**K-Hub Iteration 2 Source Route Creation Process**

**Kansas Department of Transportation**

**October 31, 2017**

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author |
| 1.0 | 09/2017 | Original document creation | Transcend |
| 1.1 | 10/2017 | Updates made per KDOT review | Transcend |

**Table of Contents**

[Introduction 3](#_Toc494460606)

[Purpose 3](#_Toc494460607)

[Environment 3](#_Toc494460608)

[Creating Source County Routes 3](#_Toc494460609)

[Source Data to be Used 3](#_Toc494460610)

[Preparing Source County Data 3](#_Toc494460611)

[Adding Divided Attribute and Creating Routes 4](#_Toc494460612)

[Adding Concurrent Routes 8](#_Toc494460613)

[Adding Exceptions 11](#_Toc494460614)

[Adding Calibration Points 15](#_Toc494460615)

[QC Source Data 17](#_Toc494460616)

[Data Reviewer Checks 17](#_Toc494460617)

[Roads and Highways Detect Non-Monotonic Routes Geoprocessing Tool 18](#_Toc494460618)

[Classify Self-Intersecting Polylines Python Script 18](#_Toc494460619)

# Introduction

## Purpose

This document explains the process Transcend used to create the source routes. The purpose of this document is to allow KDOT to recreate the process in order to run the QAQC checks provided by Transcend. By recreating the source routes in the same fashion, KDOT’s QAQC should return the same number of errors as Transcend. KDOT will be able to ensure they are cleaning their source data appropriately for iteration 2.5 process.

## Environment

The source route creation uses standard ArcGIS geoprocessing tools and Roads and Highways specific geoprocessing tools. QC is performed using Esri’s Data Reviewer batch jobs, Roads and Highways geoprocessing tools and a custom Esri Python script. All queries in the document are in VB Script unless otherwise stated.

# Creating Source County Routes

Source projection: NAD\_83\_Kansas\_Lambert\_Conformal\_Conic\_Feet

## Source Data to be Used

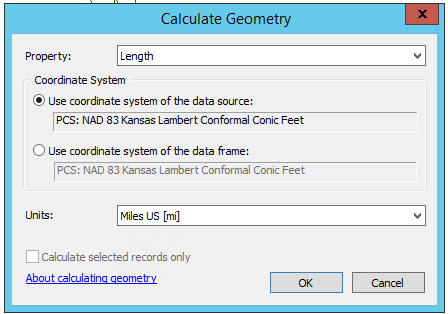
**I2\_NonPrimary\_Exor.gdb** > Highway\_NonCardinalDirectionFinal  
RouteID: KDOT\_LRS\_KEY  
Measures: Calibrated Shape Length  
Imported into working database and renamed: State\_NonPrimary\_SourceRoutes

**CalibratedStateSystem.gdb** > CalibratedStateSystem  
RouteID: KDOT\_LRS\_KEY  
Measures: Calibrated Shape Length  
Imported into working database and renamed: State\_Primary\_SourceRoutes

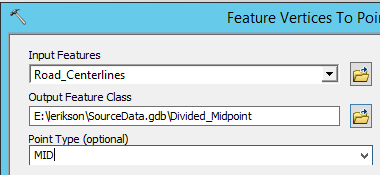
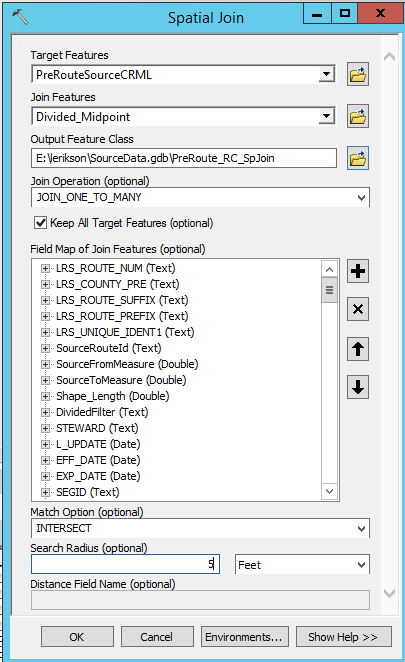
**Iteration\_2\_17D\_AllRegions\_Source.gdb** > RouteSource\_CountyLRS\_ARNOLD  
RouteID: SourceRouteId  
Measures: SourceFromMeasure and SourceToMeasures

**Iteration\_2\_17D\_UnprunedAttributes.gdb** > Road\_Centerlines  
RouteID: Non\_State\_System\_LRSKey  
Using field KDOT\_DIRECTION\_CALC to identify non-primary routes

## Preparing Source County Data

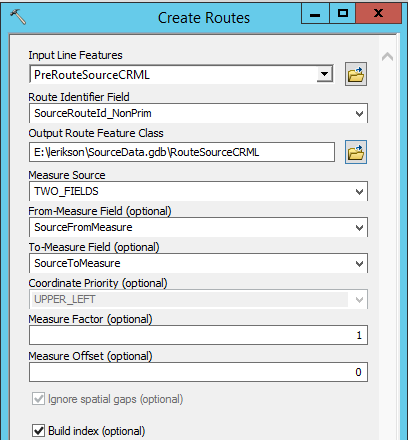
1. RouteSource\_CountyLRS\_ARNOLD - Turn off all fields except:
   1. LRS\_COUNTY\_PRE
   2. LRS\_ROUTE\_NUM
   3. LRS\_ROUTE\_PREFIX
   4. LRS\_ROUTE\_SUFFIX
   5. LRS\_UNIQUE\_IDENT1
   6. SourceFromMeasure
   7. SourceToMeasure
   8. SourceRouteId
2. Export feature class out: PreRouteSourceCRML
3. Select where SourceRouteId is null or LRS\_ROUTE\_PREFIX in (I, U, K) and export to archive db as StateRoutes\_and\_NullRoutesIDs
4. Delete selection from PreRouteSourceCRML
5. Select where SourceFromMeasure or SourceToMeasure is null and export as own feature class: PreRouteSource\_NoMeasures
6. Where SourceFromMeasure is null, populate with 0; where SourceToMeasure is null, populate with Calculate Geometry:   
   

## Adding Divided Attribute and Creating Routes

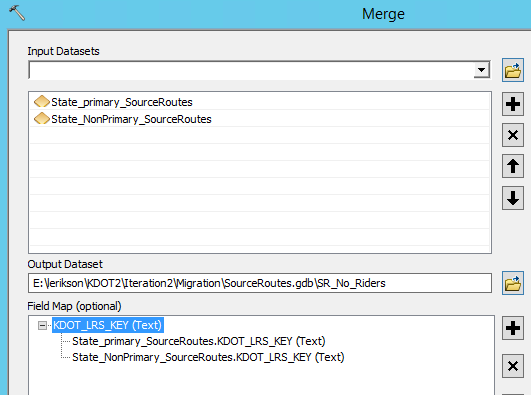
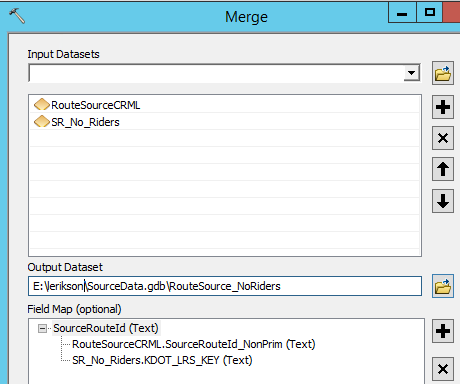
1. To Road\_Centerlines in Iteration\_2\_17D\_UnprunedAttributes.gdb, apply a definition query of a LRS\_ROUTE\_PREFIX not in (‘I’, ‘U’, ‘K’) and KDOT\_DIRECTION\_CALC = ‘1’
2. Add a new field to PreRouteSourceCRML called DividedFilter, text, length of 3.
3. Set up a relate between Road\_Centerlines to PreRouteSourceCRML by Non\_State\_System\_LRSKey and SourceRouteId
4. Select all Road\_Centerlines and relate to PreRouteSourceCRML. Populate DividedFilter in PreRouteSourceCRML as “yes” to identify routes that match the divided routes in Road\_Centerlines
5. Apply a definition query to PreRouteSourceCRML where DividedFilter = “yes”
6. Open the Feature Vertices to Points geoprocessing tool, populate the tool as follows, and run:  
   
7. Open the Spatial Join GP tool, populate the tool as follows, and run:   
   

The spatial join will accurately join the majority of the non-inventory side of the routes. For the remaining that are not joined (~23), they fall on centerlines that do not match the routeid of the Road\_Centerlines feature.

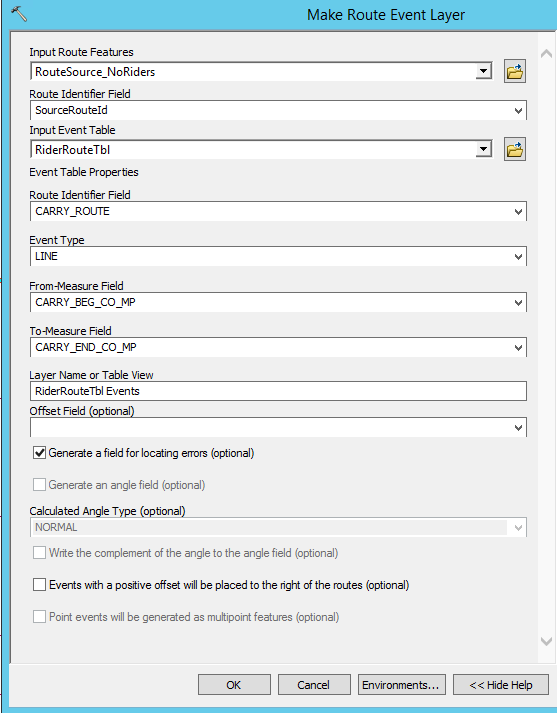
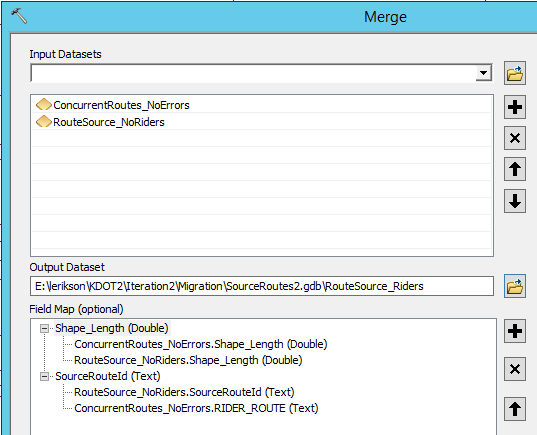
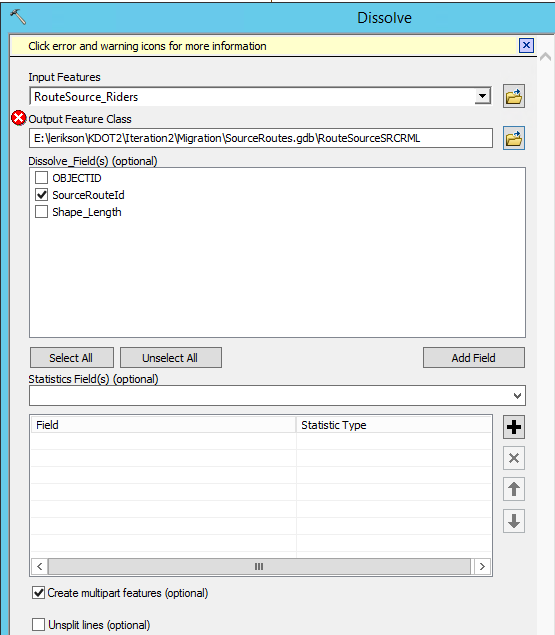
1. Add a field to PreRouteSourceCRML called KDOT\_Direction\_Calc, text, 1 character
2. Set up a relate PreRoute\_RC\_SpJoin (TARGET\_FID field) to PreRouteSourceCRML (OBJECTID field)
3. Select where SourceRouteId = Non\_State\_System\_LRSKey
4. Relate to PreRouteSourceCRML
5. Populate KDOT\_Direction\_Calc with ‘1’
6. Add a secondary routeid field named SourceRouteId\_NonPrim, text, 20. Populate the field with SourceRouteId&KDOT\_DIRECTION\_CALC. This step adds a 1 to the end of the SourceRouteId to differentiate the primary and non-primary sides for the routes, as well as for the business data.
7. Remove the definition query from PreRouteSourceCRML
8. Create routes with PreRouteSourceCRML – Route Identifier Field: SourceRouteId\_NonPrim. Measure Source: Two fields. Output: RouteSourceCRML



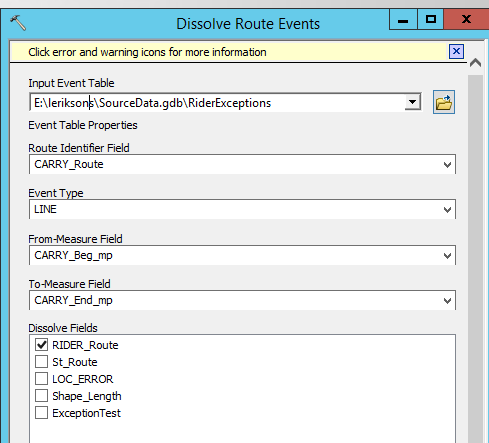
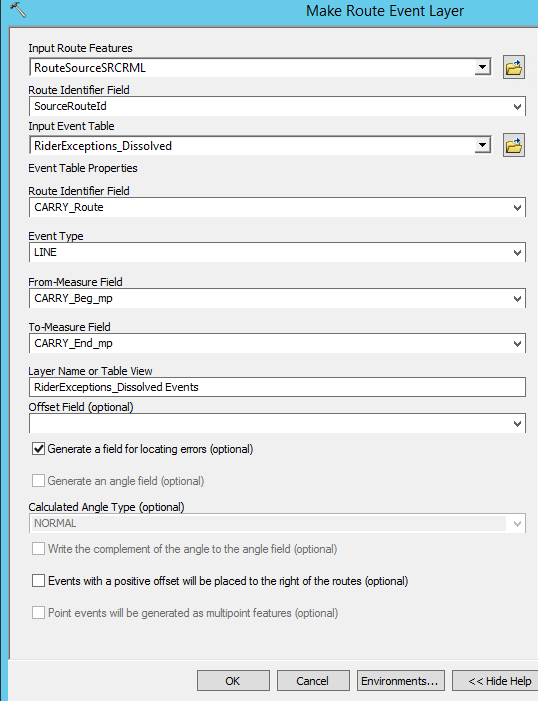
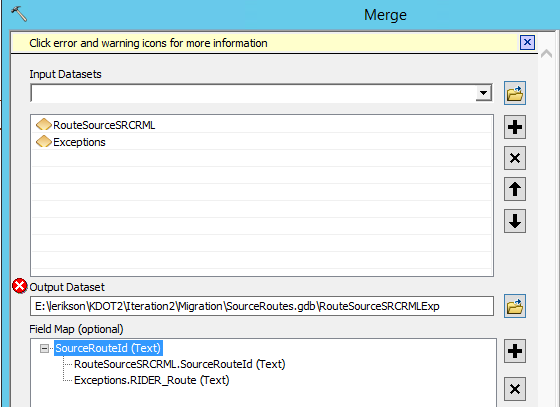
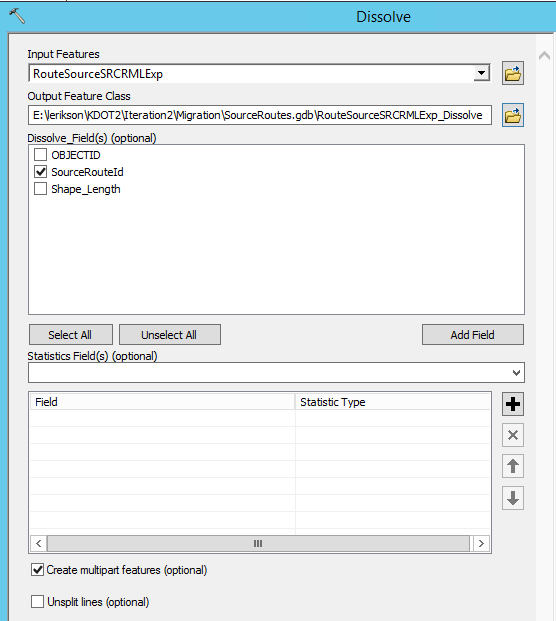
*(Joined RouteSourceCRML to PreRouteSourceCRM on the SourceRouteId\_NonPrim field. Selected where RouteSourceCRML.SourceRouteId\_NonPrim is not null to ensure all routes were created.)*

1. Merge State\_primary\_SourceRoutes and State\_NonPrimary\_SourceRoutes and keep only the necessary fields (Output: SR\_No\_Riders):  
   
2. Merge RouteSourceCRML and SR\_No\_Riders. Adjust field mapping. Output: RouteSource\_NoRiders  
   

## Adding Concurrent Routes

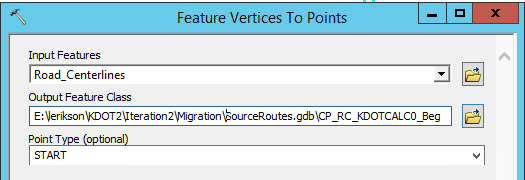
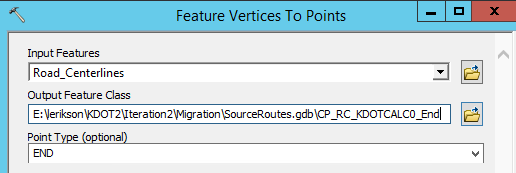
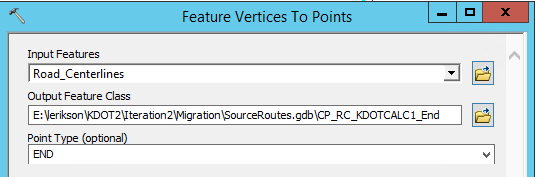
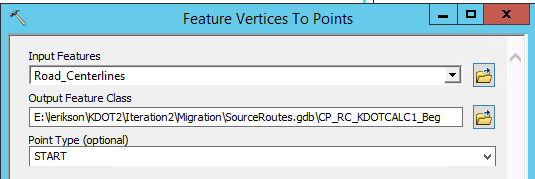
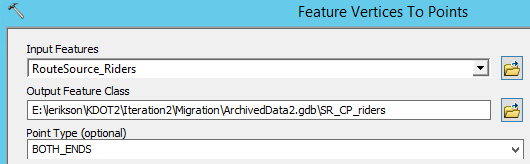
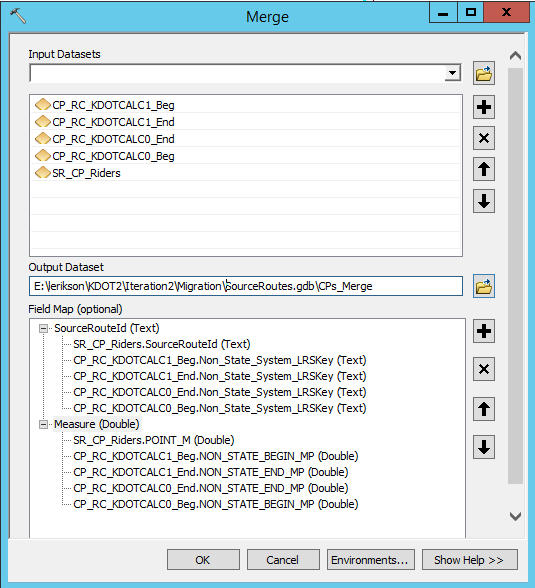
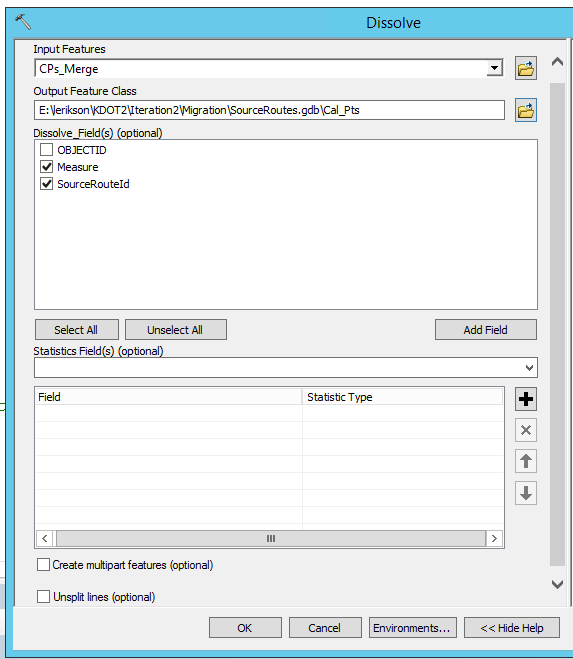
1. Import Rider Route table (Rider and Carry Routes.xls) and delete all fields except:
   1. RIDER\_ROUTE
   2. BEG\_RIDER\_CO\_LOGMI
   3. END\_RIDER\_CO\_LOGMI
   4. BEG\_RIDER\_ST\_LOGMI
   5. END\_RIDER\_ST\_LOGMI
   6. CARRY\_ROUTE
   7. CARRY\_BEG\_CO\_MP
   8. CARRY\_END\_CO\_MP
   9. CARRY\_BEG\_ST\_MP
   10. CARRY\_END\_ST\_MP
2. Run Make Route Event Layer with the following fields populated:   
   
3. Turn off all carry route fields
4. Export out all events as ConcurrentCenterlines\_total
5. Select the records with No Error for LocError and export out as ConcurrentCenterlines\_NoErrors
6. Run Create Routes GP tool using ConcurrentCenterlines\_NoErrors as source. Route Identifier Field: RIDER\_ROUTE. From Measure Field: BEG\_RIDER\_CO\_LOGMI. To Measure Field: END\_RIDER\_CO\_LOGMI
7. Merge ConcurrentRoutes\_NoErrors to RouteSource\_NoRiders (Output: RouteSource\_Riders)  
   
8. Dissolve (output: RouteSource\_SRCRML)  
   

## Adding Exceptions

1. Combine IUKroutes\_TableToExcel.xls and AllCMRsegments\_TableToExcel.xls (output: AllExceptions.xls)
2. Import AllExceptions.xls into working database (output: RiderExceptions)
3. Only fields in table should be:
   1. CARRY\_Route
   2. CARRY\_Beg\_mp
   3. CARRY\_End\_mp
   4. RIDER\_Route
4. Dissolve RiderExceptions using the Dissolve Route Events tool. (Output: RiderExceptions\_Dissolved)  
   
5. Add St\_Route field, text, 20.
6. State routes in the Carry\_Route field will need to be manually checked for direction and then populated correctly in the St\_Route field
7. When all state routes are manually checked, populate the CARRY\_Route field with corrected state routes from the St\_Route field
8. Run the Make route event layer tool and populate as follows:   
   
9. Export event (output: Exceptions)
10. Merge Exceptions and RouteSourceSRCRML (Output: RouteSourceSRCRMLExp):  
    
11. Dissolve (Output: RouteSourceSRCRMLExp\_Dissolve)   
    
12. Add a FromDate and ToDate field, type Date. Populate FromDate with 1/1/2016.

## Adding Calibration Points

Because Road\_Centerlines is completely segmented, calibration points are created from this dataset instead of RouteSource\_CountyLRS\_ARNOLD

1. Turn off all fields in Road\_Centerlines except:
   1. Non\_State\_System\_LRSKey
   2. NON\_STATE\_BEGIN\_MP
   3. KDOT\_DIRECTION\_CALC
2. Set up a relate between Road\_Centerlines and PreRouteSourceCRML on the Non\_State\_System\_LRSKey (Road\_Centerlines) and SourceRouteId (PreRouteSourceCRML)
3. Select all PreRouteSourceCRML where SourceRouteId\_NonPrim like '\_\_\_\_\_\_\_\_\_\_\_\_\_' and relate to Road\_Centerlines
4. Remove from the selection anywhere KDOT\_DIRECTION\_CALC = 1
5. Create points at the beginning of route sections (Feature Vertices To Points). Output: CP\_RC\_KDOTCALC0\_Beg  
   
6. Turn off the NON\_STATE\_BEGIN\_MP and turn on NON\_STATE\_END\_MP
7. Create points at the End of route sections (Feature Vertices To Points). Output: CP\_RC\_KDOTCALC0\_End
8. Select all PreRouteSourceCRML where SourceRouteId\_NonPrim like '\_\_\_\_\_\_\_\_\_\_\_\_\_1' and relate to Road\_Centerlines
9. Remove from the selection anywhere KDOT\_DIRECTION\_CALC = 0
10. Create points at the End of route sections (Feature Vertices To Points). Output: CP\_RC\_KDOTCALC1\_End
11. Turn off the NON\_STATE\_END\_MP and turn on NON\_STATE\_BEGIN\_MP
12. Create points at the beginning of route sections (Feature Vertices To Points). Output: CP\_RC\_KDOTCALC1\_Beg  
    
13. On RouteSource\_Riders, apply a definition query of SourceRouteId like '\_\_\_I%' or SourceRouteId like '\_\_\_U%' or SourceRouteId like '\_\_\_K%' and run Feature Vertices to Points GP tool  
    
14. Run Add XY Coordinates to obtain route measure
15. Merge Sr\_CPs, CP\_RC\_KDOTCALC0\_Beg, CP\_RC\_KDOTCALC0\_End, CP\_RC\_KDOTCALC1\_Beg, and CP\_RC\_KDOTCALC1\_End. Adjust the field mapping as shown:  
    
16. Dissolve CPs\_Merge on SourceRouteId and Measure. Uncheck Create multipart feature (Output: Cal\_Pts):  
    

## QC Source Data

### Data Reviewer Checks

#### Calibration Points

1. Duplicate CPs for same ROUTE\_ID and MEASURE
   1. ROUTEID = ROUTEID
   2. MEASURE = MEASURE
2. Duplicate CPs for same ROUTE\_ID and different MEASURE
   1. ROUTEID = ROUTEID
   2. MEASURE <> MEASURE
3. CP ROUTE\_ID is NULL
4. CP MEASURE is NULL
5. CP MEASURE < 0

#### Routes

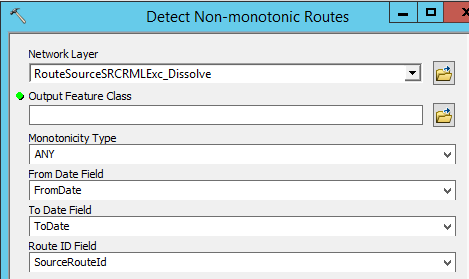
1. Invalid Geometry
2. Multipart Line
3. Non-Linear Segment
4. Polyline or Path Closes on Self
5. Duplicate Vertex (0-meter tolerance) looks for points on top of each other
6. Cutbacks (Min Angle 30 degrees)
7. Evaluate Polyline Length (polyline < 5 meters)
8. RouteId is NULL
9. Find Dangles (5-meter tolerance)
10. Orphans
11. Monotonicity (Non-Monotonic Features and Level Values)

#### Process

1. Select the Reviewer Session Manager on the Data Reviewer toolbar. Navigate to the Reviewer Workspace to store the Data Reviewer outputs.
2. Select Start Session.
3. Select Advanced. Select Reviewer Methods. Check Do not store return geometry for faster processing. Select Close.
4. Select Reviewer Batch Job Manager. Select Open and navigate to the Data Reviewer batch job provided by Transcend.
5. Double click Duplicate CPs for same ROUTE\_ID and MEASURE. Select Cal\_Pts for the Feature Class 1 and Feature Class 2.
6. Scroll down and select Select Attributes. Ensure the Error Conditions are Measure = Measure and SourceRouteId = SourceRouteId. The error conditions may need to be updated based on the field names used by KDOT.
7. Repeat the process for Duplicate CPs for same ROUTE\_ID and different MEAUSRE. Ensure the error conditions are Measure <> Measure and SourceRouteId = SourceRouteId.
8. Continue through the remaining calibration point checks selecting Cal\_Pts as the participating feature class. No other error conditions will need to be set.
9. Continue through the Routes checks, selecting RouteSourceSRCRMLExc\_Dissolve as the participating feature class.
10. Once complete, select Save As to save the checks with the updated feature classes.
11. Select Reviewer Batch Validate on the Data Reviewer toolbar. Select Full Database and select Add from File. Navigate to the checks saved in the previous step.
12. Select Validate All. If any error occurs, select Workspace and correct any broken links.
13. Finally, select Run (Note: It may be beneficial to run the Find Dangles and Orphan checks separately. These checks typically multiple hours to run)

### Roads and Highways Detect Non-Monotonic Routes Geoprocessing Tool

Populate the tool as follows:



### Classify Self-Intersecting Polylines Python Script

Update the highlighted sections to the appropriate locations and fields:

