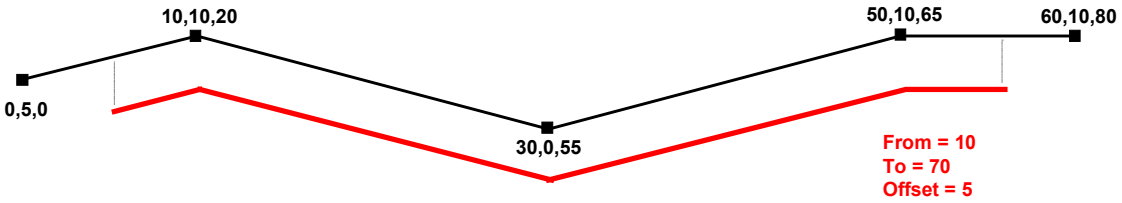


GetPointsAtM

GetSubcurveBetweenMs returns a polyline geometry corresponding to the subcurve(s) between the from- and to-m values. Note that if the m attributes do not monotonically increase along the line (check the **MMonotonic** property), the result of GetSubcurveBetweenMs can have more than one part. The following code demonstrates that GetSubcurveBetweenMs (from = 10, to = 70) will return a polyline that contains one part. Also note that GetSubcurveBetweenMs does not have an offset argument. The offsetting logic is additional.

```
Public Function GetLineLocation(pPl As IPolyline, dFrom As Double, _
    dTo As Double, dOffset As Double) As IPolyline
  Dim pMA As IMAware
  Dim pMSeg As IMSegmentation
  Dim pCCurve As IConstructCurve
  Set pMA = pPl
  If pMA.MAware Then
    Set pMSeg = pPl
    Set pPl = pMSeg.GetSubcurveBetweenMs(dFrom, dTo)
    If dOffset <> 0 Then
      Set pCCurve = New Polyline
      pCCurve.ConstructOffset pPl, dOffset
      Set GetLineLocation = pCCurve
    Else
      Set GetLineLocation = pPl
    End If
  Else
    Err.Raise vbObjectError + 11282000, "GetLineLocation", "Not m Aware"
  End If
End Function
```

J-8673



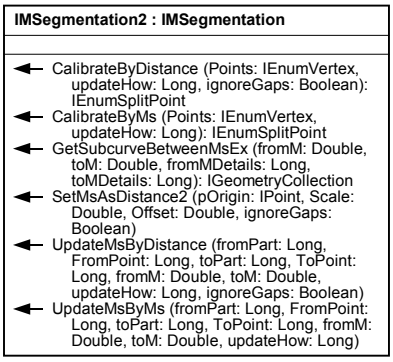
GetSubcurveBetweenMs

IMSegmentation2

The IMSegmentation2 interface was also designed to set an m-based linear reference system on a polyline. These methods offer extended ways to interpolate and update the m values on a polyline by cumulative distance and also by existing m values.

Many of the methods on this interface require an updateHow argument. This argument is given as a combination of esriGeometryUpdateMEnum values. When combining multiple values, the bitwise OR operator should always be used. This assures an error-free combination of the values (as long as the attempted combination is valid). For example, to interpolate between the input points and to extrapolate before and after the input points, use 7, which equates to

esriGeometryInterpolate OR esriGeometryExtrapolateBefore OR esriGeometryExtrapolateAfter



The IMSegmentation2 Interface

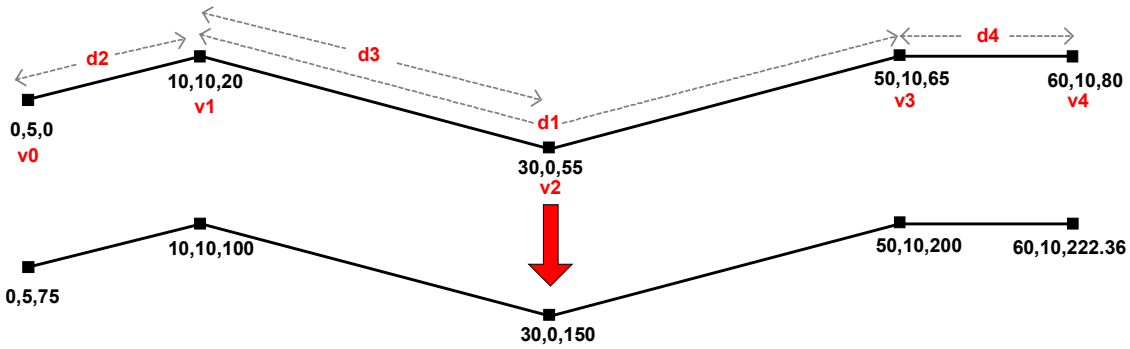
UpdateMsByDistance updates m values along the shortest path between the specified vertices. The interpolation ratio is determined by the input m values and Euclidean distance along the path between them. The update method is given as a combination of esriGeometryUpdateMEnum values.

```
Dim pGC As IGeometryCollection
Set pGC = pPl
pGC.GeometriesChanged
pPl.SimplifyNetwork

Dim pMSeg As IMSegmentation2, lUpdateHow As Long
Set pMSeg = pPl
```

J-8673

```
lUpdateHow = esriGeometryInterpolate Or esriGeometryExtrapolateBefore _
        Or esriGeometryExtrapolateAfter
pMSeg.UpdateMsByDistance 0, 1, 0, 3, 100, 200, lUpdateHow, True
```



UpdateMsByDistance

The distance (d1) between the two specified vertices (v1 and v3) is 44.72. UpdateMsByDistance explicitly sets the m difference between these vertices to be 100 (200 – 100). The distance to measure ratio, therefore, is 0.4472 (44.72/100).

By specifying esriGeometryExtrapolateBefore, the m value at v0 is calculated as follows:

- d2 = 11.18.
- 11.18 / 0.44721 = 25
- 100 – 25 = 75

By specifying esriGeometryInterpolate, the m value at v2 is calculated as follows:

- d3 = 22.36.
- 22.36 / 0.44721 = 50
- 100 + 50 = 150

By specifying esriGeometryExtrapolateAfter, the m value at v4 is calculated as follows:

- d4 = 10
- 10 / 0.44721 = 22.36
- 200 + 22.26 = 222.36

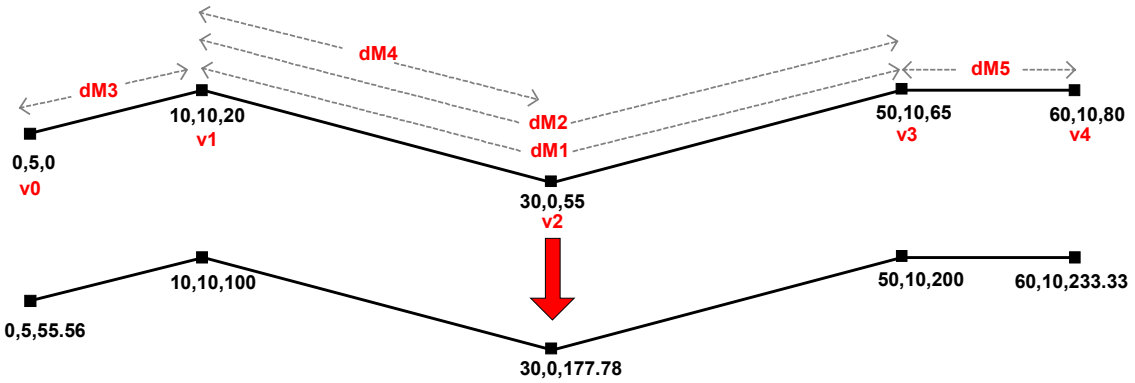
UpdateMsByMs also updates m values along the shortest path between the specified vertices. The interpolation ratio is determined by the existing m values along that path and the input m values. The update method is given as a combination of esriGeometryUpdateMEnum values.

```
Dim pGC As IGeometryCollection
Set pGC = pPl
pGC.GeometriesChanged
```

J-8673

```
pPl.SimplifyNetwork

Dim pMSeg As IMSegmentation2, lUpdateHow As Long
Set pMSeg = pPl
lUpdateHow = esriGeometryInterpolate Or esriGeometryExtrapolateBefore _
        Or esriGeometryExtrapolateAfter
        pMSeg.UpdateMsByMs 0, 1, 0, 3, 100, 200, lUpdateHow
```



UpdateMsByMs

UpdateMsByMs takes into consideration existing m values on a polyline when determining new ones. The measure distance (dM1) between the two specified vertices (v1 and v3) is 45 (65 – 20). UpdateMsByMs explicitly sets the m difference (dM2) between these vertices to be 100 (200 – 100). The new-m to old-m ratio is therefore 2.2222 (100/45).

By specifying esriGeometryExtrapolateBefore, the m value at v0 is calculated as follows:

- dM3 = 20
- 2.2222 * 20 = 44.44
- 100 – 44.4 = 55.56

By specifying esriGeometryInterpolate, the m value at v2 is calculated as follows:

- dM4 = 35
- 2.2222 * 35 = 77.78
- 100 + 77.78 = 177.78

By specifying esriGeometryExtrapolateAfter, the m value at v4 is calculated as follows:

- dM5 = 15
- 2.2222 * 15 = 33.33
- 200 + 33.33 = 233.33

CalibrateByDistance calibrates m values of existing vertices by using new m values from the input points and the shortest path distances along the polyline between those

points. New vertices are added to the polyline where there is no corresponding vertex for a given point within the MultiPoint. A tolerance is used to limit the distance the MultiPoint's points can be away from the polyline in order to be used for calibration. Points outside the tolerance are simply ignored.

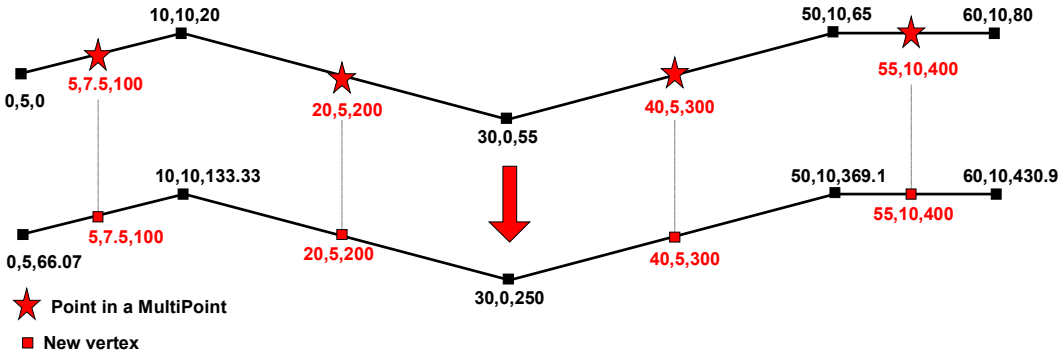
```
Public Sub CalibratePolylineByDistance(pPl As IPolyline, pMP As IMultipoint, _
                        updateHow As Long, ignoreGaps As Boolean)
    '+++ This operation must be performed on a simple geometry
  Dim pGC As IGeometryCollection
  Set pGC = pPl
  pGC.GeometriesChanged
  pPl.SimplifyNetwork

  Dim pMA As IMAware, pMSeg As IMSegmentation2

  '+++ The MultiPoint's points should have Ms
  Set pMA = pMP
  If Not pMA.MAware Then _
    Err.Raise vbObjectError + 11282000, "CalibratePolylineByDistance",  _
          "MultiPoint Not m Aware"

  '+++ The Polyline should be m Aware as well.
  Set pMA = Nothing
  Set pMA = pPl
  If Not pMA.MAware Then
    Err.Raise vbObjectError + 11282000, "CalibratePolylineByDistance", _
       "Polyline Not m Aware"
  Else
    Dim pPC As IPointCollection, pEnumV As IEnumVertex, pEnumSp As
IEnumSplitPoint
    Set pMSeg = pMA
    Set pPC = pMP
    Set pEnumV = pPC.EnumVertices
    Set pEnumSp = pMSeg.CalibrateByDistance(pEnumV, updateHow, ignoreGaps)
  End If
End Sub
```

J-8673

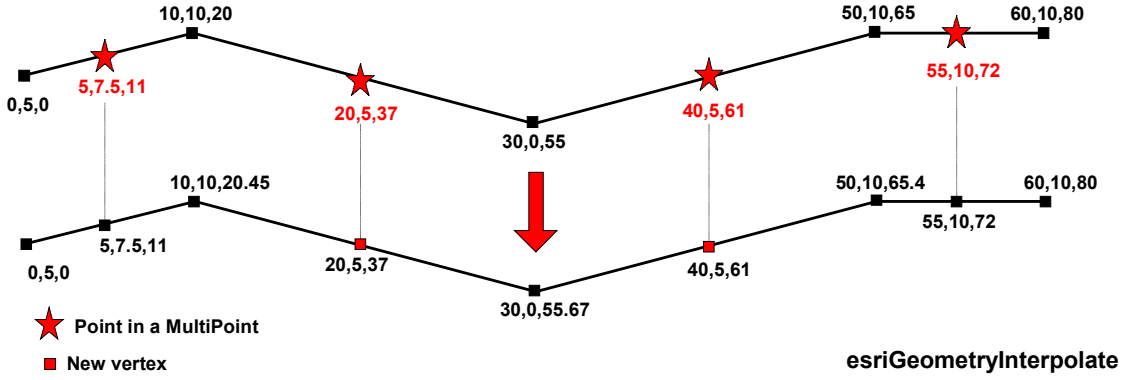


esriGeometryInterpolate OR esriGeometryExtrapolateBefore OR esriGeometryExtrapolateAfter

CalibrateByDistance

CalibrateByMs also calibrates m values of existing vertices by using new m values from the input points and the shortest path distances along the polyline between those points. New vertices are added to the polyline where there is no corresponding vertex for a given point within the MultiPoint. A tolerance is used to limit the distance the MultiPoint's points can be away from the polyline in order to be used for calibration. Points outside the tolerance are simply ignored.

CalibrateByMs differs from **CalibrateByDistance** in that existing m values on the affected vertices are used for the calibration. The m values on the affected vertices will be mapped based on the old m values and distance. This method is useful for "tweaking" an already calibrated route.



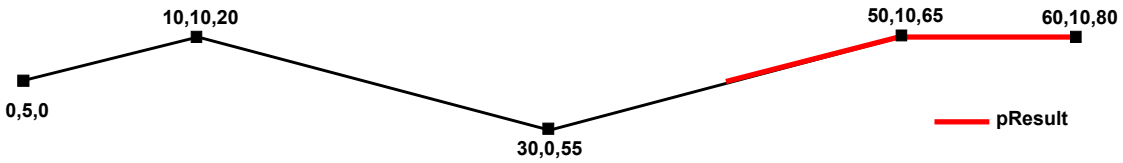
CalibrateByMs

GetSubcurveBetweenMsEx (like GetSubcurveBetweenMs) returns a polyline geometry corresponding to the subcurve(s) between the from- and to- m values. The "details" arguments are composed of esriMCurveRelationEnum values. When applied to the illustrated route, *fromDetails* will be set to **esriMBetweenMinMax** (1), and *toDetails* will be set to **esriMAboveMax** (3).

```
Dim pMSeg As IMSegmentation2
```

J-8673

```
Set pMSeg = pPl '+++ pointer to IPolyline interface
Dim fromDetails As Long, toDetails As Long
Dim pResult As IGeometryCollection
Set pResult = pMSeg.GetSubcurveBetweenMsEx(60, 85, fromDetails, toDetails)
```



GetSubcurveBetweenMsEx

SetMAsDistance2 sets the m values on vertices as scaled and offset distances from the input origin as measured along the polyline. Shortest path distances from the origin point are used. This method can be used to set m values that do not increase with the digitized direction of the polyline. This method optionally ignores distances between parts of the polyline.

The following figure illustrates the difference between the results of SetMAsDistance2 and SetMAsDistance on a multipart polyline. Note that SetMAsDistance does not use the origin point. Rather, it simply uses digitized direction.

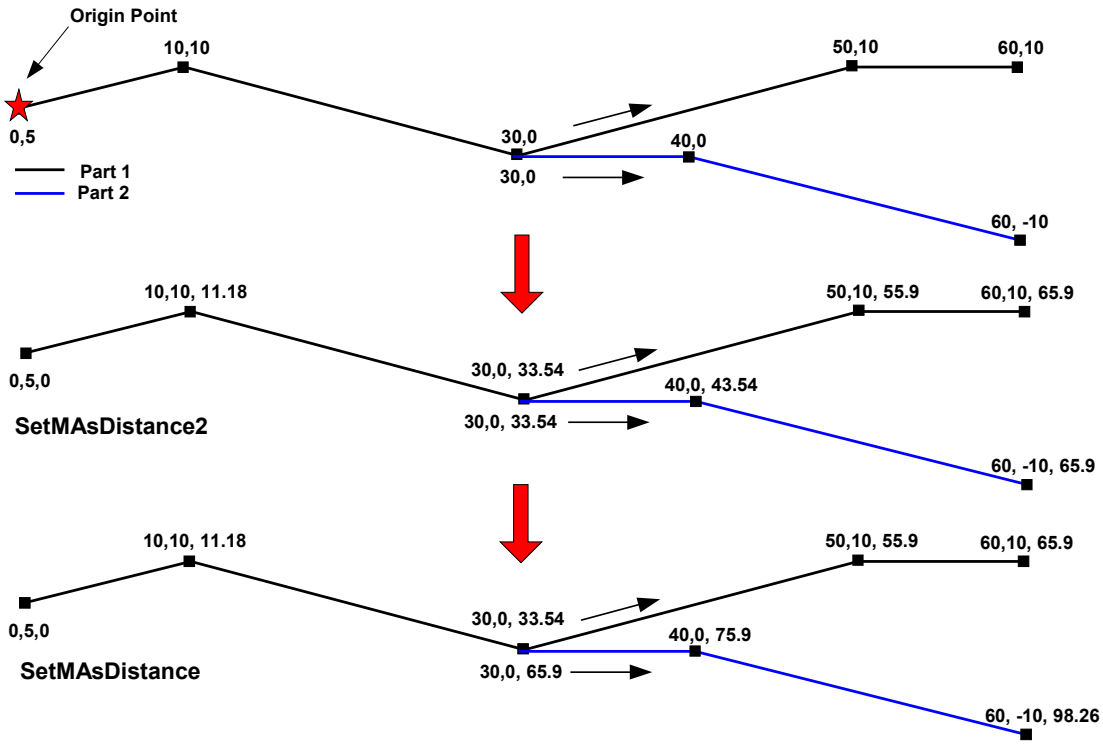
```
Dim pMA As IMAware
Set pMA = pPl '+++ pointer to IPolyline interface
If Not pMA.MAware Then pMA.MAware = True

Dim pGC As IGeometryCollection
Set pGC = pPl
pGC.GeometriesChanged
pPl.SimplifyNetwork

Dim pPt As IPoint
Set pPt = New Point
pPt.PutCoords 0, 5

Dim pMSeg As IMSegmentation2
Set pMSeg = pMA
pMSeg.SetMsAsDistance2 pPt, 1, 0, True
```

J-8673

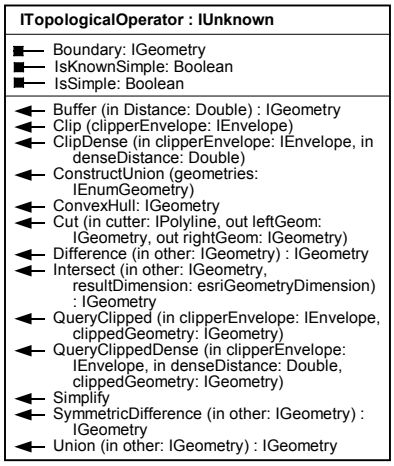


SetMAsDistance2 Versus SetMAsDistance on Multipart Polyline

J-8673

Appendix C—Topological Operators and Polylines with M Values

The **ITopologicalOperator** interface in the geometry system provides a set of methods for constructing new geometries based on topological relationships between existing geometries. Although this interface is implemented on the GeometryBag, Point, MultiPoint, Polygon, and Polyline CoClasses, what follows is a description of how some of this interface's methods behave with respect to polylines with m values. More complete descriptions of the various topological operators can be found in Chapter 6, "The Shape of Features," in *Modeling Our World* and Chapter 9, "Shaping Features with Geometry," in *Exploring ArcObjects.*



The ITopologicalOperator Interface

Many of the topological operations involve two geometries. In these cases, there is both a "primary" and a "secondary" geometry. The following rules and illustrations outline what happens to a polyline's m values when it participates in topological operations. All topological operations follow similar logic when determining how to set the m values.

- The m "awareness" of the primary geometry is dominant. This means, for example, that intersecting an MAware polyline (primary geometry) with a non-MAware Polygon (secondary geometry) will result in an MAware geometry. Conversely, intersecting a non-MAware Polygon with an MAware polyline will result in a non-MAware geometry.

J-8673

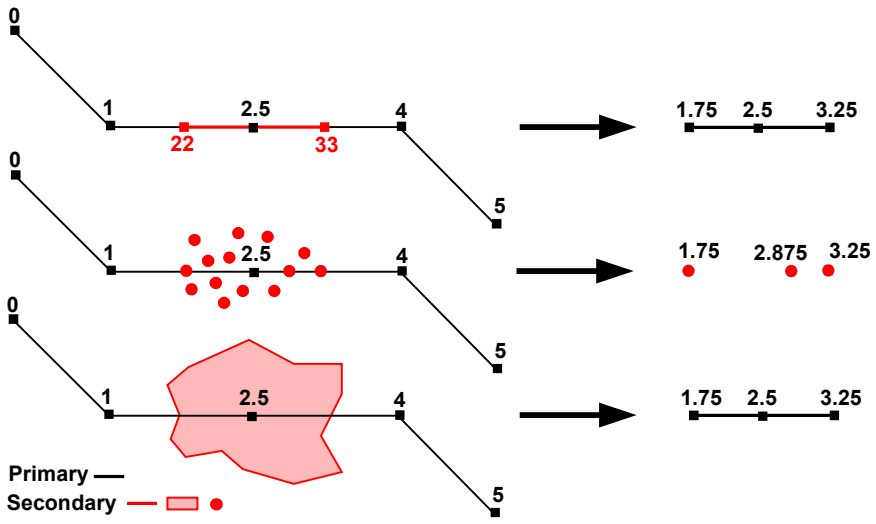
- In cases where both the primary and secondary geometries are MAware and a decision needs to be made with regard to whose m values are to be used in the output, the m values from the primary geometry are dominant. The m values from the secondary geometry are never imposed onto the primary geometry.

- Topological operations that return polygons will not preserve m values (e.g., Buffer, ConvexHull).

- Wherever necessary, m values from the primary geometry are interpolated based on length. Interpolation is possible only when the segment being split has a valid (non-NaN) m value at both ends.

- Two points are considered to be "equal" only when their x,y,m values match exactly.

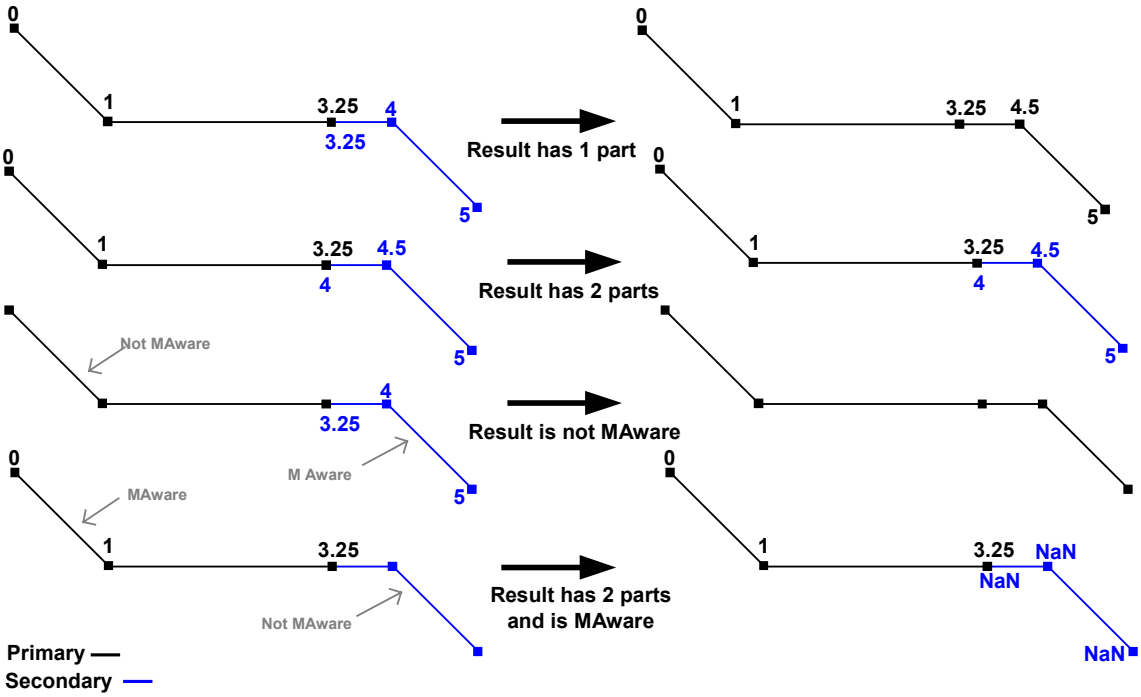
Intersect returns the logical AND of the input geometries. When intersecting geometries of different dimensions, the result dimension is less than or equal to the lower dimension.



These Intersect examples show that m values from the primary geometry are used when possible.

Union returns the logical OR of space. Only geometries of like dimension can be unioned.

J-8673



These Union examples show that the result has as few parts as possible and that the m awareness of the primary geometry determines the m awareness of the result.

Note: These examples are geometry manipulations. Trying to save a feature whose geometry is MAware (IMAware::MAware = True) to a feature class that cannot store m values (IGeometryDef::HasM = False) will strip the m values. Also, storing a feature whose geometry is not MAware in a feature class that can store m values will set the geometry set to be MAware (and all the m values will be NaN).

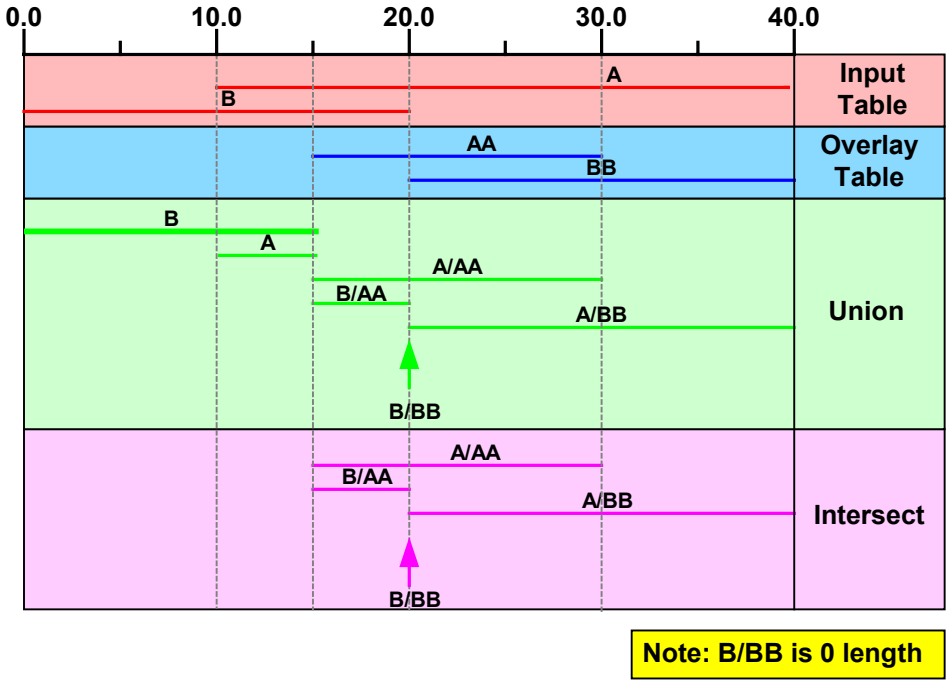
J-8673

Appendix D—Event Geoprocessing

For complete code samples on how to use the event geoprocessing functionality offered at ArcGIS 8.1, refer to the dynamic segmentation samples in ArcObjects Developer Help.

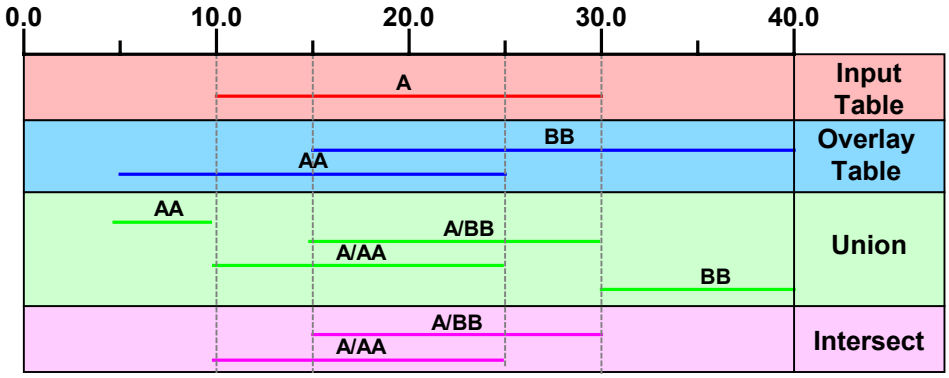
Line-on-Line Overlay

When a line-on-line event overlay is performed, a dissolve is also performed. Optionally, the output can contain 0 length events. The following illustrates various line-on-line event overlay scenarios.

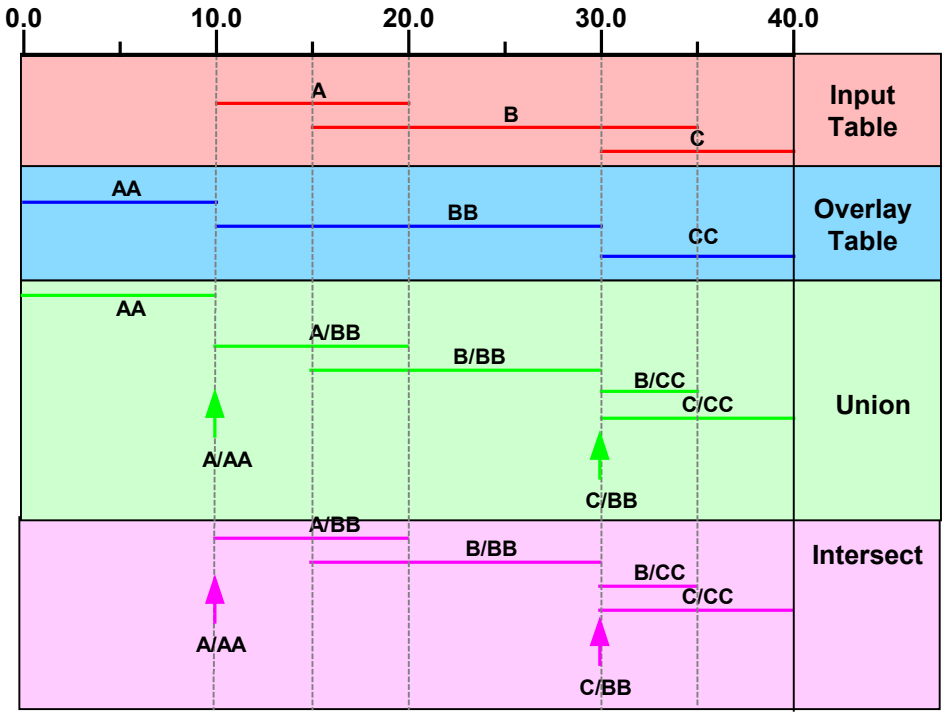


Line-on-Line Event Overlay—Example 1

J-8673



Line-on-Line Event Overlay—Example 2

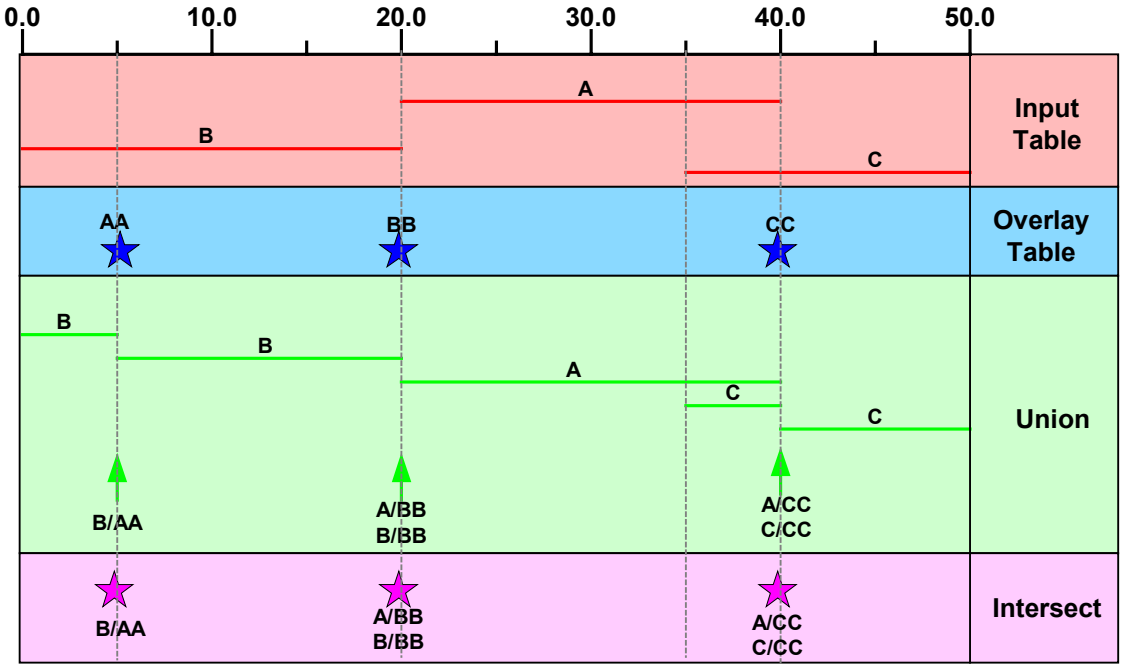


Line-on-Line Event Overlay—Example 3

Line-on-Point Overlay

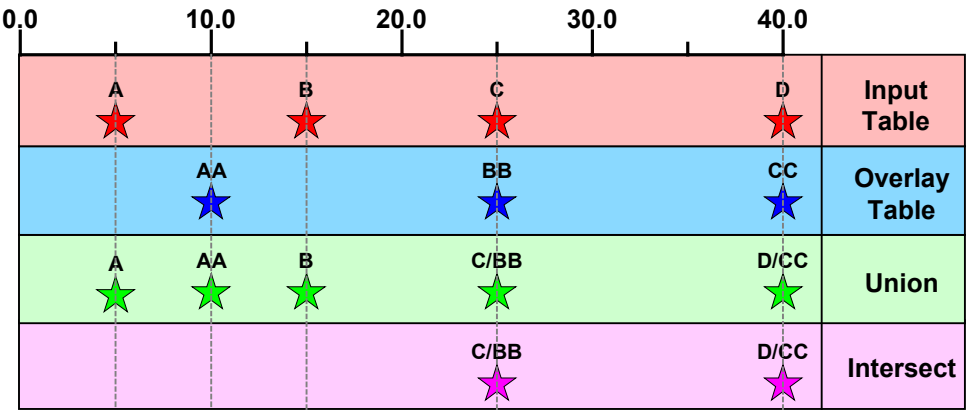
When doing an event overlay and either of the event tables is point, then the output event table will be point when doing an INTERSECT. If either event table is line, then the output event table will be line when doing a UNION. Line-on-point (or vice versa) UNION overlays often create 0 length events.

J-8673



Line-on-Point Overlay

Point-on-Point Overlay



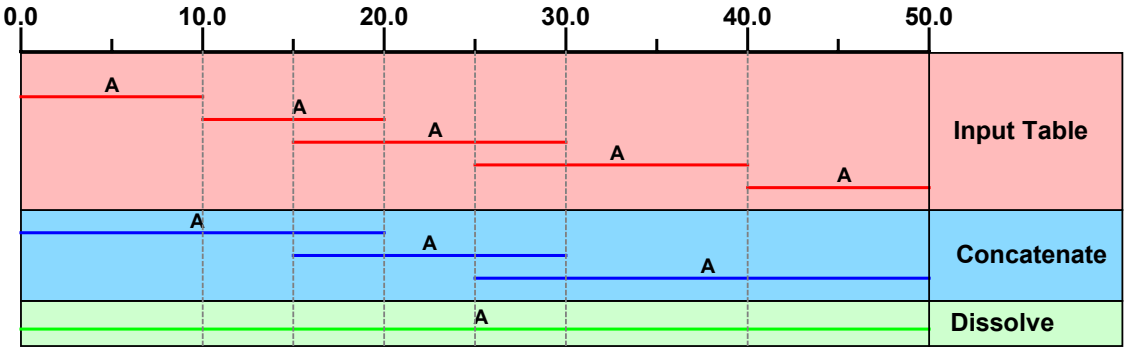
Point-on-Point Overlay

J-8673

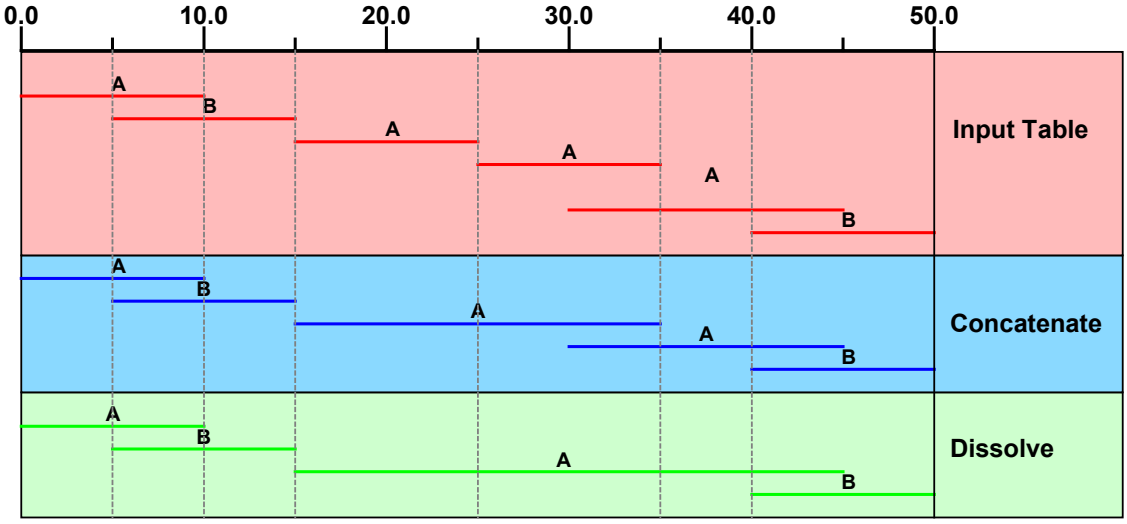
Concatenate and Dissolve

Event dissolving and event concatenating both involve combining records in line event tables if they are on the same route and have the same value for specified fields. The results are written to a new event table. The difference between dissolving and concatenating is that concatenating only combines events in situations where the to-measure of one event matches the from-measure of the next event. Dissolving events will combine events when there is measure overlap.

Only dissolve works for line or point event tables. Concatenate cannot be done on line event tables.

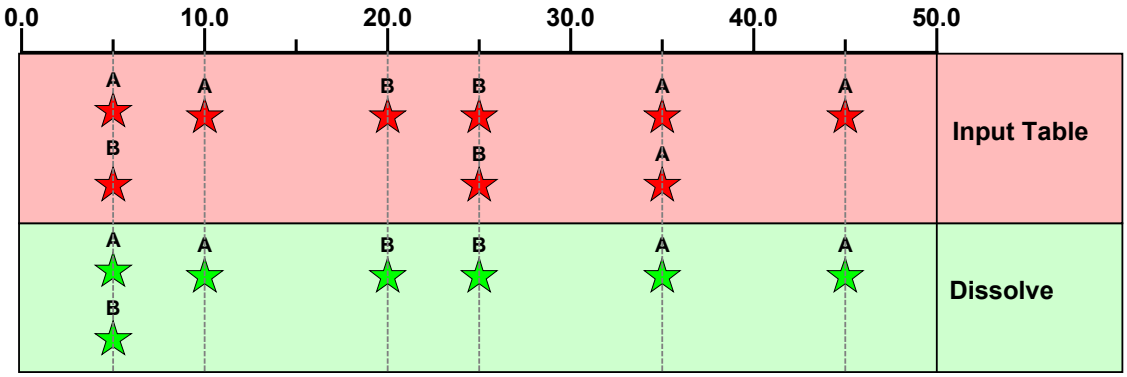


Concatenate and Dissolve—Example 1



Concatenate and Dissolve—Example 2

J-8673



Dissolve on a point event table. Concatenate point events does not work.

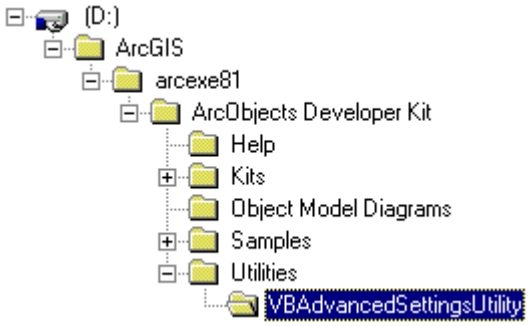
J-8673

Appendix E—Customizing the DynSeg Dialogs and Wizards

When using the dynamic segmentation user interface in ArcMap, some of the required fields are automatically populated. Taking advantage of the functionality provided by the ArcMap Advanced Settings utility can control this behavior.

Note: Using this utility requires that the ArcObjects Developer Kit was installed at the time when ArcGIS was installed.

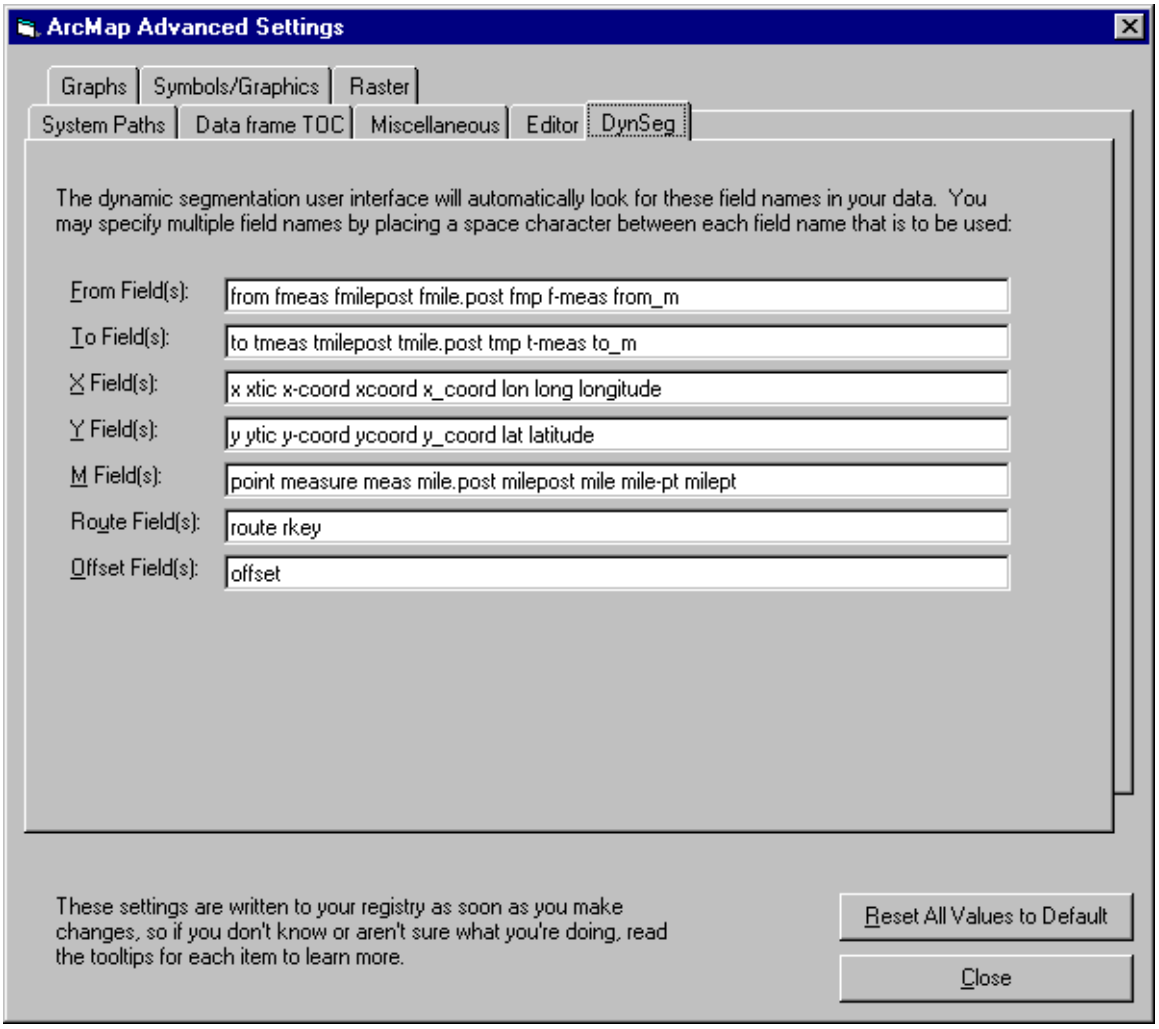
There is a folder called ArcObjects Developer Kit under the arcexe81 folder. In the Utilities\VBAdvancedSettingsUtility folder, the ArcMap Advanced Settings Utility is called AdvAMSet.exe.



AdvAMSet.exe resides in the VBAdvancedSettingsUtility folder.

To control the behavior of the Dynamic Segmentation dialogs, simply fill in as many values on the DynSeg tab as required. Note that an entry is not required on every line. For lines where there is more than one entry, simply put a space between the entries.

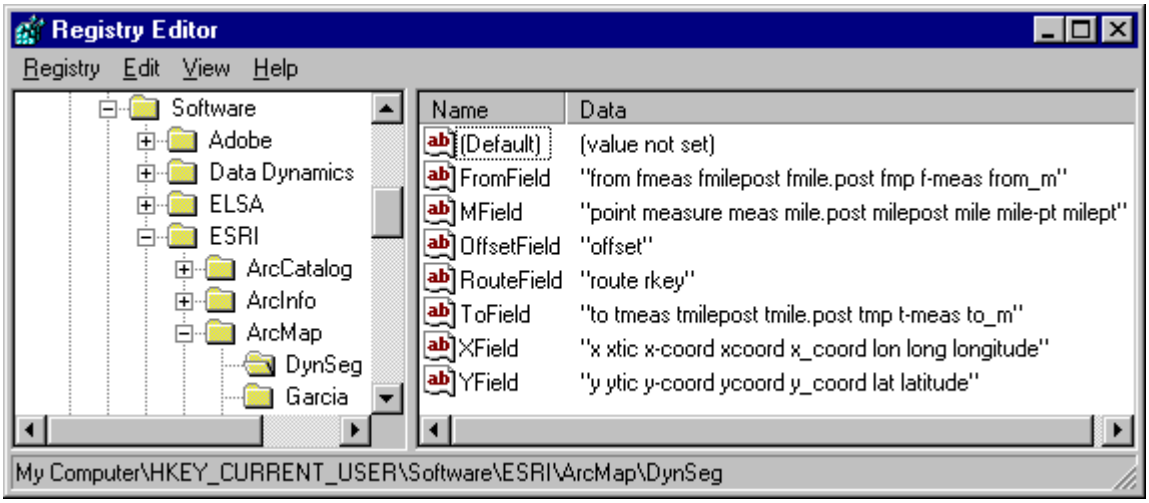
J-8673



Using the ArcMap Advanced Settings utility controls the behavior of the Dynamic Segmentation dialogs.

The ArcMap Advanced Settings Utility is simply creating a new key in the Windows Registry. If the ArcObjects Developer Kit was not installed with ArcGIS, a new Registry key can be created using, for example, the Registry Editor (regedit.exe).

J-8673



The HKEY_CURRENT_USER\Software\ESRI\ArcMap\DynSeg key can be created via the Registry Editor.