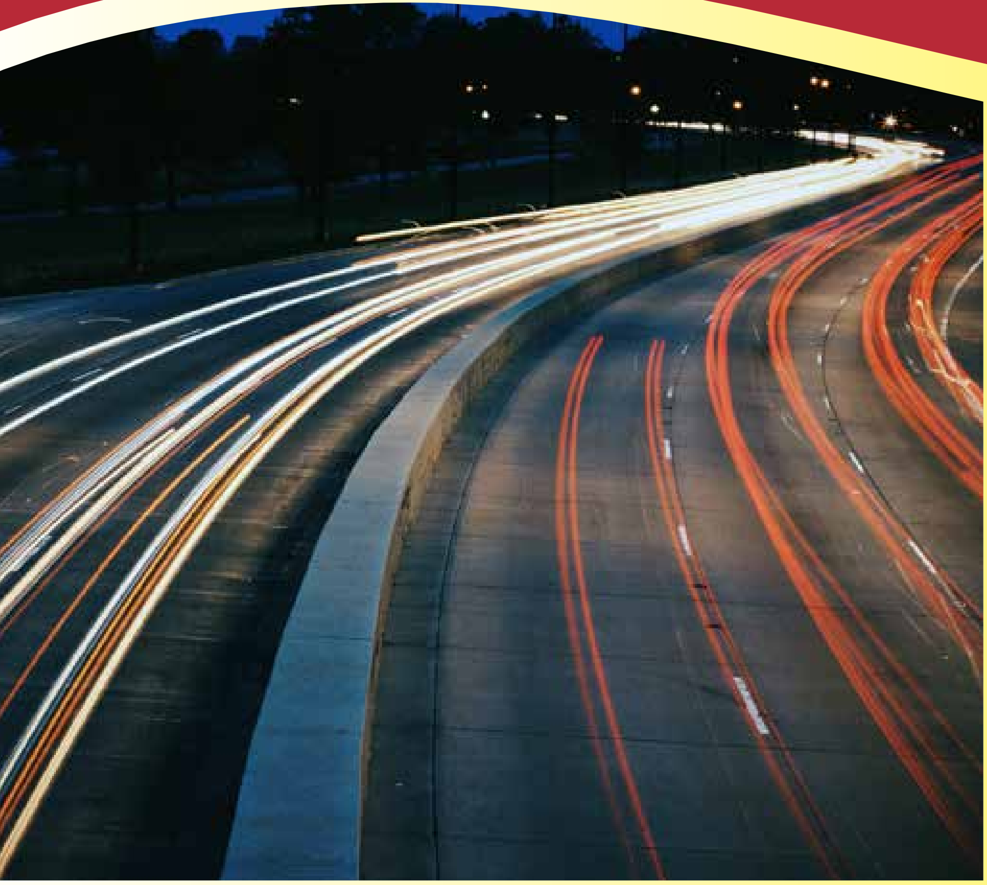
# ROADWAY DATA EXTRACTION TOOL
# Implementation and Programming Guide



Developed in Support of FHWA's Roadway Data Extraction Technical Assistance Program

U.S. Department of Transportation
**Federal Highway Administration**

Safe Roads for a Safer Future
*Investment in roadway safety saves lives*

http://safety.fhwa.dot.gov

*Cover photos: left: Thinkstock, right photo: Pexel Images, right bottom: Thinkstock*

Form DOT F 1700.7 (8-72)

| | | | |
|---|---|---|---|
| 1. Report No.<br>FHWA-SA-17-028 | 2. Government Accession No. | 3. Recipient's Catalog No. | |
| 4. Title and Subtitle<br>Roadway Data Extraction Tool<br>Implementation and Programming Guide | | 5. Report Date<br>December 2016<br>6. Performing Organizations Code | |
| 7. Authors<br>Kraus, Edgar; Le, Jerry; Sharma, Sushant | | 8. Performing Organization Report No. | |
| 9. Performing Organization Name and Address<br>Texas A&M Transportation Institute<br>1100 NW Loop 410, Suite 400<br>San Antonio, TX, 78213-2255<br><br>Leidos, Inc.<br>11251 Roger Bacon Drive<br>Reston, VA 20190 | | 10. Work Unit No. (TRAIS)<br>11. Contract or Grant No.<br>DTFH61-10-D-00024 | |
| 12. Sponsoring Agency Name and Address<br>United States Department of Transportation<br>Federal Highway Administration<br>Office of Safety<br>1200 New Jersey Avenue, SE<br>Washington, DC 20590 | | 13. Type of Report and Period Covered<br>Programming Guide<br>14. Sponsoring Agency Code<br>HSA | |
| 15. Supplementary Notes<br>FHWA Project Manager: Robert Pollack | | | |
| 16. Abstract (see below) | | | |
| 17. Key Words<br>Elements, Data Integration, Data Extraction, GIS | | 18. Distribution Statement<br>No restrictions. | |
| 19. Security Classif. (of this report)<br>Unclassified | 20. Security Classif. (of this page) Unclassified | 21. No of Pages<br>124 | 22. Price |

16. Abstract

The Roadway Data Extraction Technical Assistance Program (RDETAP) assists state and local agencies with the expansion and enhancement of roadway data inventories with regard to the Model Inventory of Roadway Elements (MIRE) and other roadway data elements. The RDETAP developed a management information system called the Roadway Data Extraction (RDE) Tool. This tool was developed to assist states extract and integrate critical data from available data sources and incorporate new value-adding data elements into existing roadway data inventories.

The objective of the RDE Tool Implementation and Programming Guide is to document and explain the steps used in extracting roadway inventory data from existing data sources to expand and enhance the roadway inventory data available to states with regard to MIRE and other roadway data. The programming guide describes the structure and code of the RDE Tool and provides examples of the process to implement the RDE Tool at state transportation agencies. The intended audience for the programming guide are advanced Geographic Information System (GIS) users and programmers as well as agency staff leading the effort to implement the RDE Tool at a transportation agency with the intent to improve roadway data for use in safety analyses. The RDE Tool Implementation and Programming Guide is a companion product to the RDE Tool User Guide that details the steps and processes to execute the RDE Tool and which is intended for transportation agency personnel that are going to use the RDE Tool as part of the agency's roadway data extraction and management workflow.

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized.

# Table of Contents

## Table of Contents (continued)

## List of Figures

## List of Figures (continued)

## List of Tables

## List of Abbreviations and Acronyms

| | |
|---|---|
| **AADT** | annual average daily traff |
| **AASHTO** | American Association of State Highway and Transportation Officials |
| **ADT** | average daily traffic |
| **CSV** | comma separated value |
| **DOT** | Department of Transportation |
| **FDE** | Fundamental Data Elements |
| **FHWA** | Federal Highway Administration |
| **GIS** | geographic information system |
| **HOV** | high-occupancy vehicle |
| **HSIP** | Highway Safety Improvement Program |
| **HSM** | Highway Safety Manual |
| **MAP-21** | Moving Ahead for Progress Act |
| **MIRE** | Model Inventory of Roadway Elements |
| **MIS** | Management Information System |
| **MoDOT** | Missouri Department of Transportation |
| **NIEM** | National Information Exchange Model |
| **RDETAP** | Roadway Data Extraction Technical Assistance Program |
| **SHSP** | Strategic Highway Safety Plan |
| **TxDOT** | Texas Department of Transportation |
| **VB.NET** | Visual Basic .NET |
| **VMT** | vehicle miles traveled |
| **WSDOT** | Washington State Department of Transportation |
| **XML** | extensible markup language |

# 1. Introduction

## Background

Both the Moving Ahead for Progress in the 21st Century (MAP-21) and the Fixing America's Surface Transportation Act (FAST) funding and authorization bills sanctioned continuation of the legacy Highway Safety Improvement Program (HSIP) as a core federal-aid program. Both legislative enactments called for advancing the capabilities of states for safety data collection, integration, and analyses to support a State's HSIP and Strategic Highway Safety Plan (SHSP). They acknowledged the importance of using multiple data sources to make effective decisions regarding resource allocation for a State's safety program. Further, both enactments underline the need for states to have in place a safety data system that can be used to conduct analyses and evaluations that support the strategic and performance based safety goals that are to be inclusive of all public roadways in the State. They also required the establishment of a subset of Model Inventory of Roadway Elements (MIRE) that would be useful for the inventory of roadway safety. The MIRE is a listing of roadway and traffic data elements to support safety management programs and analyses (*1*).

The Roadway Data Extraction Technical Assistance Program (RDETAP) within the Federal Highway Administration (FHWA) Office of Safety is intended to assist states to identify, extract, and record MIRE and other roadway data elements from commonly available existing sources of data, such as video logs, Google Earth™, Google Street View™, and Bing Maps™ Streetside™.

## RDETAP Development

From 2012 to 2013, the FHWA Office of Safety was involved in the MIRE-Management Information System (MIRE-MIS) project which sought methods to aid states with incorporation of MIRE data into state safety management practices. The MIRE-MIS had several components, one of which was the identification and development of various methods of identifying, modifying, and extracting roadway data elements. This component was known as the "Lead Agency Program."

The New Hampshire Department of Transportation participated in FHWA's MIRE-MIS Lead Agency Program. The outcome of the project was a customized tool called NH GIS-Based Tool. The NH GIS-Based Tool allowed the extraction, storage, and retrieval of roadway data elements for safety data analyses.

Recognizing the potential value of this tool and its processes to assist other states expansion and enhancement of their roadway data, the FHWA Office of Safety initiated the RDETAP. The purpose of the RDETAP was to further refine the capabilities of the NH GIS-Based Tool for processing roadway inventory data from multiple sources in an ESRI® ArcGIS® environment and attaching non-spatial attribute data (e.g., AADT and roadway width) to spatial roadway elements (e.g., intersection points and intersection legs). The data is stored in a geodatabase and can be exported in a variety of desired formats to support safety analysis. The RDETAP project has resulted in a "genericized" version of the NH GIS-Based Tool, referred to as the RDE tool. It has been enhanced to include all MIRE Version 1.0 data elements and can be further expanded to collect non-MIRE data elements. The RDE tool has been pilot tested with the states of Washington, Missouri, and Texas.

## Purpose of the Guide

The goal of the Roadway Data Extraction Technical Assistance Program (RDETAP) is to assist state and local agencies expand and enhance their roadway data inventories. The RDETAP provides guidance on how to identify, extract, and reformat critical data from available data sources and incorporate new value-adding data elements.

The purpose of this guide is to document and explain the steps used in extracting roadway inventory data from existing data sources to expand and enhance the roadway inventory data available to states with regard to the Model Inventory of Roadway Elements (MIRE) and other roadway data. Specifically, the guide provides instructions on how to modify the RDE tool (hereafter referred to as the Roadway Data Extraction (RDE) tool) that was developed for the New Hampshire Department of Transportation (DOT) and subsequently "genericized" to make it useful for other DOTs and local public agencies in the U.S.

During pilot implementation of the tool as part of the RDETAP, pilot states found that there is benefit in separating the guide into two documents, one aimed at GIS programmers, and one aimed at users of the tool. Thus as part of the RDETAP, two guides were developed, one for system programmers and one for data users. This implementation guide was developed to help state and local agencies take full advantage of the RDE tool.

Before the RDE tool can be used at a transportation agency there is a need for a significant amount of programming to adjust geodatabases, templates, and code to work with the agency's data. This programmer's guide is intended to aid with that process, while the separate Users Guide is aimed at helping a user manipulate the tool once it has been implemented at a transportation agency.

## Organization of the Guide

The guide is organized into the following eleven chapters and four appendices:

- The first chapter is this introductory chapter.
- Chapter 2 provides an overview of MIRE data elements, fundamental data elements (FDEs), and overall hierarchy of data elements in terms of collection priority.
- Chapter 3 provides a description of the RDE tool components and data management process.
- Chapter 4 provides a description of the RDE tool geodatabases, including data models and templates.
- Chapter 5 provides an overview of RDE tool toolboxes and scripts, and the business processes to create features, update features, and retire features.
- Chapter 6 provides an introduction to the RDE tool Addin, which consists of the RDE toolbar and related code.
- Chapter 7 provides general instruction on how to modify spatial templates, tools, and custom data entry interfaces to aid a transportation agency with the implementation of the RDE tool.
- Chapter 8 is a summary of the pilot implementation of the RDE tool at the Washington State DOT (WSDOT).
- Chapter 9 is a summary of the pilot implementation of the RDE tool at the Missouri DOT (MoDOT).

- Chapter 10 provides concluding remarks.
- Chapter 11 provides a listing of references used in this guide.
- Appendix I provides entity-relationship diagrams of the MIRE data model created for this project.
- Appendix II provides entity-relationship diagrams of the Safety Analyst data model created for this project.
- Appendix III provides an overview of the feature class templates that are used by the output geodatabase in the RDE tool.
- Appendix IV provides a copy of the Python script that is used to calculate the intersection angle in the RDE tool.

## 2. Model Inventory of Roadway Elements

MIRE is a guideline that provides a listing of roadway features and traffic volume elements important to safety management, and includes standardized coding for each element. MIRE Version 1.0 guide includes 202 elements grouped into three categories: roadway segments, roadway alignments, and roadway junctions. The complete listing of all the elements in the MIRE can be found at the website of the Federal Highway Administration's Office of Safety (*2*). FHWA anticipates issuing MIRE Version 2.0 in 2017 (however, all references to MIRE in this document refer to MIRE Version 1.0).

It is important to note that MIRE data elements are geared towards safety management, though having these elements could potentially benefit other State DOT business functions. There are additional data elements that states can and do collect that provide additional benefits. These non-MIRE data elements can also be incorporated into the RDE Tool.

In summary, MIRE 1.0 provides elements and attributes that are or will be needed when State and local DOTs make safety management decisions. MIRE can be broken down into categories and subcategories as shown in Table 1.

**Table 1. Categories and Subcategories of MIRE Elements.**

| MIRE Category | MIRE Subcategory |
|---|---|
| I. Roadway Segment Descriptors | I.a. Segment Location/Linkage Elements |
| | I.b. Segment Roadway Classification |
| | I.c. Segment Cross Section |
| | I.c.1. Surface Descriptors |
| | I.c.2. Lane Descriptors |
| | I.c.3. Shoulder Descriptors |
| | I.c.4. Median Descriptors |
| | I.d. Roadside Descriptors |
| | I.e. Other Segment Descriptors |
| | I.f. Segment Traffic Flow Data |
| | I.g. Segment Traffic Operations/Control Data |
| | I.h. Other Supplemental Segment Descriptors |
| II. Roadway Alignment Descriptors | II.a. Horizontal Curve Data |
| | II.b. Vertical Grade Data |
| III. Roadway Junction Descriptors | III.a. At-Grade Intersection/Junctions |
| | III.a.1. At-Grade Intersection/Junction General Descriptors |
| | III.a.2. At-Grade Intersection/Junction Descriptors (Each Approach) |
| | III.b. Interchange and Ramp Descriptors |
| | III.b.1. General Interchange Descriptors |
| | III.b.2. Interchange Ramp Descriptors |

## Fundamental Data Elements

While complete MIRE data is critical for safety, it may not be feasible for States to collect and integrate all of the MIRE data elements into their HSIP process. The MAP-21 and the FAST Act required FHWA to identify a subset of the elements in MIRE that should be integrated with crash data to conduct enhanced safety analyses in support of a State's HSIP. This subset of MIRE data elements is referred to as the MIRE Fundamental Data Elements (MIRE-FDE). The MIRE FDE are based on the elements needed to apply the HSM roadway safety management (Part B) procedures using network screening and analytical tools.

In March of 2016, FHWA issued a new HSIP regulation. This regulation identified requirements for state safety data systems that redefined FDEs based on roadway functional class and surface type (3). Effective April 14, 2016, FHWA defined three different sets of FDEs based on non-local paved roads, local paved roads, and unpaved roads (Table 2, Table 3, and Table 4). For non-local paved roads, FHWA defined FDEs for roadway segments, intersections, and interchanges/ramps. For local paved roads and unpaved roads, FHWA only defined FDEs for roadway segments. According to federal regulations, States shall incorporate specific quantifiable and measurable anticipated improvements for collection of MIRE FDEs into their State Traffic Records Strategic Plan update by July 1, 2017, and have access to the FDEs on all public roads by September 30, 2026 (*4*).

**Table 2. MIRE Fundamental Data Elements and MIRE Data Element Number for Non-Local* Paved Roads (*3*).**

| Roadway Segment | Intersection | Interchange/Ramp |
|---|---|---|
| Segment Identifier (12) | Unique Junction Identifier (120) | Unique Interchange Identifier (178) |
| Route Number (8) | Location Identifier for Road 1 Crossing Point (122) | Location Identifier for Roadway at Beginning Ramp Terminal (197) |
| Route/street Name (9) | Location Identifier for Road 2 Crossing Point (123) | Location Identifier for Roadway at Ending Ramp Terminal (201) |
| Federal Aid/ Route Type (21) | Intersection/Junction Geometry (126) | Ramp Length (187) |
| Rural/Urban Designation (20) | Intersection/Junction Traffic Control (131) | Roadway Type at Beginning Ramp Terminal (195) |
| Surface Type (23) | Average Annual Daily Traffic (79)** | Roadway Type at Ending Ramp Terminal (199) |
| Begin Point Segment Descriptor (10) | Average Annual Daily Traffic Year (80)** | Interchange Type (182) |
| End Point Segment Descriptor (11) | Unique Approach Identifier (139) | Ramp Average Annual Daily Traffic (191) |
| Segment Length (13) | | Year of Ramp Average Annual Daily Traffic (192) |
| Direction of Inventory (18) | | Functional Class (19) |
| Functional Class (19) | | Type of Governmental Ownership (4) |
| Median Type (54) | | |
| Access Control (22) | | |
| One/Two-Way Operations (91) | | |
| Number of Through Lanes (31) | | |
| Average Annual Daily Traffic (79) | | |

| Roadway Segment | Intersection | Interchange/Ramp |
|---|---|---|
| Average Annual Daily Traffic Year (80) | | |
| Type of Governmental Ownership (4) | | |

* Based on functional classification.

** For each intersecting road.

**Table 3. MIRE Fundamental Data Elements and MIRE Data Element Number for Local* Paved Roads (*3*).**

| Roadway Segment |
|---|
| Segment Identifier (12) |
| Functional Class (19) |
| Surface Type (23) |
| Type of Governmental Ownership (4) |
| Number of Through Lanes (31) |
| Average Annual Daily Traffic (79) |
| Begin Point Segment Descriptor (10) |
| End Point Segment Descriptor (11) |
| Rural/Urban Designation (20) |

* Based on functional classification.

**Table 4. MIRE Fundamental Data Elements and MIRE Data Element Number for Unpaved* Roads (*3*).**

| Roadway Segment |
|---|
| Segment Identifier (12) |
| Functional Class (19) |
| Type of Governmental Ownership (4) |
| Begin Point Segment Descriptor (10) |
| End Point Segment Descriptor (11) |

* Based on functional classification.

## MIRE Data Model

MIRE is a recommended listing and associated data dictionary of roadway inventory and traffic elements that are critical to the safety management of highways. As such, it provides the building blocks for a data model that can provide the foundation for a roadway inventory implementation based on MIRE. The following provides a description of a MIRE data model that was developed as part of the pilot MIRE-MIS implementation.

The MIRE data model depicts the data elements described in the MIRE Version 1.0 guideline and translates them to entities and attributes. The data model consists of seven main entities, which are intersection, intersection leg, segment, ramp, interchange, horizontal curve, and vertical grade. Appendix I provides entity-relationship diagrams of these and related entities. In addition to the main entities, the MIRE data model also provides numerous look-up tables that store the domain values of certain attributes in the main entities.

For the MIRE-MIS pilot implementation, the research team focused on the entities intersection, intersection leg, segment, and ramp. These entities and look-up tables were used to develop the GIS templates for the RDE tool.

# 3. Description of RDE Tool and Data Management Process

The RDE tool processes roadway inventory data from multiple sources in an ESRI ArcGIS environment and attaches non-spatial attribute data (e.g., AADT and roadway width) to spatial roadway elements (e.g., intersection points and intersection legs). The data is stored in a geodatabase and can be exported in a variety of desired formats to support a range of safety analysis tools.

## RDE Tool Components

The RDE tool consists of three main components that are briefly described below.

- RDE Tool Geodatabases
    - **InputData.gdb.** This geodatabase is used by the RDE tool to store all data that is input into the process. When user run 1. Import Data model, the model import user's data into this database and make it available to other models.
    - **IntermediateData.gdb.** This geodatabase is used by the RDE tool to temporary store data during processing.
    - **InternalData.gdb.** This geodatabase is used by the RDE tool to store internal data such as output templates.
    - **MIREProject.gdb.** This geodatabase is used by the RDE tool to store the process output, including the feature classes Intersections, IntersectionLeg, and Ramp.
    - **UpdateFeature.gdb.** This geodatabase is used by the RDE tool to temporary store data during the update feature process.
- RDE Tool Toolboxes and Scripts
    - **MIRE_3.tbx.** This ESRI toolbox contains the main models that form the RDE tool. These models can be executed by right-clicking the model and selecting "open".
    - **MIRE_support.tbx.** This ESRI toolbox contains supporting models and scripts that are used by the main models in toolbox MIRE_3.tbx. These models should not be executed directly.
    - **MIRE_update.tbx.** This ESRI toolbox contains models that can be used to update existing roadway data and roadway features.
    - **IntsectingAngle.py.** The RDE tool uses this Python script to calculate the smallest angle between two intersecting roadways.
- RDE Tool Addin
    - **MIRE Toolbar.** The *MIRE Toolbar* is an ESRI plug-in that provides buttons for a user to execute code and models within the ESRI ArcMap program.
    - **XML configuration file.** The XML configuration file provides basic configuration settings, such as field names, the path for the toolboxes, and the path to exported files. The XML configuration file must be modified before the RDE tool will work, as described below.

## RDE Tool Data Management Process

The RDE tool data management process involves the creation of several geodatabases including the output geodatabase *MIREProject.gdb* where the RDE tool output data is stored. To start, a user should combine all datasets needed for the data extraction effort into one geodatabase called *InputData.gdb*. However, this is not a required step and the RDE tool could be modified accordingly. Running the model *1 Import Data* selects the datasets of interest and moves relevant data to the geodatabase *InternalData.gdb*. Processing the data with model *2 Prepare Background Data* and subsequent models stores data in the geodatabase *IntermediateData.gdb*. When the tool finishes processing the data, all output datasets are moved to geodatabase *MIREProject.gdb*. Figure 1 provides a flow chart of the RDE tool data management process.

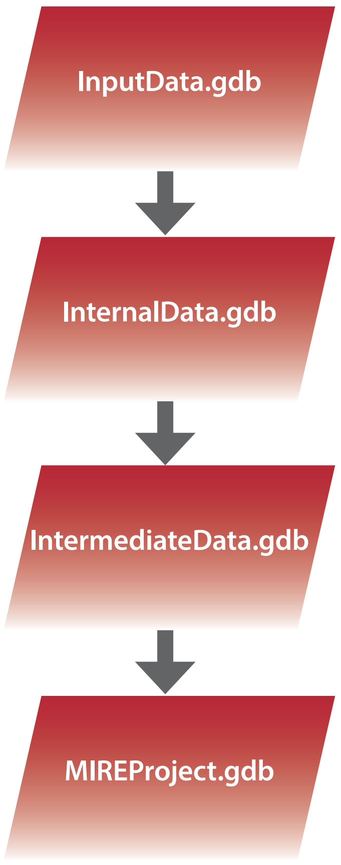


**Figure 1. Data Management Process.**

## System Requirements

The researchers tested and installed the tool on a computer with the following specification, which are the minimum requirements for the use of the RDE tool:

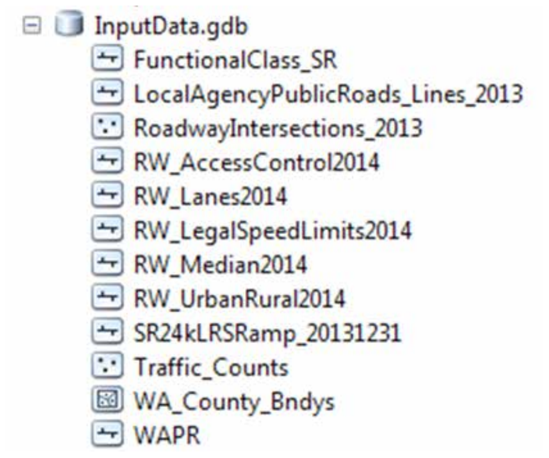
- Processor: Intel dual core or similar.
- RAM: 2 GB.
- Disk space: 3 GB.
- Operating system: Windows 7.
- ESRI Software.
- ArcMap version 9.3 or later (current compatible version is 10.4.1).
- ArcCatalog version 9.3 or later (current compatible version is 10.4.1).
- Data storage: file geodatabase version 9.3 or later or ESRI ArcSDE (personal geodatabase not supported).
- Input data: ArcGIS shapefiles, text, personal geodatabase, file geodatabase, ArcSDE, Oracle, SQL Server database (Oracle Spatial and SQL Server Spatial are not supported).
- Access to C Drive.

The following chapters provide an in-depth description of all RDE tool components. These descriptions are based on the "generic" version of the tool, which is a term the authors use to distinguish the RDE tool that should be used at the beginning of a RDE tool implementation project from a state-specific version of the RDE tool that was the result of an implementation project. However, in order for the tool to work, certain sample datasets must be available. Where needed to illustrate the functionality of the RDE tool, and with permission of the Washington State Department of Transportation (WSDOT), the authors used sample data from the WSDOT implementation effort.

# 4. RDE Tool Geodatabases

## InputData Geodatabase

The *InputData* geodatabase serves as a temporary database where the RDE tool gathers and temporarily stores all input datasets that are needed for the tool to work in one place. These are the datasets that will be used to automatically extract data and populate feature classes in the output geodatabase. Although this task is not necessary for the tool to work it makes it easier for a user to see what dataset are included in the data processing, and for what purpose. As an example, Figure 2 shows the *InputData* geodatabase in use at WSDOT.



InputData.gdb
- FunctionalClass_SR
- LocalAgencyPublicRoads_Lines_2013
- RoadwayIntersections_2013
- RW_AccessControl2014
- RW_Lanes2014
- RW_LegalSpeedLimits2014
- RW_Median2014
- RW_UrbanRural2014
- SR24kLRSRamp_20131231
- Traffic_Counts
- WA_County_Bndys
- WAPR

**Figure 2. Feature Classes in InputData Geodatabase.**

The icon in front of each feature class indicates the type of features stored in that feature class, as follows:

⁙ Point feature.

⊟ Line feature.

▨ Polygon feature.

The feature classes listed in Figure 2 provide the following data elements used by the RDE tool:

- *FunctionalClass_SR*: Provides the federal functional class code field for all state routes.
- *LocalAgencyPublicRoads_Lines_2013*: Contains the geospatial features of local public roads. For the RDE tool to work, features must have a value in the field *F_System*.
- *RoadwayIntersections_2013*: Contains intersection features that lack MIRE data. Intersection features should have an *AgencyID*, which is the intersection ID assigned by the DOT.
- *RW_AccessControl2014*: Contains the access control type code for each roadway.
- *RW_Lanes2014*: Contains the number of lanes for each roadway. The feature class should have a field for the number of lanes in decreasing and increasing direction.
- *RW_LegalSpeedLimits2014*: Contains the legal speed limit for all roadway segments.
- *RW_Median2014*: Contains a median barrier type code for each roadway.

- *RW_UrbanRural2014*: Contains an urban/rural code field for each roadway.
- *SR24kLRSRamp_20131231*: Contains geospatial features of all ramps in the state.
- *Traffic_Counts*: Contains AADT values and AADT dates for roadways.
- *WA_County_Bndys*: Contains the Washington State county boundaries.
- *WAPR*: Contains Washington State public road features and related information.

### IntermediateData Geodatabase

The *IntermediateData* geodatabase is used to store temporary data during data processing and all feature classes in *IntermedidateData* are temporary datasets. The geodatabase contains 2 feature datasets, *Intermediate* and *Temp*, and one table *intersection_tbl* (Figure 3.)

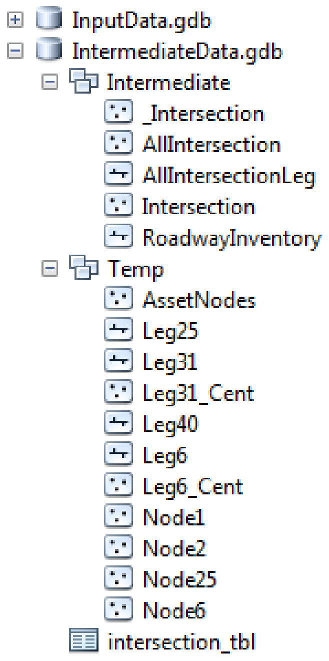
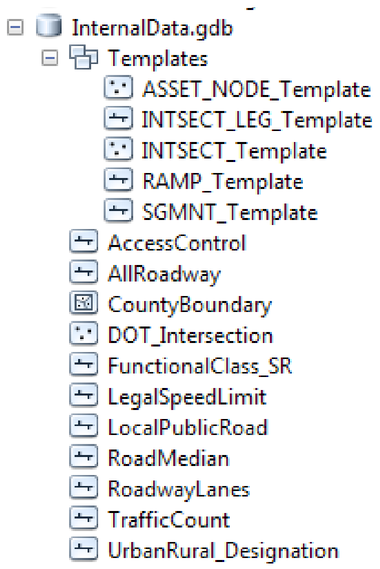

```
⊞ InputData.gdb
⊟ IntermediateData.gdb
    ⊟ Intermediate
        _Intersection
        AllIntersection
        AllIntersectionLeg
        Intersection
        RoadwayInventory
    ⊟ Temp
        AssetNodes
        Leg25
        Leg31
        Leg31_Cent
        Leg40
        Leg6
        Leg6_Cent
        Node1
        Node2
        Node25
        Node6
    intersection_tbl
```

**Figure 3. IntermediateData Geodatabase.**

The *Intermediate* feature dataset contains feature classes that are reused by several processes while the *Temp* feature dataset contains short-term feature classes. *Intermediate* and *Temp* feature datasets have the same coordinate system of the output dataset, which must be specified ahead of data processing. *Intersection_tbl* is used to store the calculated value of the intersection angle before it is moved to the final location in the *Intersection* feature class in the *MIREProject* geodatabase. To learn more about the use and purpose of each intermediate feature class, please review the description of the RDE tool models and submodels in the next chapter, which illustrate when each dataset is being used by the RDE tool.

## InternalData Geodatabase

The *InternalData* geodatabase is used to provide templates for the feature classes in the output geodatabase MIREProject.gdb and several datasets that are slight transformations of the input datasets (Figure 4). Templates are grouped into the feature dataset *Templates* that contains four feature class templates that are used to create the final output feature classes. Each template provides a listing of fields, data types for each field, indices, coordinate system definitions, subtypes as needed, resolution and tolerances, and feature extents.



```
⊟ InternalData.gdb
   ⊟ Templates
         ASSET_NODE_Template
         INTSECT_LEG_Template
         INTSECT_Template
         RAMP_Template
         SGMNT_Template
      AccessControl
      AllRoadway
      CountyBoundary
      DOT_Intersection
      FunctionalClass_SR
      LegalSpeedLimit
      LocalPublicRoad
      RoadMedian
      RoadwayLanes
      TrafficCount
      UrbanRural_Designation
```

**Figure 4. InternalData Geodatabase.**

A description of the feature class templates listed in Figure 4, and how the RDE tool makes use of them, follows. Appendix III provides a complete view of the field names and data types for each template. Note that most templates contain fields that are specific to the requirements of Safety Analyst software. Transportation agencies that use different safety analysis software would simply disregard (or delete) these fields.

- *ASSET_NODE_Template*: This template is used to create the *AssetNode* dataset which includes all nodes created by RDE tool.
- *INTSECT_Template*: This template contains all data elements depicting an intersection. It meets the requirements of Safety Analyst and the MIRE guideline by combining fields of both standards. The first 29 fields (in mixed case) originate from Safety Analyst and the remaining 18 fields (in all caps) originate from the MIRE guideline. The last two fields are additional fields that are required by RDE tool.
- *INTSECT_Leg_Template*: This template contains all data elements depicting an intersection leg. It meets the requirements of Safety Analyst and the MIRE guideline for intersection legs by combining fields of both standards. The first 15 fields (in mixed case) originate from Safety Analyst and the remaining 43 fields (in all caps) originate from the MIRE guideline. The last field is an additional fields that is required by RDE tool.

- *RAMP_Template*: This template contains all data elements depicting a ramp. It meets the requirements of Safety Analyst and the MIRE guideline for ramps by combining fields of both standards. The first 25 fields (in mixed case) originate from Safety Analyst and the remaining 20 fields (in all caps) originate from the MIRE guideline. The last field is an additional fields that is required by RDE tool.
- *SGMNT_Template*: This template contains all data elements depicting a segment. It meets the requirements of Safety Analyst and the MIRE guideline for segments by combining fields of both standards. The first 41 fields (in mixed case) originate from Safety Analyst and the remaining 96 fields (in all caps) originate from the MIRE guideline. The last field was added to the template as required by the RDE tool. The researchers included all fields from Safety Analyst in this template but not all fields of the MIRE template, because some of the MIRE elements duplicated Safety Analyst data elements.

The remaining 11 feature classes in the geodatabase *InternalData* are input datasets that are slight transformations of the original input datasets stored in the geodatabase *InputData* so that they can be processed by the RDE tool. For example, some of the input dataset names are changed to make them more meaningful.

- *AccessControl*: This is the imported version of the access control dataset provided by the agency (*RW_AccessControl2014*).
- *AllRoadway*: This is the imported version of the dataset that contains all public and private roads provided by the agency (*WAPR*).
- *CountyBoundary*: This is the imported version of the county boundary dataset provided by the agency (*WA_County_Bndys*).
- *DOT_Intersection*: This is the imported version of the intersection dataset provided by the agency (*RoadwayIntersections_2013*).
- *FunctionalClass_SR*: This is the imported version of the functional class dataset provided by the agency (*FunctionalClass_SR*).
- *LegalSpeedLimit*: This is the imported version of the legal speed limit dataset provided by the agency (*RW_LegalSpeedLimits2014*).
- *LocalPublicRoad*: This is the imported version of the local public road dataset provided by the agency (*LocalAgencyPublicRoads_Lines_2013*).
- *RoadMedian*: This is the imported version of the road median dataset provided by the agency (*RW_Median2014*).
- *RoadwayLanes*: This is the imported version of the roadway lanes dataset provided by the agency (*RW_Lanes2014*).
- *TrafficCount*: This is the imported version of the traffic count dataset provided by the agency (*Traffic_Counts*).
- *UrbanRural_Designation*: This is the imported version of the urban/rural dataset provided by the agency (*RW_UrbanRural2014*).

In addition to the templates and input datasets, the *IntermediateData.gdb* geodatabase provides a range of valid values, or domains, for a number of tables. Figure 5 provides a listing of the domain names and description that are stored in the geodatabase, along with an indication of whether they are currently used by the RDE tool. Domains that are currently not used by the RDE tool are included to facilitate future expansion of the RDE tool.

**Table 5. List of Tables with Defined Domain Values.**

| Domain Name | Domain Description | In Use |
|---|---|---|
| accessControl | Highway access control type | Segment |
| APR_PED_SGN_SPCL_FEAT_TYPE | Approach pedestrian signalization special feature type | IntersectionLeg |
| APR_RT_TRN_ON_RED_PRH_TYPE | Approach right-turn-on-red type | IntersectionLeg |
| APRCH_CRSWLK_TYPE | Approach crosswalk type | IntersectionLeg |
| APRCH_DRCT_FLOW_TYPE | Approach directional flow type | IntersectionLeg |
| APRCH_LT_RT_TURN_PRHB_TYPE | Approach left and right turn prohibition type | IntersectionLeg |
| APRCH_LT_TURN_LN_TYPE | Approach left turn lane type | IntersectionLeg |
| APRCH_LT_TURN_PROT_TYPE | Approach left turn protection type | IntersectionLeg |
| APRCH_MDN_TYPE | Approach median type | IntersectionLeg |
| APRCH_MODE | Approach transportation mode | IntersectionLeg |
| APRCH_PED_SGNLN_TYPE | Approach pedestrian signalization type | IntersectionLeg |
| APRCH_RT_TURN_CHNLZ_TYPE | Approach right turn channelization type | IntersectionLeg |
| APRCH_TRFC_CTRL_TYPE | Approach traffic control type | IntersectionLeg |
| AUX_LN_TYPE | Auxiliary lane type | Segment |
| CIR_IN_AP_EX_RT_TN_LN_TYPE | Circular intersection approach exclusive right turn lane type | IntersectionLeg |
| CIRC_INTS_BICY_FCLTY_TYPE | Circular intersection bicycle facility type | Intersection |
| CIRC_INTS_PED_FCLTY_TYPE | Circular intersection pedestrian facility type | IntersectionLeg |
| CNTRLN_RMBL_STRP_TYPE | Center lane rumble strip type | Segment |
| CNCD_RTE_TYPE | Coinciding route type | Segment |
| CURB_PRSC_TYPE | Curb presence type | Segment |
| CURB_TYPE | Curb type | Segment |
| discontinuity | Discontinuous roadway segment indicator | Segment |
| EXC_RT_TRN_LN_TRF_CTR_TYPE | Exclusive right-turn lane traffic control type | IntersectionLeg |
| FEDRL_AID_RTE_TYPE | Federal-aid route type | Segment |
| FUNC_CLASS_TYPE | Functional class type | Ramp |
| GOVT_OWNR_TYPE | Governmental ownership type | Ramp |

Table 5. List of Tables with Defined Domain Values (continued).

| Domain Name | Domain Description | In Use |
|---|---|---|
| growthSource | Growth source code | Intersection, Segment, Ramp |
| HOV_LN_TYPE | High occupancy vehicle lane type | Segment |
| interchangeInfluence | interchangeInfluence | Segment |
| intersectionType1 | Intersection type 1 | Intersection |
| intersectionType2 | Intersection type 2 | Intersection |
| INTSECT_GMTRY_TYPE | Intersection geometry type | Intersection |
| INTSECT_TRFC_CTRL_TYPE | Intersection traffic control type | Intersection |
| INTSECT_TYPE | Intersection type | Intersection |
| invalidIntersection | Invalid intersection code | Intersection |
| invalidRamp | Invalid ramp code | Ramp |
| invalidSegment | Invalid segment code | Segment |
| leftTurnPhasing | Left turn phasing code | IntersectionLeg |
| legDirection | Leg direction code | IntersectionLeg |
| legMedianType | Leg median type | IntersectionLeg |
| legType | Leg type | IntersectionLeg |
| majorRoadDirection | Major road direction | Intersection |
| MDN_BARR_TYPE | Median barrier type | Segment |
| MDN_CRSOVR_LN_TYPE | Median crossover lane type | Segment |
| medianType1 | Median type 1 | Segment |
| medianType2 | Median type 2 | Segment |
| offsetIntersection | Offset intersection code | Intersection |
| ON_ST_PRKG_PRSC_TYPE | On-street parking presence type | Segment |
| ON_ST_PRKG_TYPE | On-street parking type | Segment |
| operationWay | Operation way code | Segment |
| RAMP_METER_TYPE | Ramp meter type | Ramp |
| RAMP_TRMN_RDWY_FEAT_TYPE | Ramp terminal roadway feature type | Ramp |
| RAMP_TRMN_RDWY_TYPE | Ramp terminal roadway type | Ramp |
| RAMP_TRMN_RLTV_MNLN_TYPE | Ramp terminal relative mainlane type | Ramp |
| rampConfiguration | Ramp configuration code | Ramp |
| rampCrossroadConnection | Ramp crossroad connection code | Ramp |
| rampFreewayConnection | Ramp freeway connection code | Ramp |
| rampType | Ramp type | Ramp |
| RDWAY_LTG_TYPE | Roadway lighting type | Segment |
| roadwayClass1 | Roadway class code 1 | Segment |

**Table 5. List of Tables with Defined Domain Values (continued).**

| Domain Name | Domain Description | In Use |
|---|---|---|
| roadwayClass2 | Roadway class code 2 | Segment |
| roadwayClass3 | Roadway class code 3 | Segment |
| RTE_SIGN_QLFY_TYPE | Route signing qualifier type | Segment |
| RTE_SIGN_TYPE | Route signing type | Segment |
| RURL_URB_DSGNT | Rural-urban designation | IntersectionLeg, Segment |
| SDWALK_TYPE | Sidewalk type | Segment |
| SGNL_PROG_TYPE | Signal progression type | IntersectionLeg |
| SGNLN_TYPE | Signalization type | Intersection |
| SHLDR_RMBL_STRP_TYPE | Shoulder rumble strip type | Segment |
| SHLDR_TYPE | Shoulder type | Segment |
| SRFC_TYPE | Surface type | Segment |
| STATE_FIPS_CODE | State federal information processing standard code | Not in use |
| Status | Feature status code | Intersection |
| terrain | Terrain code | Segment |
| TOLL_FCLTY_TYPE | Toll facility type | Segment |
| trafficControl1 | Traffic control 1 | Intersection |
| travelDirection | Traffic control 2 | Segment |
| TRIBAL_RESERVATION_FIPS_CODE | Tribal reservation federal information processing standard code | Not in use |
| turnProhibitions | Turn prohibitions code | IntersectionLeg |
| YES_NO | Yes-no code | Intersection, IntersectionLeg, Ramp |

## MIREProject Geodatabase

The *MIREProject* geodatabase contains the output feature classes created by the RDE tool. After executing all models in RDE tool, the *MIREProject* geodatabase should contain 5 feature classes: *AssetNode*, *Intersection*, *IntersectionLeg*, *Ramp*, and *Segment* (Figure 5). The coordinate system of these feature classes is determined by the specific coordinate system used by each state.

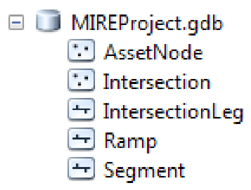


**Figure 5. MIREProject Feature Classes.**

The following provides a description of the feature classes in *MIREProject*.

- *AssetNode*: This feature class contains all possible intersecting points between two roadways in the input roadway dataset. Note that not all of these points have corresponding intersections on the ground.

- *Intersection*: This feature class contains all intersection features provided by the state DOT. The feature class uses the generic intersection template *INTSECT_Template*.
- *IntersectionLeg*: This feature class contains all intersection leg features generated by the RDE tool. The feature class uses the generic intersection leg template *INTSECT_LEG_Template*.
- *Ramp*: This feature class contains all ramp geometry provided by the state DOT. The feature class uses the generic ramp template *RAMP_Template*.
- *Segment*: This feature class contains all segment geometry provided by the state DOT. It uses the generic segment template *SGMNT_Template*.

### UpdateFeature Geodatabase

The *UpdateFeature* geodatabase is used to store temporary datasets that are created when a user runs model *6 Prepare Update Data*, *7 Update or Retire Intersections*, or *8 Update Legs and Intersections*. During the updating of the features, the RDE tool performs many alteration to the input dataset. Instead of altering the original input datasets, the RDE tool creates copies of the input datasets and make modification on the copies. At each modification stage, the RDE tool creates a separate copy of the data. Once the models complete successfully, the final output is copied into appropriate feature classes of the *MIREProject* geodatabase, and the temporary feature classes in *UpdateFeature* are deleted.

## 5. RDE Tool Toolboxes and Scripts

The RDE tool contains three toolboxes that can produce datasets formatted for use in Safety Analyst or other types of safety analysis tools, using a series of ArcGIS models:

- *MIRE_3*
- *MIRE_support*
- *MIRE_update*

The RDE tool models are capable of preparing datasets for intersections, intersection legs, segments, and ramps. Using the models, users can prepare datasets with large number of intersections, intersection legs, segments, or ramps in just one session. The models gather data from as many sources as needed, reformat the data as needed, calculate certain new data fields, and attach the data to the output datasets.

Users of the RDE tool only need to execute models in the toolbox MIRE *MIRE_3*, *MIRE_support* and *MIRE_update* contain models that are executed by models in *MIRE_3*. Figure 6 shows the eight models that are included in the *MIRE_3* toolbox.

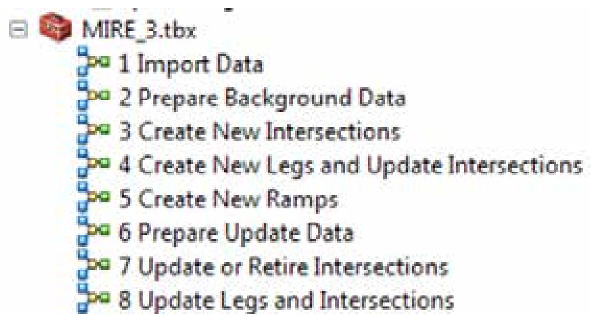

MIRE_3.tbx
1 Import Data
2 Prepare Background Data
3 Create New Intersections
4 Create New Legs and Update Intersections
5 Create New Ramps
6 Prepare Update Data
7 Update or Retire Intersections
8 Update Legs and Intersections

**Figure 6. MIRE Model Geodatabase.**

The *MIRE_3* toolbox supports three business processes:

1. Creating new features (intersections, intersection legs, segments, and ramps) that are compatible with safety analysis software, and populating the features with available data.
2. Updating a dataset of features (intersection, intersection legs, segments, and ramps) once source datasets are updated or modified.
3. Retiring intersection features if an intersection does no longer exist in the field, but the data manager wants to retain the current intersection and intersection leg data.

Models one through five support the creation of new features, while models six through 8 support the updating of features and the retiring of features. Figure 7 and Figure 8 provide an overview of the general workflow to execute the models in *MIRE_3* to create, update, or retire features.

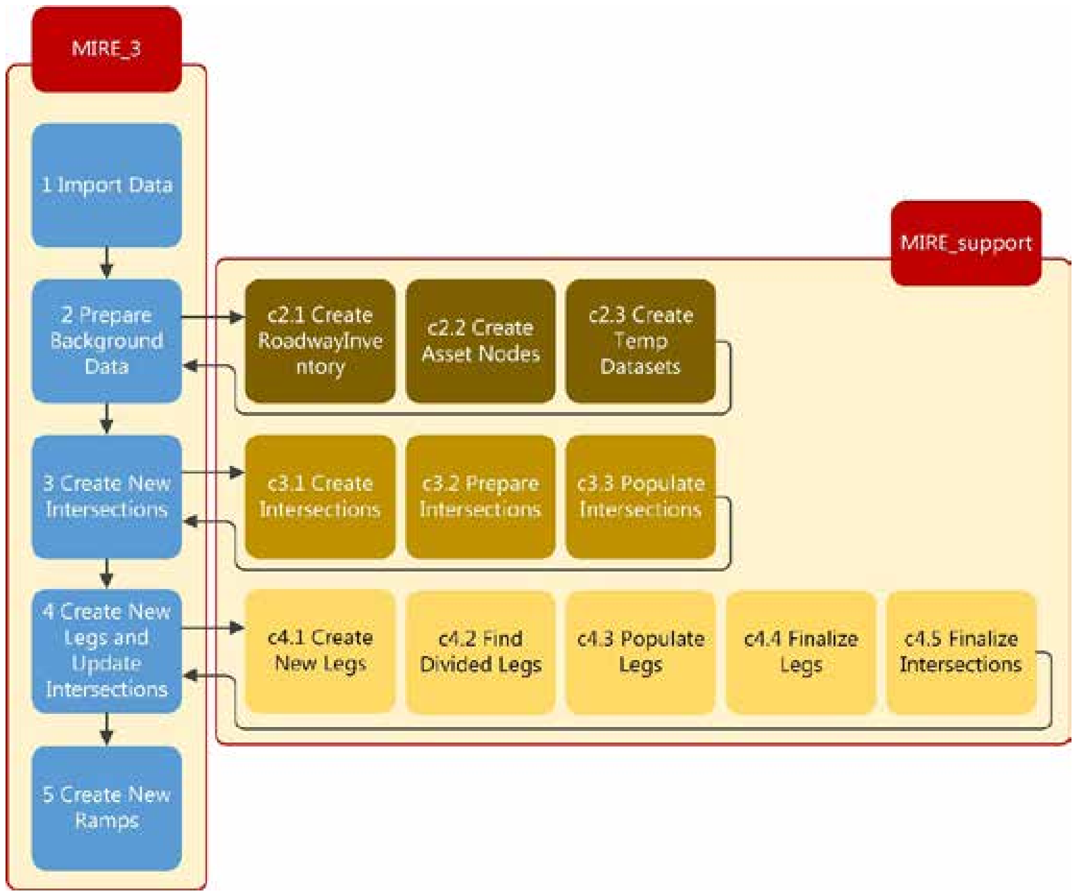

MIRE_3
1 Import Data
2 Prepare Background Data
3 Create New Intersections
4 Create New Legs and Update Intersections
5 Create New Ramps
MIRE_support
c2.1 Create RoadwayInventory
c2.2 Create Asset Nodes
c2.3 Create Temp Datasets
c3.1 Create Intersections
c3.2 Prepare Intersections
c3.3 Populate Intersections
c4.1 Create New Legs
c4.2 Find Divided Legs
c4.3 Populate Legs
c4.4 Finalize Legs
c4.5 Finalize Intersections

**Figure 7. Model Relationships and Execution Sequence for "Create" Business Process.**

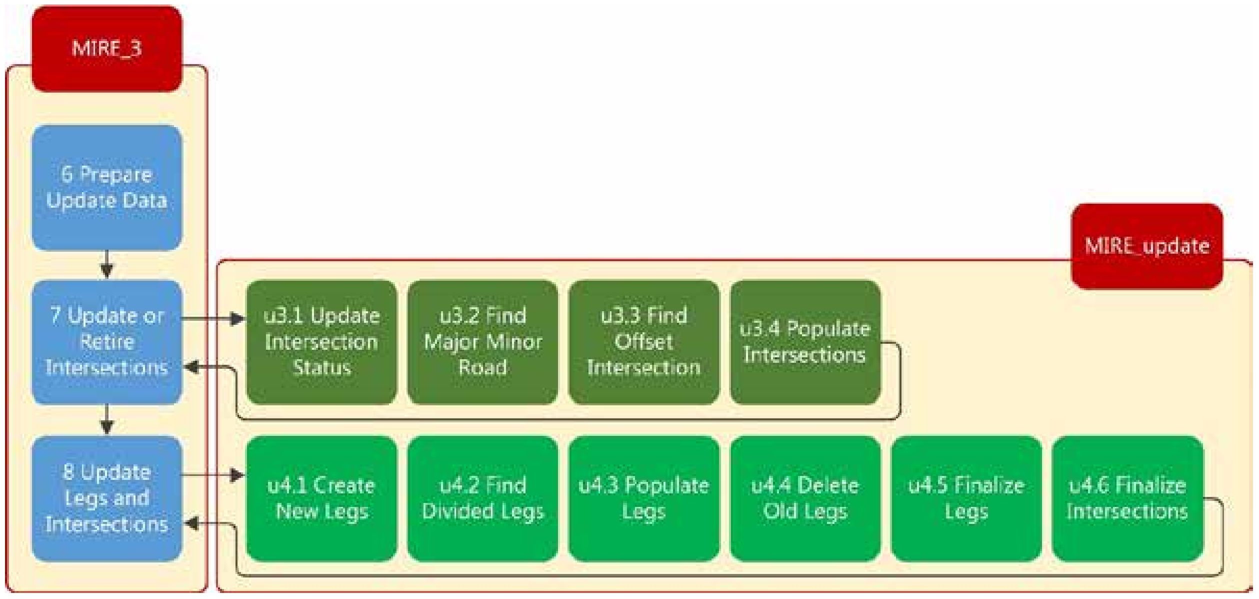

MIRE_3
6 Prepare Update Data
7 Update or Retire Intersections
8 Update Legs and Intersections
MIRE_update
u3.1 Update Intersection Status
u3.2 Find Major Minor Road
u3.3 Find Offset Intersection
u3.4 Populate Intersections
u4.1 Create New Legs
u4.2 Find Divided Legs
u4.3 Populate Legs
u4.4 Delete Old Legs
u4.5 Finalize Legs
u4.6 Finalize Intersections

**Figure 8. Model Relationships and Execution Sequence for "Update" and "Retire" Business Process.**

## Create Business Process

To create new features for safety analysis, users should execute the first five models in the *MIRE_3* toolbox. The models serve the following purpose:

- *1 Import Data*: Import input datasets from the transportation agency's geodatabase into the local geodatabase.
- *2 Prepare Background Data*: Create the *RoadwayInventory* and *AssetNode* datasets, which are needed for subsequent steps.
- *3 Create New Intersections*: Create the *Intersection* dataset from the provided input datasets. Alternatively, the tool can be configured to use an existing intersection dataset and use its geospatial intersection information, instead of creating new intersection features. However, this might create problems with the creation of new intersection legs if the existing intersection features do not coincide with the geospatial roadway data provided.
- *4 Create New Legs and Update Intersections*: Create the *IntersectionLeg* dataset based on input data and geospatial roadway line work.
- *5 Create New Ramps*: Create and populate the *Ramp* dataset using input datasets.

A detailed description of the main functions within each model follow below.

### *Model 1 Import Data*

Figure 9 provides an overview of the model's functions and relationships as seen in model builder. The main purpose of this model is to bring all input datasets from various data sources into one single geodatabase. As shown, the model uses the ArcGIS *Feature Class to Feature Class* tool to import the transportation agency's datasets into the RDE tools *InternalData geodatabase. The imported datasets can then be used by subsequent models. 1 Import Data* does not execute any submodels.

Note that it is important to ensure that all datasets that will be imported use the same coordinate system. In addition, each dataset has fields that are required for the model to execute. These fields are dependent on required fields in other subsequent models, but can be adjusted as needed. The "P" adjacent to a model input indicates that the input is a parameter that can be changed by the user on the interface when executing the model.

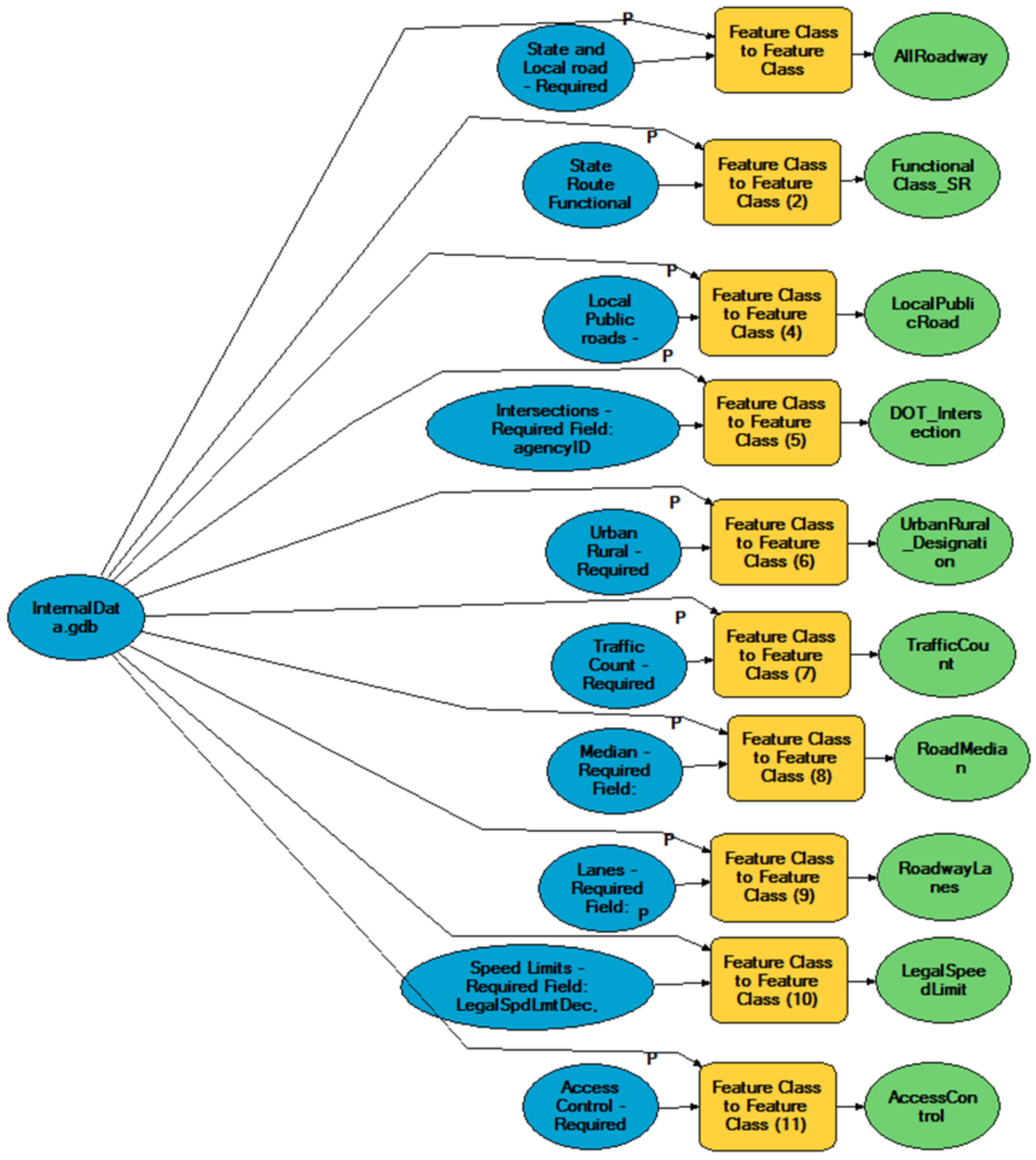

InternalDat a.gdb
State and Local road - Required
Feature Class to Feature Class
AllRoadway
State Route Functional
Feature Class to Feature Class (2)
FunctionalClass_SR
Local Public roads -
Feature Class to Feature Class (4)
LocalPublicRoad
Intersections - Required Field: agencyID
Feature Class to Feature Class (5)
DOT_Intersection
Urban Rural - Required
Feature Class to Feature Class (6)
UrbanRural_Designation
Traffic Count - Required
Feature Class to Feature Class (7)
TrafficCount
Median - Required Field:
Feature Class to Feature Class (8)
RoadMedian
Lanes - Required Field:
Feature Class to Feature Class (9)
RoadwayLanes
Speed Limits - Required Field: LegalSpdLmtDec.
Feature Class to Feature Class (10)
LegalSpeedLimit
Access Control - Required
Feature Class to Feature Class (11)
AccessControl
P

**Figure 9. Model 1 Import Data.**

### *Model 2 Prepare Background Data*

This model creates background datasets that are used by subsequent models in order to create intersections and intersection legs. Due to the complexity of the model, it is split into three submodels that are stored in the *MIRE_support* toolbox: *c2.1 Create RoadwayInventory*, *c2.2 Create Asset Nodes*, and *c2.3Create Temp Datasets*. Figure 10 provides an overview of the model and submodel relationships. Note that Figure 10 includes both solid lines and dashed lines between model components. Solid lines indicate that data or information is being transferred, while dashed lines indicate that a component is a precondition to a function, meaning that it has to complete before the next function can execute.

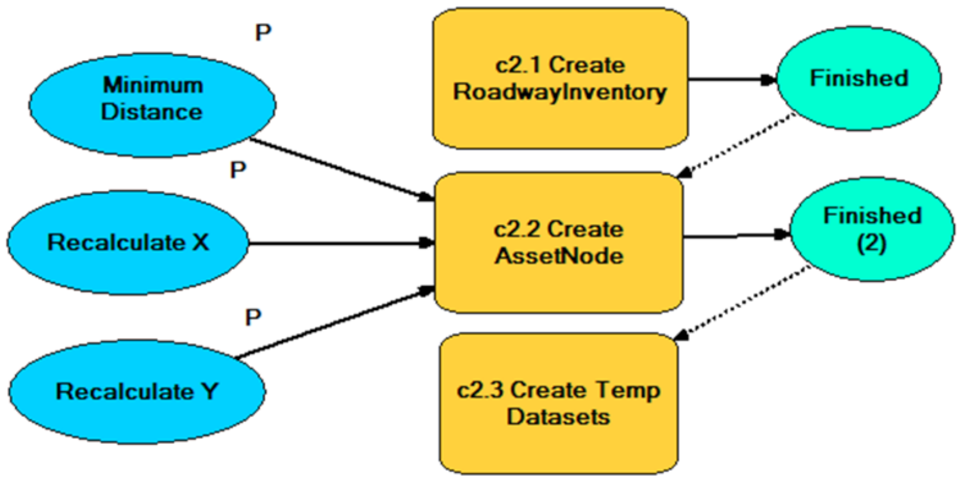


**Figure 10. Model 2 Prepare Background Data.**

### Submodel c2.1 Create Roadway Inventory

This submodel uses geometry from the roadway dataset and attribute data from various datasets to create the *RoadwayInventory* feature class. The submodel is too large and complex to show in this guide, therefore programmers are advised to review the model in ArcCatalog using ArcGIS ModelBuilder.

The model gathers some data that are needed to populate the *Intersection* and *IntersectionLeg* feature classes and to determine the major versus minor road at an intersection. Attribute data used to determine major versus minor road include the roadway's Federal Functional Class, F_System code, route type, AADT, State route number, and serveral others. Determination of major versus minor roadway should be adjusted based on the data available at the transportation agency.

### Submodel c2.2 Create Asset Nodes

This submodel creates the *AssetNode* dataset which contains all possible intersecting points between roads, based on the transportation agency's roadway network data. The submodel is too large and complex to show in this guide, therefore programmers are advised to review the model in ArcCatalog using ArcGIS ModelBuilder.

The submodel automatically removes spatially duplicate points and removes points that are too close to each other, since these points are unlikely to have corresponding intersections in the field. The default minimum distance between nodes is set to 10 feet but can be adjusted as needed. The RDE tool uses asset nodes to create intersection features (if enabled) and to create intersection legs that are connected precisely to each intersection.

### Submodel c2.3 Create Temp Datasets

This submodel creates temporary datasets that are needed to execute model *3 Create New Intersections* and model *4 Create New Legs and Update Intersections*. The temporary datasets are created using intermediate data and the templates *INTSECT_Template* and *INTSECT_LEG_Template*. Figure 11 provides an overview of the submodel.

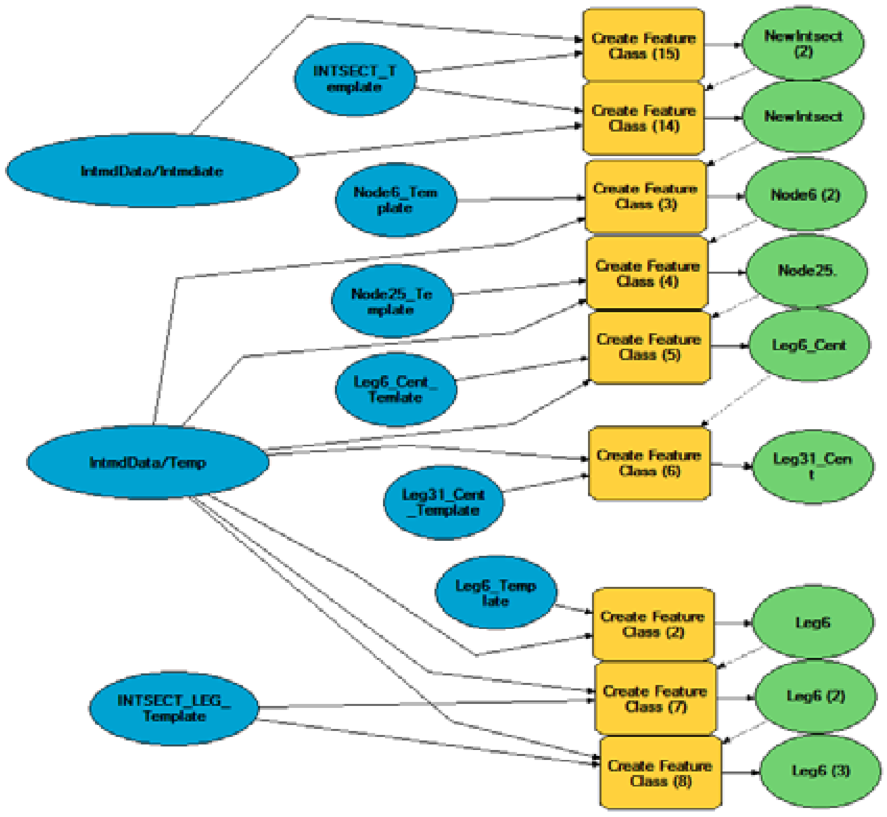


**Figure 11. Submodel c2.3 Create Temp Datasets.**

### *Model 3 Create New Intersections*

This model uses intersection features provided by the transportation agency and combines the data with several input datasets to create the final output *Intersection* dataset. Due to the complexity of the model it is split into three submodels that are stored in the *MIRE_support* toolbox: *c3.1 Create Intersections*, *c3.2 Prepare Intersections*, and *c3.3 Populate Intersections*. Figure 12 provides an overview of the model and submodel relationships.

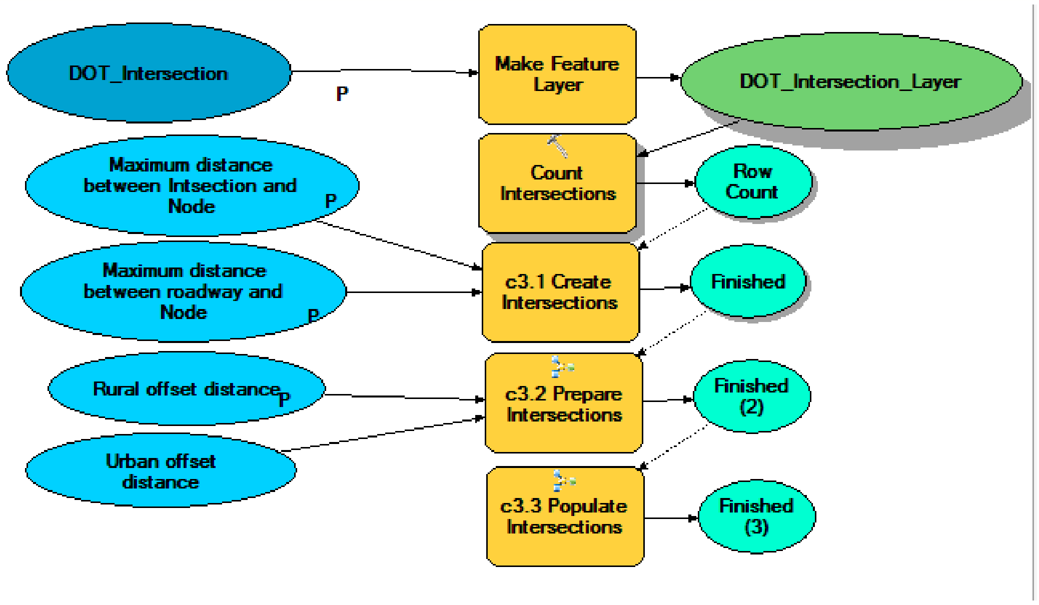

DOT_Intersection
P
Make Feature Layer
DOT_Intersection_Layer
Maximum distance between Intsection and Node
P
Count Intersections
Row Count
Maximum distance between roadway and Node
P
c3.1 Create Intersections
Finished
Rural offset distance
P
c3.2 Prepare Intersections
Finished (2)
Urban offset distance
c3.3 Populate Intersections
Finished (3)

**Figure 12. Model 3 Create New Intersections.**

As shown in Figure 12, the model counts the number of intersection features of the *DOT_Intersection* feature class that was created in an earlier step based on various intersection datasets provided by the transportation agency. If *DOT_Intersection* contains any intersection features, the model will execute, otherwise it will stop. The output of the model is the *Intersection* feature class populated with attribute data from the input datasets.

## Submodel c3.1 Create Intersections

This submodel scans all nodes in the *AssetNode* dataset and selects the nodes that are closest to the intersection features provided by the transportation agency's dataset. Nodes in the *AssetNode* dataset are intersecting points between two roadways, so not all of these nodes represent actual intersections in the field. A user can specify the maximum distance between a node and a nearby intersection parameter before executing the model, the default value is 30 feet. If a node does not have any nearby intersection, it will not be selected. After selecting nodes with nearby intersections, the tool imports data from the *RoadwayInventory* dataset to populate the Intersection dataset. As such it must find roadway features in the vicinity of each node. The distance threshold between a node and a roadway can be set in the "Maximum distance between roadway and Node" parameter, the default value is 10 feet.

Note that this model uses certain attribute data in the *RoadwayInventory* dataset to determine major versus minor roads at each intersection. Since states use different roadway attributes to determine major and minor roadways, including AADT, route type, and federal highway classification, it is important to review the code and ensure that it executes correctly. The submodel is too large and complex to show in this guide, therefore programmers are advised to review the model in ArcCatalog using ArcGIS ModelBuilder.

### Submodel c3.2 Prepare Intersections

This submodel attempts to find offset intersections and measure the actual offset distance for each intersection, which is the distance between the centerlines of the intersecting legs at an intersection. If the offset distance is greater than zero, the intersection is identified as an offset intersection. Separate minimums can be specified for urban and rural areas at which an intersection is considered an offset intersection, before executing the submodel. The default value for urban areas is 50 feet and for rural areas 100 feet. The submodel is too large and complex to show in this guide, therefore programmers are advised to review the model in ArcCatalog using ArcGIS ModelBuilder.

### Submodel c3.3 Populate Intersections

This submodel combines intersection features provided by the transportation agency, attribute data from the previous submodels, and the RDE tool templates to create the *Intersection* dataset. Once created, the submodel copies the *Intersection* dataset to the *MIREProject* geodatabase. Figure 13 provides an overview of the submodel and relationships.

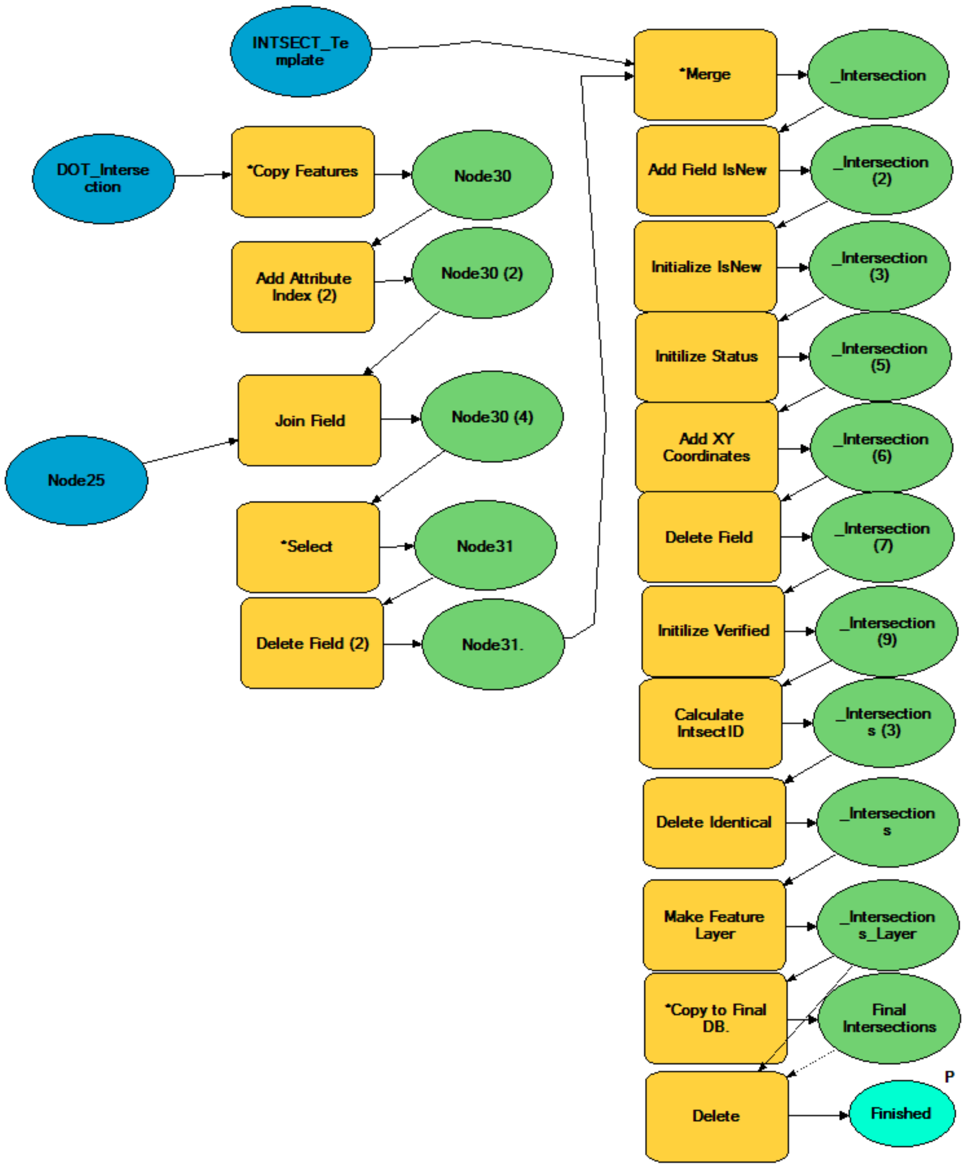


**Figure 13. Submodel c3.3 Populate Intersections.**

### *Model 4 Create New Legs and Update Intersections*

This model creates new intersection legs around each intersection created by model *3 Create New Intersections* and populates these legs with data from input datasets. Due to the complexity of the model, it is split into five submodels that are stored in the MIRE_support toolbox: *c4.1 Create New Legs, c4.2 Find Divided Legs, c4.3 Populate Legs, c4.4 Finalize Legs*, and *c4.5 Finalize Intersections*. Figure 14 provides an overview of the model's functions and relationships as seen in model builder.

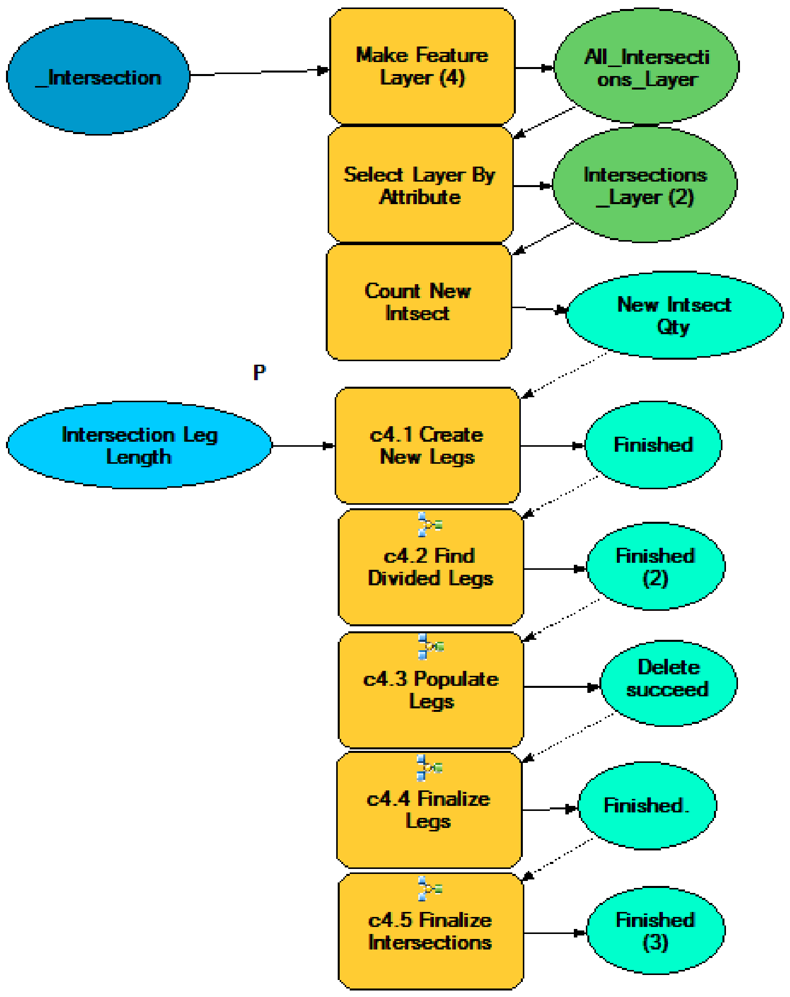


**Figure 14. Create New Legs and Update Intersections.**

The model first counts the number of new intersection points that are available. If there is at least one intersection, the model will proceed to create legs, otherwise the model will stop. The following provides a description of the purpose and process of each submodel shown in Figure 14. With the exception of submodel *c4.5 Finalize Intersection*s, all submodels are too large and complex to show in this guide, therefore programmers are advised to review the models in ArcCatalog using ArcGIS ModelBuilder.

### Submodel c4.1 Create New Legs

This submodel creates a 50 feet buffer around each asset node, which determines the length of the intersection legs. The buffer length can be changed as a model parameter before executing the model.

The submodel intersects the buffer with the *RoadwayInventor*y feature class to create a set of potential or temporary intersection legs. The submodel then selects intersection legs that are at least 50 feet long and less than four feet away from any intersection, which become the permanent or final intersection legs. The model adds the field *LegID* to all permanent legs and sets the *LegID* value equal to the value in the field *OBJECTID*. The submodel then intersects the intersection legs with the *RoadwayInventory* feature class to get attribute data such route ID, federal functional class, route unique ID, route type, and any other attributes that are needed to populate intersection legs.

### Submodel c4.2 Find Divided Legs

This submodel applies when a transportation agency needs to identify intersection legs on divided highways. A requirement for the submodel to work is that the *RoadwayInventory* dataset has information to distinguish a divided highway from an undivided highway. The submodel spatially joins intersection legs with the *RoadwayInventory* feature class to determine if a leg was created from a divided highway or not. If the leg was created from a divided highway feature, the leg is considered a divided highway leg.

### Submodel c4.3 Populate Legs

Depending on the available input datasets, the submodel spatially joins intersection leg features with input datasets such as road median, number of lanes, traffic count, speed limit, access control, and similar geospatial data. As a result, the submodel populates the intersection leg fields with data from the spatial joins.

### Submodel c4.4 Finalize Legs

This submodel merges all intersection leg features created in previous steps with the intersection leg template to create the *IntersectionLeg* feature class. The submodel also spatially joins the intersection legs with the urban/rural dataset to determine if a leg is in an urban or rural area. The submodel then copies the *IntersectionLeg* feature class to the final location, i.e., the *MIREProject* geodatabase.

### Submodel c4.5 Finalize Intersections

This submodel calculate the intersection angle value, which is the smallest angle between any two legs at an intersection. This angle can only be calculated after the legs have been crated. The submodel resets the field *IsNew* to “No” because at this point all intersections have been updated and are no longer considered new intersections. The model also extracts the *agencyID* from the input data and copies the value from *NodeID* into the field *intersectionI*D. The submodel then merges the intersection features with the intersection template (*INTSECT_Template*) to create the final Intersections feature class, and copies the feature class to the final location, i.e. the *MIREProject* geodatabase. Figure 15 provides an overview of the submodel’s functions and relationships as seen in model builder.

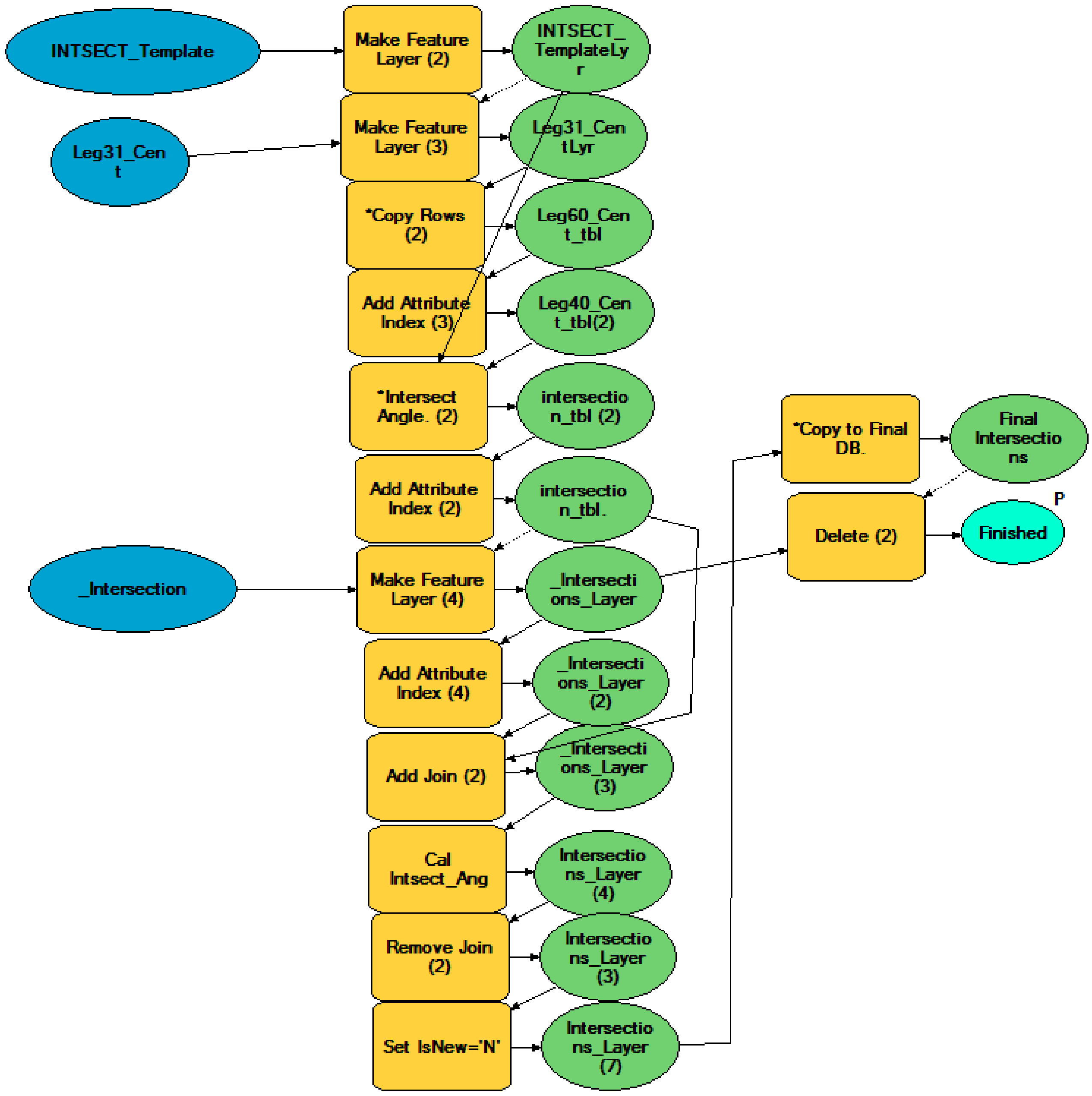


Figure 15. Submodel c4.5 Finalize Intersections.

### *Model 5 Create New Ramps*

This model merges the ramp template with the ramp input dataset from transportation agency to create the feature class *Ramp*. The model also calculates the length of the ramp feature and sets the value of the field *Verified* to “No.” Figure 16 provides an overview of the model’s functions and relationships as seen in model builder. Model 5 *Create New Ramps* does not execute any submodels.

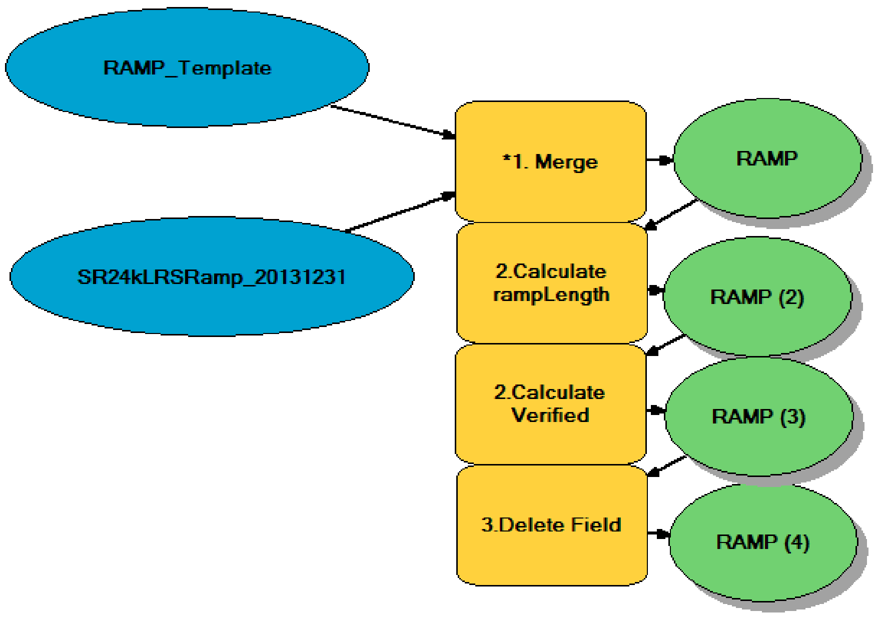


**Figure 16. Model 5 Create New Ramps.**

## Update Features Business Process

To update existing features for safety analysis when new data becomes available in the source datasets, the last three models in the *MIRE_3* toolbox should be executed. When updating features, these models serve the following purpose:

- *6 Prepare Update Data*: This model creates several temporary datasets necessary for subsequence models.
- *7 Update or Retire Intersections*: This model updates attribute values and can mark certain intersections as "Retired."
- *8 Update Legs and Intersections*: This model creates new intersection legs for newly created intersections and calculates certain values such as intersection angle.

A detailed description of the main functions within each model follow below.

### *Model 6 Prepare Update Data*

Before executing a model, ArcGIS needs to validate the model. The validation process requires a number of temporary datasets that are used by all submodels. *Model 6 Prepare Update Data* creates all temporary datasets that are needed to validate model *7 Updated or Retire Intersections* and model *8 Update Legs and Intersections*.

The model is too large and complex to show in this guide, therefore programmers are advised to review the models in ArcCatalog using ArcGIS ModelBuilder. *6 Prepare Update Data* does not execute any submodels.

### *Model 7 Update or Retire Intersections*

When a user intends to retire intersections by removing the underlying nodes, model 7 will update the intersection status to “Retired.” If a user deletes intersections, it is not required to execute model 6, 7, or 8. Similarly, if a user changes attribute data of intersections or intersection legs, model 6, 7, and 8 would not need to be executed. However, execution of these models is mandatory when a user creates a new intersection using the RDE tool toolbar. When new intersections are added to the intersection dataset using this procedure, most of the intersection attributes are missing. Executing model *7 Update or Retire Intersections* creates a set consisting of only new intersections (field *IsNew* = "*Yes*") and populates these new intersections the same way Model 3 does.

Due to the complexity of the model, it is split into four submodels that are stored in the MIRE_update toolbox: *u3.1 Update Intersection Status, u3.2 Find Major Minor Road, u3.3 Find Offset Intersection, u3.4 Populate Intersections*. Figure 17 provides an overview of the model and submodel relationships.

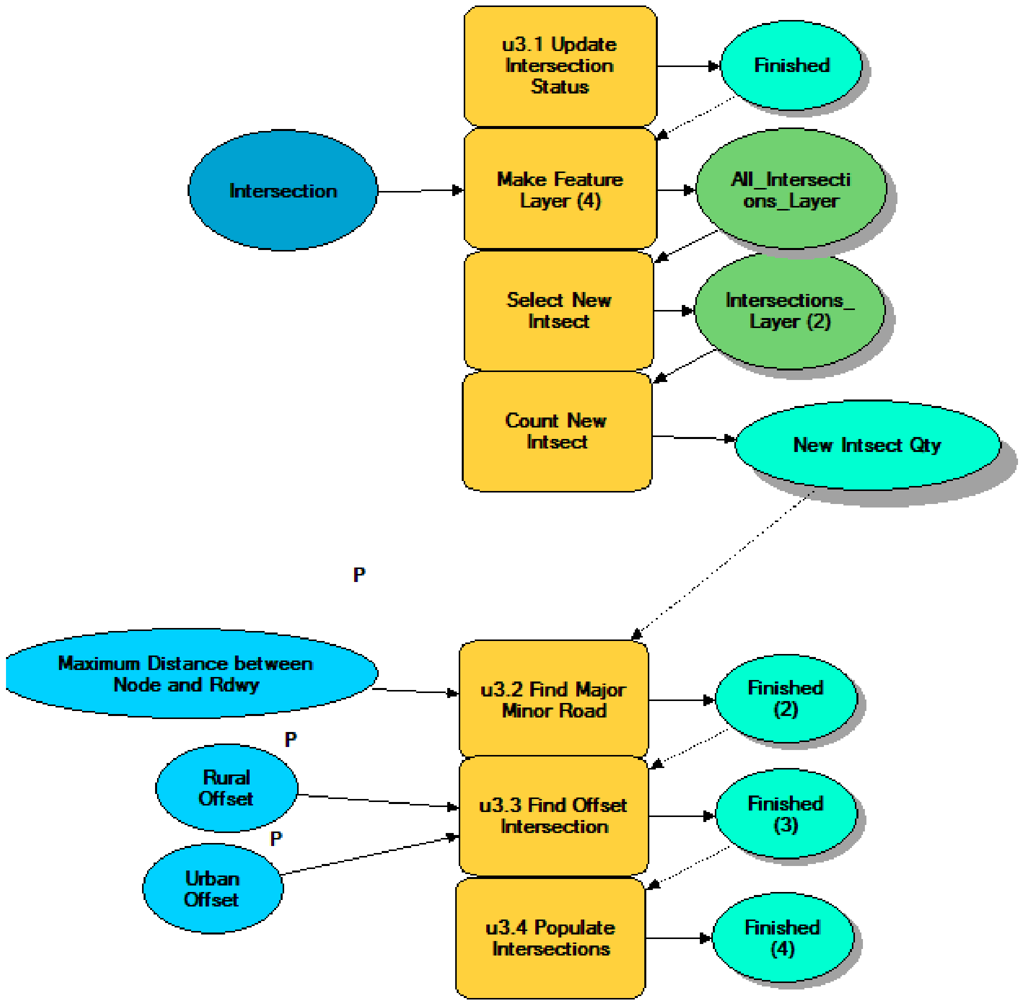


**Figure 17. Model 7 Update or Retire Intersections.**

### Submodel u3.1 Update Intersection Status

This submodel looks creates a set of intersections that do not have an underlying asset node. For all intersection features in this set the model will update the *Status* field to "Retired." Figure 18 provides an overview of the submodel.

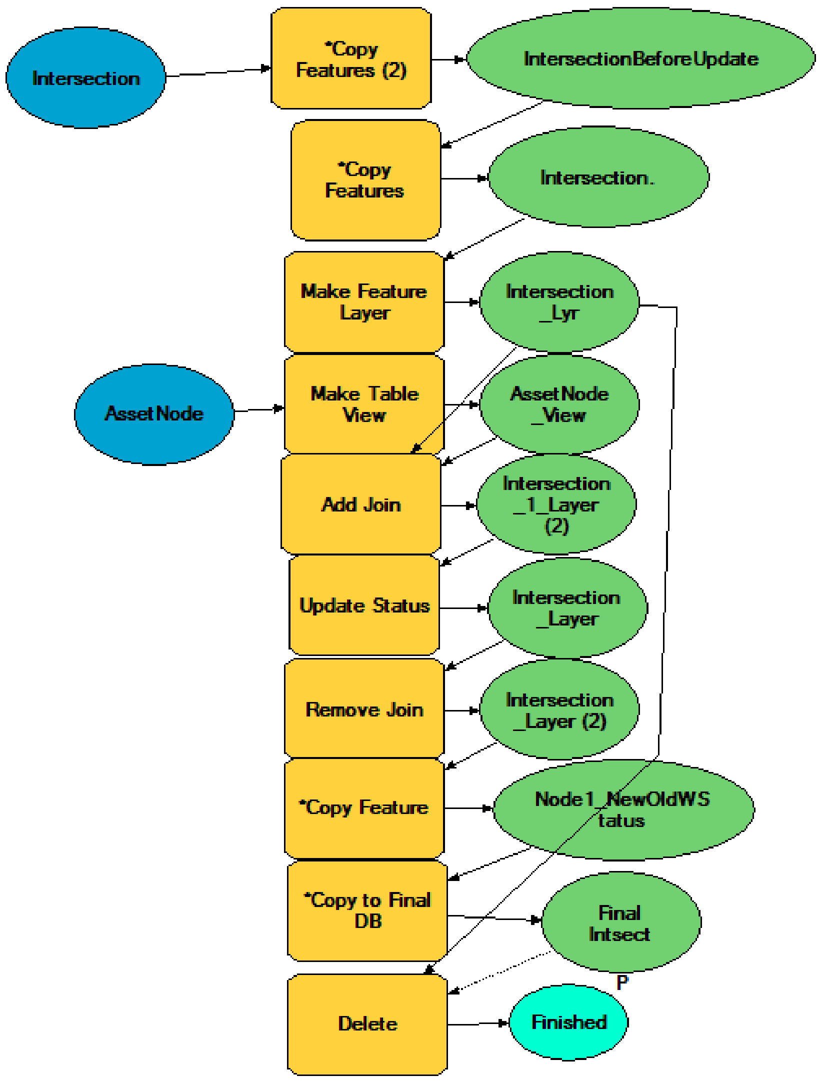


**Figure 18. Submodel u3.1 Update Intersection Status.**

### Submodel u3.2 Find Major Minor Road

This submodel finds the nearest roadways for each new intersection. Each intersection feature usually has at least two nearby roadways. The submodel uses several attributes of the nearest roadways to determine which road is the major and which is the minor road. These attributes should be adjusted based on available data, and different states use different attributes to determine major/minor roads. Attributes currently used include route type, federal functional class, route ID, F_System, etc. The model updates only new intersections (*IsNew* = "Yes.") The submodel is too large and complex to show in this guide, therefore programmers are advised to review the model in ArcCatalog using ArcGIS ModelBuilder.

### Submodel u3.3 Find Offset Intersection

This submodel attempts to locate offset intersections for the new intersections only. It works the same way as submodel *c3.2 Prepare Intersections*. The submodel attempts to find offset intersections and measure the actual offset distance for each intersection, which is the distance between the centerlines of the intersecting legs at an intersection. If the offset distance is greater than zero, the intersection is identified as an offset intersection. Separate minimums can be specified for urban and rural areas at which an intersection is considered an offset intersection, before executing the submodel. The default value for urban areas is 50 feet and for rural areas 100 feet. The submodel is too large and complex to show in this guide, therefore programmers are advised to review the model in ArcCatalog using ArcGIS ModelBuilder.

### Submodel u3.4 Populate Intersections

This submodel attempts to populate attribute data for new intersections only. It works the same way as submodel *c3.3 Populate Intersections*. The submodel combines intersection features provided by the transportation agency, attribute data from the previous submodels, and the RDE tool templates to create the *Intersection* dataset. Once created, the submodel copies the *Intersection* dataset to the *MIREProject* geodatabase. Figure 19 provides an overview of the submodel.

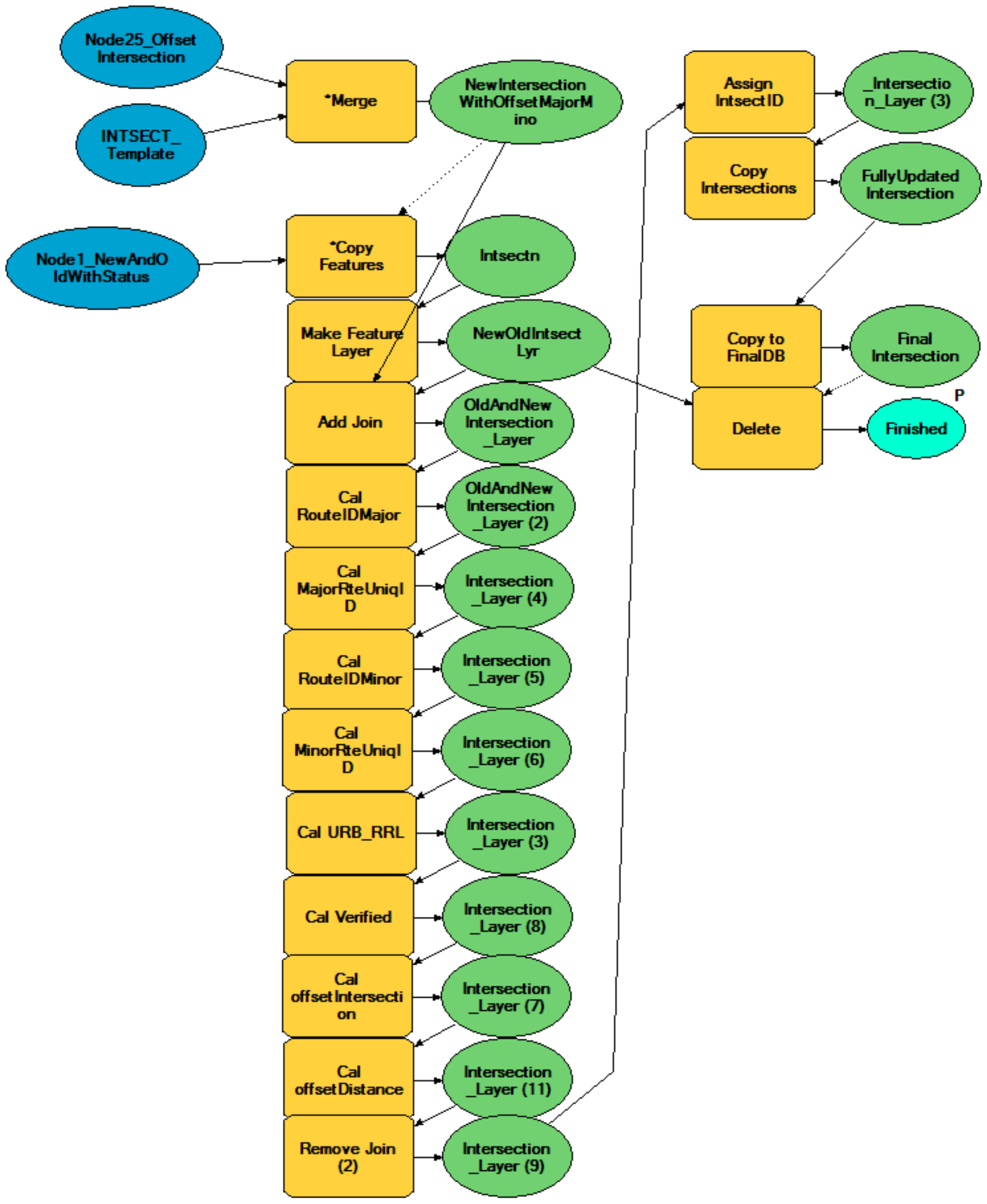


**Figure 19. Submodel u3.4 Populate Intersections.**

### *Model 8 Update Legs and Intersections*

This model attempts to find newly created intersections (field *IsNew* = "Yes") and creates legs for these intersections. Due to the complexity of the model, it is split into five submodels that are stored in the *MIRE_update* toolbox: *u4.1 Create New Legs*, *u4.2 Find Divided Legs*, *u4.3 Populate Legs*, *u4.4 Finalize Legs*, and *u4.5 Finalize Intersections*. If the model cannot find any new intersections it will stop executing. If the model finds any new intersections it will execute all submodels. Figure 20 provides an overview of the model and submodel relationships.

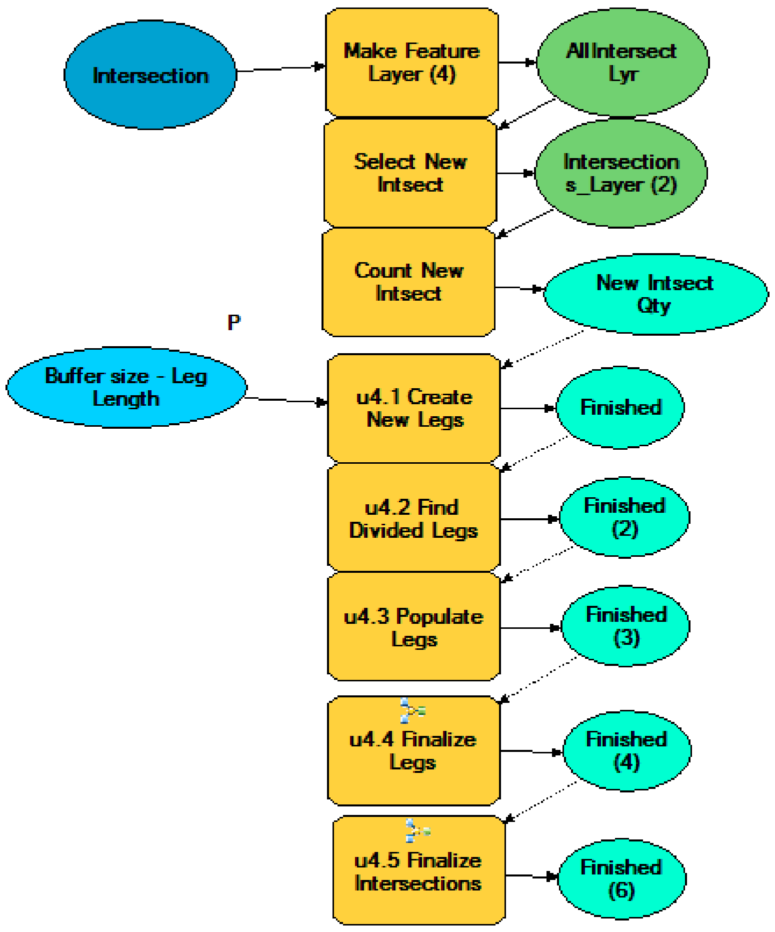


**Figure 20. Model 8 Update Legs and Intersections.**

The submodels are too large and complex to show in this guide, therefore programmers are advised to review the models in ArcCatalog using ArcGIS ModelBuilder.

### Submodel u4.1 Create New Legs

This submodel creates new legs around new intersections the same way as submodel c4.1

### Submodel u4.2 Find Divided Legs

This submodel finds divided legs among new legs and works the same way as submodel c4.2.

### Submodel u4.3 Populate Legs

This submodel populate new intersection legs and works the same way as submodel c4.3.

### Submodel u4.4 Finalize Legs

This submodel creates the final version of the intersection legs and copies them to the *IntersectionLeg* feature class in the *MIREProject* geodatabase. It works the same way as submodel c4.4.

### Submodel u4.5 Finalize Intersections

This submodel calculates the intersection angle for new intersections, merges old intersections with new intersections to create the final *Intersections* feature class, and copies the feature class to the final location, i.e. the *MIREProject* geodatabase.

## Retire Intersection Features Business Process

To retire existing intersection features, users should execute model *6 Prepare Update Data* and model *7 Update or Retire Intersections* in the *MIRE_3* toolbox. When retiring intersection features, the models serve the following purpose:

- *6 Prepare Update Data*: Creates a temporary datasets to allow model validation.
- *7 Update or Retire Intersections*: Retire intersection features.

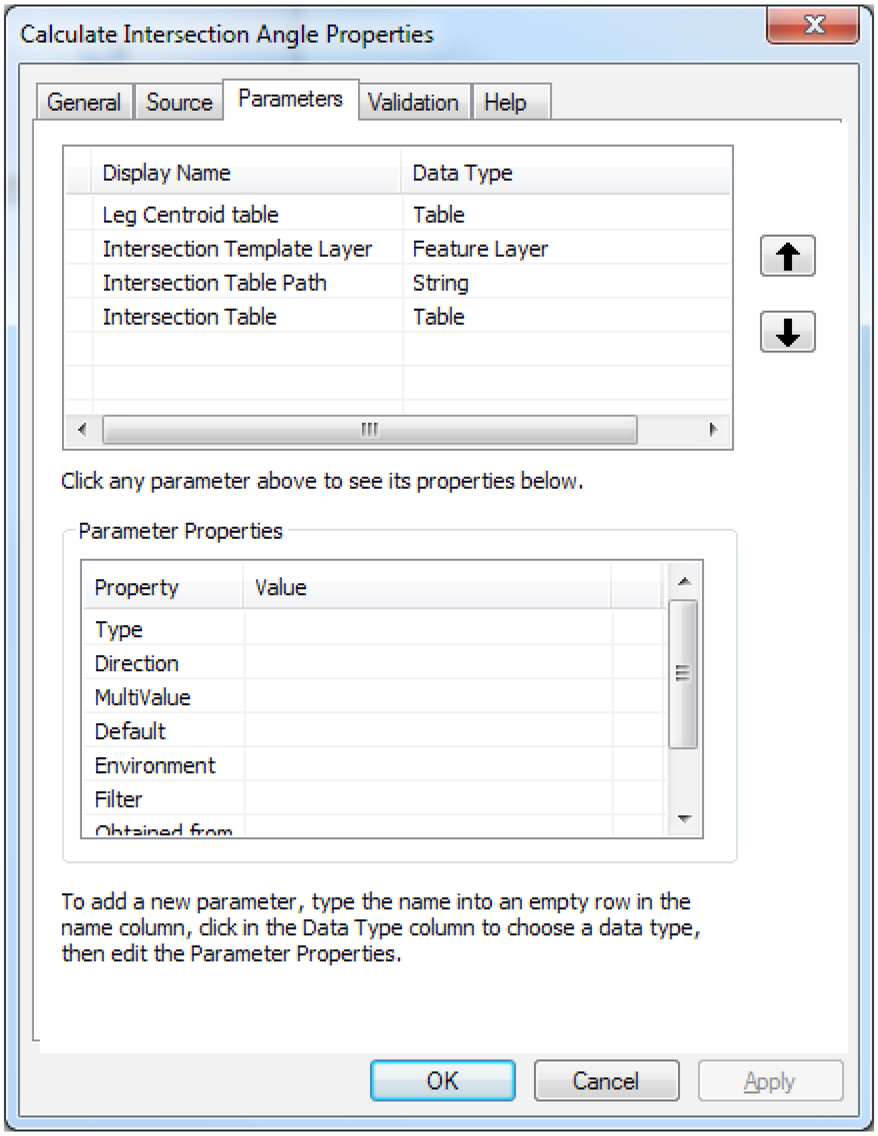
The business process to retire intersection features uses the same models and submodels that are described in the section *Update Features Business Process*, with the exception that model *8 Update Legs and Intersections* does not need to be executed. Note also that only intersection features will be marked as retired, intersection legs do not have a status field to indicate whether they have been retired or not.

## Script Calculate Intersection Angle

The intersection angle is the smallest acute angle between any two legs at an intersection. The script *CalculateIntersectionAngle.py* in the toolbox *MIRE_support* uses the Python programming language to find the intersection angle at each intersection. The script is stored at *C:\MIRE_Tool\Scripts\CalculateIntersectionAngle.py* and is included in its entirety in Appendix IV.

The script is used by model *c4.5 Finalize Intersections* and model *u4.5 Finalize Intersections*. The script uses the following parameters (Figure 21):

- *Leg_Centroid_table*. This table provides the attributes *LegID*, *LegAngle*, and *NodeID* for the script. *LegID* is the unique identifier for each intersection leg. *NodeID* is the unique identifier of the node that the leg is connected to. *LegAngle* is the angle between a leg and the north direction going clockwise.
- *Intersection_Template_Layer*. This feature layer is the layer created from the intersection template in the *Templates* feature dataset.
- *Intersection_Table_Path*. This string specifies the location of the output table.
- *intersection_tbl*: This table is the output of the script. The table contains the fields *intersectionID* and *INTSECT_ANG_MS* (intersection angle measurement.)



| Display Name | Data Type |
|---|---|
| Leg Centroid table | Table |
| Intersection Template Layer | Feature Layer |
| Intersection Table Path | String |
| Intersection Table | Table |

Click any parameter above to see its properties below.

Parameter Properties

| Property | Value |
|---|---|
| Type | |
| Direction | |
| MultiValue | |
| Default | |
| Environment | |
| Filter | |
| Obtained from | |

To add a new parameter, type the name into an empty row in the name column, click in the Data Type column to choose a data type, then edit the Parameter Properties.

**Figure 21. Parameters of Script Calculate Intersection Angle.**

The output of the script is a table *intersection_tbl* which, depending on the model that called the script, might be stored in the geodatabases *IntermediateData* or *UpdateFeature*.

# 6. RDE Tool Addin

The RDE tool includes an ArcGIS custom toolbar with buttons that execute ArcGIS ArcToolBox models described in the previous chapter, ArcGIS custom data entry interfaces, and a data export algorithm, among other features. The ArcToolBox models add attribute data to existing intersection point features, create intersection leg line features, and attach attribute data to both. The data entry interfaces allow the manual entry of data that is not available in a database format, such as video logs. Figure 22 provides a screenshot of the MIRE toolbar.

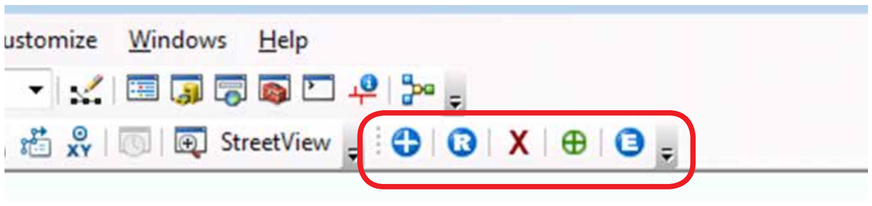
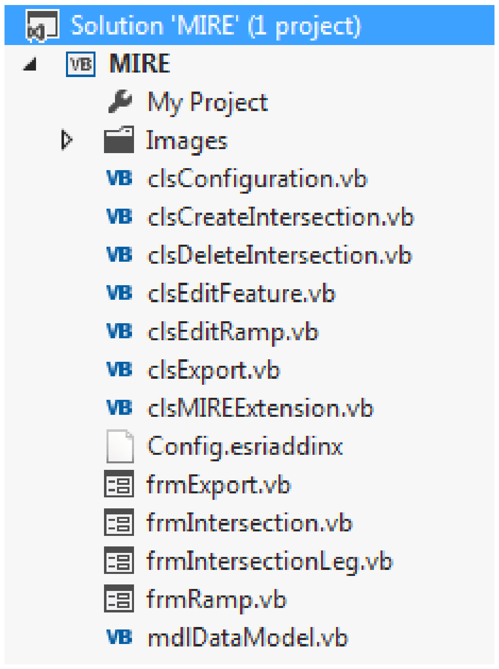


**Figure 22. MIRE Toolbar.**

## Code Structure

The MIRE toolbar is an ArcGIS Addin, created with ArcGIS libraries and VB.NET. The tool includes several classes and forms as shown in Figure 23.

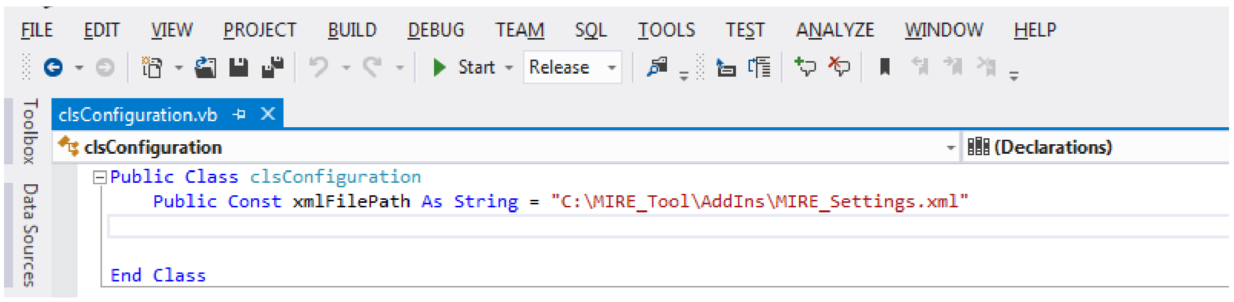


Solution 'MIRE' (1 project)
MIRE
My Project
Images
clsConfiguration.vb
clsCreateIntersection.vb
clsDeleteIntersection.vb
clsEditFeature.vb
clsEditRamp.vb
clsExport.vb
clsMIREExtension.vb
Config.esriaddinx
frmExport.vb
frmIntersection.vb
frmIntersectionLeg.vb
frmRamp.vb
mdlDataModel.vb

**Figure 23. MIRE Toolbar Solution.**

The Addin includes four forms: Intersection (*frmIntersection.vb*), Intersection Leg (*frmIntersectionLeg.vb*), Ramp (*frmRamp.vb*), and Export (*frmExport.vb*). The first three forms allow users to edit intersections, intersection legs, or ramps. The fourth form allows users to export intersection and intersection leg data to a comma separated value (CSV) format.

The tool reads setting information in the *MIRE_Settings.xml* file that is located in the folder *C:\MIRE_Tool\AddIns*. If a user needs to change the location of the XML file, it is necessary to change its path in the class *clsConfiguarion.vb* as shown in Figure 24.

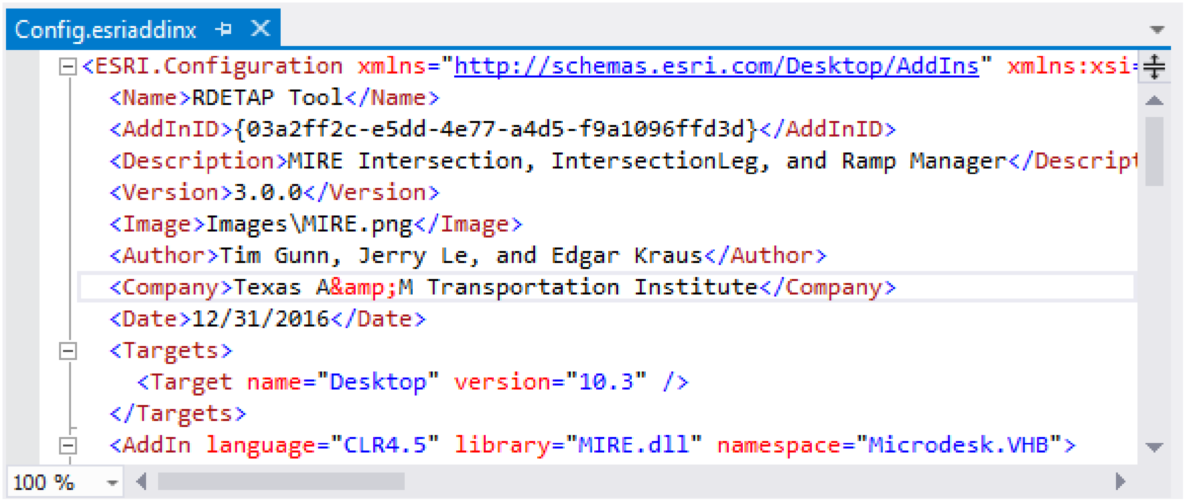


```
Public Class clsConfiguration
    Public Const xmlFilePath As String = "C:\MIRE_Tool\AddIns\MIRE_Settings.xml"

End Class
```

**Figure 24. Class clsConfiguration.**

After changing any file in the MIRE tool source code it is necessary to recompile the code. Note that this version of the code has been compiled with Microsoft Visual Studio 2012. Other versions of Microsoft Visual Studio might not work with the tool. The ArcGIS custom toolbar code will be upgraded during the next RDETAP project to allow for expansion and compilation with the latest version of Microsoft Visual Studio.

The configuration file *Config.esriaddinx* contains information about the MIRE tool such as tool name, tool description, version, and tool icon file path as shown in Figure 25. Any of this information could be modified, however the *AddInID* should not be changed because each addin needs to have a unique ID



```
<ESRI.Configuration xmlns="http://schemas.esri.com/Desktop/AddIns" xmlns:xsi=
  <Name>RDETAP Tool</Name>
  <AddInID>{03a2ff2c-e5dd-4e77-a4d5-f9a1096ffd3d}</AddInID>
  <Description>MIRE Intersection, IntersectionLeg, and Ramp Manager</Descript
  <Version>3.0.0</Version>
  <Image>Images\MIRE.png</Image>
  <Author>Tim Gunn, Jerry Le, and Edgar Kraus</Author>
  <Company>Texas A&amp;M Transportation Institute</Company>
  <Date>12/31/2016</Date>
  <Targets>
    <Target name="Desktop" version="10.3" />
  </Targets>
  <AddIn language="CLR4.5" library="MIRE.dll" namespace="Microdesk.VHB">
```

**Figure 25. MIRE Tool Configuration File.**

Other elements of the configuration file contain information about classes in the tool which should not be changed.

### XML Configuration File

The XML configuration file *MIRE_Settings.xml* is located at *C:\MIRE_Tool\AddIns* and provides basic settings for the RDE tool. The XML configuration file contains the following elements:

- *Layers*. All *Layer* elements have the attributes *Label* and *DatabaseName*. *Label* defines how an element is displayed on the screen in ArcMap, while *DatabaseName* is the name of the feature class as stored in the geodatabase. Note that the attribute *Label* should not be modified. There are four *Layer* elements.
    - Layer *AssetNode*.
    - Layer *Intersection*.
    - Layer *IntersectionLeg*.
    - Layer *Ramp*.

  Each *Layer* element has the element *Field* with the attributes *Label*, *DatabaseName*, and *Enabled*. Each field element describes a field or column in the corresponding feature class. *Label* defines how an element is displayed on the screen in ArcMap, *DatabaseName* is the name of the field as stored in the feature class of the geodatabase, and *Enabled* defines whether a field can be edited using the RDE tool toolbar. If *Enabled* is set to "True" it can be edited through the interface, if it set to "False" it is greyed-out and cannot be edited.
- *Models*. There are two models with the attributes *Label*, *ToolboxPath*, and *ToolName*.
    - Model *UpdateIntersections*.
    - Model *PopulateNewIntersections*.

  *Label* defines how a model is displayed on the screen in ArcMap, *ToolboxPath* describes the location on the local computer where the MIRE toolbox is stored, and *ToolName* specifies the name of the tool in the MIRE toolbox.
- *Paths*. There is one path stored in the MIRE XML configuration file, which is the location of the RDE tool's exported files. The name of the element is *InitialCSV_ExportPath*. The path does not have any attributes.
- *SelectTolerance*. This element has one sub element called *ExpandMapUnitsBy*, with a value that is set to "20." ArcMap map units are the units in which the spatial data in the map or scene are drawn. Map units are determined by the linear coordinate system of the map or local scene. When a user tries to select an intersection, intersection leg, segment, or ramp by clicking on it, the *ExpandMapUnitsBy* value will determine the area in which the tool will search for a feature. With a default value of "20," the RDE tool will try to find and select the first feature in the area with a radius equal to 20 map units around the location of the click.

### Use of MIRE Toolbar

While the RDE tool toolboxes and scripts allow users to process thousands of intersections and intersection legs in one session, the toolbar is intended to add, delete, or edit one intersection or intersection leg at a time. This feature is useful when add or update a small number of intersections or intersection legs, or when adding data that is not easily accessible in a geospatial format. The MIRE toolbar consists of five tools:

- Edit intersection or intersection leg.
- Edit Ramp.
- Delete Intersection.
- Create Intersection.
- Export Intersection and Approach Data.

The following provides a detailed description of each tool.

***Edit Intersection or Intersection Leg***

Clicking on the *Edit Intersection* or *Intersection Leg* button and then on an intersection feature will produce a window with intersection attributes and various dropdown menus, as shown in Figure 26. A user can update the information in this window as desired and click *OK* to save the changes. The contents, i.e. fields and labels of this window can be adjusted by a programmer, for example some fields that are not editable or are not needed could be hidden from view.

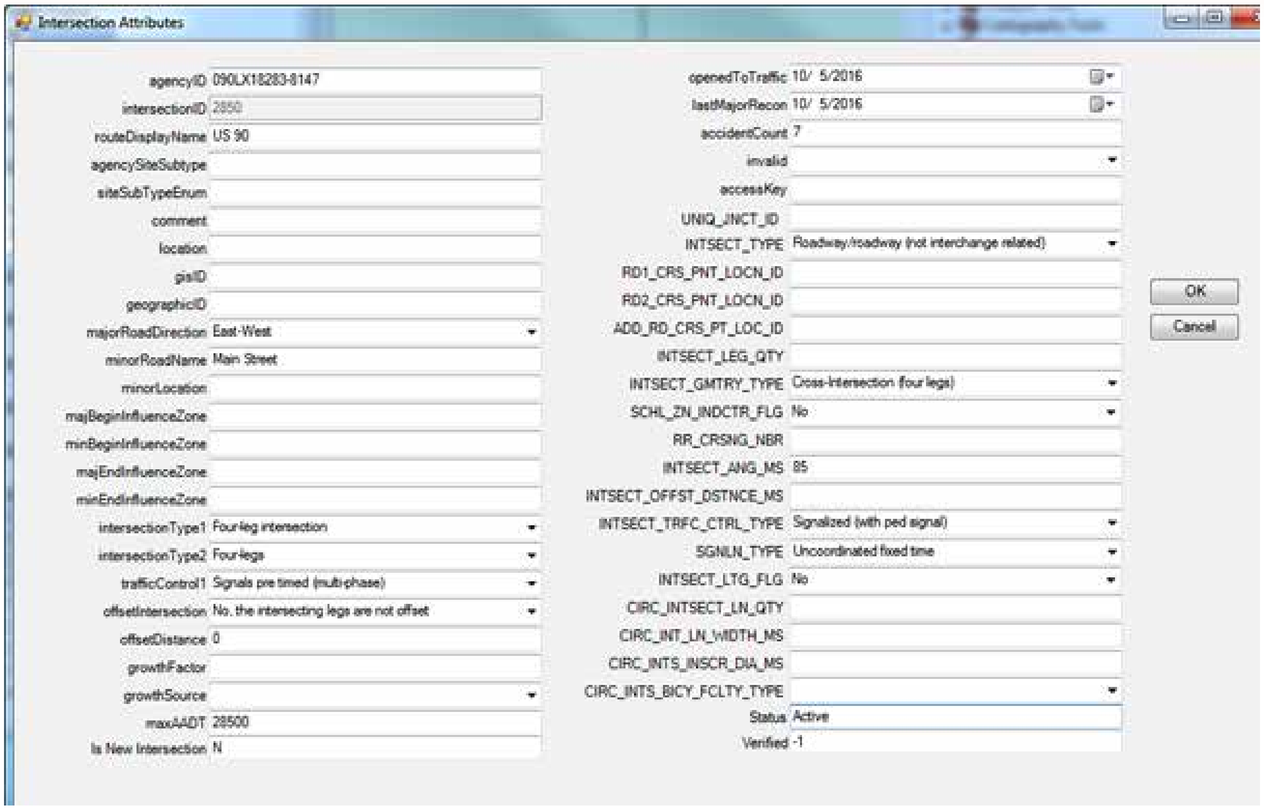
The *MIRE_Settings.xml* configuration file can be used to prevent edits to certain fields. For example, the field *intersectionID* is greyed out and cannot be edited by a user, as shown in Figure 26. To prevent a field from being edited by user, open the *MIRE_Settings.xml* configuration file in a text editor, search for the field name that needs to be greyed out, and change the *Enabled* attribute from *True* to *False*. The following is an example of how to prevent edits to the field *agencyID* shown in Figure 26:

**Original code**

```
[...]

<Layer Label=”Intersections” DatabaseName=”Intersections”>

<!--Field Names-->

 <Field Label=”agencyID” DatabaseName=”agencyID” Enabled=”True”/>

 <Field Label=”intersectionID” DatabaseName=”intersectionID”
Enabled=”False”/>

[...]
```

**Modified Code**

```
[...]

<Layer Label=”Intersections” DatabaseName=”Intersections”>

<!--Field Names-->

 <Field Label=”agencyID” DatabaseName=”agencyID” Enabled=”False”/>

 <Field Label=”intersectionID” DatabaseName=”intersectionID”
Enabled=”False”/>

[...]
```

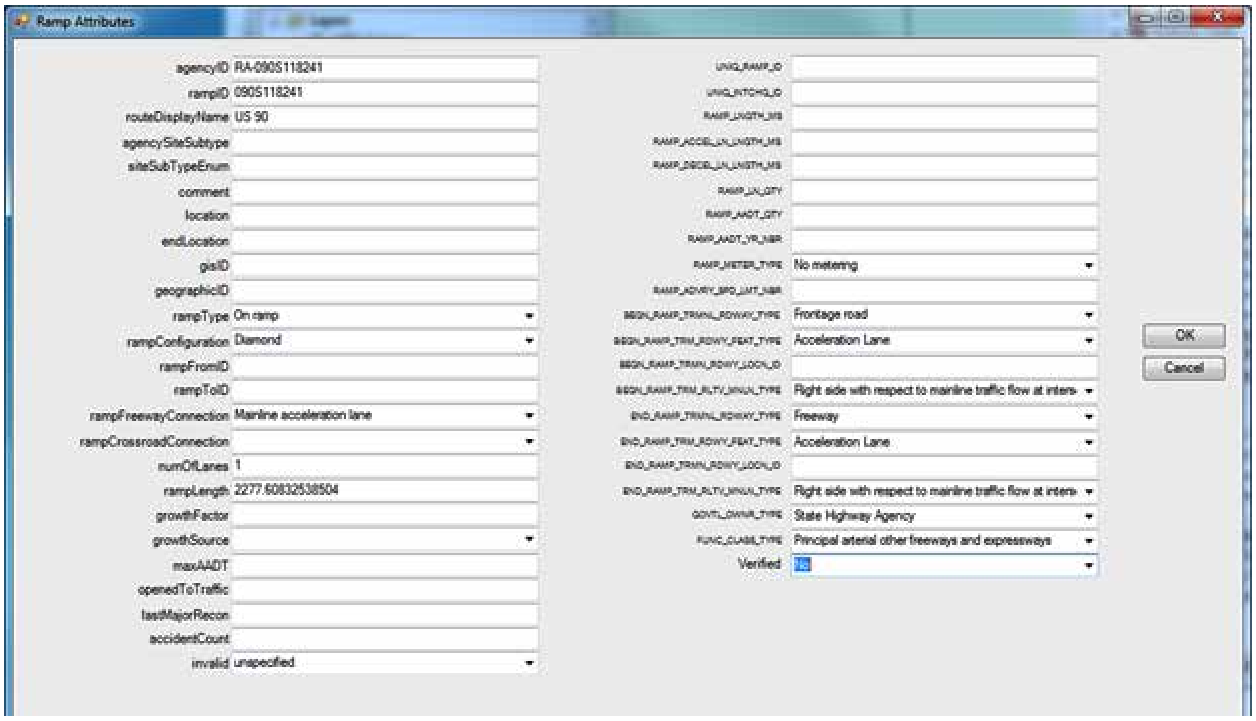


| | | | |
|---|---|---|---|
| agencyID | 090LX18283-8147 | openedToTraffic | 10/ 5/2016 |
| intersectionID | 2850 | lastMajorRecon | 10/ 5/2016 |
| routeDisplayName | US 90 | accidentCount | 7 |
| agencySiteSubtype | | invalid | |
| siteSubTypeEnum | | accessKey | |
| comment | | UNIQ_JNCT_ID | |
| location | | INTSECT_TYPE | Roadway/roadway (not interchange related) |
| gisID | | RD1_CRS_PNT_LOCN_ID | |
| geographicID | | RD2_CRS_PNT_LOCN_ID | |
| majorRoadDirection | East-West | ADD_RD_CRS_PT_LOC_ID | |
| minorRoadName | Main Street | INTSECT_LEG_QTY | |
| minorLocation | | INTSECT_GMTRY_TYPE | Cross-Intersection (four legs) |
| majBeginInfluenceZone | | SCHL_ZN_INDCTR_FLG | No |
| minBeginInfluenceZone | | RR_CRSNG_NBR | |
| majEndInfluenceZone | | INTSECT_ANG_MS | 85 |
| minEndInfluenceZone | | INTSECT_OFFST_DSTNCE_MS | |
| intersectionType1 | Four-leg intersection | INTSECT_TRFC_CTRL_TYPE | Signalized (with ped signal) |
| intersectionType2 | Four-legs | SGNLN_TYPE | Uncoordinated fixed time |
| trafficControl1 | Signals pre timed (multi-phase) | INTSECT_LTG_FLG | No |
| offsetIntersection | No, the intersecting legs are not offset | CIRC_INTSECT_LN_QTY | |
| offsetDistance | 0 | CIRC_INT_LN_WIDTH_MS | |
| growthFactor | | CIRC_INTS_INSCR_DIA_MS | |
| growthSource | | CIRC_INTS_BICY_FCLTY_TYPE | |
| maxAADT | 28500 | Status | Active |
| Is New Intersection | N | Verified | -1 |

**Figure 26. Intersection Attributes.**

### *Edit Ramp*

Clicking on the *Edit Ramp* button and then on a ramp feature produces a window with ramp attributes and various dropdown menus, as shown in Figure 27. A user can update the information in this window as desired and click *OK* to save the changes. The contents, i.e. fields and labels of this window can be adjusted by a programmer, for example some fields that are not editable or are not needed could be hidden from view.



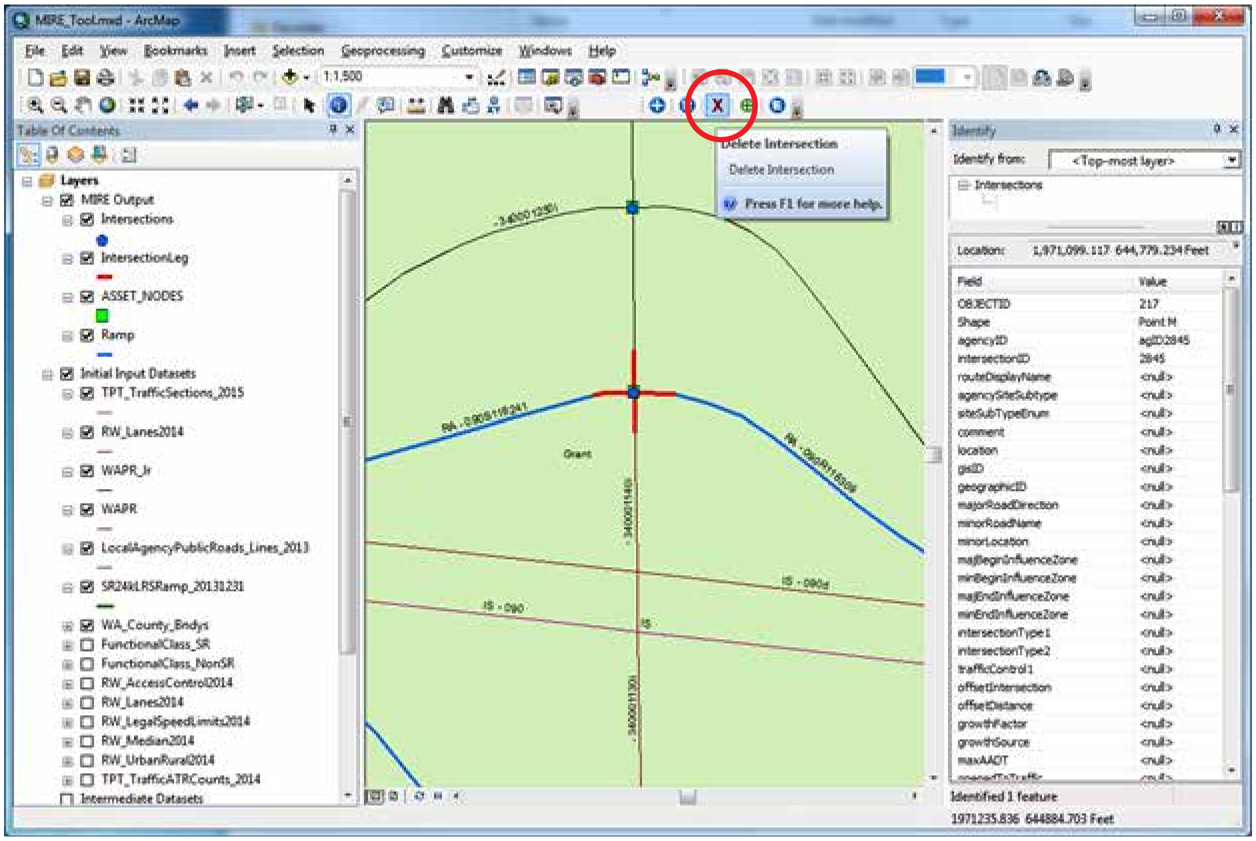
| | |
|---|---|
| agencyID | RA-090S118241 |
| rampID | 090S118241 |
| routeDisplayName | US 90 |
| agencySiteSubtype | |
| siteSubTypeEnum | |
| comment | |
| location | |
| endLocation | |
| gisID | |
| geographicID | |
| rampType | On ramp |
| rampConfiguration | Diamond |
| rampFromID | |
| rampToID | |
| rampFreewayConnection | Mainline acceleration lane |
| rampCrossroadConnection | |
| numOfLanes | 1 |
| rampLength | 2277.60832538504 |
| growthFactor | |
| growthSource | |
| maxAADT | |
| openedToTraffic | |
| lastMajorRecon | |
| accidentCount | |
| invalid | unspecified |

| | |
|---|---|
| UNIQ_RAMP_ID | |
| UNIQ_INTCHG_ID | |
| RAMP_LNGTH_MS | |
| RAMP_ACCEL_LN_LNGTH_MS | |
| RAMP_DECEL_LN_LNGTH_MS | |
| RAMP_LN_QTY | |
| RAMP_AADT_QTY | |
| RAMP_AADT_YR_NBR | |
| RAMP_METER_TYPE | No metering |
| RAMP_ADVRY_SPD_LMT_NBR | |
| BEGN_RAMP_TRMNL_RDWAY_TYPE | Frontage road |
| BEGN_RAMP_TRM_RDWY_FEAT_TYPE | Acceleration Lane |
| BEGN_RAMP_TRMN_RDWY_LOCN_ID | |
| BEGN_RAMP_TRM_RLTV_MNLN_TYPE | Right side with respect to mainline traffic flow at inters |
| END_RAMP_TRMNL_RDWAY_TYPE | Freeway |
| END_RAMP_TRM_RDWY_FEAT_TYPE | Acceleration Lane |
| END_RAMP_TRMN_RDWY_LOCN_ID | |
| END_RAMP_TRM_RLTV_MNLN_TYPE | Right side with respect to mainline traffic flow at inters |
| GOVT_OWNR_TYPE | State Highway Agency |
| FUNC_CLASS_TYPE | Principal arterial other freeways and expressways |
| Verified | [illegible] |

**Figure 27. Ramp Attributes.**

### *Delete Intersection*

The *Delete Intersection* button on the MIRE toolbar allows a user to delete an intersection feature manually. The tool deletes the intersection feature and record in the feature class *Intersections*, all associated intersection legs and records in the feature class *IntersectionLeg*, but not the related node feature in the feature class *AssetNode*. To manually delete an intersection, click on the *Delete Intersection* button and then on the intersection (Figure 28).



Location: 1,971,099.117 644,779.234 Feet

| Field | Value |
|---|---|
| OBJECTID | 217 |
| Shape | Point M |
| agencyID | agID2845 |
| intersectionID | 2845 |
| routeDisplayName | <null> |
| agencySiteSubtype | <null> |
| siteSubTypeEnum | <null> |
| comment | <null> |
| location | <null> |
| gisID | <null> |
| geographicID | <null> |
| majorRoadDirection | <null> |
| minorRoadName | <null> |
| minorLocation | <null> |
| majBeginInfluenceZone | <null> |
| minBeginInfluenceZone | <null> |
| majEndInfluenceZone | <null> |
| minEndInfluenceZone | <null> |
| intersectionType1 | <null> |
| intersectionType2 | <null> |
| trafficControl1 | <null> |
| offsetIntersection | <null> |
| offsetDistance | <null> |
| growthFactor | <null> |
| growthSource | <null> |
| maxAADT | <null> |

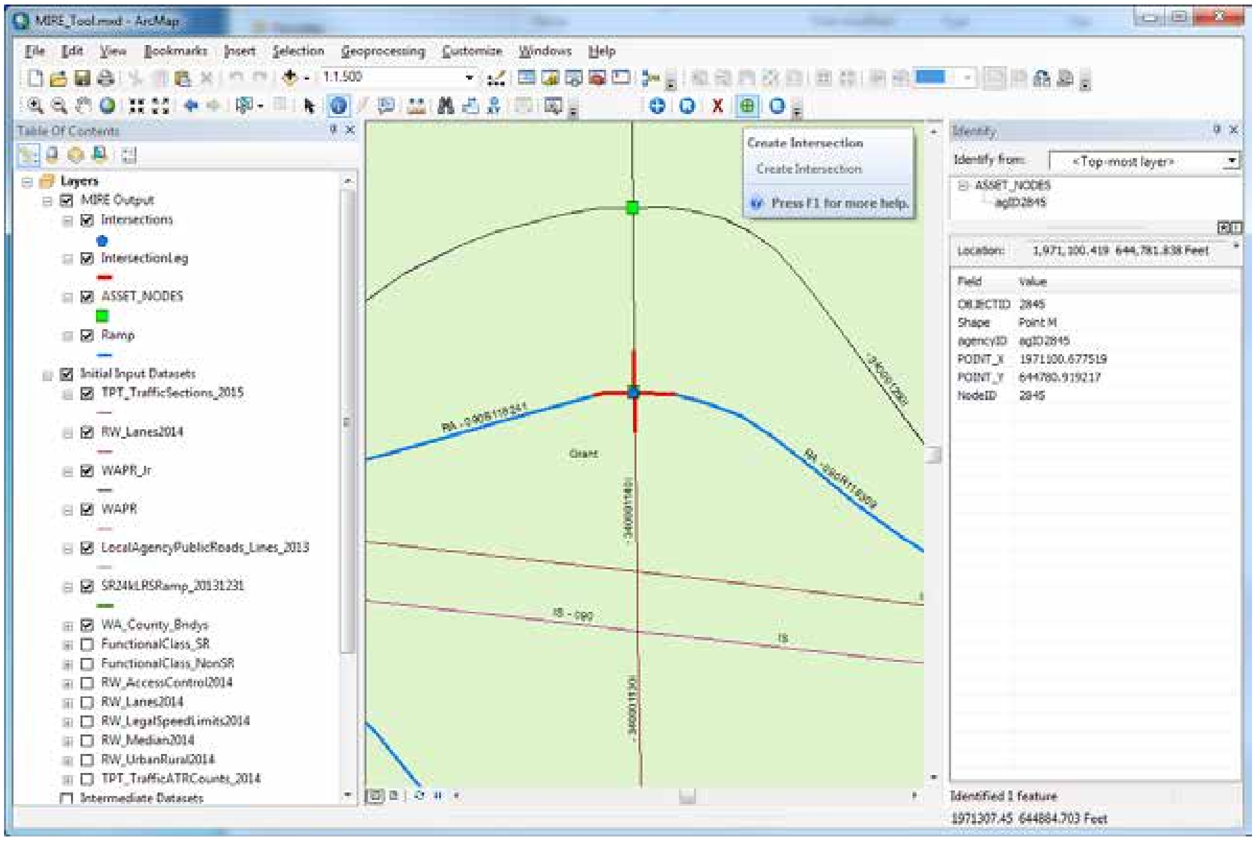
Identified 1 feature

1971235.836 644884.703 Feet

**Figure 28. Delete Intersection Button on the MIRE Toolbar.**

### *Create Intersection*

The *Create Intersection* button on the MIRE toolbar allows a user to create a new intersection feature in the *Intersection* feature class (Figure 29). The *Create Intersection* tool requires that an asset node is located where the intersection is going to be created, or more specifically, a record in the feature class *AssetNode*. This is necessary because both spatial location and intersection ID of the intersection are managed by the asset node.



Location: 1,971,100.419 644,781.838 Feet

| Field | Value |
|---|---|
| OBJECTID | 2845 |
| Shape | Point M |
| agencyID | agID2845 |
| POINT_X | 1971100.677519 |
| POINT_Y | 644780.919217 |
| NodeID | 2845 |

**Figure 29. Create Intersection Button on MIRE Toolbar.**

Once the intersection feature is created, a user might run the models *6 Update Intersections* and *7 Update Legs* to add data to the new intersection feature, and create and update the intersection leg features, as described previously.

### *Export Intersection and Approach Data*

The *Export Intersection and Approach Data* button on the MIRE toolbar allows a user to export all or selected intersections in the *Intersection* feature class, and all or selected intersection legs in the *IntersectionLeg* feature class (Figure 30).

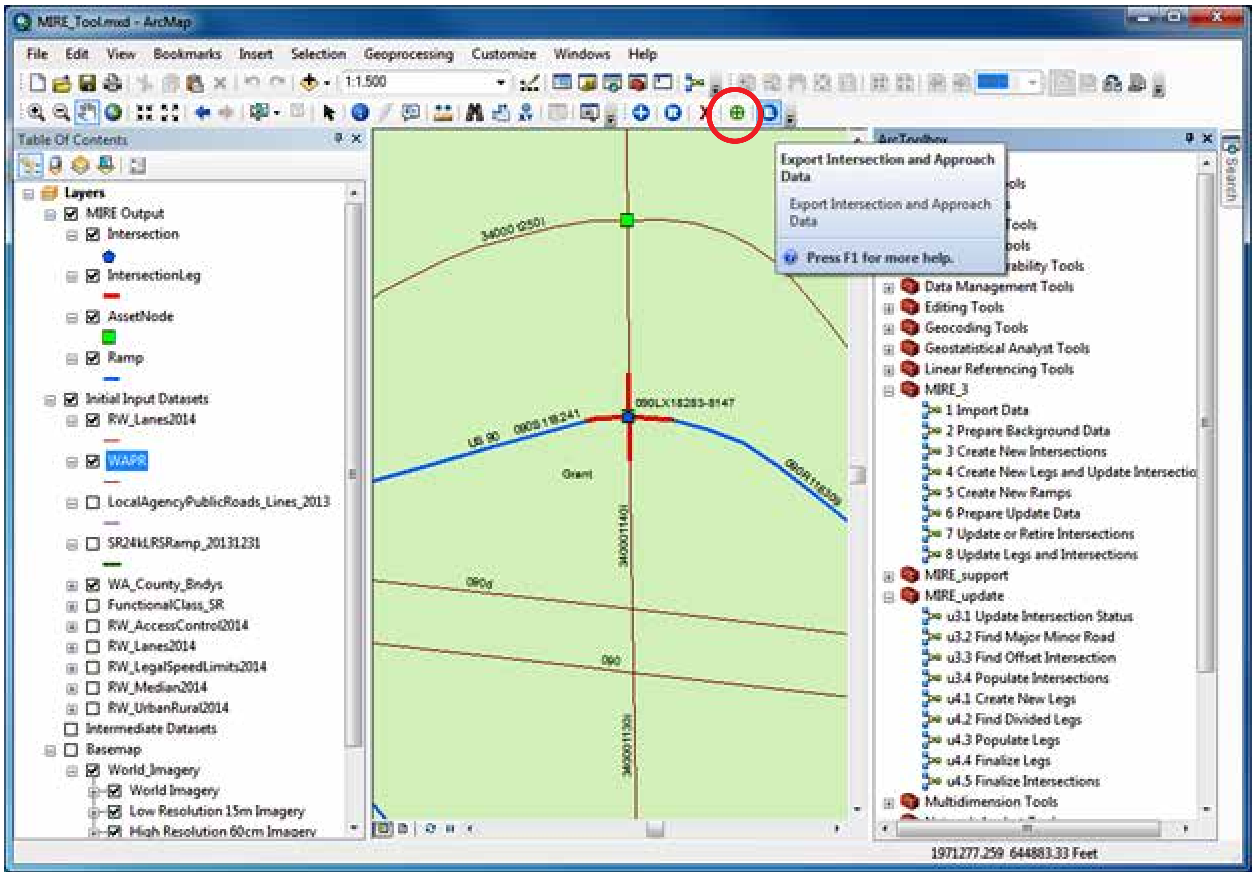
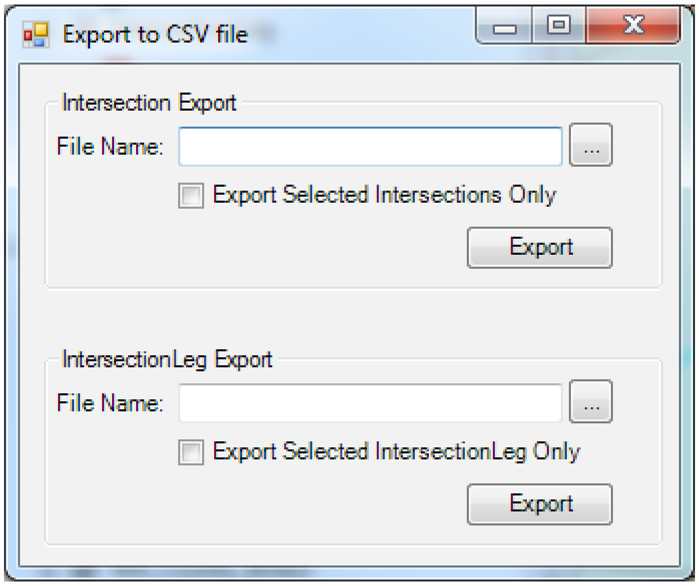


**Figure 30. Export Intersection and Approach Data Button on Mire Toolbar.**

After clicking on the *Export Intersection and Approach Data* button, a window appears that lets a user export intersections, intersection legs, or both (Figure 31).


Export to CSV file

Intersection Export

File Name: ...

☐ Export Selected Intersections Only

Export

IntersectionLeg Export

File Name: ...

☐ Export Selected IntersectionLeg Only

Export

**Figure 31. Export Intersections and Intersection Leg Data.**

Clicking on the top path (...) button allows a user to specify a CSV file, or provide a new file name for export of intersections. Clicking on the bottom path (...) button allows a user to specify a CSV file, or provide a new file name for the export of intersection legs. By default, all records in the *Intersection* and *IntersectionLeg* feature classes are exported to the location specified. The export can be limited to features that were previously selected in ArcMap using the *Select* tool, and checking the boxes "Export Selected Intersections Only" and "Export Selected IntersectionLeg Only."

## Default Location for Exported Files

The default location for exported files is *C:\MIRE_Tool\Export*. The default location is determined by the XML configuration file *MIRE_Settings.xml*, which is located at *C:\MIRE_Tool\AddIns*. To modify the location, search for the following text in the XML configuration file, and make changes accordingly.

```
<Paths>

<InitialCSV _ ExportPath>C:\MIRE _ Tool\Export</InitialCSV _ ExportPath>

</Paths>
```

# 7. RDE Tool Modification Instructions

The RDE tool consists of many components: data model, geodatabases, toolboxes, and a toolbar. All components can be adjusted to the needs and datasets available at the transportation agency. However, all components are related, so modifying one component will requires some modifications to other components. Note that the current version of the RDE tool toolbar uses Visual Studio 2012. Recompiling the tool with a more recent version of Visual Studio might cause some errors and might require some code modifications and upgrades. As mentioned earlier, the next RDETAP project will upgrade the ArcGIS custom toolbar code to allow modification and compilation with the latest version of Microsoft Visual Studio.

This chapter provides an overview of generic steps for an implementation project, followed by in-depth guidance on how to modify the components of the RDE tool.

## Recommended Process to Implement the RDE Tool

Each implementation at a transportation agency is a unique project with unique goals and objectives. However, based on past experiences, there are several steps that a transportation agency could consider that have proven beneficial to past implementation projects, as follows:

- Form implementation team.
  - Determine if local agencies should be involved.
- Establish goals and objectives in general terms.
  - Gather input datasets as feasible.
  - Analyze input datasets and develop detailed data mapping.
  - Develop draft state-specific data templates.
  - Prepare materials for implementation workshop.
- Conduct implementation team workshop.
  - Establish detailed goals and objectives.
  - Discuss analysis of input datasets and data mapping.
  - Discuss state-specific data characteristics, linear referencing methods, data management approach.
  - Gather additional input datasets as needed.
- Develop RDE tool modification plan, timeline, and milestones.
- Modify RDE tool.
- Conduct implementation team meetings (webinars) at milestones.
- Deliver modified RDE tool.
  - Deliver draft version for agency review.

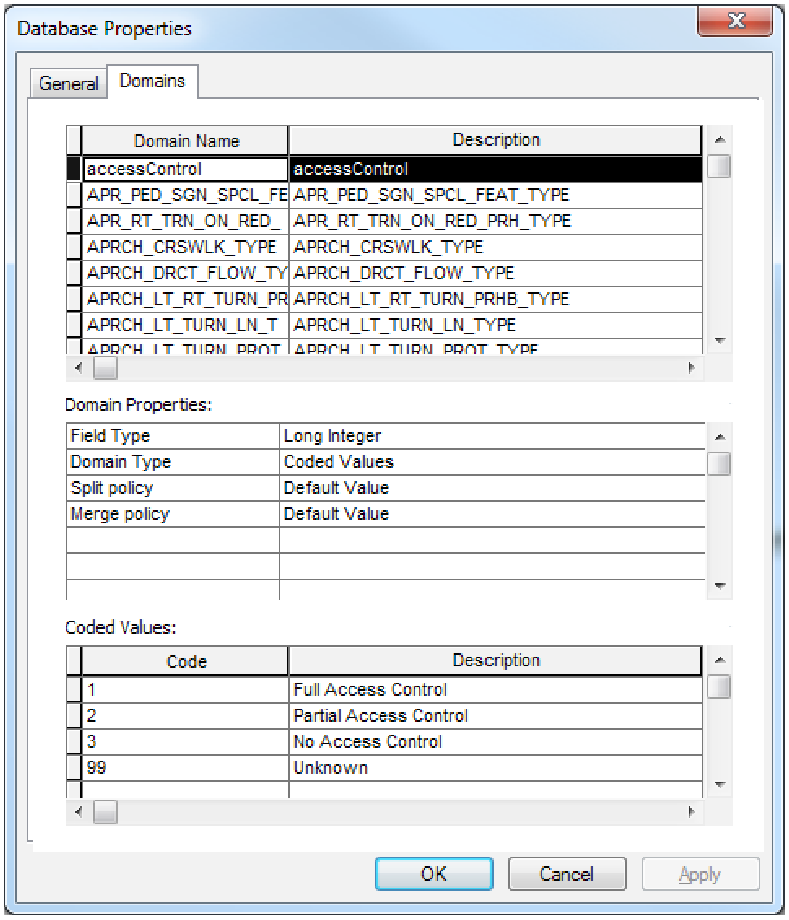
- Make requested changes and deliver final version.

- Provide ongoing support as needed.

## Changes to RDE Tool Data Model

The current data model used for the RDE tool includes the fields that most transportation agency would require to conduct roadway safety analysis. Additional fields that a transportation agency might require can be added to the final output datasets (i.e. *Intersection, IntersectionLeg, Ramp*, and *Segment*) without modifying the data model.

Templates for the output datasets are provided in appendix III. A transportation agency can modify domains for attributes in the geodatabases as needed. Values of a domain can be modified directly in a geodatabase by right clicking on a geodatabase, selecting *Properties*, and then selecting the *Domain* tab (Figure 32). A list of domains will be displayed in the top half of the screen, while the coded values for each domain will be displayed on the bottom half. By clicking on any domain, coded values can be added, modified, or deleted.



| Domain Name | Description |
|---|---|
| accessControl | accessControl |
| APR_PED_SGN_SPCL_FE | APR_PED_SGN_SPCL_FEAT_TYPE |
| APR_RT_TRN_ON_RED_ | APR_RT_TRN_ON_RED_PRH_TYPE |
| APRCH_CRSWLK_TYPE | APRCH_CRSWLK_TYPE |
| APRCH_DRCT_FLOW_TY | APRCH_DRCT_FLOW_TYPE |
| APRCH_LT_RT_TURN_PR | APRCH_LT_RT_TURN_PRHB_TYPE |
| APRCH_LT_TURN_LN_T | APRCH_LT_TURN_LN_TYPE |
| APRCH_LT_TURN_PROT | APRCH_LT_TURN_PROT_TYPE |

Domain Properties:

| | |
|---|---|
| Field Type | Long Integer |
| Domain Type | Coded Values |
| Split policy | Default Value |
| Merge policy | Default Value |

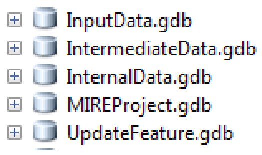
Coded Values:

| Code | Description |
|---|---|
| 1 | Full Access Control |
| 2 | Partial Access Control |
| 3 | No Access Control |
| 99 | Unknown |

Figure 32. RDE Tool Database Domain Values.

## Changes to RDE Tool Geodatabases

As described in Chapter D, the RDE tool consists of five geodatabases (Figure 33). Geodatabases are used to store input datasets, RDE tool templates, intermediate data, and the final output data. Refer to the Chapter D to learn more about RDE tool geodatabases.

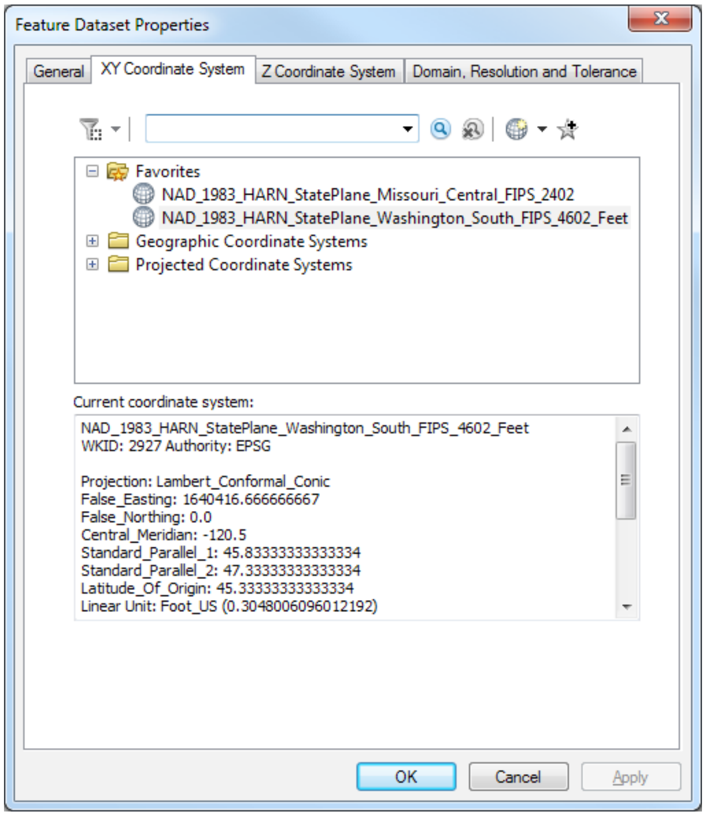


InputData.gdb
IntermediateData.gdb
InternalData.gdb
MIREProject.gdb
UpdateFeature.gdb

**Figure 33. RDE Tool Geodatabases.**

These geodatabases have feature datasets which require a coordinate system. If a coordinate system needs to be changed, a user can change the coordinate system by following the steps below (Figure 34):

- Right-click on a feature dataset in ArcMap.
- Select *Properties*.
- Select the *XY Coordinate System* tab.
- Click on the down arrow next to the globe.
- Select *New* or *Import* to enter a new coordinate system or import a coordinate system from an existing feature class.


Feature Dataset Properties
General
XY Coordinate System
Z Coordinate System
Domain, Resolution and Tolerance
Favorites
Geographic Coordinate Systems
Projected Coordinate Systems
Current coordinate system:
OK
Cancel
Apply

NAD_1983_HARN_StatePlane_Missouri_Central_FIPS_2402
NAD_1983_HARN_StatePlane_Washington_South_FIPS_4602_Feet

```
NAD_1983_HARN_StatePlane_Washington_South_FIPS_4602_Feet
WKID: 2927 Authority: EPSG

Projection: Lambert_Conformal_Conic
False_Easting: 1640416.666666667
False_Northing: 0.0
Central_Meridian: -120.5
Standard_Parallel_1: 45.83333333333334
Standard_Parallel_2: 47.33333333333334
Latitude_Of_Origin: 45.33333333333334
Linear Unit: Foot_US (0.3048006096012192)
```

**Figure 34. ArcMap Feature Dataset Properties.**

To change a geodatabase to store temporary data, input data, or final output data, it is necessary to modify several paths in the RDE tool toolbox code. Renaming any of these geodatabases also requires modification of several paths in the RDE tool toolbox code.

To use the templates in the *InternalData* geodatabase, copy a template from the *Templates* feature dataset and remove the suffix *_Template*. For example, *INTSECT_LEG_Template* would become *INTSECT_LEG*. Other datasets can be added to this geodatabase as needed. However, all spatial and non-spatial datasets must be imported into the *MIREProject* geodatabase before they can be used with the RDE tool. For example, datasets in Excel or relational databases should be converted to a geodatabase before using the data with the RDE tool.

### Changes to RDE Toolboxes

As described in Chapter E, the RDE tool uses three toolboxes: *MIRE_3*, *MIRE_support*, and *MIRE_update*. Refer to Chapter E for more information before making any changes to the toolboxes. Note that models in the *MIRE_3* toolbox call several submodels in *MIRE_support* and *MIRE_update* toolboxes. Changing the names of submodels will not affect the models in *MIRE_3*. However, the names of submodels do not update automatically. If a submodel in *MIRE_support* and *MIRE_update* toolboxes is removed, models in *MIRE_3* might not work. Processes inside submodels can be changed if needed, however, proper planning and testing are required before making any changes to the submodels. Changing the *MIRE_3* toolbox name requires changing this name in the RDE tool XML configuration file where the RDE toolbox is referenced.

### Changes to RDE Tool Toolbar

The RDE tool toolbar was originally written in VB.NET version 2008. The RDETAP pilot implementation project modified the code and recompiled it with VB.NET version 2012 in Visual Studio .NET 2012. However, Visual Studio 2012 does not fully support this toolbar add-in. As a result, the toolbar can be recompiled, but the code but cannot be debugged in Visual Studio 2012. The next RDETAP project will upgrade the toolbar code to be fully compatible with the latest version of Visual Studio.

Before making any modification to the code, it is necessary to review Chapter F that describes the RDE tool toolbar and related code. Changing the code will require a solid knowledge of VB.NET and ArcGIS add-ins for .NET.

# 8. Case Study: Implementation of RDE Tool at Washington State DOT

This chapter provides a summary of the pilot RDE tool implementation at WSDOT and the process the RDETAP team followed to facilitate implementation. The intent of the chapter is to provide future transportation agencies an idea of the resources and efforts needed to conduct a RDE tool implementation by providing a real-life implementation example.

The project team essentially followed the recommended implementation process described in the previous chapter. This chapter provides a summary of activities at each step of the implementation project.

## Form Implementation Team

The pilot implementation team consisted of the following:

- Project team, consisting of FHWA project manager, Leidos project manager, TTI implementation manager, and TTI programmer.
- WSDOT staff, including WSDOT GIS data manager, WSDOT data management supervisor, WSDOT traffic data manager, and lead WSDOT Safety Analyst programmer.

Once the implementation team was formed, the project team conducted a kickoff meeting to introduce the RDE tool and purpose of the project, introduce important project contacts, and discuss the necessary steps to complete the next step of the implementation process.

## Establish Goals and Objectives in General Terms

Following the kickoff meeting, TTI collected all requested input datasets with the intent to come up with a general implementation plan. WSDOT's main goal for the RDETAP project was to integrate several roadway datasets internal to WSDOT and datasets developed or provided by external agencies, in an effort to create an integrated dataset that could be exported to AASHTOWare Safety Analyst, the roadway safety analysis tool in use at the DOT. WSDOT also requested to expand the RDE tool code to allow for the calculation of certain MIRE data elements not yet included in any datasets available to WSDOT. TTI analyzed the contents of the input datasets and created an Excel spreadsheet that produced a preliminary mapping of existing state data to the available MIRE/Safety Analyst templates. The result of this task was a detailed description of all data elements and their transcription into draft, WSDOT-specific output data templates.

In addition, the project team developed data models and presentation materials for the implementation workshops. The project team developed the following objectives:

- Create three MIRE-compatible GIS layers for the feature types intersections, intersection legs, and ramps.
- Extract roadway inventory data from WSDOT GIS linear and point data, and transfer the data into the MIRE-compatible GIS layers stored in a geodatabase.
- Expand the ArcGIS models to calculate data fields that can be extracted using the GIS roadway geometry.

- Document necessary changes to RDE tool based on WSDOT data management process and requirements, define input data and output templates, and modify tool accordingly.
- Provide a bug-free and working RDE tool to WSDOT.
- Provide technical support to implement the tool at WSDOT.
- Provide all available documentation related to the RDE tool.

### Conduct Implementation Team Workshop

The project team set up a series of workshops over the course of two days to discuss the output data templates, and methods to derive the data. The meetings were scheduled as follows:

- Day 1: Meet with WSDOT staff, including GIS data management staff, IT management, and safety data analysts to discuss the following:
    - Meet in the morning to discuss the project overall, goals and objectives, current implementation status.
    - Meet in the afternoon to discuss WSDOT data collection and analysis process, safety data analysis process using tool output including custom interfaces, and WSDOT's plans for future process enhancements. Discuss sample data provided by MoDOT and data mapping.
- Day 2: Meet with WSDOT staff, including GIS data management staff, IT management, and safety data analysts to discuss the following:
    - Meet in the morning to continue discussion on future improvements, including roadway segment and custom interfaces, and discuss tool implementation schedule, trial runs, and tests.
    - Meet in the afternoon with Safety Analyst team to discuss input requirements, participate in Safety Analyst demo, and discuss GIS data modification and export/import business process.

The implementation team dedicated a significant amount of time at the workshop meetings to discuss the data mapping document in detail. For example, team members discussed which data elements to extract, which domain values to retain, and where to store the data elements in the output template. Team members also discussed possible options to generate or calculate new data elements that did not exist previously in a state database. As a result, team members described a plan to calculate intersection offset and intersection angle values.

### Develop RDE Tool Modification Plan, Timeline, and Milestones

Following the workshop meetings, the project team revised the data mapping document and developed a detailed implementation plan. The plan included tentative milestones for the following:

- RDE tool modifications and initial submission to WSDOT.
- RDE tool installation on WSDOT server.
- RDE tool testing and follow-up phase.
- RDE tool development workshop.
- Additional RDE tool modifications and final tool submission to WSDOT.

In addition to the modification plan, the project team developed a task list that described the RDE tool modifications in detail. This allowed the project team to schedule resources and update meetings at project milestones. Table 6 provides an overview of the timeline for the implementation of the RDE tool at WSDOT along with a description of major project development milestones.

**Table 6. WSDOT RDE Tool Pilot Implementation Timeline.**

| Date | Milestone |
|---|---|
| 11/2014 | Project team delivers webinar to WSDOT to demo RDE tool and discuss potential pilot. |
| 01/2015 | WSDOT becomes first state to participate in RDE tool implementation, implementation team formed. |
| 03/2015 | TTI receives required WSDOT input datasets and starts data analysis and data mapping. |
| 05/2015 | Project team and WSDOT agree on RDE tool modifications. |
| 07/2015 | TTI delivered the first version of the modified RDE tool. |
| 08/2015 | TTI delivered the second version of the modified RDE tool. |
| 09/2015 | Workshops at WSDOT offices to discuss RDE tool upgrades. |
| 11/2015 | TTI delivers update to modified RDE tool. |
| 01/2016 | Project team and WSDOT discuss further RDE tool modifications. |
| 04/2016 | TTI delivers draft final RDE tool. |
| 07/2016 | TTI delivers final RDE tool. |

## Conduct Implementation Team Meetings (Webinars) at Milestones

The project team conducted approximately monthly update meetings to discuss progress and recent RDE tool modifications. Typically, the project team would deliver an updated version of the RDE tool to WSDOT, provide follow-up to assist with the installation of the tool, provide instruction on the use of the new features, and then allow for some testing by the transportation agency. Once WSDOT was ready to provide feedback, the implementation team conducted a progress meeting. The implementation team made frequent use of web conferencing, which allowed WSDOT to review tool upgrades remotely, and engage in detailed discussion while reviewing the tool remotely.

## RDE Tool Modifications

### *WSDOT Input Data*

TTI received several input datasets from WSDOT, including geodatabases, shapefiles, and tabular data. Figure 35 shows a list of shapefiles, and Figure 36 shows a list of WSDOT feature classes and tables. As shown in Figure 36, WSDOT stored most of the state highway attribute data in separate feature classes. For example, acceleration lanes, speed limits, and special use lanes are separate feature classes. In addition, there are typically annual updates for each feature class, so it is important to track feature class versions.

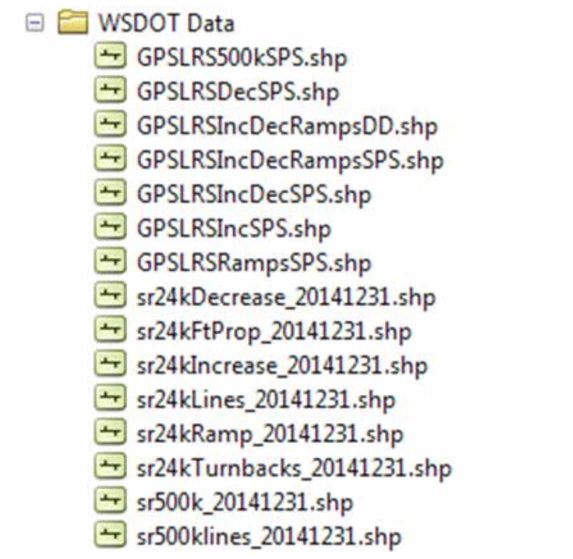

WSDOT Data
- GPSLRS500kSPS.shp
- GPSLRSDecSPS.shp
- GPSLRSIncDecRampsDD.shp
- GPSLRSIncDecRampsSPS.shp
- GPSLRSIncDecSPS.shp
- GPSLRSIncSPS.shp
- GPSLRSRampsSPS.shp
- sr24kDecrease_20141231.shp
- sr24kFtProp_20141231.shp
- sr24kIncrease_20141231.shp
- sr24kLines_20141231.shp
- sr24kRamp_20141231.shp
- sr24kTurnbacks_20141231.shp
- sr500k_20141231.shp
- sr500klines_20141231.shp

Figure 35. WSDOT Shapefile Geospatial Data.

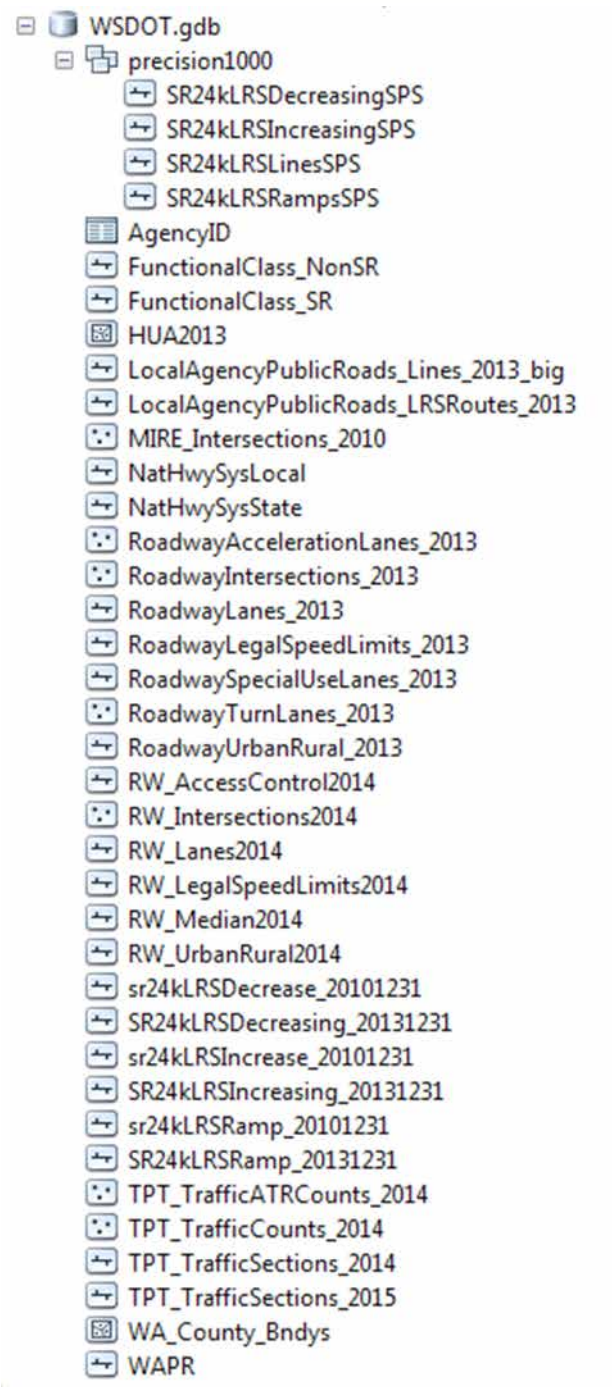

WSDOT.gdb
- precision1000
  - SR24kLRSDecreasingSPS
  - SR24kLRSIncreasingSPS
  - SR24kLRSLinesSPS
  - SR24kLRSRampsSPS
- AgencyID
- FunctionalClass_NonSR
- FunctionalClass_SR
- HUA2013
- LocalAgencyPublicRoads_Lines_2013_big
- LocalAgencyPublicRoads_LRSRoutes_2013
- MIRE_Intersections_2010
- NatHwySysLocal
- NatHwySysState
- RoadwayAccelerationLanes_2013
- RoadwayIntersections_2013
- RoadwayLanes_2013
- RoadwayLegalSpeedLimits_2013
- RoadwaySpecialUseLanes_2013
- RoadwayTurnLanes_2013
- RoadwayUrbanRural_2013
- RW_AccessControl2014
- RW_Intersections2014
- RW_Lanes2014
- RW_LegalSpeedLimits2014
- RW_Median2014
- RW_UrbanRural2014
- sr24kLRSDecrease_20101231
- SR24kLRSDecreasing_20131231
- sr24kLRSIncrease_20101231
- SR24kLRSIncreasing_20131231
- sr24kLRSRamp_20101231
- SR24kLRSRamp_20131231
- TPT_TrafficATRCounts_2014
- TPT_TrafficCounts_2014
- TPT_TrafficSections_2014
- TPT_TrafficSections_2015
- WA_County_Bndys
- WAPR

Figure 36. WSDOT Feature Class Geospatial Data and Tabular Data.

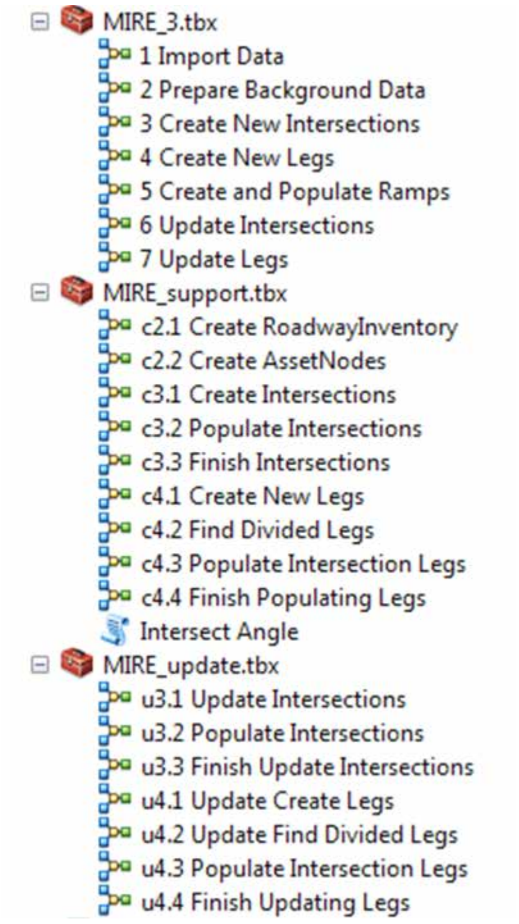
### *Initial Processing*

Most of the datasets were projected to the South Washington State Plane coordinate system (NAD 1983 HARN State Washington South FIPS 4602 Feet). However, some of the datasets were unprojected, for example *MIRE_Intersections_2010*, so TTI projected these datasets to the same South Washington State Plane coordinate system.

WSDOT provided numerous linear roadway asset datasets and two intersection datasets: *MIRE_Intersections_2010* and *RoadwayIntersections_201*3. There was some overlap between the two intersection datasets, with the majority of spatial features in the *MIRE_Intersections_2010* dataset included in the *RoadwayIntersections_201*3 dataset. As a result, TTI modified the RDE tool to merge attribute data from both datasets and avoid duplication of intersection features. Accordingly, the resulting intersection dataset included some intersections with attributes from either input dataset, and some intersections with data from both input datasets.

### *RDE Tool Toolbox Modifications*

The RDE tool is comprised of models that are used to process intersections and intersection legs datasets. The original RDE tool consisted of four models, the project team added numerous models and submodels and organized them into three toolboxes to create the models for the WSDOT RDE tool, as shown in Figure 37.



- MIRE_3.tbx
  - 1 Import Data
  - 2 Prepare Background Data
  - 3 Create New Intersections
  - 4 Create New Legs
  - 5 Create and Populate Ramps
  - 6 Update Intersections
  - 7 Update Legs
- MIRE_support.tbx
  - c2.1 Create RoadwayInventory
  - c2.2 Create AssetNodes
  - c3.1 Create Intersections
  - c3.2 Populate Intersections
  - c3.3 Finish Intersections
  - c4.1 Create New Legs
  - c4.2 Find Divided Legs
  - c4.3 Populate Intersection Legs
  - c4.4 Finish Populating Legs
  - Intersect Angle
- MIRE_update.tbx
  - u3.1 Update Intersections
  - u3.2 Populate Intersections
  - u3.3 Finish Update Intersections
  - u4.1 Update Create Legs
  - u4.2 Update Find Divided Legs
  - u4.3 Populate Intersection Legs
  - u4.4 Finish Updating Legs

**Figure 37. WSDOT RDE Tool Models.**

The first five models in the *MIRE_3* toolbox are used for creating new intersection, intersection leg, and ramp feature classes, while models six and seven are used to update existing spatial features. Figure 38 and Figure 39 provide a schema of the create business process and the update business process, and illustrate how the various models and submodels relate to each other.

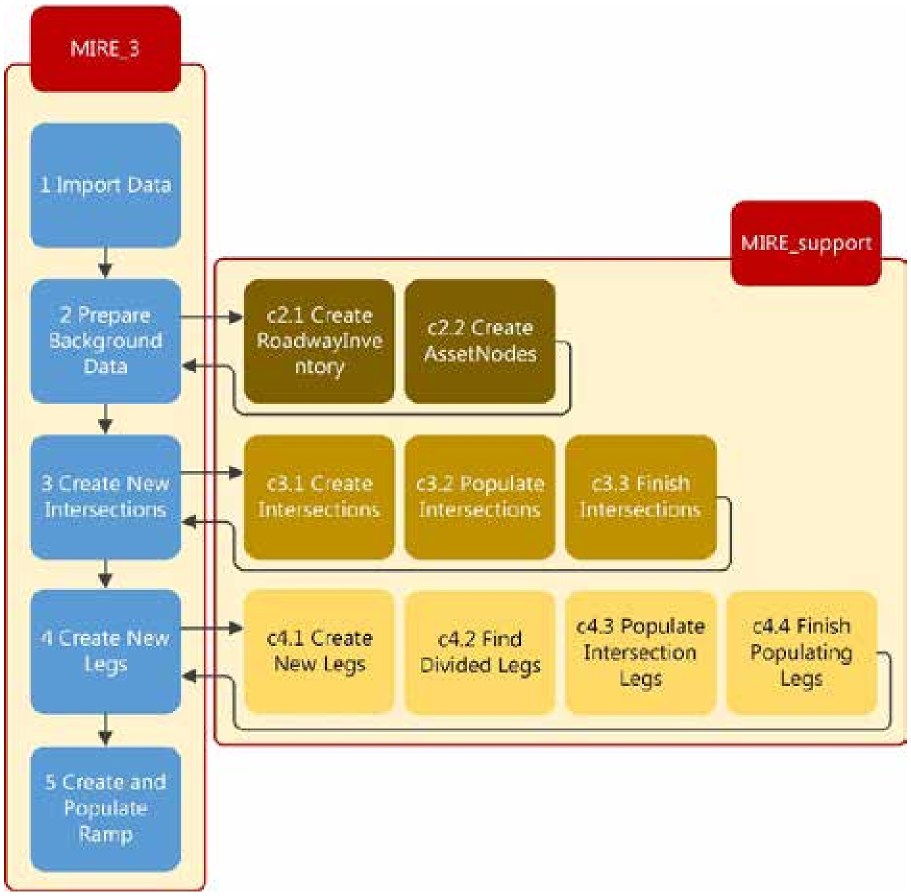

MIRE_3
1 Import Data
2 Prepare Background Data
3 Create New Intersections
4 Create New Legs
5 Create and Populate Ramp
MIRE_support
c2.1 Create RoadwayInventory
c2.2 Create AssetNodes
c3.1 Create Intersections
c3.2 Populate Intersections
c3.3 Finish Intersections
c4.1 Create New Legs
c4.2 Find Divided Legs
c4.3 Populate Intersection Legs
c4.4 Finish Populating Legs

**Figure 38. Model Relationships and Execution Sequence for WSDOT "Create" Business Process.**

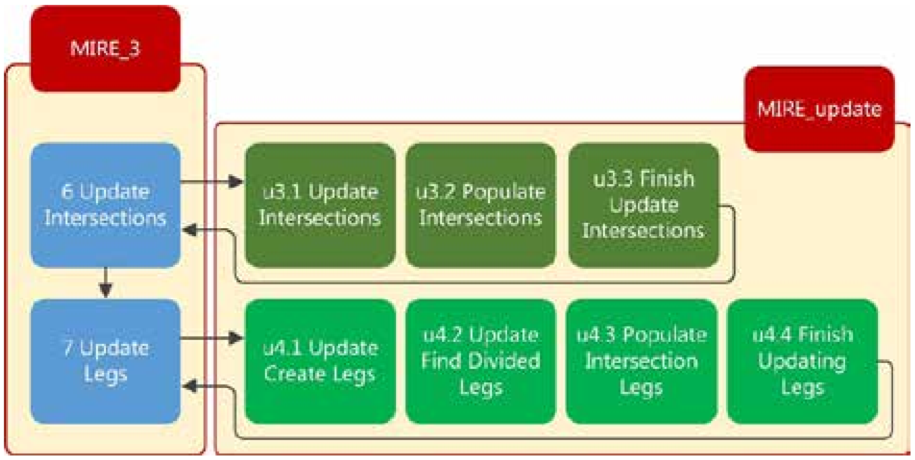

MIRE_3
6 Update Intersections
7 Update Legs
MIRE_update
u3.1 Update Intersections
u3.2 Populate Intersections
u3.3 Finish Update Intersections
u4.1 Update Create Legs
u4.2 Update Find Divided Legs
u4.3 Populate Intersection Legs
u4.4 Finish Updating Legs

**Figure 39. Model Relationships and Execution Sequence for WSDOT "Update" Business Process.**

***Create Business Process Models***

### Model 1 Import Data

This model imports all input datasets from various data sources into one single geodatabase. The model uses the ArcGIS Feature Class to Feature Class tool to import the transportation agency's datasets into the RDE tools InternalData geodatabase. The imported datasets can then be used by subsequent models. *1 Import Dat*a does not execute any submodels.

Note that it is important to ensure that all datasets that will be imported using the same coordinate system. In addition, each dataset has fields that are required for the model to execute. These fields are dependent on required fields in other subsequent models, but can be adjusted as needed.

### Model 2 Prepare Background Data

This model executes the submodels *c2.1 Create RoadwayInventory* and *c2.2 Create AssetNodes*. The model combines features from the dataset *LocalAgencyPublicRoads_Lines_2013* and the dataset *SR24kLRSLinesSPS*, which contain roadway features from the local network and the state route network. In addition, the model adds the field *SOURCE* to the output feature class to keep track of which dataset a feature originated from. The model assign the value "local" to features from the dataset *LocalAgencyPublicRoads_Lines_2013*, and the value "state" to features from the dataset *SR24kLRSLinesSPS*. The model also adds several other fields needed for data processing. Below is the list of the tasks completed by the model:

- Add and assign *SOURCE* field to combined state and local route dataset.
- Add and assign *JURISDICTION* field to state route dataset.
- Add and assign *RT_TYPES*, *RT_TYPEB*, and *LOCAL_UNIQ_ID* to local route dataset.
- Add *STATE_UNIQ_ID* to state route dataset.
- Merge datasets and add and assign *uniqueID* field.
- Add and assign Route name.

### Model 3 Create New Intersections

This model executes the submodels *c3.1 Create Intersections*, *c3.2 Populate Intersections*, and *c3.3 Finish Intersections*. The inputs of the model are *RoadwayInventory*, *DOT_Intersection*, and *AssetNode* datasets. The *RoadwayInventory* and *DOT_Intersection* feature class are created by previous models. The model copies attribute data from the *RoadwayInventory* dataset to the *DOT_Intersection* dataset. Below is the list of tasks completed by the model:

- Spatial intersect between *RoadwayInventory* dataset and *DOT_Intersection* dataset.
- Determine major and minor roads based highway hierarchy in *RT_TYPEB* field.
- Calculate unique ID for major and minor roads.
- Calculate intersection route type based on *RT_TYPEA*.
- Determine major and minor road name.
- Calculate jurisdiction.
- Calculate major and minor road offset.

- Determine the number of legs at each intersection.
- Run a linear directional mean to calculate route directions.
- Calculate the beginning and ending of a major influence zone.
- Calculate the beginning and ending of a minor influence zone.

### Model 4 Create New Legs

This model executes the submodels *c4.1 Create New Legs*, *c4.2 Find Divided Legs*, *c4.3 Populate Intersection Legs*, and *c4.4 Finish Populating Legs*. The model creates new intersection legs around each feature in the *Intersections* dataset. As a result, intersection legs can only be created after features in *Intersections* have been created. By default, each leg is 50 feet in length, which is a value that can be adjusted before running the model. After crating the leg features, the model populates attribute data for all intersection legs. Below is a list of tasks completed by the model:

- Select all intersections with the field *New*=-1.
- Create a 50 feet buffer around each intersection feature and create leg features.
- Set the value in *Agency ID* to the value in *NodeID.*
- Identify major and minor roads at each intersection.
- Add the field *LegID* and calculate its value.
- Calculate leg direction and leg offsets.
- Populate the speed limit value.
- Populate the leg median type.

### Model 5 Create and Populate Ramp

This model merges the *Ramp* template created by the project team with *SR24kLRSRamp_20131231*, the ramp dataset provided by WSDOT. The model also assigns an ID to each ramp feature.

## *Update Business Process Models*

### Model 6 Update Intersections

This model executes the submodels *u3.1 Update Intersections*, *u3.2 Populate Intersections*, and *u3.3 Finish Update Intersections*. The model updates attribute data for all intersection features in the *Intersection* dataset. This model should be executed when input datasets have been changed, e.g., some intersections have been retired, or attribute data in the *RoadwayInventory* dataset (or its sources) has been changed.

### Model 7 Update Legs

This model executes the submodels *u4.1 Update Create Legs*, *u4.2 Update Find Divided Legs*, *u4.3 Populate Intersection Legs*, and *u4.4 Finish Updating Legs*. The model updates attribute data for all intersection leg features in the *IntersectionLeg* dataset. This model should be executed when input datasets have been changed.

### *Manual Data Entry Interfaces*

The project team implemented the generic data entry interfaces that were described in detail in a previous chapter. The main effort of the implementation was to make changes to the interfaces in terms of the fields that are shown and can be edited. Since the number of fields in the intersection and intersection leg templates increased substantially, there was some discussion whether all fields should be shown on the interfaces, or just a subset. In the end, WSDOT preferred that all fields be shown on the manual data entry interfaces. As a result, any of the fields can be edited on the interfaces as long as the computer using the RDE tool has a sufficiently high monitor resolution.

## Outlook and Lessons Learned

The RDE tool implementation at WSDOT was the first pilot implementation using the upgraded RDE tool. The project team took note of the following lessons learned:

- **Have the in-person workshops early in the implementation process.** Initially, the project team handled most of the project communications and planning through web conferences and phone calls. While this is useful to some degree, the project reached a point where it became necessary to meet in person to discuss project development details. For future implementations, the project team recommends to have that in-person meeting early in the process, possibly within the first three months of a one-year implementation project. The project team noted that once everybody got to know each other, communications were much improved and overall development cycles were much shorter.
- **Have a dedicated contact for RDE tool updates.** Before the in-person meeting there was some uncertainty about who to include in regular tool update communications. After the in-person meeting, it became clear who to contact in which type of development scenario, and WSDOT provided a dedicated contact for all technical updates and questions. This made a significant and positive impact on tool development cycles.
- **Data assembly, stakeholder feedback, and tool modification might take longer than anticipated.** Assembling input datasets, creating a data mapping document, determining data template contents, and providing feedback at various stages of the tool implementation process are time consuming and involve collaboration from a diverse group of transportation agency individuals. Usually, most of these individuals are users of the data, while few individuals manage the data and are familiar with data management and processing issues in Geographic Information Systems. As a result, decisions on data management strategies as well as feedback on preliminary tool improvements might take longer than anticipated. This issue is compounded when outside local transportation agencies are involved to provide or share data with a state agency, and the number of people involved in providing and processing datasets increases. Furthermore, some of the RDE tool upgrades had never been implemented, so it was a challenge for the project team to provide accurate estimates of required resources.

WSDOT indicated that as a follow-up to the implementation, the DOT is interested in a new project that would involve data extraction and exchange from local data sources.

# 9. Case Study: Implementation of RDE Tool at Missouri State DOT

This chapter provides a summary of the pilot RDE tool implementation at MoDOT and the process the RDETAP team followed to facilitate implementation. The intent of the chapter is to provide future transportation agencies an idea of the resources and efforts needed to conduct a RDE tool implementation by providing a real-life implementation example.

The project team essentially followed the recommended implementation process described in Chapter 7. This chapter provides a summary of activities at each step of the implementation project.

## Form Implementation Team

The pilot implementation team consisted of the following groups:

- Project team, consisting of FHWA project manager, Leidos project manager, TTI implementation manager, and TTI programmer.
- MoDOT staff, including MoDOT GIS data manager, MoDOT data management supervisor, and MoDOT Safety Analyst programmer.
- City of Springfield, Missouri, GIS data manager and traffic data manager.
- St. Louis County, Missouri, GIS data manager and traffic data manager.

Once the implementation team was formed, the project team conducted a kickoff meeting to introduce the RDE tool and purpose of the project, introduce important project contacts, and discuss the necessary steps to complete the next step of the implementation process.

## Establish Goals and Objectives in General Terms

Following the kickoff meeting, TTI collected requested input datasets with the intent to come up with a general implementation plan. MoDOT's main goal for the RDETAP project was to integrate several internal and local roadway datasets to create an integrated roadway dataset that could be exported to AASHTOWare Safety Analyst, the roadway safety analysis tool currently being implemented at the DOT. MoDOT was also interested to determine whether the RDE tool could calculate certain MIRE data elements not available from currently available data sources. TTI analyzed the contents of the input datasets and created an Excel spreadsheet that produced a preliminary mapping of existing state data to the available MIRE/Safety Analyst templates. The result of this task was a detailed description of all data elements and their transcription into draft, MoDOT-specific output data templates.

In addition, the project team developed data models and presentation materials for the implementation workshops. The project team developed the following objectives:

- Create four MIRE-compatible GIS layers for the feature types intersections, intersection legs, ramps, and segments.
- Extract roadway inventory data from MoDOT GIS linear and point data, and transfer the data into the MIRE-compatible GIS layers stored in a geodatabase.
- Expand the ArcGIS models to calculate data fields that can be extracted using the GIS roadway geometry.

- Extract traffic counts from locally available datasets and attach data to the intersection layer using a neighborhood algorithm.
- Investigate the possibility to extract and transfer other locally available data to the MoDOT data layers.
- Document necessary changes to RDE tool based on MoDOT data management process and requirements, define input data and output templates, and modify tool accordingly.
- Provide a bug-free and working RDE tool to MoDOT.
- Provide technical support to implement the tool at MoDOT.
- Provide all available documentation related to the RDE tool.

### Conduct Implementation Team Workshop

The project team then set up a series of workshops over the course of a week to discuss the output data templates, and methods to derive the data. The meetings were scheduled as follows:

- Day 1: Meet with MoDOT staff, including GIS data management staff, IT management, and safety data analysts to discuss the following:
    - Meet in the morning to discuss the project overall, goals and objectives, current implementation status.
    - Meet in the afternoon to discuss MoDOT's data collection and analysis process, safety data analysis process using tool output including custom interfaces, and MoDOT's plans for future process enhancements. Discuss sample data provided by MoDOT and data mapping.
- Day 2: Meet with MoDOT staff, including GIS data management staff, IT management, and safety data analysts to discuss the following:
    - Meet in the morning to continue discussion on future improvements, including roadway segment and custom interfaces, and discuss tool implementation schedule, trial runs, and tests.
    - Meet in the afternoon with Safety Analyst team to discuss input requirements, participate in Safety Analyst demo, and discuss GIS data modification and export/import business process.
- Day 3: Meet with representatives from City of Springfield including GIS staff, safety analysts, and IT management. Include representatives from MoDOT, in person or via conference phone, as available.
    - Provide overview/description of project, goals and objectives, anticipated outcomes, and potential benefits to local agencies.
    - Discuss City of Springfield safety data collection and analysis process.
    - Discuss data needs and potential project collaboration.
    - Discuss expectations and schedule.
- Day 4: Meet with representatives from St. Louis County including GIS staff, safety analysts, and IT management. Include representatives from MoDOT, in person or via conference phone, as available.

- Provide overview/description of project, goals and objectives, anticipated outcomes, and potential benefits to local agencies.
- Discuss St. Louis County safety data collection and analysis process.
- Discuss data needs and potential project collaboration.
- Discuss expectations and schedule.

In addition, the project team developed data models and presentation materials for the implementation workshops.

### Develop RDE Tool Modification Plan, Timeline, and Milestones

Following the workshop meetings, the project team revised the data mapping document and developed a detailed implementation plan. The plan included tentative milestones for the following:

- Review of MoDOT Safety Analyst requirements as described in Oracle export files.
- Review of local datasets and development of data integration plan.
- RDE tool modifications and initial submission to MoDOT.
- RDE tool installation on MoDOT server.
- RDE tool testing and follow-up phase.
- Additional RDE tool modifications and final tool submission to MoDOT.

In addition to the modification plan, the project team developed a task list that described the RDE tool modifications in detail. This allowed the project team to schedule resources and update meetings at project milestones. Table 7 provides an overview of the timeline for the implementation of the RDE tool at MoDOT along with a description of major project development milestones.

**Table 7. MoDOT RDE Tool Pilot Implementation Timeline.**

| Date | Milestone |
|---|---|
| 12/2015 | Project team delivers webinar to WSDOT to demo RDE tool and discuss potential pilot. |
| 12/2015 | MoDOT becomes second state to participate in RDE tool implementation, implementation team formed. |
| 12/2015 | TTI receives required MoDOT input datasets and starts data analysis and data mapping. |
| 01/2016 | Project team meets with MoDOT and local transportation agencies to discuss goal and objectives, review available data. |
| 03/2016 | Project team agrees on project implementation plan and plan to integrate local data. |
| 08/2016 | TTI delivers data models and templates to MoDOT. |
| 10/2016 | TTI delivered the first version of the modified RDE tool. |
| 10/2016 | Conference call with MoDOT to discuss RDE tool and upgrades. |
| 12/2016 | TTI delivers update to modified RDE tool. |

## Conduct Implementation Team Meetings (Webinars) at Milestones

The project team conducted several update meetings to discuss project progress and RDE tool modifications. Once MoDOT staff was ready to provide feedback, the implementation team conducted a progress meeting. As with the WSDOT implementation, the implementation team made frequent use of web conferencing, which allowed the transportation agency to review tool upgrades remotely, and engage in detailed discussion while reviewing the tool remotely.

## RDE Tool Modifications

### *MoDOT Input Data*

TTI received several files from MoDOT to include in the RDE tool processing. Table 8 provides an overview of the MoDOT input files that TTI reviewed and included in the data mapping file.

**Table 8. MoDOT Input Files Included in Data Integration Review.**

| Owner | File Name | File Type | File Content Description |
|---|---|---|---|
| MoDOT | MoDOT.mdb | ESRI personal geodatabase | TW_INTERSECTION, SS_SEGMENT, and SS_PAVEMENT_CURRENT_TO_ INTERCHANGE layers |
| MoDOT | MODOT_Export.gdb | ESRI file geodatabase | SS_PAVEMENT_CURRENT and SS_ INTERSECTION_CURRENT layers |
| MoDOT | arcsde_data.gdb | ESRI file geodatabase | City, county, district, and other polygons |

Upon completion of the review, TTI created a data mapping spreadsheet that listed file information and specific data about each MoDOT feature class. For example, Table 9 provides an overview of the MoDOT feature class *SS_PAVEMENT_CURRENT* that was provided to the implementation team in geodatabase *MODOT.gdb*.

**Table 9. Overview of MoDOT Feature Class *SS_PAVEMENT_CURRENT*.**

| Item | Description |
|---|---|
| Owner | MoDOT |
| File Name | MODOT.gdb |
| File Date | 06/17/2015 |
| Table/Feature | SS_PAVEMENT_CURRENT |
| Feature Type | Polyline |
| Geographical Extent | Area around City of Springfield and St. Louis County. |
| Count of Records | 124,761 |

Table 10 provides a selection of the *SS_PAVEMENT_CURRENT* attributes and shows how some of these attributes were transcribed to RDE tool datasets and fields. If an attribute does not appear in the columns RDE Tool Dataset Name or RDE Tool Dataset Field Name, the attribute was not included in the data extraction process.

**Table 10. Data Extraction Mapping for *SS_PAVEMENT_CURRENT* Feature Class.**

| Attribute | Data Type | RDE Tool Dataset Name | RDE Tool Dataset Field Name |
|---|---|---|---|
| SS_PAVEMENT_ID | Object ID | | |
| TRAVELWAY_ID | Long Integer | | |
| YEAR | Short Integer | | |
| TRF_INFO_SEG_ID | Long Integer | Segment | agencyID |
| TRF_INFO_SEG_DESC | Text | | |
| TRAVELWAY_DESG | Text | | |
| TRAVELWAY_NAME | Text | Segment, Intersection | routeDisplayName, minorRoadName |
| TRAVELWAY_DIR | Text | Segment, Intersection | majorRoadDirection |
| TRAVELWAY_OFST_DIR | Text | | |
| DISTRICT | Short Integer | | |
| COUNTY_NAME | Text | Segment | CNTY_NM |
| CNTL_TW_ID | Long Integer | | |
| CNTL_TW_DESG | Text | | |
| AREA_DESG_NAME | Text | | |
| NUMBER_OF_LANES | Short Integer | Segment | numThruLaneTotal |
| AADT | Double | Segment | maxAADT |
| SURFACE_TYPE | Text | Segment | SRFC_TYPE |
| SURFACE_DATE | Date | | |
| FUNC_CLASS_NAME | Text | Segment | roadwayClass1 |

### *Data from Other Sources*

TTI also received several input data files from local transportation agencies, as shown in Table 11. TTI reviewed these files to determine whether there was a method to integrate the data with MoDOT geospatial data.

**Table 11. List of Local Transportation Agency Input Files Included in Data Integration Review.**

| Owner | File Name | File Type | File Content Description |
|---|---|---|---|
| City of Springfield | Springfield Base Model.sim | Trafficware SimTraffic simulation file | City of Springfield traffic simulation data |
| City of Springfield | Springfield Base Model.syn | Trafficware Synchro file | Traffic control, intersection, and lane data for City of Springfield |
| City of Springfield | TempWork.gdb | ESRI file geodatabase | Intersections, streets, traffic signals, and other City of Springfield feature classes. |
| St. Louis County | Traffic_Counts.shp | ESRI shapefile | Inventory of traffic counts and locations in St. Louis County. |
| St. Louis County | Municipal_Boundaries.shp | ESRI shapefile | St. Louis County municipal boundary limits. |
| St. Louis County | Parcels.shp | ESRI shapefile | St. Louis County property parcel boundary limits. |

### *Integration of State and Local Datasets*

In contrast to the RDE tool implementation at WSDOT, the MoDOT implementation focused on the integration of data from two local transportation agencies, the City of Springfield, Missouri, and St. Louis County, Missouri. Whereas integration of data stored in ESRI shapefiles and geodatabases is relatively straight forward, the project team was unsure to what degree data could be integrated stored in the data in Synchro and SimTraffic files. TTI was able to export roadway data from the City of Springfield Synchro and SimTraffic files in CSV format. However, both input files were created in Synchro without the geospatial referencing feature enabled, therefore TTI was unable to link the lane, layout, timing, and volume data included in the CSV files to the MoDOT roadway network.

Upon completion of the review of local input files, three files were selected to be included in the state/local data integration effort: the St. Louis County *Traffic_Counts* shapefile, a City of Springfield *street* feature class, and a City of Springfield *TRAFFIC_COUNTS* feature class. Table 12 provides an overview of the City of Springfield feature class *street* that was provided to the implementation team in geodatabase *TempWork.gdb*.

**Table 12. Overview of City of Springfield Feature Class *street*.**

| Item | Description |
|---|---|
| Owner | City of Springfield, Missouri |
| File Name | TempWork.gdb |
| File Date | 01/22/2016 |
| Table/Feature | Street |
| Feature Type | Polyline |
| Geographical Extent | City of Springfield, Missouri |
| Count of Records | 30,368 |

Table 13 provides a segment of the *street* attributes, and shows how some of its attributes were transcribed to RDE tool datasets and fields.

**Table 13. Data Extraction Mapping for City of Springfield *street* Feature Class.**

| Attribute | Data Type | RDE Tool Dataset Name | RDE Tool Dataset Field Name |
|---|---|---|---|
| OBJECTID | Object ID | | |
| Shape | Geometry | | |
| CLASS | Text | Segment | roadwayClass1 |
| STREET_NAM | Text | Segment | routeDisplayName |
| STREET_TYP | Text | | |
| SUF_DIR | Text | | |
| L_COUNTY | Text | | |
| EDITDATE | Date | | |
| NAME | Text | Segment | routeDisplayName |

Table 14 provides an overview of the St. Louis County shapefile *Traffic_Counts* that was provided to the implementation team.

**Table 14. Overview of St. Louis County Feature Class *Traffic_Counts*.**

| Item | Description |
|---|---|
| Owner | St. Louis County, Missouri |
| File Name | Traffic_Counts.shp |
| File Date | 01/22/2016 |
| Table/Feature | n/a |
| Feature Type | Point |
| Geographical Extent | St. Louis County, Missouri |
| Count of Records | 1,043 |

Table 15 provides a segment of the *Traffic_Counts* attributes, and shows how some of its attributes were transcribed to RDE tool datasets and fields. Note that the implementation team opted to change the attribute names of the *Traffic_Counts* feature class to indicate the source, St. Louis County.

**Table 15. Data Extraction Mapping for St. Louis County *Traffic_Counts* Feature Class.**

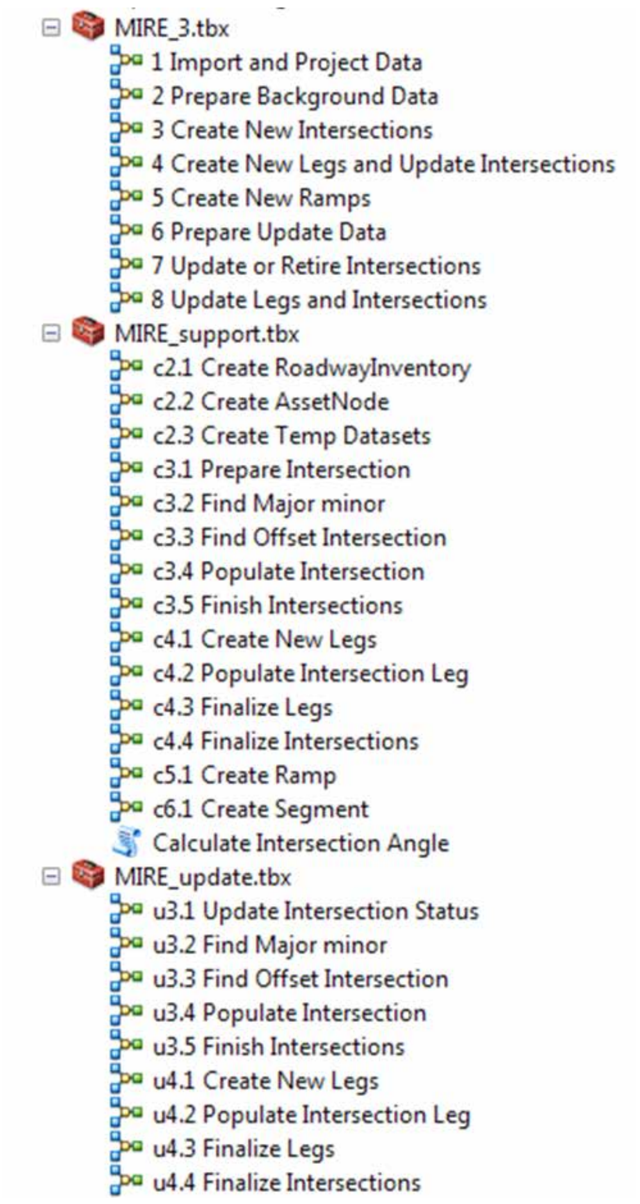
| Attribute | Data Type | RDE Tool Dataset Name | RDE Tool Dataset Field Name |
|---|---|---|---|
| FID | Object ID | | |
| Shape | Geometry | | |
| Onstreet | Text | Intersection | SLC_Onstreet |
| Atstreet | Text | Intersection | SLC_Atstreet |
| Month | Text | Intersection | SLC_Month |
| Bridge_ID | Text | | |
| DirFrom | Text | Intersection | SLC_DirFrom |
| LocationID | Text | | |
| PkHrVol | Short Integer | Intersection | SLC_PkHrVol |
| Max24HrDay | Text | Intersection | SLC_Max24HrDay |
| PkHrDay | Text | Intersection | SLC_PkHrDay |
| PkHrTime | Text | Intersection | SLC_PkHrTime |
| PkHrAMPM | Text | Intersection | SLC_PkHrAMPM |
| Month2 | Text | Intersection | SLC_Month2 |
| AWT | Long Integer | Intersection | SLC_AWT |
| Max24HrVol | Long Integer | Intersection | SLC_Max24HrVol |
| YearTxt | Text | Intersection | SLC_YearTxt |
| Year | Double | Intersection | SLC_Year |

### *Initial Processing*

The input datasets used different projected and unprojected coordinate systems. For the processed datasets stored in the *InternalData* geodatabase, TTI projected all datasets to the same NAD 1983 HARN State Plane Missouri Central FIPS 2402 coordinate system.

***RDE Tool Toolbox Modifications***

The RDE tool is comprised of models that are used to process geospatial features, for example intersections and intersection legs datasets. The project team made use of the modifications created for WSDOT, where appropriate, which resulted in 31 models and submodels, organized into three toolboxes as shown in Figure 40.



MIRE_3.tbx
- 1 Import and Project Data
- 2 Prepare Background Data
- 3 Create New Intersections
- 4 Create New Legs and Update Intersections
- 5 Create New Ramps
- 6 Prepare Update Data
- 7 Update or Retire Intersections
- 8 Update Legs and Intersections

MIRE_support.tbx
- c2.1 Create RoadwayInventory
- c2.2 Create AssetNode
- c2.3 Create Temp Datasets
- c3.1 Prepare Intersection
- c3.2 Find Major minor
- c3.3 Find Offset Intersection
- c3.4 Populate Intersection
- c3.5 Finish Intersections
- c4.1 Create New Legs
- c4.2 Populate Intersection Leg
- c4.3 Finalize Legs
- c4.4 Finalize Intersections
- c5.1 Create Ramp
- c6.1 Create Segment
- Calculate Intersection Angle

MIRE_update.tbx
- u3.1 Update Intersection Status
- u3.2 Find Major minor
- u3.3 Find Offset Intersection
- u3.4 Populate Intersection
- u3.5 Finish Intersections
- u4.1 Create New Legs
- u4.2 Populate Intersection Leg
- u4.3 Finalize Legs
- u4.4 Finalize Intersections

**Figure 40. MoDOT RDE Tool Models.**

The first five models in the *MIRE_3* toolbox are used for creating new intersection, intersection leg, and ramp feature classes, while models six, seven, and eight are used to update existing spatial features. Figure 41 and Figure 42 provide a schema of the create business process and the update business process, and illustrate how the various models and submodels relate to each other.

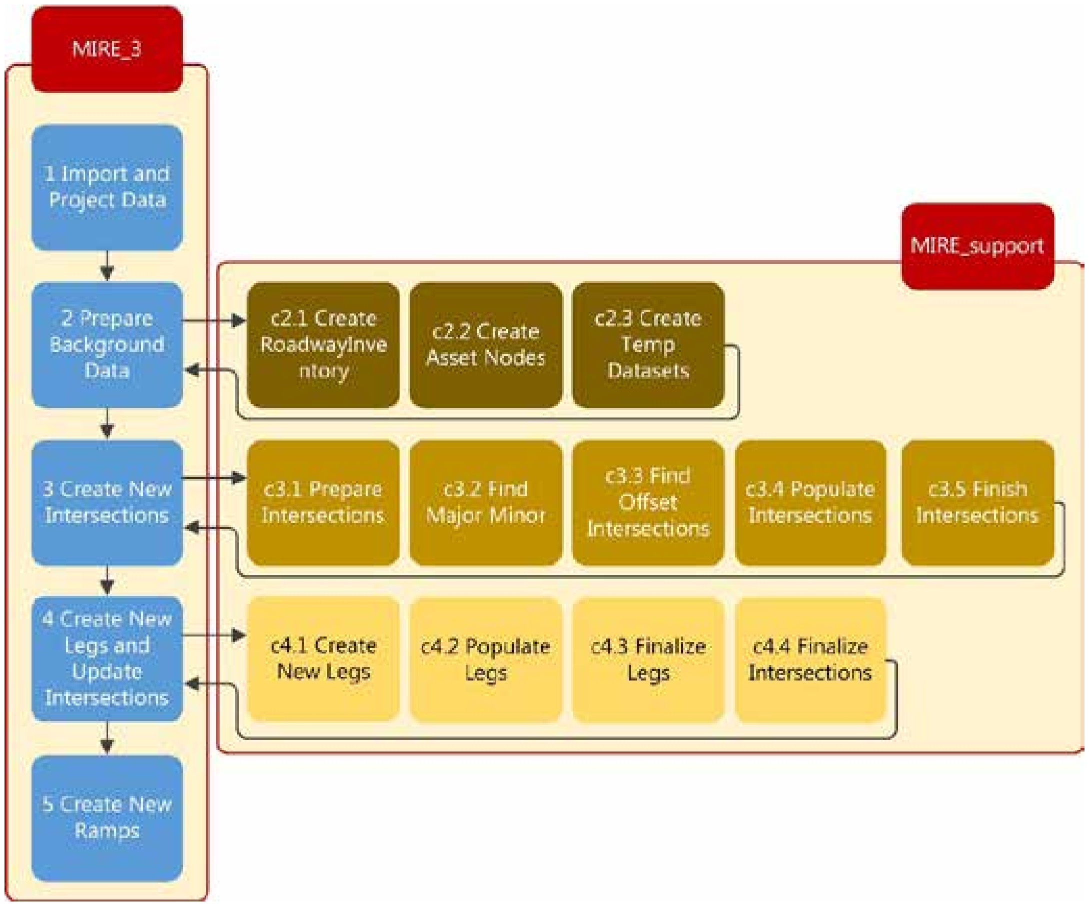

MIRE_3
1 Import and Project Data
2 Prepare Background Data
3 Create New Intersections
4 Create New Legs and Update Intersections
5 Create New Ramps
MIRE_support
c2.1 Create RoadwayInventory
c2.2 Create Asset Nodes
c2.3 Create Temp Datasets
c3.1 Prepare Intersections
c3.2 Find Major Minor
c3.3 Find Offset Intersections
c3.4 Populate Intersections
c3.5 Finish Intersections
c4.1 Create New Legs
c4.2 Populate Legs
c4.3 Finalize Legs
c4.4 Finalize Intersections

**Figure 41. Model Relationships and Execution Sequence for MoDOT "Create" Business Process.**

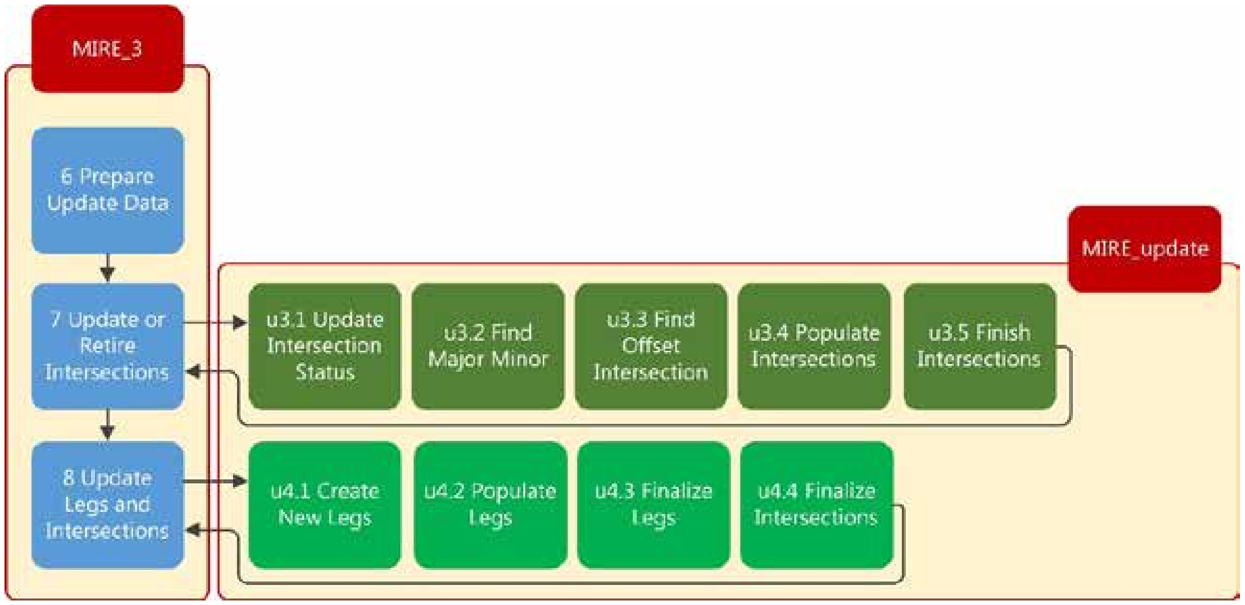

MIRE_3
6 Prepare Update Data
7 Update or Retire Intersections
8 Update Legs and Intersections
MIRE_update
u3.1 Update Intersection Status
u3.2 Find Major Minor
u3.3 Find Offset Intersection
u3.4 Populate Intersections
u3.5 Finish Intersections
u4.1 Create New Legs
u4.2 Populate Legs
u4.3 Finalize Legs
u4.4 Finalize Intersections

**Figure 42. Model Relationships and Execution Sequence for MoDOT "Update" Business Process.**

***Create Business Process Models***

## Model 1 Import and Project Data

This model imports all input datasets from various data sources into one single geodatabase, *InternalData.gdb*. The model uses the ArcGIS Feature Class to Feature Class tool to import the transportation agency's datasets into the RDE tools geodatabase. The imported datasets can then be used by subsequent models. Model *1 Import and Project Data* does not execute any submodels.

Note that it is important to ensure that all datasets that will be imported using the same coordinate system. In addition, each dataset has fields that are required for the model to execute. These fields are dependent on required fields in other subsequent models, but can be adjusted as needed.

## Model 2 Prepare Background Data

This model executes the submodels *c2.1 Create RoadwayInventory* and *c2.2 Create Asset Nodes*. The model uses features from the dataset *Pavement_Current* stored in the *InternalData* geodatabase, which contains roadway features from the Missouri local network and state route network. The model adds several fields needed for data processing. Below is the list of the tasks completed by the model:

- Remove duplicate lines in *Pavement_Current* dataset.
- Add and assign *RTE_TYPE* field based on *TRAVELWAY_DESG* field in *Pavement_Current* dataset.
- Add and assign *RouteID* based on *TRAVELWAY_ID* field in *Pavement_Current* dataset.
- Assign *AADT* field based on *TOTAL_AADT* field in *Pavement_Current* dataset.
- Add and assign *RteUniqID* to *RoadwayInventory* dataset.
- Create spatial intersect between all features in RoadwayInventory dataset to create AssetNode dataset.
- Remove duplicate features in *AssetNode* dataset.
- Assign *NodeID* to each feature in *AssetNode* dataset.

## Model 3 Create New Intersections

This model executes the submodels *c3.1 Prepare Intersections, c3.2 Find Major Minor, c3.3 Find Offset Intersections, c3.4 Populate Intersections*, and *c3.5 Finish Intersections*. The inputs of the model are *RoadwayInventory, Intersection_curren*t, and *AssetNode* datasets. The *RoadwayInventory* and *Intersection_current* feature class are created by previous models. The model copies attribute data from the *RoadwayInventory* dataset to the *Intersection_current* dataset. Below is the list of tasks completed by the model:

- Create temporary node dataset in *IntermediateData* geodatabase based on *Intersection_current* dataset and *AssetNode* dataset.
- Find major and minor roads at each feature in temporary node dataset.
- Find offset intersections and calculate the offset between intersecting roads.
- Populate features in temporary node dataset with data from input datasets.
- Merge features in temporary node dataset with *INTSECT_Template* in *InternalData* geodatabase.

- Copy the features created in the previous step to the *Intersection* dataset in the *MIREProject* geodatabase.

### Model 4 Create New Legs and Update Intersections

This model executes the submodels *c4.1 Create New Legs, c4.2 Populate Legs, c4.3 Finalize Legs*, and *c4.4 Finalize Intersections*. The model creates new intersection legs around each feature in the *Intersection* dataset. As a result, intersection legs can only be created after features in *Intersection* have been created. By default, each leg is 50 feet in length, which is a value that can be adjusted before running the model. After crating the leg features, the model populates attribute data for all intersection legs. Below is a list of tasks completed by the model:

- Select all features in *Intersection* dataset with the field *IsNew* equal to "*Yes*".
- Create a 50 feet buffer around each selected feature.
- Intersect the above buffer with the *RoadwayInventory* dataset to create new intersection leg features in a temporary dataset located in the *IntermediateData* geodatabase.
- Add the field *LegID* to the intersection leg features and calculate its value.
- Calculate the leg direction value for each intersection leg feature.
- Populate AADT and the speed limit value for each intersection leg feature.
- Populate the leg median type for each intersection leg feature.
- Merge the intersection leg features with *INTSECT_LEG_Template* to create the final output dataset *IntersectionLeg*.

### Model 5 Create New Ramps

This model selects ramp features from the *RoadwayInventory* dataset and merges the features with *RAMP_Template* in the *InternalData* geodatabase. Below is a list of tasks completed by the model:

- Select features from *RoadwayInventory* dataset marked as ramps.
- Merge the ramp features with *RAMP_Template* to create final output dataset *Ramp*.
- Populate data fields in Ramp such as *agencyID, numOfLanes, rampLength, RAMP_AADT_YR_NBR, RAMP_ADVRY_SPD_LMT_NBR, FUNC_CLASS_TYPE, UNIQ_INTCHG_ID, and UNIQ_RAMP_ID.*

### Model 6 Create New Segments

This model creates the segment dataset from *RoadwayInventory* and segment template. Below is a list of tasks completed by the model:

- Select features from *RoadwayInventory* dataset marked as segments.
- Merge the segment features with *SGMNT_Template* to create final output dataset Segment.
- Populate data fields in Segment such as agencyID, route display name, travel direction, county name, number of lanes, lane width, shoulder type, shoulder width, AADT, surface type, road class, county code, speed limit, bridge number, segment ID.

#### *Update Business Process Models*

#### Model 7 Prepare Update Data

This model creates several temporary datasets needed for verifying and running models 8 and 9.

#### Model 8 Update or Retire Intersections

This model executes the submodels *u3.1 Update Intersection Status, u3.2 Find Major Minor, u3.3 Find Offset Intersections, u3.4 Populate Intersections,* and *u3.5 Finish Intersections*. The model updates attribute data for all intersection features in the *Intersectio*n dataset. This model should be executed when input datasets have been changed, e.g., some intersections have been retired, or attribute data in the *RoadwayInventory* dataset (or its sources) has been changed. Model *7 Prepare Update Data* should be executed before executing model *8 Update or Retire Intersections*.

#### Model 9 Update Legs and Intersections

This model executes the submodels *u4.1 Create New Legs, u4.2 Populate Legs, u4.3 Finalize Legs,* and *u4.4 Finalize Intersections*. The model updates attribute data for all intersection leg features in the *IntersectionLeg* dataset. This model should be executed when input datasets have been changed and after model 8 *Update or Retire Intersections* has been executed.

### *Manual Data Entry Interfaces*

The project team implemented the generic data entry interfaces that were described in detail in a previous chapter. The main effort of the implementation was to make changes to the interfaces in terms of the fields that are shown and can be edited. Since the number of fields in the intersection and intersection leg templates increased substantially, there was some discussion whether all fields should be shown on the interfaces, or just a subset. MoDOT preferred that all fields be shown on the manual data entry interfaces. As a result, any of the fields can be edited on the interface as long as the computer using the RDE tool has a sufficiently high monitor resolution.

## Outlook and Lessons Learned

The RDE tool implementation at MoDOT was the second pilot implementation using the upgraded RDE tool. As a result, the implementation project benefited from some of the lessons learned during the first pilot with WSDOT. As a result, the implementation team opted to have a series of meetings with state and local stakeholders early in the implementation project. These meetings helped get the project kicked off on track and established clear communication channels for all parties involved. The project also adapted some of the code developed for WSDOT, which considerably sped up development work. The project also developed new code that might be of benefit for the RDE tool developed for WSDOT (and future implementation states).

# 10. Concluding Remarks

This Implementation and Programming Guide is intended to be a guide for the adaptation and implementation of the RDE tool components. It is written from the perspective of the project team that developed separate, i.e., state-specific versions of the RDE tool for the pilot states. Therefore, a certain module or feature of the RDE tool always depends on a characteristic of an input dataset used at a pilot state.

The project team attempted to make the data processing as transparent as possible; as a result, the RDE tool consists of several sub models that could be combined to streamline the model execution process. The project team decided to leave this task up to each pilot state to make modifications to best meet the needs of the DOT users.

The project team also discussed the development of an additional interface to edit segments in addition to the interfaces for intersections, intersection legs, and ramps. The project team did not pursue the development of the segment interface due to compatibility issues with the software development environment. The next RDETAP project is expected to rewrite the code for all current interfaces to make the code compatible with current software development environments, or potentially use ArcGIS native code, to avoid the compatibility and debugging issues that the project team encountered. The next RDETAP project is expected to expand the code to include the segment interface.

Readers should take note that MIRE version 1.0, which is the foundation of this MIRE pilot implementation, is in the process of being upgraded to version 2.0. Although the changes in version 2.0 appear to be minor, the templates used in this RDE tool should be compared to the final version of MIRE 2.0 to ensure their compatibility. In addition, there are plans to include MIRE in the surface transportation domain of the National Information Exchange Model (NIEM). Once the MIRE data elements are included in NIEM, it will provide another option to develop an information exchange that could transfer roadway data between state and local transportation agencies.

## 11. References

1. *MIRE MIS.* MIRE Management Information System, Office of Safety Programs, Federal Highway Administration, Washington, D.C., 2014. Available at http://safety.fhwa.dot.gov/rsdp/feasibility.cfm. Accessed on September 30, 2016.
2. Roadway Safety Data Program, Federal Highway Administration, U.S. Department of Transportation. http://safety.fhwa.dot.gov/rsdp/mire.aspx. Accessed on September 30, 2016.
3. *Guidance on State Safety Data Systems.* Federal Highway Administration, Office of Safety, Washington, D.C., March 15 2016. Available at http://safety.fhwa.dot.gov/legislationandpolicy/fast/docs/ssds_guidance.pdf. Accessed on September 30, 2016.
4. *Highway Safety Improvement Program, Implementation.* 23 CFR Section 924.11(b) (2016). Available at https://www.gpo.gov/fdsys/pkg/CFR-2016-title23-vol1/xml/CFR-2016-title23-vol1-sec924-11.xml. Accessed on September 30, 2016.

# Appendix I. MIRE Data Model Entity-Relationship Diagrams

## Intersection Entity-Relationship Diagram

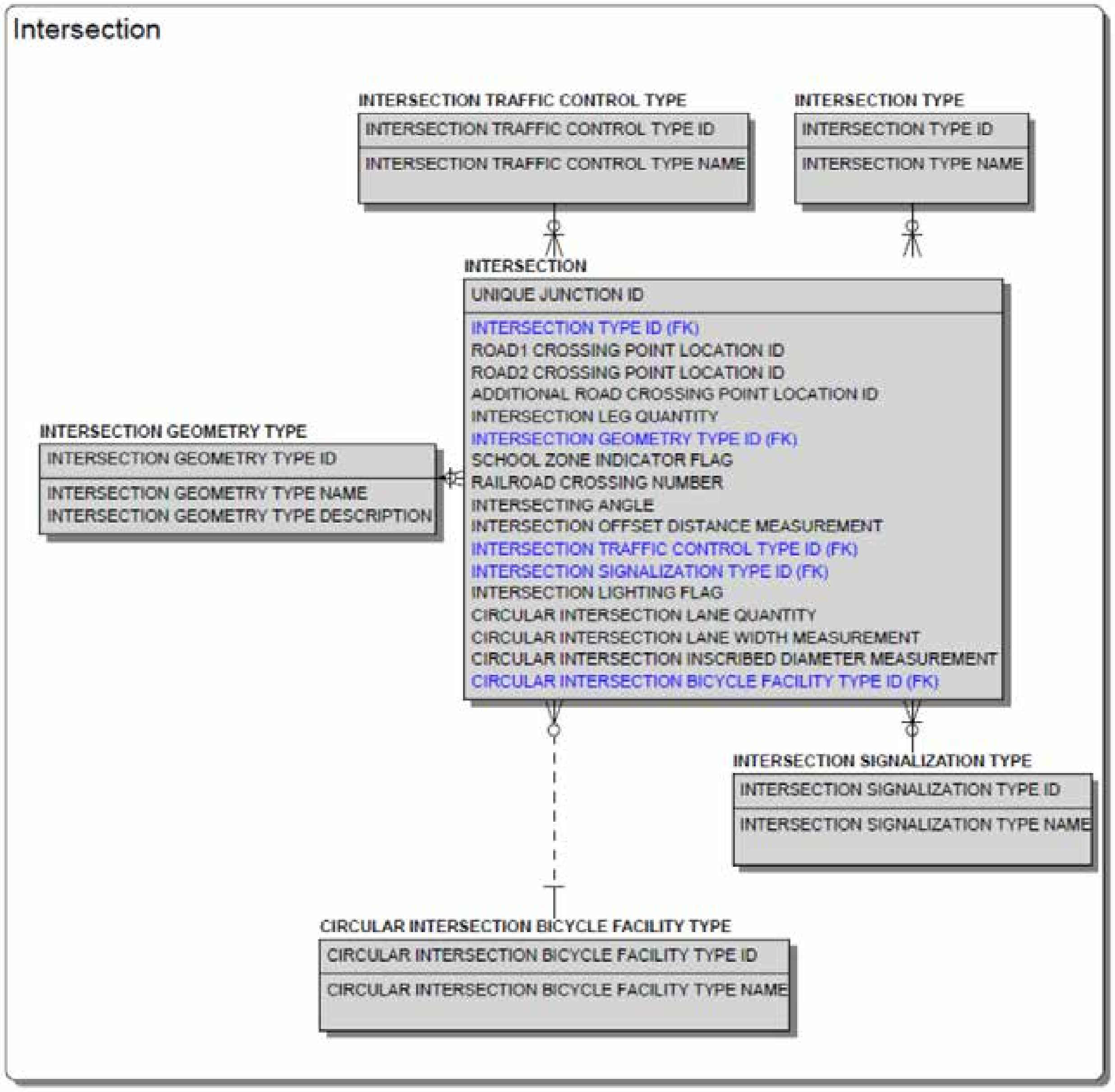

Intersection
INTERSECTION TRAFFIC CONTROL TYPE
INTERSECTION TRAFFIC CONTROL TYPE ID
INTERSECTION TRAFFIC CONTROL TYPE NAME
INTERSECTION TYPE
INTERSECTION TYPE ID
INTERSECTION TYPE NAME
INTERSECTION
UNIQUE JUNCTION ID
INTERSECTION TYPE ID (FK)
ROAD1 CROSSING POINT LOCATION ID
ROAD2 CROSSING POINT LOCATION ID
ADDITIONAL ROAD CROSSING POINT LOCATION ID
INTERSECTION LEG QUANTITY
INTERSECTION GEOMETRY TYPE ID (FK)
SCHOOL ZONE INDICATOR FLAG
RAILROAD CROSSING NUMBER
INTERSECTING ANGLE
INTERSECTION OFFSET DISTANCE MEASUREMENT
INTERSECTION TRAFFIC CONTROL TYPE ID (FK)
INTERSECTION SIGNALIZATION TYPE ID (FK)
INTERSECTION LIGHTING FLAG
CIRCULAR INTERSECTION LANE QUANTITY
CIRCULAR INTERSECTION LANE WIDTH MEASUREMENT
CIRCULAR INTERSECTION INSCRIBED DIAMETER MEASUREMENT
CIRCULAR INTERSECTION BICYCLE FACILITY TYPE ID (FK)
INTERSECTION GEOMETRY TYPE
INTERSECTION GEOMETRY TYPE ID
INTERSECTION GEOMETRY TYPE NAME
INTERSECTION GEOMETRY TYPE DESCRIPTION
INTERSECTION SIGNALIZATION TYPE
INTERSECTION SIGNALIZATION TYPE ID
INTERSECTION SIGNALIZATION TYPE NAME
CIRCULAR INTERSECTION BICYCLE FACILITY TYPE
CIRCULAR INTERSECTION BICYCLE FACILITY TYPE ID
CIRCULAR INTERSECTION BICYCLE FACILITY TYPE NAME

**Figure 43. Intersection Entity-Relationship Diagram in MIRE Data Model.**

## Intersection Leg Entity-Relationship Diagram

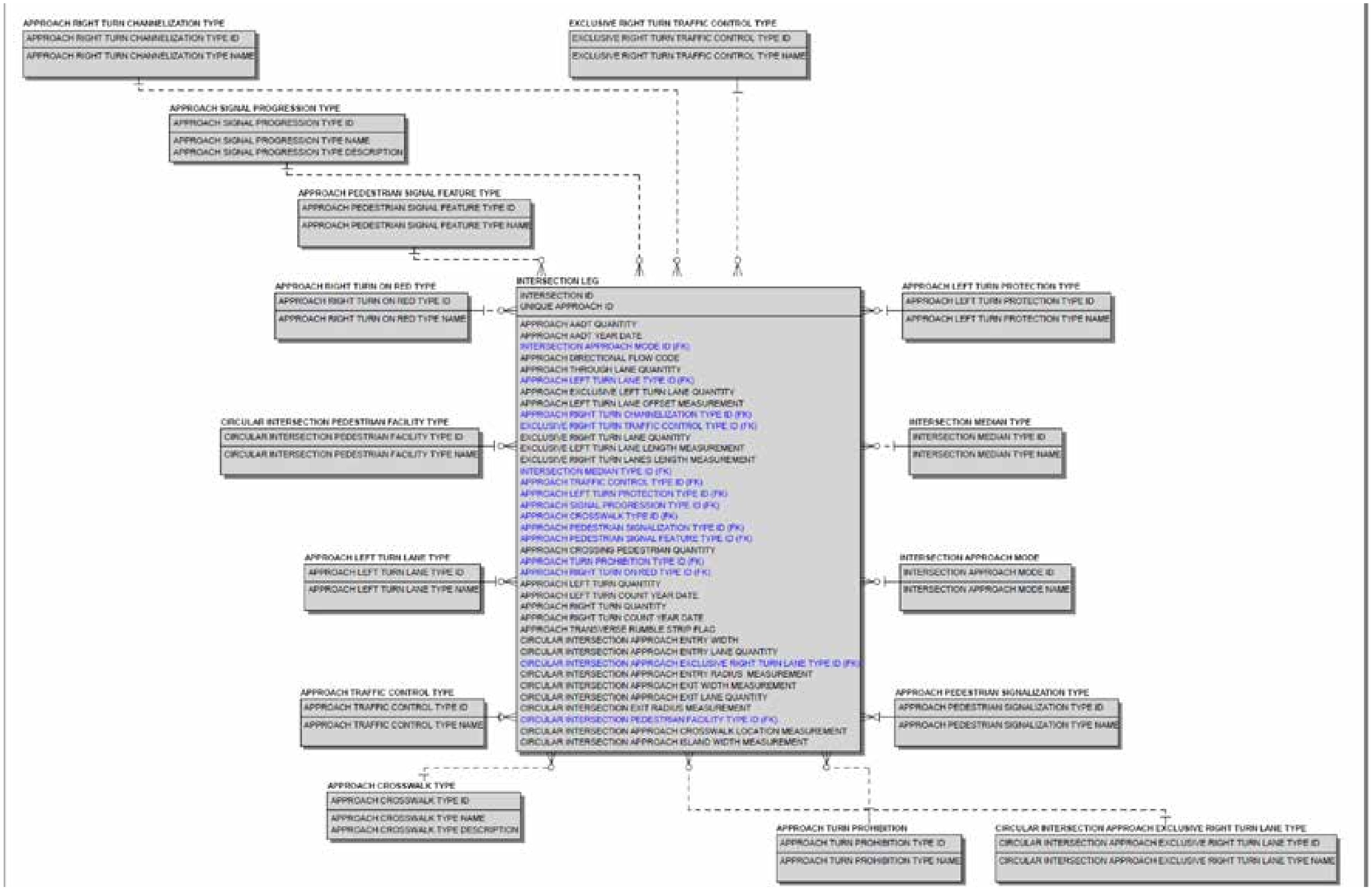

APPROACH RIGHT TURN CHANNELIZATION TYPE
APPROACH RIGHT TURN CHANNELIZATION TYPE ID
APPROACH RIGHT TURN CHANNELIZATION TYPE NAME
EXCLUSIVE RIGHT TURN TRAFFIC CONTROL TYPE
EXCLUSIVE RIGHT TURN TRAFFIC CONTROL TYPE ID
EXCLUSIVE RIGHT TURN TRAFFIC CONTROL TYPE NAME
APPROACH SIGNAL PROGRESSION TYPE
APPROACH SIGNAL PROGRESSION TYPE ID
APPROACH SIGNAL PROGRESSION TYPE NAME
APPROACH SIGNAL PROGRESSION TYPE DESCRIPTION
APPROACH PEDESTRIAN SIGNAL FEATURE TYPE
APPROACH PEDESTRIAN SIGNAL FEATURE TYPE ID
APPROACH PEDESTRIAN SIGNAL FEATURE TYPE NAME
APPROACH RIGHT TURN ON RED TYPE
APPROACH RIGHT TURN ON RED TYPE ID
APPROACH RIGHT TURN ON RED TYPE NAME
INTERSECTION LEG
INTERSECTION ID
UNIQUE APPROACH ID
APPROACH AADT QUANTITY
APPROACH AADT YEAR DATE
INTERSECTION APPROACH MODE ID (FK)
APPROACH DIRECTIONAL FLOW CODE
APPROACH THROUGH LANE QUANTITY
APPROACH LEFT TURN LANE TYPE ID (FK)
APPROACH EXCLUSIVE LEFT TURN LANE QUANTITY
APPROACH LEFT TURN LANE OFFSET MEASUREMENT
APPROACH RIGHT TURN CHANNELIZATION TYPE ID (FK)
EXCLUSIVE RIGHT TURN TRAFFIC CONTROL TYPE ID (FK)
EXCLUSIVE RIGHT TURN LANE QUANTITY
EXCLUSIVE LEFT TURN LANE LENGTH MEASUREMENT
EXCLUSIVE RIGHT TURN LANES LENGTH MEASUREMENT
INTERSECTION MEDIAN TYPE ID (FK)
APPROACH TRAFFIC CONTROL TYPE ID (FK)
APPROACH LEFT TURN PROTECTION TYPE ID (FK)
APPROACH SIGNAL PROGRESSION TYPE ID (FK)
APPROACH CROSSWALK TYPE ID (FK)
APPROACH PEDESTRIAN SIGNALIZATION TYPE ID (FK)
APPROACH PEDESTRIAN SIGNAL FEATURE TYPE ID (FK)
APPROACH CROSSING PEDESTRIAN QUANTITY
APPROACH TURN PROHIBITION TYPE ID (FK)
APPROACH RIGHT TURN ON RED TYPE ID (FK)
APPROACH LEFT TURN QUANTITY
APPROACH LEFT TURN COUNT YEAR DATE
APPROACH RIGHT TURN QUANTITY
APPROACH RIGHT TURN COUNT YEAR DATE
APPROACH TRANSVERSE RUMBLE STRIP FLAG
CIRCULAR INTERSECTION APPROACH ENTRY WIDTH
CIRCULAR INTERSECTION APPROACH ENTRY LANE QUANTITY
CIRCULAR INTERSECTION APPROACH EXCLUSIVE RIGHT TURN LANE TYPE ID (FK)
CIRCULAR INTERSECTION APPROACH ENTRY RADIUS MEASUREMENT
CIRCULAR INTERSECTION APPROACH EXIT WIDTH MEASUREMENT
CIRCULAR INTERSECTION APPROACH EXIT LANE QUANTITY
CIRCULAR INTERSECTION EXIT RADIUS MEASUREMENT
CIRCULAR INTERSECTION PEDESTRIAN FACILITY TYPE ID (FK)
CIRCULAR INTERSECTION APPROACH CROSSWALK LOCATION MEASUREMENT
CIRCULAR INTERSECTION APPROACH ISLAND WIDTH MEASUREMENT
APPROACH LEFT TURN PROTECTION TYPE
APPROACH LEFT TURN PROTECTION TYPE ID
APPROACH LEFT TURN PROTECTION TYPE NAME
CIRCULAR INTERSECTION PEDESTRIAN FACILITY TYPE
CIRCULAR INTERSECTION PEDESTRIAN FACILITY TYPE ID
CIRCULAR INTERSECTION PEDESTRIAN FACILITY TYPE NAME
INTERSECTION MEDIAN TYPE
INTERSECTION MEDIAN TYPE ID
INTERSECTION MEDIAN TYPE NAME
APPROACH LEFT TURN LANE TYPE
APPROACH LEFT TURN LANE TYPE ID
APPROACH LEFT TURN LANE TYPE NAME
INTERSECTION APPROACH MODE
INTERSECTION APPROACH MODE ID
INTERSECTION APPROACH MODE NAME
APPROACH TRAFFIC CONTROL TYPE
APPROACH TRAFFIC CONTROL TYPE ID
APPROACH TRAFFIC CONTROL TYPE NAME
APPROACH PEDESTRIAN SIGNALIZATION TYPE
APPROACH PEDESTRIAN SIGNALIZATION TYPE ID
APPROACH PEDESTRIAN SIGNALIZATION TYPE NAME
APPROACH CROSSWALK TYPE
APPROACH CROSSWALK TYPE ID
APPROACH CROSSWALK TYPE NAME
APPROACH CROSSWALK TYPE DESCRIPTION
APPROACH TURN PROHIBITION
APPROACH TURN PROHIBITION TYPE ID
APPROACH TURN PROHIBITION TYPE NAME
CIRCULAR INTERSECTION APPROACH EXCLUSIVE RIGHT TURN LANE TYPE
CIRCULAR INTERSECTION APPROACH EXCLUSIVE RIGHT TURN LANE TYPE ID
CIRCULAR INTERSECTION APPROACH EXCLUSIVE RIGHT TURN LANE TYPE NAME

Figure 44. Intersection Leg Entity-Relationship Diagram in MIRE Data Model.

## Segment Entity-Relationship Diagram

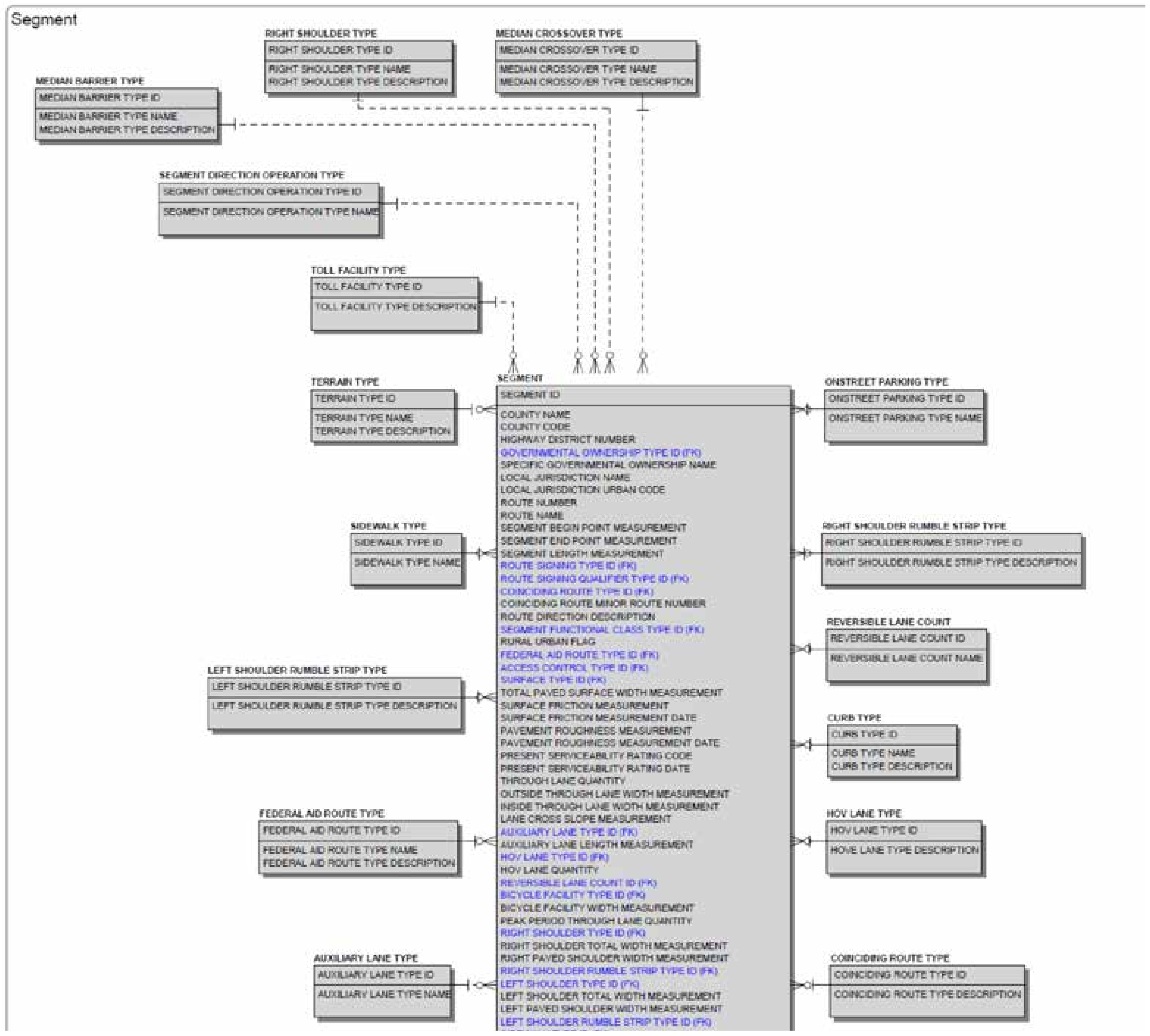

Segment
MEDIAN BARRIER TYPE
MEDIAN BARRIER TYPE ID
MEDIAN BARRIER TYPE NAME
MEDIAN BARRIER TYPE DESCRIPTION
RIGHT SHOULDER TYPE
RIGHT SHOULDER TYPE ID
RIGHT SHOULDER TYPE NAME
RIGHT SHOULDER TYPE DESCRIPTION
MEDIAN CROSSOVER TYPE
MEDIAN CROSSOVER TYPE ID
MEDIAN CROSSOVER TYPE NAME
MEDIAN CROSSOVER TYPE DESCRIPTION
SEGMENT DIRECTION OPERATION TYPE
SEGMENT DIRECTION OPERATION TYPE ID
SEGMENT DIRECTION OPERATION TYPE NAME
TOLL FACILITY TYPE
TOLL FACILITY TYPE ID
TOLL FACILITY TYPE DESCRIPTION
TERRAIN TYPE
TERRAIN TYPE ID
TERRAIN TYPE NAME
TERRAIN TYPE DESCRIPTION
SIDEWALK TYPE
SIDEWALK TYPE ID
SIDEWALK TYPE NAME
LEFT SHOULDER RUMBLE STRIP TYPE
LEFT SHOULDER RUMBLE STRIP TYPE ID
LEFT SHOULDER RUMBLE STRIP TYPE DESCRIPTION
FEDERAL AID ROUTE TYPE
FEDERAL AID ROUTE TYPE ID
FEDERAL AID ROUTE TYPE NAME
FEDERAL AID ROUTE TYPE DESCRIPTION
AUXILIARY LANE TYPE
AUXILIARY LANE TYPE ID
AUXILIARY LANE TYPE NAME
SEGMENT
SEGMENT ID
COUNTY NAME
COUNTY CODE
HIGHWAY DISTRICT NUMBER
GOVERNMENTAL OWNERSHIP TYPE ID (FK)
SPECIFIC GOVERNMENTAL OWNERSHIP NAME
LOCAL JURISDICTION NAME
LOCAL JURISDICTION URBAN CODE
ROUTE NUMBER
ROUTE NAME
SEGMENT BEGIN POINT MEASUREMENT
SEGMENT END POINT MEASUREMENT
SEGMENT LENGTH MEASUREMENT
ROUTE SIGNING TYPE ID (FK)
ROUTE SIGNING QUALIFIER TYPE ID (FK)
COINCIDING ROUTE TYPE ID (FK)
COINCIDING ROUTE MINOR ROUTE NUMBER
ROUTE DIRECTION DESCRIPTION
SEGMENT FUNCTIONAL CLASS TYPE ID (FK)
RURAL URBAN FLAG
FEDERAL AID ROUTE TYPE ID (FK)
ACCESS CONTROL TYPE ID (FK)
SURFACE TYPE ID (FK)
TOTAL PAVED SURFACE WIDTH MEASUREMENT
SURFACE FRICTION MEASUREMENT
SURFACE FRICTION MEASUREMENT DATE
PAVEMENT ROUGHNESS MEASUREMENT
PAVEMENT ROUGHNESS MEASUREMENT DATE
PRESENT SERVICEABILITY RATING CODE
PRESENT SERVICEABILITY RATING DATE
THROUGH LANE QUANTITY
OUTSIDE THROUGH LANE WIDTH MEASUREMENT
INSIDE THROUGH LANE WIDTH MEASUREMENT
LANE CROSS SLOPE MEASUREMENT
AUXILIARY LANE TYPE ID (FK)
AUXILIARY LANE LENGTH MEASUREMENT
HOV LANE TYPE ID (FK)
HOV LANE QUANTITY
REVERSIBLE LANE COUNT ID (FK)
BICYCLE FACILITY TYPE ID (FK)
BICYCLE FACILITY WIDTH MEASUREMENT
PEAK PERIOD THROUGH LANE QUANTITY
RIGHT SHOULDER TYPE ID (FK)
RIGHT SHOULDER TOTAL WIDTH MEASUREMENT
RIGHT PAVED SHOULDER WIDTH MEASUREMENT
RIGHT SHOULDER RUMBLE STRIP TYPE ID (FK)
LEFT SHOULDER TYPE ID (FK)
LEFT SHOULDER TOTAL WIDTH MEASUREMENT
LEFT PAVED SHOULDER WIDTH MEASUREMENT
LEFT SHOULDER RUMBLE STRIP TYPE ID (FK)
ONSTREET PARKING TYPE
ONSTREET PARKING TYPE ID
ONSTREET PARKING TYPE NAME
RIGHT SHOULDER RUMBLE STRIP TYPE
RIGHT SHOULDER RUMBLE STRIP TYPE ID
RIGHT SHOULDER RUMBLE STRIP TYPE DESCRIPTION
REVERSIBLE LANE COUNT
REVERSIBLE LANE COUNT ID
REVERSIBLE LANE COUNT NAME
CURB TYPE
CURB TYPE ID
CURB TYPE NAME
CURB TYPE DESCRIPTION
HOV LANE TYPE
HOV LANE TYPE ID
HOVE LANE TYPE DESCRIPTION
COINCIDING ROUTE TYPE
COINCIDING ROUTE TYPE ID
COINCIDING ROUTE TYPE DESCRIPTION

Figure 45. Segment Entity-Relationship Diagram in MIRE Data Model.

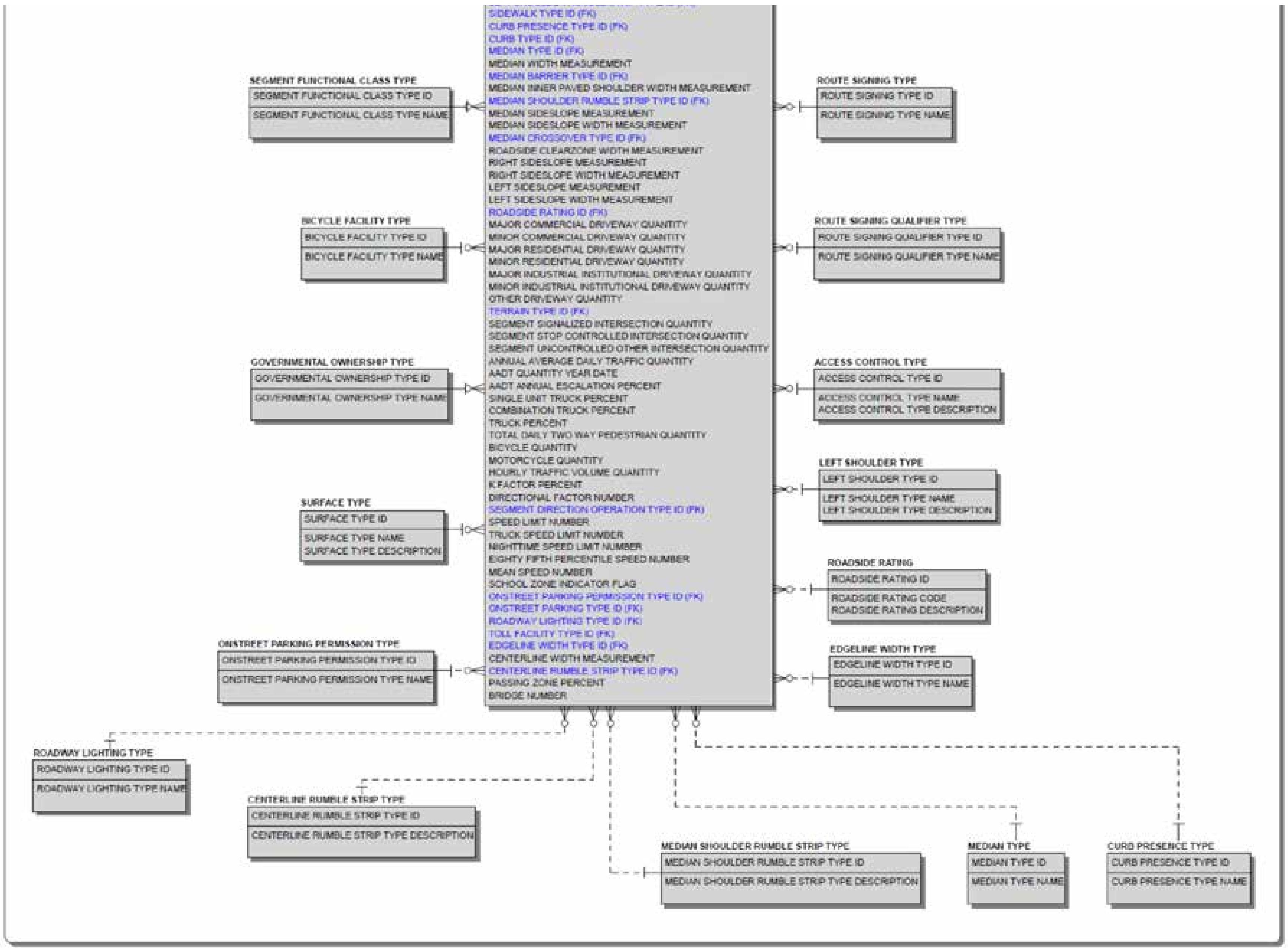

SIDEWALK TYPE ID (FK)
CURB PRESENCE TYPE ID (FK)
CURB TYPE ID (FK)
MEDIAN TYPE ID (FK)
MEDIAN WIDTH MEASUREMENT
MEDIAN BARRIER TYPE ID (FK)
MEDIAN INNER PAVED SHOULDER WIDTH MEASUREMENT
MEDIAN SHOULDER RUMBLE STRIP TYPE ID (FK)
MEDIAN SIDESLOPE MEASUREMENT
MEDIAN SIDESLOPE WIDTH MEASUREMENT
MEDIAN CROSSOVER TYPE ID (FK)
ROADSIDE CLEARZONE WIDTH MEASUREMENT
RIGHT SIDESLOPE MEASUREMENT
RIGHT SIDESLOPE WIDTH MEASUREMENT
LEFT SIDESLOPE MEASUREMENT
LEFT SIDESLOPE WIDTH MEASUREMENT
ROADSIDE RATING ID (FK)
MAJOR COMMERCIAL DRIVEWAY QUANTITY
MINOR COMMERCIAL DRIVEWAY QUANTITY
MAJOR RESIDENTIAL DRIVEWAY QUANTITY
MINOR RESIDENTIAL DRIVEWAY QUANTITY
MAJOR INDUSTRIAL INSTITUTIONAL DRIVEWAY QUANTITY
MINOR INDUSTRIAL INSTITUTIONAL DRIVEWAY QUANTITY
OTHER DRIVEWAY QUANTITY
TERRAIN TYPE ID (FK)
SEGMENT SIGNALIZED INTERSECTION QUANTITY
SEGMENT STOP CONTROLLED INTERSECTION QUANTITY
SEGMENT UNCONTROLLED OTHER INTERSECTION QUANTITY
ANNUAL AVERAGE DAILY TRAFFIC QUANTITY
AADT QUANTITY YEAR DATE
AADT ANNUAL ESCALATION PERCENT
SINGLE UNIT TRUCK PERCENT
COMBINATION TRUCK PERCENT
TRUCK PERCENT
TOTAL DAILY TWO WAY PEDESTRIAN QUANTITY
BICYCLE QUANTITY
MOTORCYCLE QUANTITY
HOURLY TRAFFIC VOLUME QUANTITY
K FACTOR PERCENT
DIRECTIONAL FACTOR NUMBER
SEGMENT DIRECTION OPERATION TYPE ID (FK)
SPEED LIMIT NUMBER
TRUCK SPEED LIMIT NUMBER
NIGHTTIME SPEED LIMIT NUMBER
EIGHTY FIFTH PERCENTILE SPEED NUMBER
MEAN SPEED NUMBER
SCHOOL ZONE INDICATOR FLAG
ONSTREET PARKING PERMISSION TYPE ID (FK)
ONSTREET PARKING TYPE ID (FK)
ROADWAY LIGHTING TYPE ID (FK)
TOLL FACILITY TYPE ID (FK)
EDGELINE WIDTH TYPE ID (FK)
CENTERLINE WIDTH MEASUREMENT
CENTERLINE RUMBLE STRIP TYPE ID (FK)
PASSING ZONE PERCENT
BRIDGE NUMBER
SEGMENT FUNCTIONAL CLASS TYPE
SEGMENT FUNCTIONAL CLASS TYPE ID
SEGMENT FUNCTIONAL CLASS TYPE NAME
BICYCLE FACILITY TYPE
BICYCLE FACILITY TYPE ID
BICYCLE FACILITY TYPE NAME
GOVERNMENTAL OWNERSHIP TYPE
GOVERNMENTAL OWNERSHIP TYPE ID
GOVERNMENTAL OWNERSHIP TYPE NAME
SURFACE TYPE
SURFACE TYPE ID
SURFACE TYPE NAME
SURFACE TYPE DESCRIPTION
ONSTREET PARKING PERMISSION TYPE
ONSTREET PARKING PERMISSION TYPE ID
ONSTREET PARKING PERMISSION TYPE NAME
ROUTE SIGNING TYPE
ROUTE SIGNING TYPE ID
ROUTE SIGNING TYPE NAME
ROUTE SIGNING QUALIFIER TYPE
ROUTE SIGNING QUALIFIER TYPE ID
ROUTE SIGNING QUALIFIER TYPE NAME
ACCESS CONTROL TYPE
ACCESS CONTROL TYPE ID
ACCESS CONTROL TYPE NAME
ACCESS CONTROL TYPE DESCRIPTION
LEFT SHOULDER TYPE
LEFT SHOULDER TYPE ID
LEFT SHOULDER TYPE NAME
LEFT SHOULDER TYPE DESCRIPTION
ROADSIDE RATING
ROADSIDE RATING ID
ROADSIDE RATING CODE
ROADSIDE RATING DESCRIPTION
EDGELINE WIDTH TYPE
EDGELINE WIDTH TYPE ID
EDGELINE WIDTH TYPE NAME
ROADWAY LIGHTING TYPE
ROADWAY LIGHTING TYPE ID
ROADWAY LIGHTING TYPE NAME
CENTERLINE RUMBLE STRIP TYPE
CENTERLINE RUMBLE STRIP TYPE ID
CENTERLINE RUMBLE STRIP TYPE DESCRIPTION
MEDIAN SHOULDER RUMBLE STRIP TYPE
MEDIAN SHOULDER RUMBLE STRIP TYPE ID
MEDIAN SHOULDER RUMBLE STRIP TYPE DESCRIPTION
MEDIAN TYPE
MEDIAN TYPE ID
MEDIAN TYPE NAME
CURB PRESENCE TYPE
CURB PRESENCE TYPE ID
CURB PRESENCE TYPE NAME

Figure 45 (Continued). Segment Entity-Relationship Diagram in MIRE Data Model.

## Ramp Entity-Relationship Diagram

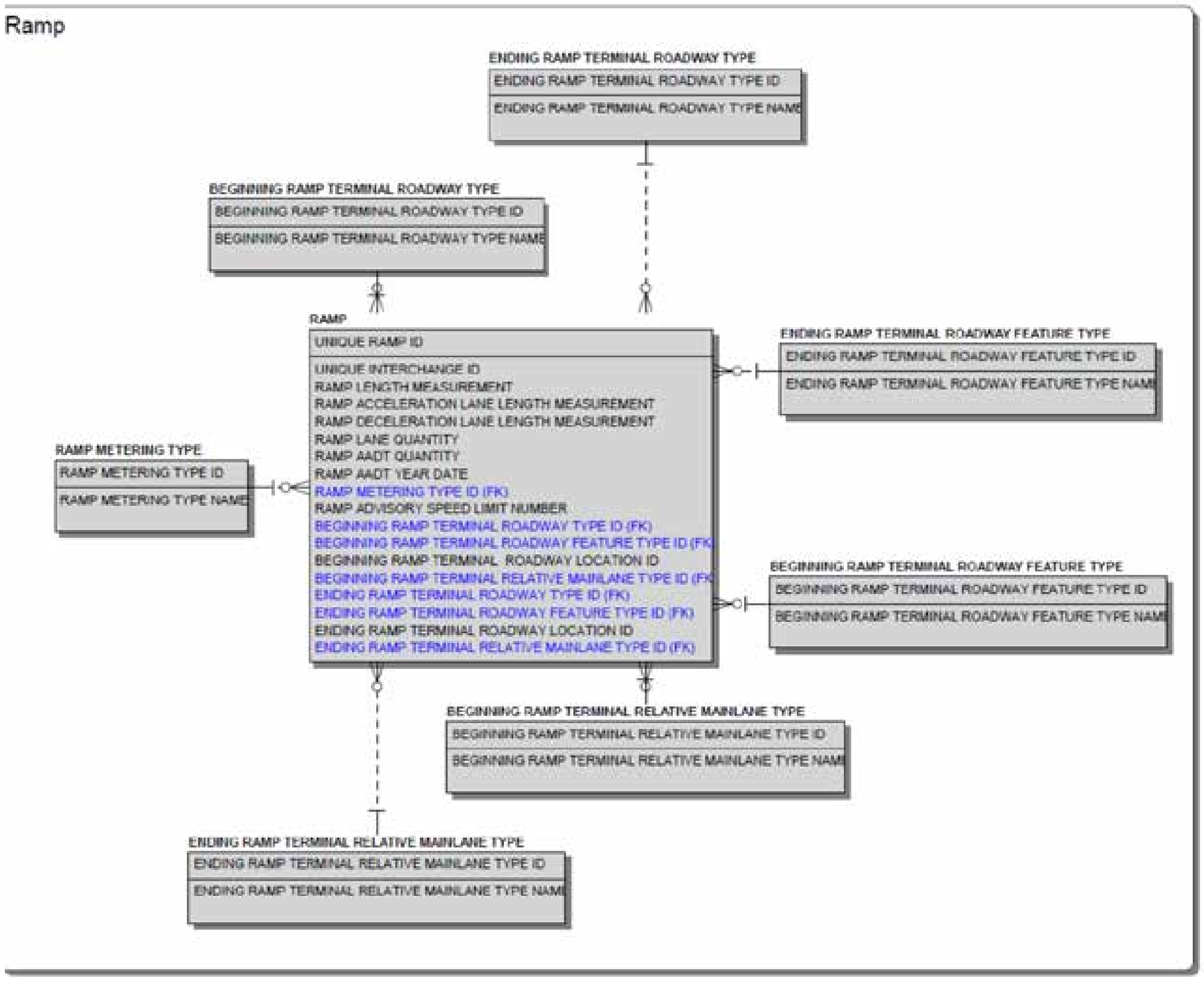

Ramp
ENDING RAMP TERMINAL ROADWAY TYPE
ENDING RAMP TERMINAL ROADWAY TYPE ID
ENDING RAMP TERMINAL ROADWAY TYPE NAME
BEGINNING RAMP TERMINAL ROADWAY TYPE
BEGINNING RAMP TERMINAL ROADWAY TYPE ID
BEGINNING RAMP TERMINAL ROADWAY TYPE NAME
RAMP
UNIQUE RAMP ID
UNIQUE INTERCHANGE ID
RAMP LENGTH MEASUREMENT
RAMP ACCELERATION LANE LENGTH MEASUREMENT
RAMP DECELERATION LANE LENGTH MEASUREMENT
RAMP LANE QUANTITY
RAMP AADT QUANTITY
RAMP AADT YEAR DATE
RAMP METERING TYPE ID (FK)
RAMP ADVISORY SPEED LIMIT NUMBER
BEGINNING RAMP TERMINAL ROADWAY TYPE ID (FK)
BEGINNING RAMP TERMINAL ROADWAY FEATURE TYPE ID (FK)
BEGINNING RAMP TERMINAL ROADWAY LOCATION ID
BEGINNING RAMP TERMINAL RELATIVE MAINLANE TYPE ID (FK)
ENDING RAMP TERMINAL ROADWAY TYPE ID (FK)
ENDING RAMP TERMINAL ROADWAY FEATURE TYPE ID (FK)
ENDING RAMP TERMINAL ROADWAY LOCATION ID
ENDING RAMP TERMINAL RELATIVE MAINLANE TYPE ID (FK)
ENDING RAMP TERMINAL ROADWAY FEATURE TYPE
ENDING RAMP TERMINAL ROADWAY FEATURE TYPE ID
ENDING RAMP TERMINAL ROADWAY FEATURE TYPE NAME
RAMP METERING TYPE
RAMP METERING TYPE ID
RAMP METERING TYPE NAME
BEGINNING RAMP TERMINAL ROADWAY FEATURE TYPE
BEGINNING RAMP TERMINAL ROADWAY FEATURE TYPE ID
BEGINNING RAMP TERMINAL ROADWAY FEATURE TYPE NAME
BEGINNING RAMP TERMINAL RELATIVE MAINLANE TYPE
BEGINNING RAMP TERMINAL RELATIVE MAINLANE TYPE ID
BEGINNING RAMP TERMINAL RELATIVE MAINLANE TYPE NAME
ENDING RAMP TERMINAL RELATIVE MAINLANE TYPE
ENDING RAMP TERMINAL RELATIVE MAINLANE TYPE ID
ENDING RAMP TERMINAL RELATIVE MAINLANE TYPE NAME

**Figure 46. Ramp Entity-Relationship Diagram in MIRE Data Model.**

## Interchange Entity-Relationship Diagram

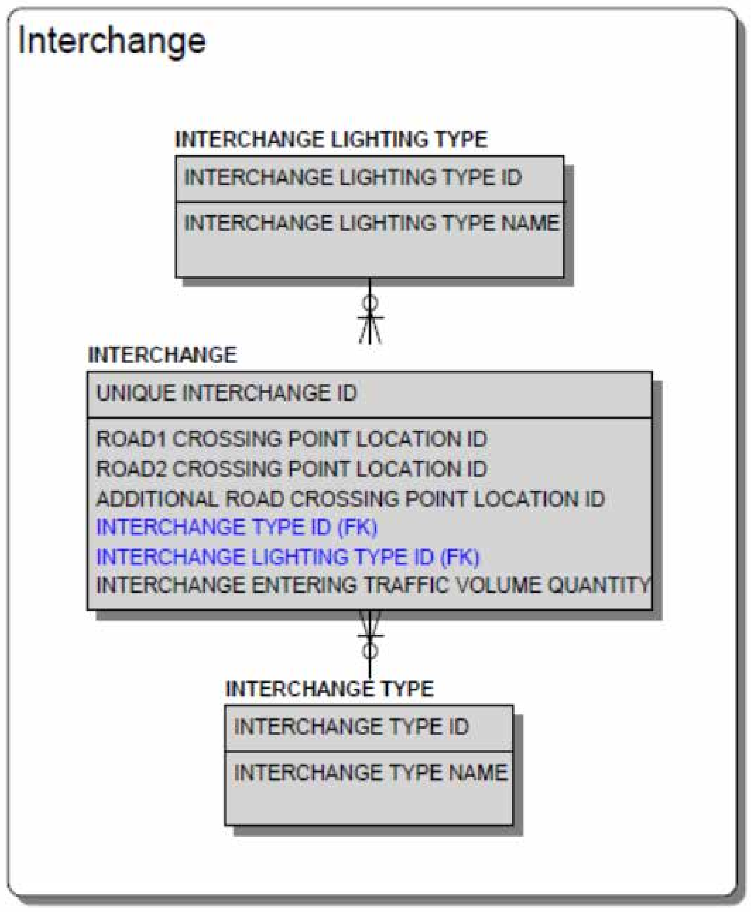

Interchange
INTERCHANGE LIGHTING TYPE
INTERCHANGE LIGHTING TYPE ID
INTERCHANGE LIGHTING TYPE NAME
INTERCHANGE
UNIQUE INTERCHANGE ID
ROAD1 CROSSING POINT LOCATION ID
ROAD2 CROSSING POINT LOCATION ID
ADDITIONAL ROAD CROSSING POINT LOCATION ID
INTERCHANGE TYPE ID (FK)
INTERCHANGE LIGHTING TYPE ID (FK)
INTERCHANGE ENTERING TRAFFIC VOLUME QUANTITY
INTERCHANGE TYPE
INTERCHANGE TYPE ID
INTERCHANGE TYPE NAME

**Figure 47. Interchange Entity-Relationship Diagram in MIRE Data Model.**

## Horizontal Curve Entity-Relationship Diagram

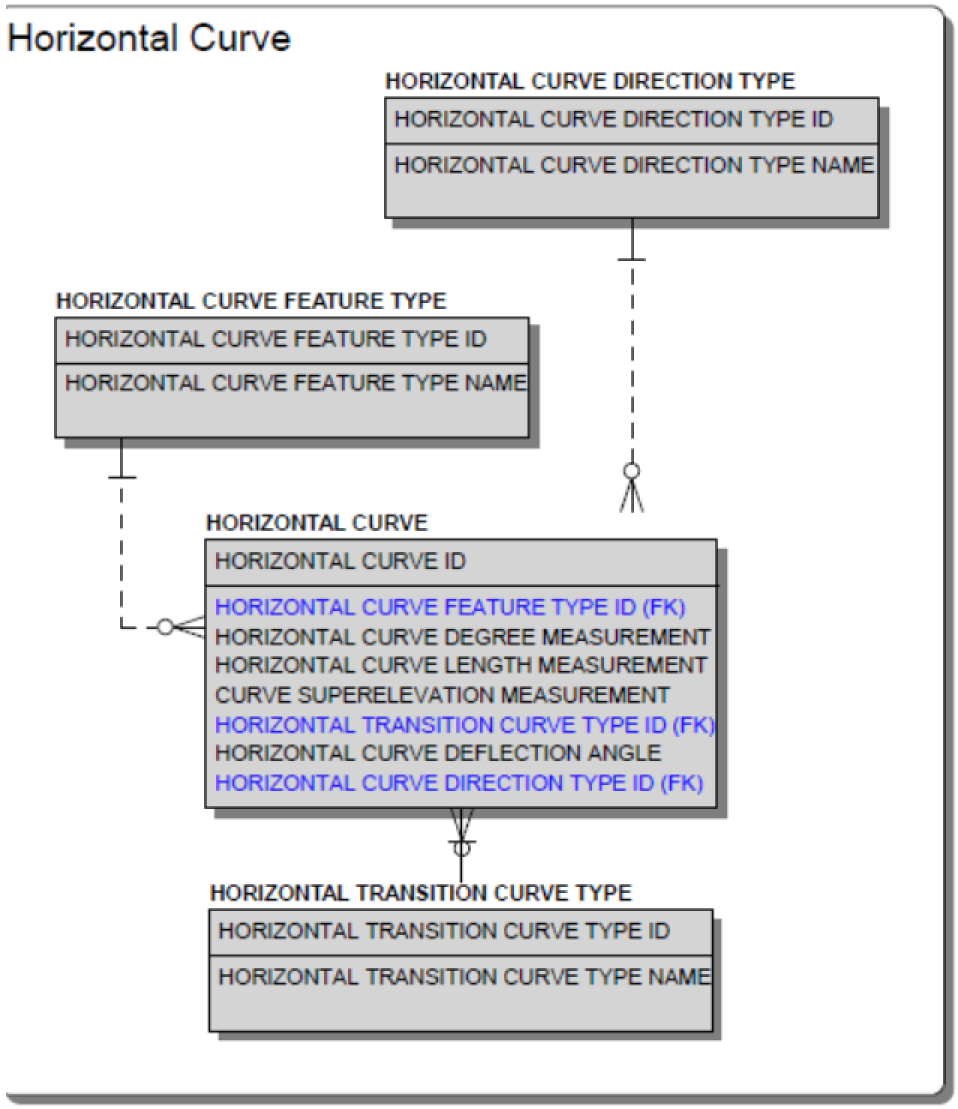

Horizontal Curve
HORIZONTAL CURVE DIRECTION TYPE
HORIZONTAL CURVE DIRECTION TYPE ID
HORIZONTAL CURVE DIRECTION TYPE NAME
HORIZONTAL CURVE FEATURE TYPE
HORIZONTAL CURVE FEATURE TYPE ID
HORIZONTAL CURVE FEATURE TYPE NAME
HORIZONTAL CURVE
HORIZONTAL CURVE ID
HORIZONTAL CURVE FEATURE TYPE ID (FK)
HORIZONTAL CURVE DEGREE MEASUREMENT
HORIZONTAL CURVE LENGTH MEASUREMENT
CURVE SUPERELEVATION MEASUREMENT
HORIZONTAL TRANSITION CURVE TYPE ID (FK)
HORIZONTAL CURVE DEFLECTION ANGLE
HORIZONTAL CURVE DIRECTION TYPE ID (FK)
HORIZONTAL TRANSITION CURVE TYPE
HORIZONTAL TRANSITION CURVE TYPE ID
HORIZONTAL TRANSITION CURVE TYPE NAME

**Figure 48. Horizontal Curve Entity-Relationship Diagram in MIRE Data Model.**

## Vertical Grade Entity-Relationship Diagram

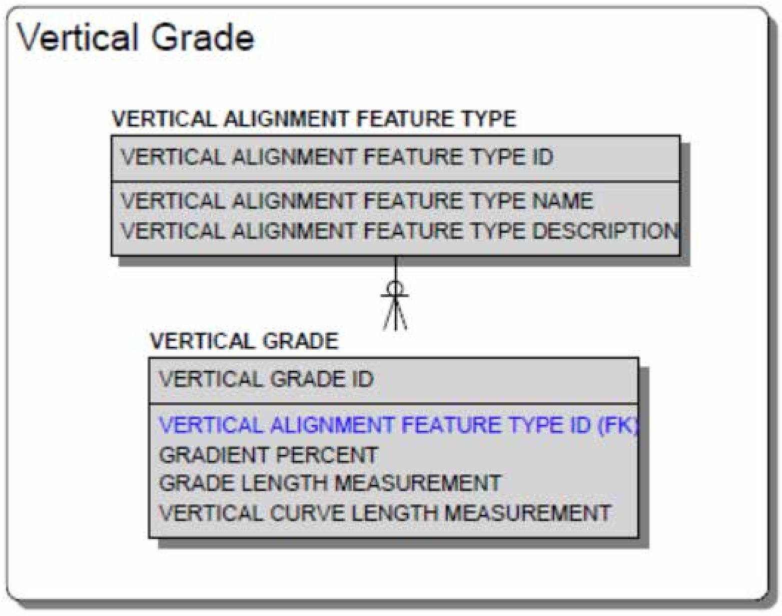

Vertical Grade
VERTICAL ALIGNMENT FEATURE TYPE
VERTICAL ALIGNMENT FEATURE TYPE ID
VERTICAL ALIGNMENT FEATURE TYPE NAME
VERTICAL ALIGNMENT FEATURE TYPE DESCRIPTION
VERTICAL GRADE
VERTICAL GRADE ID
VERTICAL ALIGNMENT FEATURE TYPE ID (FK)
GRADIENT PERCENT
GRADE LENGTH MEASUREMENT
VERTICAL CURVE LENGTH MEASUREMENT

**Figure 49. Vertical Grade Entity-Relationship Diagram in MIRE Data Model.**

# Appendix II. Safety Analyst Data Model Entity-Relationship Diagrams

The output of the RDE tool is intended to be used with safety analysis tools such as the Highway Safety Manual (HSM), the Interactive Highway Safety Design Model, and AASHTOWare Safety Analyst. The MIRE-MIS pilot implementation focused on Safety Analyst as a potential tool for analyzing data produced by the RDE tool because it was requested by the NHDOT. Two of the pilot states in the RDETAP project are also exploring potential uses of the RDE Tool data output for safety analysis programs.

Safety Analyst uses spatially referenced data and is data intensive. The RDE tool is able to support the specific and extensive data requirements of Safety Analyst. However, as mentioned above, the RDE tool was designed to supply data for any common safety analysis tool. Accordingly, a transportation agency using a less data intensive safety analysis tool would simply disregard (or delete) specific attribution required by Safety Analyst.

Safety Analyst defines attribution for the main MIRE data entities, specifically intersections, intersection legs, segments, and ramps. This appendix shows the complete model and related look-up tables.

To make the RDE tool fully compatible with the requirements of states implementing Safety Analyst, the researchers reviewed domain values for all elements in both MIRE and Safety Analyst data models and developed compatible geodatabase templates.

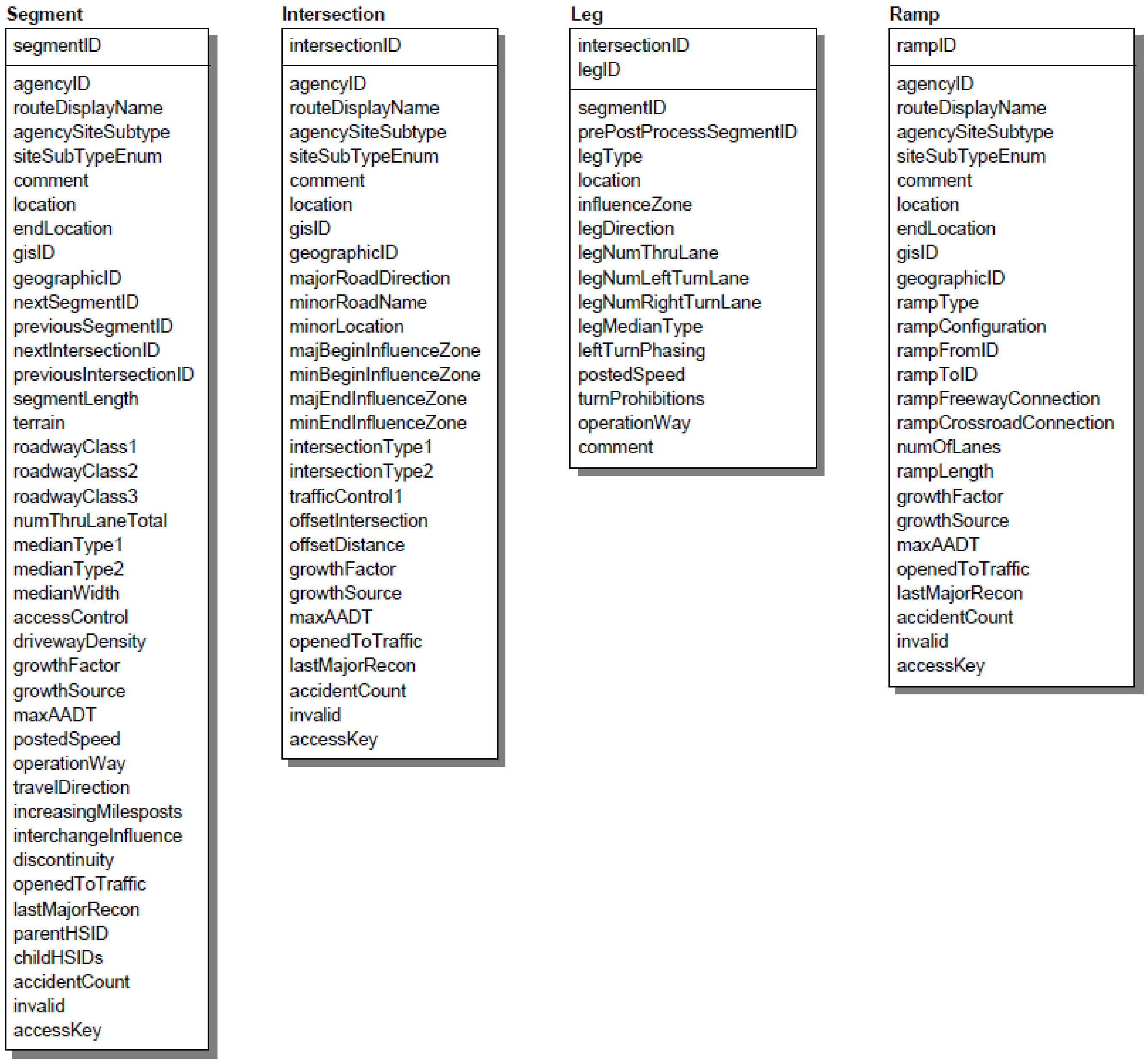


**Figure 50. Main Entities in Safety Analyst Data Model.**

## Intersection Entity-Relationship Diagram

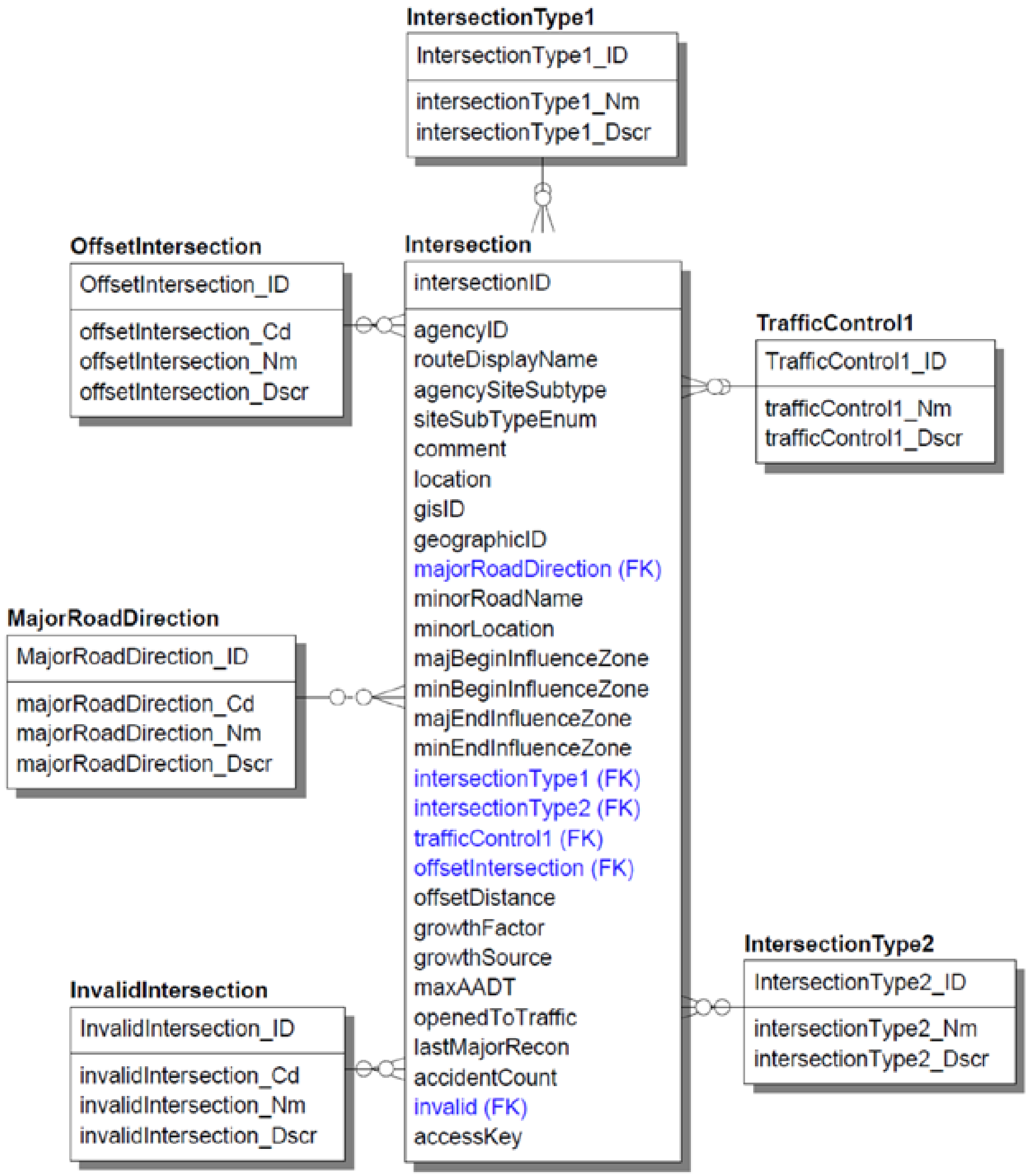


**Figure 51. Intersection Entity-Relationship Diagram in Safety Analyst Data Model.**

## Intersection Leg Entity-Relationship Diagram

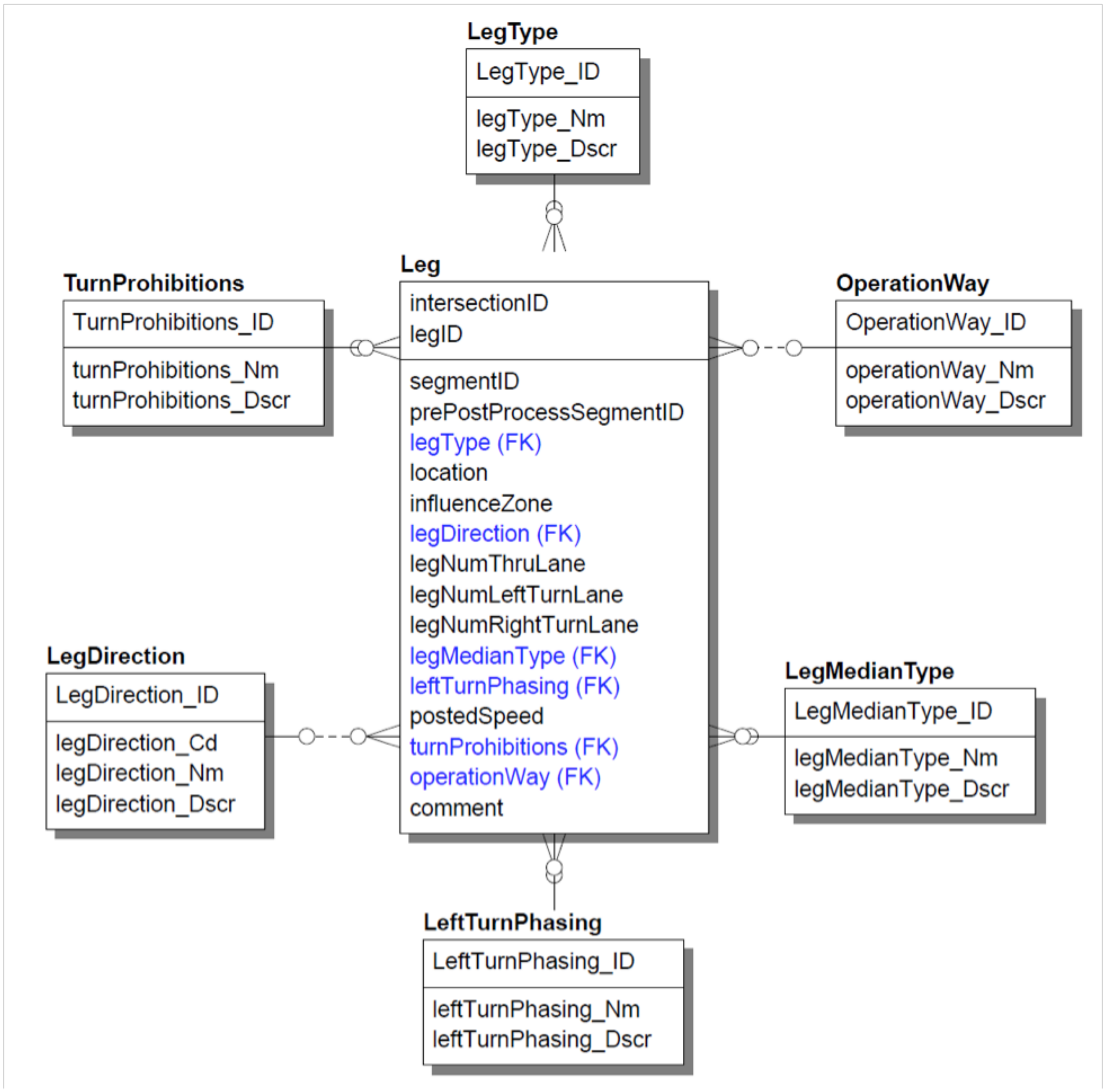


**Figure 52. Intersection Entity-Relationship Diagram in Safety Analyst Data Model.**

## Segment Entity-Relationship Diagram

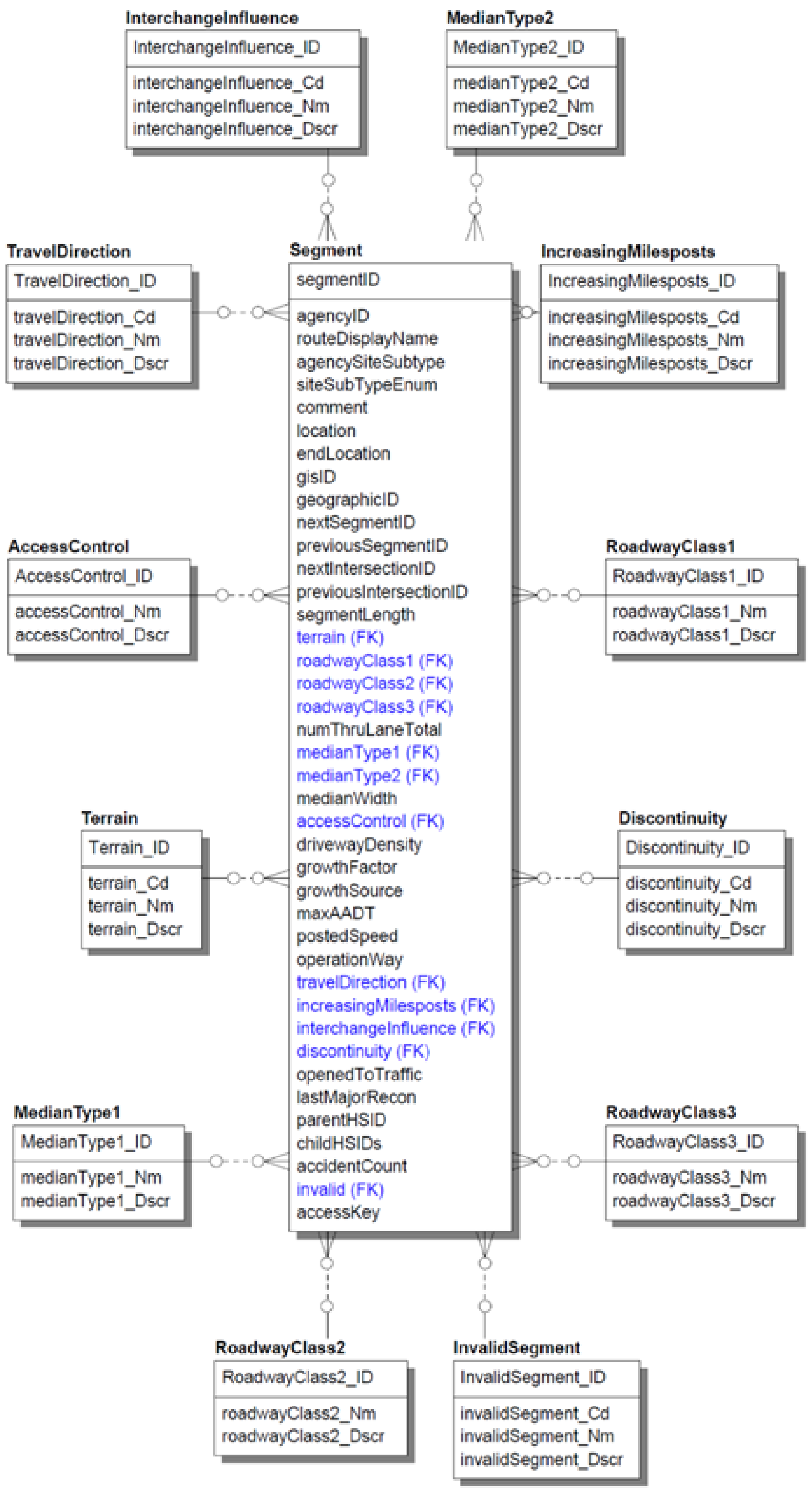


**Figure 53. Segment Entity-Relationship Diagram in Safety Analyst Data Model.**

## Ramp Entity-Relationship Diagram

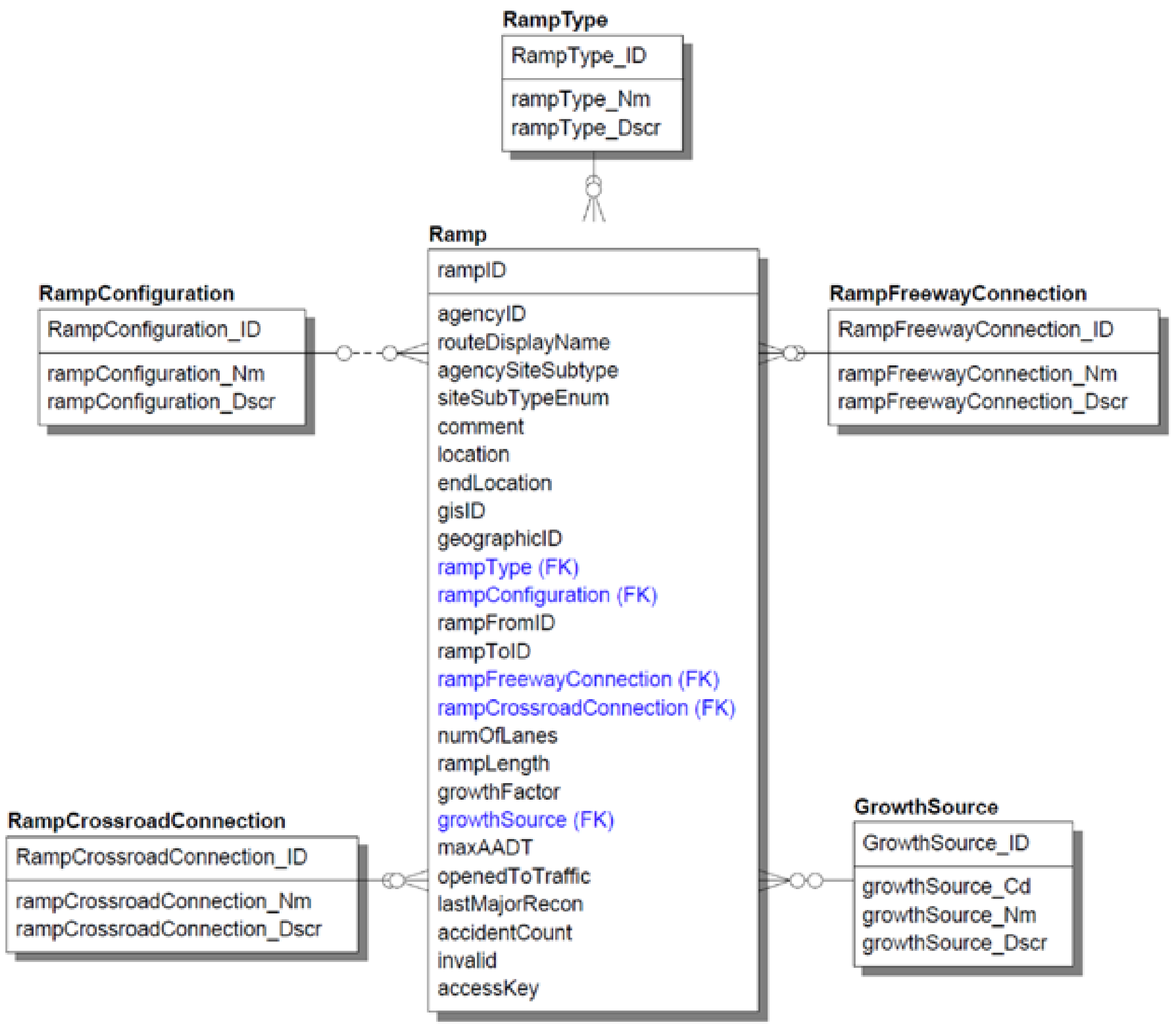


**Figure 54. Ramp Entity-Relationship Diagram in Safety Analyst Data Model.**

# Appendix III. Feature Class Templates for Output Geodatabase

## Generic Asset Node Template

**Table 16. Generic Asset Node Template.**

| Field Name | Data Type |
|---|---|
| OBJECTID | Object ID |
| SHAPE | Geometry |
| agencyID | Text |
| POINT_X | Double |
| POINT_Y | Double |
| NodeID | Long Integer |
| Count of Records | 30,368 |

## Generic Intersection Template

**Table 17. Generic Intersection Template.**

| Field Name | Data Type |
|---|---|
| agencyID | Text |
| intersectionID | Long Integer |
| routeDisplayName | Text |
| agencySiteSubtype | Text |
| siteSubTypeEnum | Text |
| comment | Text |
| location | Text |
| gisID | Text |
| geographicID | Text |
| majorRoadDirection | Text |
| minorRoadName | Text |
| minorLocation | Text |
| majBeginInfluenceZone | Double |
| minBeginInfluenceZone | Double |
| majEndInfluenceZone | Double |
| minEndInfluenceZone | Double |
| intersectionType1 | Text |
| intersectionType2 | Text |
| trafficControl1 | Text |
| offsetIntersection | Text |
| offsetDistance | Double |

**Table 17. Generic Intersection Template (continued).**

| Field Name | Data Type |
|---|---|
| growthFactor | Double |
| growthSource | Text |
| maxAADT | Double |
| openedToTraffic | Date |
| lastMajorRecon | Date |
| accidentCount | Long Integer |
| Invalid | Text |
| accessKey | Text |
| UNIQ_JNCT_ID | Long Integer |
| INTSECT_TYPE | Text |
| ROAD1_CRSNG_PNT_LOCN_ID | Long Integer |
| ROAD2_CRSNG_PNT_LOCN_ID | Long Integer |
| ADDL_RD_CRSNG_PNT_LOCN_ID | Long Integer |
| INTSECT_LEG_QTY | Short Integer |
| INTSECT_GMTRY_TYPE | Text |
| SCHL_ZN_INDCTR_FLG | Text |
| RR_CRSNG_NBR | Short Integer |
| INTSECT_ANG_MS | Float |
| INTSECT_OFFST_DSTNCE_MS | Double |
| INTSECT_TRFC_CTRL_TYPE | Text |
| SGNLN_TYPE | Text |
| INTSECT_LTG_FLG | Text |
| CIRC_INTSECT_LN_QTY | Short Integer |
| CIRC_INTSECT_LN_WIDTH_MS | Double |
| CIRC_INTSECT_INSCR_DIA_MS | Double |
| CIRC_INTS_BICY_FCLTY_TYPE | Text |
| Status | Text |
| Verified | Text |

## Generic IntersectionLeg Template

**Table 18 Generic Intersection Leg Template.**

| Field Name | Data Type |
|---|---|
| intersectionID | Long Integer |
| legID | Long Integer |
| legType | Text |
| location | Text |
| influenceZone | Float |
| legDirection | Text |
| legNumThruLane | Short Integer |
| legNumLeftTurnLane | Short Integer |
| legNumRightTurnLane | Short Integer |
| legMedianType | Text |
| leftTurnPhasing | Text |
| postedSpeed | Short Integer |
| turnProhibitions | Text |
| operationWay | Text |
| comment | Text |
| INTSECT_ID | Text |
| UNIQ_APRCH_ID | Long Integer |
| APRCH_AADT_QTY | Long Integer |
| APRCH_AADT_YR_NBR | Long Integer |
| APRCH_MODE | Text |
| APRCH_DRCT_FLOW_TYPE | Text |
| APRCH_THRGH_LN_QTY | Short Integer |
| APRCH_LT_TURN_LN_TYPE | Text |
| APRCH_EXCLV_LT_TURN_LN_QTY | Short Integer |
| APRCH_LT_TURN_LN_OFFST_MS | Double |
| APRCH_RT_TURN_CHNLZ_TYPE | Text |
| EXC_RT_TRN_LN_TRF_CTR_TYPE | Text |
| EXCLV_RT_TURN_LN_QTY | Short Integer |
| EXCLV_LT_TURN_LN_LNGTH_MS | Double |
| EXCLV_RT_TURN_LN_LNGTH_MS | Double |
| APRCH_MDN_TYPE | Text |

**Table 18 Generic Intersection Leg Template (continued).**

| Field Name | Data Type |
|---|---|
| APRCH_TRFC_CTRL_TYPE | Text |
| APRCH_LT_TURN_PROT_TYPE | Text |
| SGNL_PROG_TYPE | Text |
| APRCH_CRSWLK_TYPE | Text |
| APRCH_PED_SGNLN_TYPE | Text |
| APR_PED_SGN_SPCL_FEAT_TYPE | Text |
| APRCH_CRSNG_PED_QTY | Short Integer |
| APRCH_LT_RT_TURN_PRHB_TYPE | Text |
| APR_RT_TRN_ON_RED_PRH_TYPE | Text |
| APRCH_LT_TURN_QTY | Short Integer |
| APRCH_LT_TURN_CNT_YR_NBR | Date |
| APRCH_RT_TURN_QTY | Short Integer |
| APRCH_RT_TURN_CNT_YR_NBR | Date |
| APRCH_TRNSVRS_RMBL_STRP_FLG | Text |
| CIRC_INTS_APRCH_ENTRY_WIDTH | Double |
| CIRC_INTS_APRCH_ENTRY_LN_QTY | Short Integer |
| CIR_IN_AP_EX_RT_TN_LN_TYPE | Text |
| CIRC_INTS_APRCH_ENTRY_RAD_MS | Double |
| CIRC_INTS_APRCH_EXIT_WIDTH_MS | Double |
| CIRC_INTS_APRCH_EXIT_LN_QTY | Short Integer |
| CIRC_INTS_EXIT_RAD_MS | Double |
| CIRC_INTS_PED_FCLTY_TYPE | Text |
| CIRC_INTS_APR_CRSWLK_LOCN_MS | Double |
| CIRC_INTS_APRCH_ISLND_WIDTH_MS | Double |
| APRCH_SKEW_ANG_MS | Float |
| RURL_URB_DSGNT | Text |
| RDWAY_OWNR | Text |
| Verified | Text |

## Generic Ramp Template

**Table 19. Generic Ramp Template.**

| Field Name | Data Type |
|---|---|
| agencyID | Text |
| rampID | Text |
| routeDisplayName | Text |
| agencySiteSubtype | Text |
| siteSubTypeEnum | Text |
| comment | Text |
| location | Text |
| endLocation | Text |
| gisID | Text |
| geographicID | Text |
| rampType | Text |
| rampConfiguration | Text |
| rampFromID | Text |
| rampToID | Text |
| rampFreewayConnection | Text |
| rampCrossroadConnection | Text |
| numOfLanes | Short Integer |
| rampLength | Double |
| growthFactor | Double |
| growthSource | Text |
| maxAADT | Long Integer |
| openedToTraffic | Date |
| lastMajorRecon | Date |
| accidentCount | Long Integer |
| invalid | Text |
| UNIQ_RAMP_ID | Text |
| UNIQ_INTCHG_ID | Text |
| RAMP_LNGTH_MS | Double |
| RAMP_ACCEL_LN_LNGTH_MS | Double |
| RAMP_DECEL_LN_LNGTH_MS | Double |
| RAMP_LN_QTY | Short Integer |
| RAMP_AADT_QTY | Long Integer |

Table 19. Generic Ramp Template (continued).

| Field Name | Data Type |
|---|---|
| RAMP_AADT_YR_NBR | Long Integer |
| RAMP_METER_TYPE | Text |
| RAMP_ADVRY_SPD_LMT_NBR | Short Integer |
| BEGN_RAMP_TRMN_RDWY_TYPE | Text |
| BEGN_RAMP_TRMN_RDWY_FEAT_TYPE | Text |
| BEGN_RAMP_TRMN_RDWY_LOCN_ID | Text |
| BEGN_RAMP_TRMN_RLTV_MNLN_TYPE | Text |
| END_RAMP_TRMN_RDWY_TYPE | Text |
| END_RAMP_TRMN_RDWY_FEAT_TYPE | Text |
| END_RAMP_TRMN_RDWY_LOCN_ID | Text |
| END_RAMP_TRMN_RLTV_MNLN_TYPE | Text |
| GOVTL_OWNR_TYPE | Text |
| FUNC_CLASS_TYPE | Text |
| Verified | Text |

## Generic Roadway Segment Template

**Table 20. Generic Segment Template.**

| Field Name | Data Type |
|---|---|
| agencyID | Text |
| segmentID | Text |
| routeDisplayName | Text |
| agencySiteSubtype | Text |
| siteSubTypeEnum | Text |
| comment | Text |
| location | Text |
| endLocation | Text |
| gisID | Text |
| geographicID | Text |
| nextSegmentID | Text |
| previousSegmentID | Text |
| nextIntersectionID | Text |
| previousIntersectionID | Text |
| segmentLength | Double |
| terrain | Text |
| roadwayClass1 | Text |
| roadwayClass2 | Text |
| roadwayClass3 | Text |
| numThruLaneTotal | Short Integer |
| medianType1 | Text |
| medianType2 | Text |
| medianWidth | Double |
| accessControl | Text |
| drivewayDensity | Double |
| growthFactor | Double |
| growthSource | Text |
| maxAADT | Double |
| postedSpeed | Double |
| operationWay | Text |
| travelDirection | Text |
| increasingMilesposts | Text |
| interchangeInfluence | Text |
| discontinuity | Text |
| openedToTraffic | Date |

**Table 20. Generic Segment Template (continued).**

| Field Name | Data Type |
|---|---|
| lastMajorRecon | Date |
| parentHSID | Text |
| childHSIDs | Text |
| accidentCount | Long Integer |
| invalid | Text |
| accessKey | Text |
| CNTY_NM | Text |
| CNTY_CD | Text |
| HWY_DIST_NBR | Text |
| GOVTL_OWNR_TYPE | Text |
| SPECF_GOVTL_OWNR_NM | Text |
| LCL_JURIS_NM | Text |
| LCL_JURIS_URB_CD | Text |
| RTE_NBR | Text |
| RTE_NM | Text |
| SGMNT_BEG_PNT_MS | Double |
| SGMNT_END_PNT_MS | Double |
| RTE_SIGN_TYPE | Text |
| RTE_SIGN_QLFY_TYPE | Text |
| COCDN_RTE_TYPE | Text |
| COCDN_RTE_MINR_RTE_NBR | Text |
| DRCT_OF_INV | Text |
| RURL_URB_DSGNT | Text |
| FEDRL_AID_RTE_TYPE | Text |
| SRFC_TYPE | Text |
| TOTL_PVD_SRFC_WIDTH_MS | Double |
| SRFC_FRCT_MS | Double |
| SRFC_FRCT_MS_DT | Date |
| PVMT_RGHNS_MS | Double |
| PVMT_RGHNS_MS_DT | Date |
| PRSNT_SRCVB_RTNG_CD | Text |
| PRSNT_SRCVB_RTNG_DT | Date |
| THRGH_LN_QTY | Short Integer |
| OUTSD_THRGH_LN_WIDTH_MS | Double |
| INSD_THRGH_LN_WIDTH_MS | Double |

**Table 20. Generic Segment Template (continued).**

| Field Name | Data Type |
|---|---|
| LN_CROSS_SLP_MS | Double |
| AUX_LN_TYPE | Text |
| AUX_LN_LNGTH_MS | Double |
| HOV_LN_TYPE | Text |
| HOV_LN_QTY | Short Integer |
| CROSS_LN_QTY | Short Integer |
| BICY_FCLTY_TYPE | Text |
| BICY_FCLTY_WIDTH_MS | Double |
| PEAK_PER_THRGH_LN_QTY | Short Integer |
| RT_SHLDR_TYPE | Text |
| RT_SHLDR_TOTL_WIDTH_MS | Double |
| RT_PVD_SHLDR_WIDTH_MS | Double |
| RT_SHLDR_RMBL_STRP_TYPE | Text |
| LT_SHLDR_TYPE | Text |
| LT_SHLDR_TOTL_WIDTH_MS | Double |
| LT_PVD_SHLDR_WIDTH_MS | Double |
| LT_SHLDR_RMBL_STRP_TYPE | Text |
| SDWALK_TYPE | Text |
| CURB_PRSC_TYPE | Text |
| CURB_TYPE | Text |
| MDN_BARR_TYPE | Text |
| MDN_INNER_PVD_SHLDR_WIDTH_MS | Double |
| MDN_SHLDR_RMBL_STRP_TYPE | Text |
| MDN_SDSLP_MS | Double |
| MDN_SDSLP_WIDTH_MS | Text |
| MDN_CRSOVR_LN_TYPE | Text |
| RDSD_CLEARZN_WIDTH_MS | Double |
| RT_SDSLP_MS | Double |
| RT_SDSLP_WIDTH_MS | Double |
| LT_SDSLP_MS | Double |
| LT_SDSLP_WIDTH_MS | Double |
| RDSD_RTNG_CD | Text |
| MAJR_COML_DRWY_QTY | Short Integer |
| MINR_COML_DRWY_QTY | Short Integer |
| MAJR_RESD_DRWY_QTY | Short Integer |

**Table 20. Generic Segment Template (continued).**

| Field Name | Data Type |
|---|---|
| MINR_RESD_DRWY_QTY | Short Integer |
| MAJR_IND_INSTNL_DRWY_QTY | Short Integer |
| MINR_IND_INSTNL_DRWY_QTY | Short Integer |
| OTHR_DRWY_QTY | Short Integer |
| SGMNT_SGNLZ_INTSECT_QTY | Short Integer |
| SGMNT_STP_CTRLD_INTSECT_QTY | Short Integer |
| SGMNT_UNCTRL_OTHR_INTSECT_QTY | Short Integer |
| AADT_QTY_YR_NBR | Long Integer |
| AADT_ANNL_ESCAL_PCT | Short Integer |
| SUT_PCT | Short Integer |
| COMBN_TRCK_PCT | Short Integer |
| TRCK_PCT | Short Integer |
| TOTL_DAILY_TWO_WAY_PED_QTY | Long Integer |
| BICY_QTY | Long Integer |
| MTRCYCL_QTY | Long Integer |
| HRLY_TRFC_VOL_QTY | Long Integer |
| K_FCTR_PCT | Short Integer |
| DRCTL_FCTR_NBR | Long Integer |
| TRCK_SPD_LMT_NBR | Short Integer |
| NGTME_SPD_LMT_NBR | Short Integer |
| EIGHTY_FIFTH_PCTL_SPD_NBR | Short Integer |
| MEAN_SPD_NBR | Short Integer |
| SCHL_ZN_INDCTR_FLG | Text |
| ON_ST_PRKG_PRSC_TYPE | Text |
| ON_ST_PRKG_TYPE | Text |
| RDWAY_LTG_TYPE | Text |
| TOLL_FCLTY_TYPE | Text |
| EDGLN_WIDTH_MS | Double |
| CNTRLN_WIDTH_MS | Double |
| CNTRLN_RMBL_STRP_TYPE | Text |
| PASS_ZN_PCT | Short Integer |
| BRDG_NBR | Text |
| Verified | Text |

## Appendix IV. Python Script Calculate Intersection Angle

```
import arcpy

import math

from arcpy import env


def getWorkspaceFromPath(aPath):

      lastSlash = aPath.rfind('.gdb')

      workspace = aPath[:lastSlash + 4]

      return workspace


def findMinAngle(InputTable,outputTable):

      whereClause = "NodeID > 0 And NodeID <= 2"

      whereClause = ''

      spatialReference= ""

      fields = "LegID, LegAngle, NodeID "

      sort _ fields = "NodeID"

      rows = arcpy.SearchCursor(InputTable, whereClause,
spatialReference, fields ,sort _ fields)

      print "Leg table OPEN SUCCESSFUL"

      outputRows = arcpy.InsertCursor(outputTable)

      #print "outputPrmts OPEN SUCCESSFUL"

      maxLegNum = 8

      curNodeID = "0"

      legIndex = 0

      arLegAngle = [1 for x in range(maxLegNum )]

      bFirstTime = True

      mtLegAngle = [[1 for x in range(maxLegNum )] for y in range(
maxLegNum )]

      mtLegAngle[1][1] = 1

      iLegCount = 0

      isNewNode = False
      for row in rows:
            NodeID = row.NodeID
            LegID = row.LegID
            LegAngle = int(float(row.LegAngle))

if NodeID != curNodeID:
                  if not bFirstTime:
                        #printArr(arLegAngle, iLegCount)
      INTSECT _ ANG _ MS = calculateLegAngles(arLegAngle,mtLegAngle,
iLegCount)
                        updateOutput(outputRows,curNodeID, INTSECT _ ANG _
MS)
                  curNodeID = NodeID
                  legIndex = 1
                  isNewNode = True
                  bFirstTime = False
                  iLegCount = 1
                  arLegAngle[legIndex] = LegAngle
            else: # same NodeID
                  legIndex = legIndex + 1
                  #arLegAngle.append(Angle)
                  arLegAngle[legIndex] = LegAngle
                  iLegCount = iLegCount + 1

            if isNewNode:
                  #print "\nNew Node %s" % NodeID
                  isNewNode = False

      #print "NodeID = {0},LegID ={1},Angle={2},legIndex={3},iLegCount= {4}:
".format(NodeID,LegID, Angle, legIndex, iLegCount)

      #printArr(arLegAngle, iLegCount)
```

```
    #Out of for loop, calculate the last time
    INTSECT _ ANG _ MS = calculateLegAngles(arLegAngle,mtLegAngle,
iLegCount)
    updateOutput(outputRows,NodeID,INTSECT _ ANG _ MS)
    #arcpy.Delete _ management(InputTable)


def updateOutput(outputRows,NodeID,INTSECT _ ANG _ MS):
    newRow = outputRows.newRow()
    newRow.IntersectionID = NodeID
    newRow.INTSECT _ ANG _ MS = INTSECT _ ANG _ MS
    outputRows.insertRow(newRow)


def calculateLegAngles(arLegAngle,mtLegAngle, iLegCount):
    #print "Inside calculateLegAngles function"
    for iRow in range(1, iLegCount + 1):
            #mtLegAngle.append([]) # append new row
            for iCol in range(1, iLegCount + 1 ):
                if iRow < iCol:
mtLegAngle[iRow][iCol] = (abs(arLegAngle[iRow] - arLegAngle[iCol])) #
append new col
                    if mtLegAngle[iRow][iCol] > 180:
mtLegAngle[iRow][iCol] = 360 - mtLegAngle[iRow][iCol]
                    #print mtLegAngle[iRow][iCol],
                #else:
                    #print "",
            #print ("")


    minAngle = 360
    for iRow in range(1 , iLegCount + 1):
            for iCol in range(1, iLegCount + 1):
if mtLegAngle[iRow][iCol] > 0 and mtLegAngle[iRow][iCol] < minAngle and
iRow < iCol:
```

```
                        minAngle = mtLegAngle[iRow][iCol]
    if minAngle > 90:
        minAngle = minAngle - 90
    print "Min Angle: ", minAngle
    return minAngle

def printArr(myArr, iLegCount):
    for iRow in range(1, iLegCount + 1):
            print myArr[iRow]

def deleteAllFieldsExcept(inputDS, fieldsToKeep):
    print('deleteAllFieldsExcept')
    allFields = arcpy.ListFields(inputDS)

    fieldsToDelete = []
    for field in allFields:
        if field.name not in fieldsToKeep:
            fieldsToDelete.append(field.name)
            print("{0} ".format(field.name))

    if fieldsToDelete:
        arcpy.DeleteField_management(inputDS, fieldsToDelete)
    return

def copyTable(featClassLayer, intsectTbl, intsectFieldsToKeep):
    env.workspace = workspace
    if arcpy.Exists(intsectTbl):
        arcpy.Delete_management(intsectTbl)

    arcpy.CopyRows_management (featClassLayer, intsectTbl )
    deleteAllFieldsExcept(intsectTbl, intsectFieldsToKeep)
```

```
      return intsectTbl


def joinWithIntsect(intersectionFC, intsectTbl):
      arcpy.AddIndex _ management (intsectTbl, ‘intersectionID’,
‘intsectTbl _ intsectID _ Idx’, ‘UNIQUE’,’ASCENDING’)
      arcpy.AlterField _ management (intsectTbl, ‘INTSECT _ ANG _
MS’,’INTSECT _ ANG _ MS2’,’INTSECT _ ANG _ MS2’)


      intsectLayer = intersectionFC + ‘ _ Lyr’
      arcpy.MakeFeatureLayer _ management (intersectionFC, intsectLayer)


      arcpy.AddJoin _ management(intsectLayer,’intersectionID’,intsectTbl,
‘intersectionID’,’KEEP _ COMMON’ )


      expression = “getIntsectAngle(!INTSECT _ ANG _ MS2!)”
      codeBlock = “””def getIntsectAngle(INTSECT _ ANG _ MS2):
                  return INTSECT _ ANG _ MS2
       “””


      arcpy.CalculateField _ management (intsectLayer, ‘INTSECT _ ANG _ MS’,
expression, “PYTHON _ 9.3”, codeBlock)
      arcpy.RemoveJoin _ management(intsectLayer)
      return True


intsectFieldsToKeep = [‘OBJECTID’,’intersectionID’,’INTSECT _ ANG _ MS’ ]


legCentroidTbl = arcpy.GetParameterAsText(0)
intsectTemplate = arcpy.GetParameterAsText(1)
intsectTablePath = arcpy.GetParameterAsText(2)


workspace = getWorkspaceFromPath(intsectTablePath)


intsectTbl = copyTable(intsectTemplate, intsectTablePath,
intsectFieldsToKeep)
LegCount = arcpy.GetCount _ management(legCentroidTbl)
iLegCount = int(LegCount.getOutput(0))

if iLegCount > 0:
      findMinAngle(legCentroidTbl, intsectTablePath)
      arcpy.SetParameter(3, intsectTbl)
```

## For More Information:

http://safety.fhwa.dot.gov

## FHWA, Office of Safety

Robert Pollack
Robert.Pollack@dot.gov
(202) 366-5019

U.S. Department of Transportation
**Federal Highway Administration**

Safe Roads for a Safer Future
*Investment in roadway safety saves lives*

December 2016
FHWA-SA-17-028

http://safety.fhwa.dot.gov