# Commute GeoCalculator:

## A GIS Server Extension for Comparing Automobile and Transit Travel Costs

by

John Siddhartha Pedigo

A Thesis Project Presented to the
FACULTY OF THE USC DORNSIFE COLLEGE OF LETTERS, ARTS AND SCIENCES
University of Southern California
In Partial Fulfillment of the
Requirements for the Degree
MASTER OF SCIENCE
(GEOGRAPHIC INFORMATION SCIENCE AND TECHNOLOGY)

May 2023

To my wonderful fiancé, Dee.
May we live a good story.

# Acknowledgements

I am grateful to my mentors, Professors Darren Ruddell, and Vanessa Osbourne, for the direction I needed and to Dr. Elisabeth Sedano for instructive writing support and advice when I needed it. I would like to thank Mr. Richard Tsung, former System Administrator at the Spatial Sciences Institute USC, and Mr. Jason Henderson, GIS Supervisor at the Texas Department of Transportation, who both supported me on numerous occasions with technical needs in the cloud-based development environments. I am also grateful to Streetlight Data, Inc. for the assistance provided by Daniel Katleman and Adham Kalila in the licensed acquisition of critical traffic source data.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| | |
|---|---|
| AADT | Average annual daily traffic |
| ABM | Activity-based model |
| API | Application Program Interface |
| AWS | Amazon Web Services |
| COM | Component Object Model |
| COTS | Commercial off-the-shelf |
| EPSG | European Petroleum Survey Group |
| FSM | Four-stage model |
| FTA | Federal Transit Administration |
| GIS | Geographic information system |
| GISci | Geographic information science |
| GTFS | General Transit Feed Specification |
| GUID | Globally unique identifier |
| IIS | Internet Information Services |
| JSON | JavaScript Object Notation |
| MAP-21 | Moving Ahead for Progress in the 21st Century Act |
| MLRS | Multi-level Linear Referencing System |
| MPPM | Modernized portfolio and project management |
| MWAAS | Middleware-as-a-service |
| MWCOG | Metropolitan Washington Council of Governments |
| NCHRP | National Cooperative Highway Research Program |
| NCRTPB | National Capital Region Transportation Planning Board |

| | |
|---|---|
| NEC | Northeast Corridor (US) |
| NU | New urbanism |
| OD | Origin-destination |
| REST | Representational State Transfer |
| SOE | Server Object Extension |
| TAZ | Traffic Analysis Zone |
| TDM | Travel demand management |
| TOD | Transit-oriented development |
| TRB | Transportation Research Board |
| UI/UX | User interface / user experience |
| VGI | Volunteered geographic information |
| VMT | Vehicle miles travelled |
| WMATA | Washington Metropolitan Area Transit Authority |

# Abstract

Most research literature on aggregate travel behavior and the built environment indicates that a dense, mixed-use, and transit-friendly settlement pattern generates lower automobile miles travelled than a traditional suburban development. By the same comparison, a substantial portion of research shows that any shift away from this ideal neo-urbanist community to more general urbanized areas exhibits only marginal – if any – influence upon travel behavior. Additionally, the commuter who must traverse such complex urban landscapes lacks information about the daily end-to-end costs associated with each practical mode of travel. This project's GIS service package models the costs of driving versus transit, in minutes and dollars, for individual commutes from the perspective of a traveler. To sufficiently provide these spatial results, a network dataset was constructed for each travel mode – driving and multimodal transit – in the central metropolitan area of Washington D.C. The applied variables for driving included the cost of fuel per mile, travel time, parking time, and average parking fee. For bus, rail and pedestrian modes, the variables include average transit fares, walking, waiting and in-vehicle times, as well as these same inputs applied to any transfers. Commute times are summarized for each mode alongside corresponding dollar totals. For this cost conversion, annual income is extracted from location-based probabilistic income in traveler demographic data provided by StreetLight Data, Inc. Through web services development this thesis investigates a new approach for web GIS to model travel-cost information for individual commutes. These interactive services facilitate several use cases for research and transportation management, particularly if applied to invoke a commuter's quantitative and qualitative response to mode choice. Where uncertainty currently prevails in modeling travel behavior, such empirical mode-choice data volumes become quite valuable.

# Chapter 1 Introduction

In terms of cost, there is a fundamental realization to consider in the way that an individual views the impact of their daily commute. It is widely accepted that automobile transportation produces harmful greenhouse gases which contribute to air pollution, global warming, and climate change, as well as respiratory ailments (Centers for Disease Control and Prevention 2022). Yet the trends in automobile-driven fossil fuel consumption continue to increase each year (Energy Information Administration 2007). For the individual driver, inactivity in conjunction with long commutes increases the rate of cardiovascular illness and obesity, two of the leading causes of death in the United States (Hoehner et al. 2012). Before the onset of the COVID-19 pandemic in the US, fatal automobile accidents were listed among the top three leading causes of death (Centers for Disease Control and Prevention 2021). Socioeconomic research finds additional deleterious issues linking automobile dependency to the lack of municipal tax revenue, wasted federal subsidies (Calthorpe 1993), ageing infrastructure (Duany and Plater-Zyberk 1992), inner-city crime (Kushner 2005), job loss (Kuby, Barranda, and Upchurch 2004), classism and even racism (Hanson 2001). Despite some level of public awareness about these costly outcomes, on any given day most commuters in the US are not likely to be dissuaded from driving.

Within a broader initiative to address this problem space, the goal of the *Commute GeoCalculator* is to provide web GIS services that calculate and visualize the "perceived" costs of individual commutes by transit and driving. For each travel mode, perceived costs are direct "out-of-pocket" expenses measured in minutes and dollars for the purpose of invoking a mode choice response from an application end user. However, while this purpose serves the long-term goal of the project, the immediate results generated by these services do not necessarily require a

mode choice response. Various applications in research and industry will benefit from the project's automation of comparative travel costs, modeled over a chosen geography.

As for the long-run goal of the project, economic feasibility forms the quantitative basis for the travel mode decision-making process via a public-facing application that would leverage these web services. The end results of this process, empirical mode choice data, are valuable to government organizations in forming transport policy, or perhaps useful to researchers in testing spatial analyses. Particularly where changes to the built environment promote the use of transit in a heterogenous settlement pattern, these empirical mode-choice data present a new opportunity toward intervention against automobile dependency.

## 1.1. Project Motivation and Background

New urbanism – often called smart growth – is an interdisciplinary planning movement dedicated to counteracting the costly effects of sprawl and automobile dependency by applying new policies to land-use and urban form, including transit-oriented developments (TODs) (Calthorpe 1993). Attraction to transit is a pervasive topic in the NU movement, and there is an enduring consensus among proponents that land-use and urban form policies continue to control, manage, and shape the travel behavior of individuals. However, after a multitude of various studies on this topic, it is generally unclear if these types of policy changes to the built environment necessarily translate into changes in travel behavior. The end goal and motivation of the project is to construct an effective response to research calling for application components that apply authoritative methods and user inputs to help reveal how policy changes impact the attraction of transit (Tallis 2014).

The data on travel behavior in prior literature, such as vehicle hours, are typically aggregated geographically and applied to spatial analyses. Such efforts are referred to, here, as

aggregate studies. Inconclusiveness persists even where careful treatment is given to confounding variables and the classification of the built environment, in efforts to control for spurious correlations in heterogeneous settlement patterns (Higgins and Kanaroglou 2016). The challenges to establishing forward causality of the built environment to travel behavior have not eluded researchers. Quite to the contrary, multiple aspects of the built environment, population, economy, and regional effects have been realized and controlled by research scientists. But in the majority of case studies, the author finds vast urban geographies where forward causality remains inconclusive. At the core of the NU approach is the issue of whether the true effects from the built environment on travel demand are causal, associative or a mixture of both (Bhat and Eluru 2009). This distinction is centrally important to the intended future use of the project results as ground-truth data for testing travel behavior models. This aspiration of the project is examined more closely in Chapter 2 with four methodological problems that have been identified in travel behavior studies.

The manner in which NU-related achievements and critiques have developed since the early 1990s provides the underlying foundation and impetus for the current project. The key finding from these urban studies is the opportunity for new developments in the approach to data heterogeneity and uncertainty in travel behavior. This is not to suggest that aggregate studies on travel behavior carry little value, but rather that there are disconnects between the research and what is on the minds of individual daily travelers. The motivation behind the *Commute GeoCalculator* is to facilitate the alleviation of these disconnects through a mutually beneficial exchange of services and information among commuters, researchers, and transportation authorities.

## 1.2. Project Overview

The project creates middleware-as-a-service (MWAAS) that models the travel cost between origin X and destination Y of a user-defined commute, with the option of unlimited intermediate stops for analyzing a tour. The quantitative information provided to the user consists of peak and non-peak travel costs of driving and multimodal transit between origin and destination (OD) points. The intention is to enable a mapping application to assist individuals in their decision-making process regarding OD travel options. Based on OD locations, authoritative input parameters, and the economic principle of marginal utility, the web application translates time into dollar amounts for walking, waiting, transferring, driving, parking, and riding transit. Fuel costs and transit fares are added, where applicable. This spatialized translation reveals the perceived cost of each travel mode for a commuter. If these direct costs can be readily shown for any individual's specific travel pattern, the author finds it likely that more constructive attention will be given to mode choice by the travelling public. Further, where these modeled travel costs are collected, a number of valuable insights unfold about the performance of urban policy on the transportation system, itself, and the travel behavior therein.

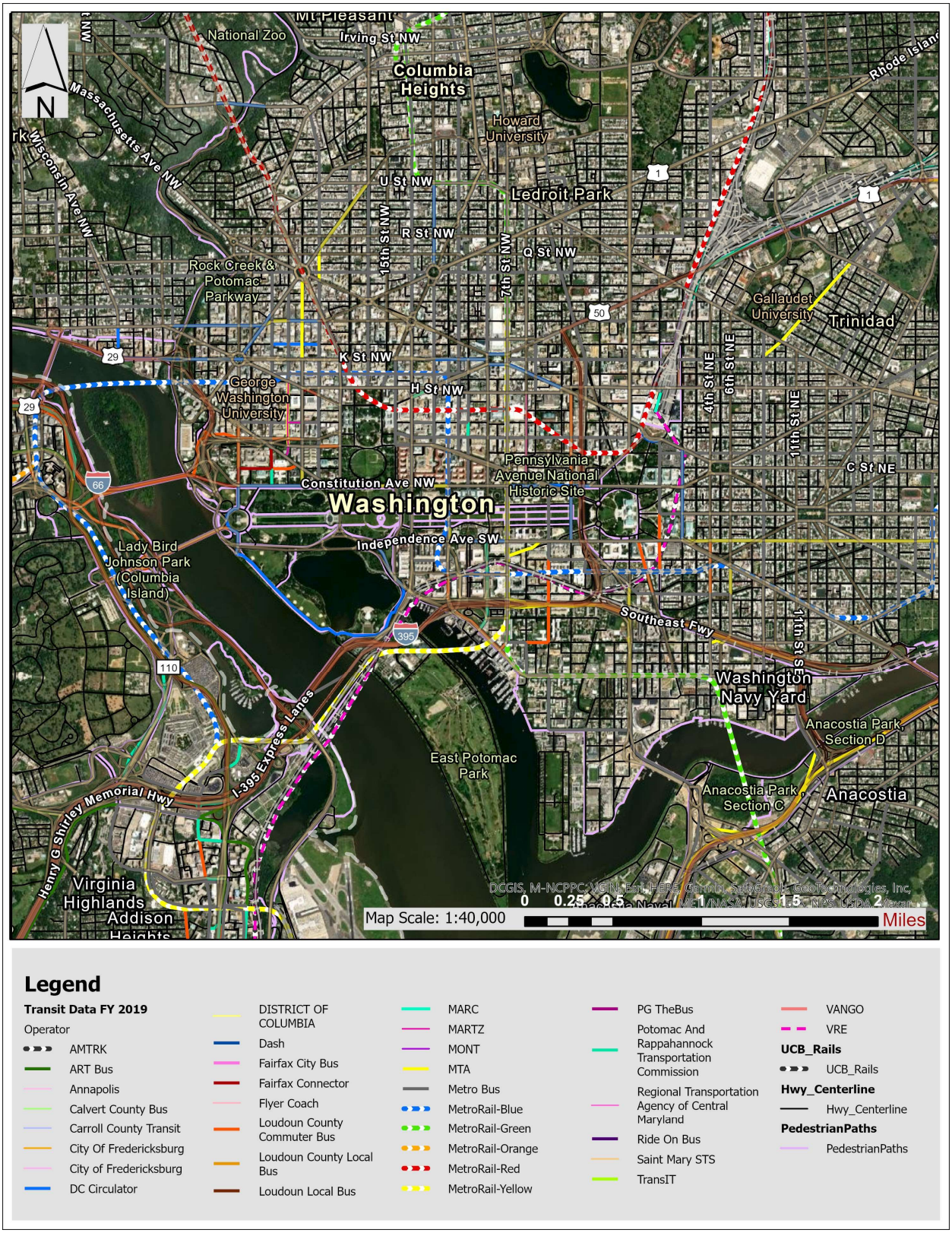
Regarding ongoing efforts to hypothesize travel behavior, some probabilistic models used to classify elements of the built environment assume that population variables are unobserved. In such cases, these variables are often represented by clusters of other manifest variables, which may easily skew results. The absence of empirical population data is problematic for studies that seek to establish typologies of the built environment as a basis for modeling travel behavior (Higgins and Kanaroglou 2016). The *Commute GeoCalculator* developed herein is designed to support a web client that obtains end user responses to the travel cost returns from the service endpoints. User responses may indicate which travel mode is chosen for each individual

commute, and why it is chosen. VGI of quantitative and qualitative travel preference data across a growing population is the future goal of development for the current application. The project carries the potential to expose underlying factors that transcend the current limitations imposed by predetermined typologies of the built environment. Where applied, this exposure of travel mode efficiency may lead to better informed TDM (travel demand management) strategies or service improvements (Litman 2010).

The application of marginal utility in travel behavior studies can be found in early research literature of the 1970s – the age of neotraditional environmentalism, predecessor of NU. Shunk and Bouchard (1970) point to the development of an independent decision variable based upon marginal "disutilities" of travel by competing modes as perceived by the traveler. Their study shows how a model that is basically more behavioristic in nature rather than simulative conceivably could be an approach to more effective modal-choice prediction procedures (Shunk and Bouchard 1970). The project closely considers this behavioral approach by providing a web-based toolset designed for gathering direct feedback from a commuter's point of view.

## 1.3. Study Area

The central metropolitan Washington D.C. area is ideal for the development of a scientific tool to assess and compare travel data, because mode choice is a pervasive characteristic of the geography. Local and regional bus and rail services are widely available throughout most of metropolitan area. Rail and road data are sufficient in this area for building the respective network datasets. Settlement patterns are heterogenous – from transit-oriented developments (TOD) to traditional suburban neighborhoods and commercial strips. The study area, shown in Figure 1, contains a diverse population, a wide range of employment densities, and a very active presence of urban planners and policy makers as it is anchored by the US Capitol complex.




## Legend

**Transit Data FY 2019**

Operator

- AMTRK
- ART Bus
- Annapolis
- Calvert County Bus
- Carroll County Transit
- City Of Fredericksburg
- City of Fredericksburg
- DC Circulator
- DISTRICT OF COLUMBIA
- Dash
- Fairfax City Bus
- Fairfax Connector
- Flyer Coach
- Loudoun County Commuter Bus
- Loudoun County Local Bus
- Loudoun Local Bus
- MARC
- MARTZ
- MONT
- MTA
- Metro Bus
- MetroRail-Blue
- MetroRail-Green
- MetroRail-Orange
- MetroRail-Red
- MetroRail-Yellow
- PG TheBus
- Potomac And Rappahannock Transportation Commission
- Regional Transportation Agency of Central Maryland
- Ride On Bus
- Saint Mary STS
- TransIT
- VANGO
- VRE

**UCB_Rails**

- UCB_Rails

**Hwy_Centerline**

- Hwy_Centerline

**PedestrianPaths**

- PedestrianPaths

Figure 1. Study Area – Central Metropolitan Washington D.C.

Furthermore, the Metropolitan Washington Council of Governments (MWCOG) has territorial jurisdiction within the NEC (Northeastern Corridor) rail network, which can facilitate future growth of the application onto the megaregional scale (Mitch 2021). All required data are contained within this boundary encompassing the District of Columbia (Washington, D.C.), which includes parcels of Arlington, Fairfax, Montgomery, and Prince George's counties. All data in the study area are processed in the Web Mercator Auxiliary Sphere, WGS 1984 datum, but then published to the MWAAS as unprojected GCS in North American Datum (NAD) 1983. This transformation assists with uniform routing and enables any spatial reference in the service.

## 1.4. Methodological Overview

Applying a two-tier architecture, the project provides commute cost information from at least one origin to one destination point defined by the service user. A substantial amount of data crunching occurs before the user or application visits the REST endpoint and enters parameters to plot commute events. These steps are detailed in Chapter 3, preceding software development in Chapter 4. The overall conceptual design of the MWAAS, its purpose and composition, is best understood through a brief examination of the elements in each tier – starting with the data tier.

To enable the creation of both transit and driving route paths with associated costs across a typical seven-day week, a specific temporal scale is applied to the source data in Tier 1. For the project, fully processed traffic volumes and household income are averaged on weekends and weekday peak/non-peak times in 2019. Specifically, trip speed and traveler annual income are provisioned by StreetLight Data, Inc. (SL) in US-standard traffic analysis zones (TAZ). Through Tier-1 data automation, these metrics are linearized to street centerlines and integrated into the model. A cross-sectional view of all base metrics on one roadway segment is depicted in Table 1. Data sources and source formats are indicated in the summary columns near the top of this table,

while the rows present precisely how each attribute is organized on the project's temporal scale. The column headings at the bottom of Table 1 convey which attributes are assigned as costs for determining path routing within a network dataset, and which attributes are path-dependent metrics assigned to variables after routing.

Table 1. Tier 1 Sample Data: Cross-Sectional View

| US Standard Traffic Analysis Zones | | | US Census Streets | Bus and Rail Stops |
|---|---|---|---|---|
| Day Type:<br>1. Weekday<br>2. Weekend | StreetLight Traffic and Traveller Metrics (GPS and LBS) | | Employment Density | GTFS Calibration Data |
| | AUTOS and TRANSIT | AUTOS and TRANSIT | AUTOS | TRANSIT |
| | Trip Speed (mph) | Traveller Income ($) | Parking Penalty (mins) | Transfer Wait (mins) |
| Day Part:<br>0. All Day | 19.3 | 90100 | 4 | 7 |
| 1. Early AM | 24.5 | 68000 | 2 | 0 |
| 2. Peak AM | 14.2 | 118000 | 5 | 10 |
| 3. Mid-Day | 16.8 | 79000 | 7 | 15 |
| 4. Peak PM | 12.5 | 113500 | 5 | 5 |
| 5. Late PM | 28.4 | 72000 | 1 | 5 |
| Data Period:<br>2019 | Applied to Centerlines | | Applied to Network Datasets | |

The applied modes are automobile, pedestrian, bus, and rail. Bicycle travel is planned for the next project phase. For the automobile mode, SL includes trip speed and traveler income on both car and truck movements, tracked via GPS, then measured against authoritative traffic counts. For transit modes, SL data includes the same variables by indices calculated from location-based services (LBS) (StreetLight 2022), and the project then applies transit schedules from the general transit feed specification (GTFS) data to geolocate transfer locations and to compute average transfer wait times for bus and rail. Pedestrian traffic indexes are applied to the model as is. Average parking penalties are calculated from US Census data on population density multiplied by the percentage of employment by county, with WMCOG authoritative coefficients

(in minutes) assigned to the resulting employment density measures. The extract, transform, and load (ETL) automation processes all time-based variables into the data model, and facilitates the manual build of both network datasets using Esri Network Analyst.

In Tier 2, object-oriented code leverages the spatial data model and enhances the capabilities of a map service to deliver linear referencing functions for each categorized mode of travel – transit and driving. These enhanced capabilities are applied to individual user-defined commutes and include linear referencing with computation of the total travel cost for each travel mode. When an API call via REST (Representational State Transfer) network protocol is sent from a web client in a prospective Tier 3, or directly from an end user at the REST page, the logic module exploits the routing functions of each network dataset to provide path geometry for each travel mode on a commute. Almost simultaneously, Tier-2 code applies the path geometry to spatially extract travel speed, parking penalties, transit wait times, transit fares, and inferred traveler income. These are the operations of the "as is" state represented in Figure 2.

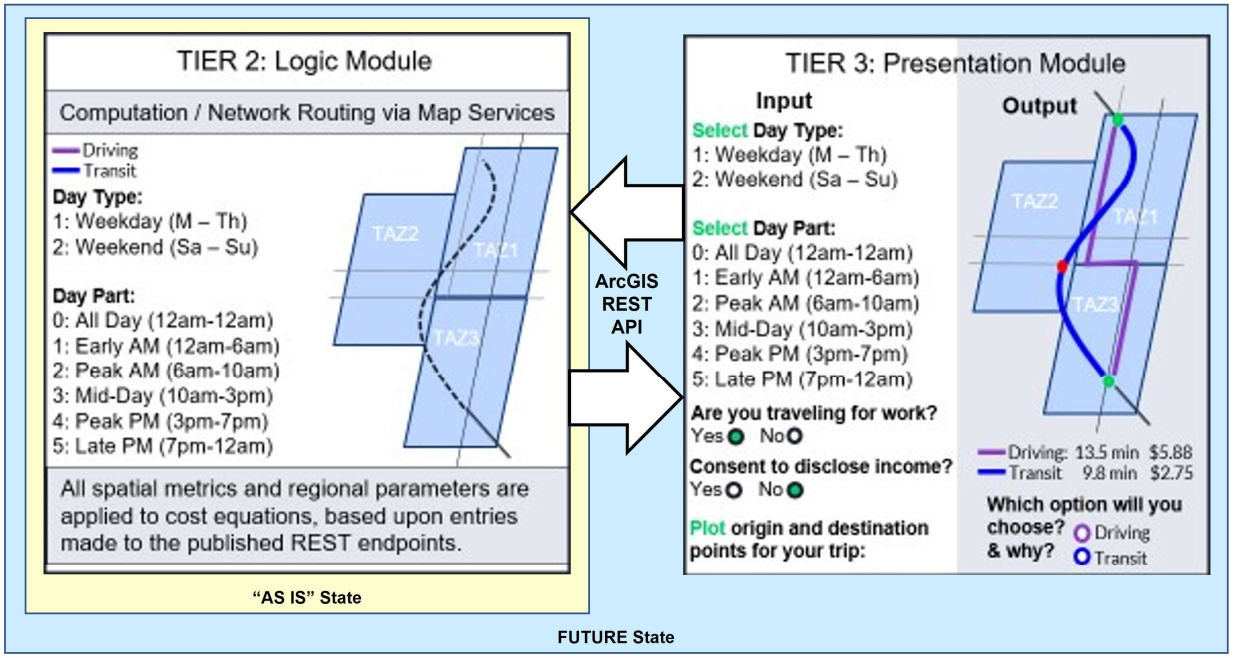


Figure 2. Tier 2 and 3 Cross-Sectional View

With these routed variables, the middleware computes the total cost of driving versus transit using authoritative coefficients and direct cost estimates for parking fees and fuel. In Figure 2, a hypothetical web client interacts with the *Commute GeoCalculator* web services created in the project. By design, the user or application invokes this entire process through the REST API of the logic modules by selecting a day type, day part, with other parameters and plotting origin X and destination Y points for a commute query. After this submittal, what the user receives back are each of the travel mode paths with dollar and minute cost values. Results at the REST page are returned in JSON or HTML format, readily usable by a web client (Fu 2020), and the resulting path geometries are also written to the local application server as feature layer collection for visualization purposes outside of a web client. Alternatively, these results may be consumed by a scripted process, or through direct user interface by someone who is interested in experimenting with the services.

## 1.5. User Requirements

The intended users of the middleware services and data model are GIS application developers who intend to script a travel cost analysis or build a web client that extends travel cost information to the travelling public. Scientists who want to gather samples of mode-specific travel cost data in the service area will also benefit from the project's services. However, the majority of user requirements are defined from the standpoint of the everyday commuter. Commute GeoCalculator enables a web client to more easily accommodate commuters who may have never used GIS software before. In this context, the key functions in the project's middleware address temporal scale, performance, spatial cognition, and thus promote an attractive user experience. Transportation researchers, planning authorities, and policy makers

may also take interest in the resulting mode choice data, given that these user-driven data would be collected over time.

The end goal for user requirements is two-fold in that the project hopes to directly support a web GIS that encourages people to use more sustainable travel modes than gas-powered automobiles, and it endeavors to collect data that policy makers and researchers can use to enhance their understanding of urban travel behavior. Accordingly, the commuting public are the intended end users of the project's services operating behind a web application, while the intended beneficiaries include urban researchers as well as urbanites that benefit from improved decision-making.

To that end, the middleware provides general travel cost estimates based on user-inputted start- and end- points of a planned trip. Plus, the desired time of day for each commute and any preferred spatial reference for the output must also be entered. When implemented in a web client, users should be prompted to share – as VGI – the mode and path of travel they choose for their trip, and why the mode was chosen. In the future workflow, this is the point at which the user will have the opportunity to make their location-specific views known to transportation researchers, planning authorities, and policy makers. By design, the intended application would allow only these authorized parties to access the web client for the purpose of viewing and later downloading volunteered commuter data. For the purposes of fulfilling review and demonstration of the project, the initial users shall be authorized students, faculty, and staff of the Spatial Sciences Institute at the University of Southern California.

## 1.6. Thesis Overview

The ensuing chapters detail the progression of project stages, following a review of related works and literature in Chapter 2. In Chapter 3, the criteria, requirements, and components of the

data tier are discussed in detail. From data acquisition through each data processing step built into ETL automation, this chapter provides sufficient information for replicating the application's data model in Tier 1. Chapter 4 covers the same proposition for the development of Tier 2, including the purpose, function, and form of the system architecture – software, versions, and resources – as well as the database design and object-oriented code engineered by the author. Chapter 5 summarizes the overall project results, with a walk-through for using the finished application at the REST page. References to user help documents are provided as well as an in-depth discussion of the pertinent use cases and test results. Chapter 6 provides a conclusion of the design choices and principles applied, as well as project limitations, development challenges, recommendations for future work, and the prospect of expanding *Commute GeoCalculator* to other geographic locations.

# Chapter 2 Related Literature

The review of related works begins with research on cost-based utilities in travel mode choice, followed by common methodological problems in travel behavior studies and research design principles that address such issues. Additionally, current regulations and policy trends in transportation management are assessed in this chapter, and a brief review of related applications is offered. In a comparison and contrast of related applications, the author refers to the spectrum of programs and software that leverage GIS to model or analyze travel behavior. Finally, all research findings are drawn together into a summary of insights that have influenced the design and development work of the project.

## 2.1. Utility Theory and Costs of Travel

In this project, utility theory is applied in the construction of additive cost equations to establish a fundamental basis of choice by individual travelers. Mode choice is an aspect of travel demand analysis that determines the number or percentages of trips between zones that are made by automobile and by transit. Whether one travel mode is preferred to another depends on how much utility, or satisfaction, it yields relative to its alternatives (Case and Fair 2004). Economic feasibility based on out-of-pocket costs, itself, may very well not be enough to guide the probable mode choice of most commuters in any given urban setting. However, it can be reasonably expected that most people who frequently commute would at least consider the economic feasibility of alternatives as a decision-making factor. This is a common inference in travel cost models. And with this in mind, the research approach to utility theory in this project presumes that the benefit of travel is defined solely as the destination, itself, to which all mode choices have access. Therefore, the benefit is set to zero and the utility of travel is defined only

by the cost of transportation, in an indirect mathematical relationship expressed as the following:

$$B_T + C_T = U_T \quad (1)$$

where $B_T = 0$, as any benefit of transportation beyond just reaching the destination; $C_T$ is the cost of transportation, and $U_T$ is the utility of transportation.

In this context of travel costs, there are a number of utility formulae for mode choice found in the scientific literature. Researchers, Koppelman and Bhat, contribute a disaggregate discrete-choice model of high relevance to the project. The authors assert no direct benefits of a given travel mode, itself, beyond the destinations involved. Despite occasional mention of intrinsic benefits found in the travel mode preferences of certain population groups, the paper remains cost centric in its analysis. These subject matter experts do point out certain advantages of a disaggregate approach to modeling group behavior over an aggregate approach. Here, the aggregate approach is referred to as directly modeling the combined share of all or a segment of decision makers choosing each alternative as a function of grouped elements. The disaggregate approach recognizes that aggregate behavior is a result of numerous individual decisions based on available elements, and thus individual choice responses are modeled (Koppelman and Bhat 2006).

The aforementioned elements of travel behavior applied in Koppelman and Bhat (2006) include the following: (1) attributes of the travel alternatives, (2) utility biases due to excluded variables, (3) a characteristic variable of the traveler, and (4) interactions relating the traveler to their mode preference. The attributes of travel alternatives encompass aspects of the built environment – path length, travel cost, waiting and walking times, parking time, the level of mobility and service frequency, for example. Excluded variables may include socio-demographic factors at a different geographic scale, or qualitative data regarding safety, comfort, and

reliability of travel alternatives. Characteristic variables of the traveler often include household income, age, automobile ownership, and the purpose of a trip. Interactions between the traveler and their mode preference refers to network accessibility, employment or residential density, proximity to carpool lanes or other transportation facilities. These elements are also found in the equations issued by the National Capital Region Transportation Planning Board (NCRTPB) and adopted by MWCOG in calculating travel costs at the regional level (NCRTPB 2020). This basic construct is expressed as the following:

$$V_{i,t} = V(S_t) + V(X_i) + V(S_t,X_i) \quad (2)$$

where $V_{i,t}$ is the systematic portion of utility in the alternative *I* for individual *t*; $V(S_t)$ is the portion of utility associated with characteristics of the individual *t*; $V(X_i)$ is the portion of utility in the alternative *I* associated with the built environment; and $V(S_t,X_i)$ is the portion of the utility which results from interactions between attributes of alternative *I* and the characteristics of individual *t*. This additive cost model can produce a cost value associated with each travel mode – multimodal transit as well as automobile. A larger output value indicates a less convenient commute, whereas a smaller value reflects the opposite scenario. The difference between applied outputs assists in determining the level of convenience associated with each mode, again, in terms of cost. In particular, the inclusion of multiple characteristic variables of the traveler is shown to significantly reduce residual outcomes of uncertainty in the modeling process presented by Koppelman and Bhat (2006). Traveler and trip related data, relating specifically to the actual mode choice of the traveler, are generally obtained by surveying a sample of travelers from the population of interest. The most common of these travel surveys are household, workplace, and intercept surveys; each of which involve direct contact with travelers by the researcher.

Applicable to the above additive method is a family of multinomial equations, known as logit models. Logit models are multi-factor logistic regression, and these models are often implemented to analyze daily travel behavior in terms of relative probabilities of auto and transit alternatives used between predetermined attraction zones. In the most basic description, the utility function $U_i$ is composed a deterministic part, $a_iX_i$, and a random part, $\varepsilon_i$ the unknown residuals (Bai, Li, and Sun 2017). The fundamental form is $U_i = a_iX_i + \varepsilon_i$, where $a_i$ represents the set of weighted coefficients assigned to known costs $X_i$ of a given travel mode: path length, waiting time, walking time, transfer time, parking time, traffic conditions, service frequency, and socioeconomic factors as well. In practice, a basic logit model is often expressed as:

$$U_x = \sum_{i=1}^{n} a_iX_i + \varepsilon_i \qquad (3)$$

where $U_x$ is the utility of a given travel mode; $X_n$ represents the total number of attributes; $X_i$ is the attribute value of time, cost, or other factor; and $a_i$ is the coefficient value for attribute *i*. With given value inputs, the $U_x$ result for each mode applied to this equation is then fashioned into a proportion of exponentials equal to one (1), for the probability that a commuter will choose one mode over the other in a given travel scenario. For example, the probability of mode choice in auto P(*A*), applied from the above $U_x$ result for automobile utility, $U_A$, is:

$$P(A) = \frac{e^{U_A}}{e^{U_A} + e^{U_T}} \qquad (4)$$

Conversely, then the $U_x$ result for transit utility, $U_T$, is the prerequisite for solving this example for P(*A*). This proportion may be solved for the probability of mode choice in transit by very simple algebra, P(*T*) = 1 – P(*A*). The example depicts only a basic logit model for mode choice.

A substantial number of travel behavior studies apply advanced logit models for the purpose of traffic forecasting or predictive modeling. Among these methods is the trip-based approach grounded in the Four-Stage Model (FSM) that is often seen in commercial-off-the-shelf (COTS) traffic simulation software for performing trip generation, trip distribution, mode choice, and traffic assignment. Then, there is the tour-based approach, rooted in the Activity-Based Model (ABM) of travel demands. The latter of these two logit model applications is of acute interest to the project, because ABM is aligned with dynamic discrete choice modeling (DDCM) with explicit consideration of state dependence and expectation feedback. State dependence in this context refers to the position of the traveler in route on the transportation systems, and the conditions present – traffic, weather, vehicle access, time of day, etc. Expectation feedback is the expectation of the next trip's mode choice, obviously during a tour. Empirical data for these two variables of ABM are relatively rare but may be captured through intercept surveys in route. Generally, however, these data are inferred through statistical methods (Hasnine and Habib 2018).

To build cost utility models for multimodal transit and driving, the qualifying elements found in research include a value of time parameter that is based on combined characteristics of the individual. Plus, in-vehicle travel time, walking time, transfer waiting time and fare costs apply to transit, while fuel cost per mile, driving time, parking time and parking fees apply to driving. Finally, a random part, $\mathcal{E}_i$ for non-deterministic residual factors improves the accuracy and reliability of a travel cost utility model. The overall aim asserted to cost-based utilities in the current project, is that any elements applied from research are also firmly rooted in authoritative methods applied within the region of interest.

## 2.2. Principles and Problems in Travel Behavior Research Design

Many traffic and travel behavior studies include critical assessments of efforts to establish forward causality from the built environment to travel behavior. Where inconclusive or conflicting results are most often found in this research literature, four methodological problems are identified: (1) self-selection; (2) spatial autocorrelation; (3) inter-trip dependency; and (4) geographic scale. Across a range of methodological issues found in critical review, these four challenges stand out as the most impactful to proving forward causality and correlation of the built environment upon the travel behavior of commuters (Hong, Shen, and Zhang 2014). However, it is noteworthy to mention that through this process significant gaps in the treatment of disaggregate traveler data have also been discovered. The most notable example of this finding in existing literature is the lack of attention to behavioral changes in individuals who move to new environments, where work and leisure destinations are closer and transportation options are abundant. Most travel studies are cross-sectional rather than longitudinal, but researchers Hong, Shen, and Zhang (2014) identify this common scenario from the perspective of the traveler as a significant factor that should be manageable in non-longitudinal travel studies.

Of the four most predominant methodological problems, self-selection occurs when households direct themselves into neighborhoods, either neo-urbanist or traditional suburban, based on their own VMT preferences. This dilemma erodes the randomness of observations due to unattended linkages between residential location choice and travel patterns. Some relatively recent research that is focused on this topic finds evidence of significant interplay between self-selection and geographic scale. In this regard, an important empirical question concerns whether ZIP codes are delineated at an appropriate spatial scale for capturing behavioral processes. Too

large an area will dilute the effects of urban form, while too small an area will potentially omit important effects (Vance and Hedel 2007). Empirical evidence collected across a range of geographic scales indicates there is substantial scope for land use measures to influence mobility behavior, when the sampling frame is set to an area smaller than zip codes (Grazi, Van de Bergh, and Van Ommeren 2008; Vance and Hedel 2007).

The main criticism of spatial autocorrelation states that observations are no longer independent when nearby locations tend to have similar characteristics (Bhat and Eluru 2009; Hong, Shen, and Zhang 2014). This is noted as a common problem for linear regression based on non-zero spatial autocorrelation, whereby biased estimators can debase any inference of significance (LeSage 1997; Miller 1999). Multiple researchers refer to a simple hierarchical framework of empirical data sampling on both dependent and independent variables as a way of mitigating against this problem (Duncan and Jones 2000; Bhat and Zhao 2002; Bottai, Salvati, and Orsini 2006; Antipova, Wang, and Wilmont 2011; Chaix et al. 2005). In principle, scientific observations must be independent and random in any sampling phase. A hierarchical framework provides a convenient means of organizing geographic data samples by distinct characteristics, before applying these to variables in spatial analyses.

The problem with inter-trip dependency is the fixation of trip-based models upon isolated trips without considering possible interdependency between trips that would comprise a complete tour (Hong, Shen, and Zhang 2014). As a result, conventional trip-based analysis is criticized as unsatisfactory for explaining the fundamental forces behind travel behavior (Krizek 2003). Critical analysis suggests that the more stops involved with connected trips, the more people choose to use their automobiles.

Finally, the modifiable areal unit problem (MAUP) generates inconsistent results when data are measured at different geographic scales (Hong, Shen, and Zhang 2014). Goodchild (2011) puts forth an in-depth discussion of scale, in which geographic or spatial scale is emphasized as a representative fraction of some natural or social phenomena under analysis. He speaks to the process of decision-making, itself, by noting that data for any physical process should first be defined and tested free of any scale whatsoever, to the extent possible. Sorting through these aspects of a given analysis upfront during the decision-making process is imperative to correctly interpreting the results (Goodchild 2011). With travel cost results kept in disaggregate form, the author finds little reason to anticipate the MAUP in the long-term direct use of the project's output data. However, Section 2.5 discusses how certain abstractions asserted within the two-part cost model do invoke errors that are related to the MAUP.

## 2.3. MAP-21 Regulations and Policy Trends in Transportation Management

Moving Ahead for Progress in the 21st Century Act (MAP-21) allocates federal dollars to the Federal Highway Administration (FHWA) and the Federal Transit Administration (FTA) to issue grants to state and local governments for the purpose of upgrading transportation network facilities. Public safety is the overarching justification upon which this legislation was passed in July of 2012. Under MAP-21, federal funding of multimodal projects in metropolitan areas, including fixed guideway rail, metro bus, pedestrian, and biking improvements, requires project justification based on "before and after studies" that meet local priorities and criteria set forth by the metropolitan planning organization (MPO). Funding for highway projects requires that all principle arterial roads connected to the project meet specific criteria to be designated on the national highway system (NHS). The MPOs also provide oversight for this set of criteria on highway projects. For each respective travel mode (automobile or transit), the MPO is expected

to enforce higher safety standards with all public and private operators seeking MAP-21 funding. Since MAP-21, related reports indicate that a significantly higher level of coordination is taking place between MPO's and government officials for approval and funding of transportation initiatives. A substantial part of this relatively recent development is that reporting requirements have become more stringent with specific criteria mandated for GIS and public participation components on some projects (Kirk et al. 2012).

The regulations under MAP-21 commenced with the charge to improve public safety, but there are additional project criteria set forth that pertain to locational accuracy, data quality, local priorities, and public participation in some cases. Many local and state transportation authorities have adopted public participatory GIS (PPGIS) and modernized portfolio and project management (MPPM) tools, respectively, to better operate in this relatively new coordination space (Kirk et al. 2012; Giuffrida et al. 2019). However, MPPM implementations typically lack the capability to interact with the public in a mutually beneficial capacity when it comes to empirical assessment of mode choice. PPGIS is shown to fill this capacity for local transportation initiatives, and there are other public participation strategies actively utilized by state and local authorities, such as public workshops, websites, and hearings where community feedback is acquired. After previous research on these outreach efforts, the author finds no evidence of a mechanism in place that allows the general public to readily compare the cost-based utilities of their own Individual commute patterns and options. In short, the author finds a gap in these new policy trends wherein the commuting public could have a more effective toolset for engaging all stakeholders regarding local priorities in transportation.

## 2.4. Related Applications

A number of sophisticated COTS software packages apply customizable travel-demand modeling techniques that combine transportation analysis and micro-simulation through GIS, including TransCAD, Emme4, Cube, PTV Vissum, and TranSIM. Some of these products even mitigate against the aforementioned methodical problems, but the respective business and technical reach of all such products are far out-of-scope for the criteria of the current project. First and foremost, each of these travel modeling software packages generate mode choice data from statistically modeled trip distribution and not from explicitly volunteered traveler inputs. Similar to the project, each of these products will work with static metrics as input, but the workflows do not diverge at network routing to collect empirical mode choice data. Instead, these platforms complete all prescribed FSM or ABM tasks to then provide an array of options for animated simulations on the network of interest.

Some of the more popular highway traffic and navigation applications offer trip routing for automobiles with the estimated travel times calculated from near real-time traffic feeds – Google Maps, Waze, and CoPilot GPS are leading examples (Sari et al. 2017). Google Maps offers routing by multiple alternative travel modes based on transit operator data, but one limitation worth noting is that each selected mode may only be routed on separate map views. There is no combined view of all selected travel modes on one map UI, at the time of this current research. Textual directions are optionally displayed with travel times between turns, as well as bus and train operator head signs at transfer points, plus street-view photographs are provided along pedestrian paths. All such features are presented on well-annotated vector tile maps or labelled satellite imagery. However, each mode must be submitted and viewed individually. Figure 3

illustrates this user experience (UX) in Google Maps, for a tourist in Las Vegas, Nevada who is evaluating travel alternatives between the airport and destination hotel.

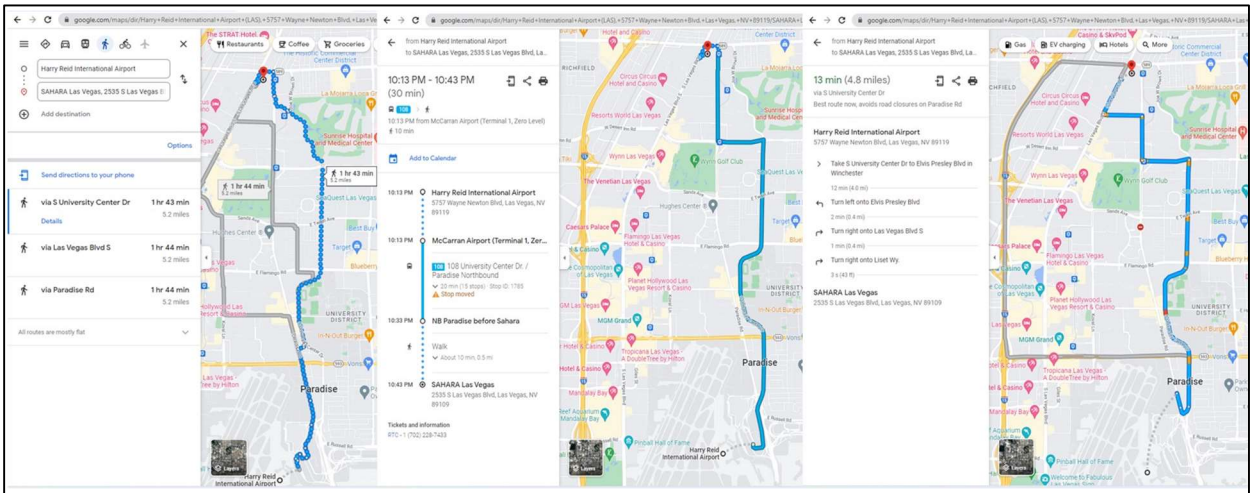



Figure 3. Google Maps Directions by Travel Mode

In this example, the results do not include transit fares, parking time, fuel cost per mile, or any monetary value of time, because the application functions as a traveler information system without any concern for saving and measuring the results of many traversals. Waze and CoPilot GPS are designed to enhance travel planning and navigation only for driving, and they perform very well in that problem space. Waze in particular stands out in this traffic and navigation category, not only for its sheer performance in geolocating capabilities, but for the fact that it is also a sustainable VGI system.

To help further clarify the project's position in the market of transportation apps, the author has identified a similar web GIS that relates to the project's criteria more closely. The "Yay Transit!" application, developed by Melinda Morang and Patrick Stevens of Esri, provides several tools to study and simulate schedule-aware transit trips (Morang and Stevens 2013). However, just as the title suggests, this Esri-based application focuses solely on transit networks whereas this thesis project incorporates certain parallel functions for the roadway (driving)

network in the study area. As with the current project, Yay Transit! leverages the ArcGIS Network Analyst extension for several functions that engage users in transit information, including visualization of static GTFS data on a map. Their innovative toolset extends this capability onto real-time GTFS data, while the project's application exposes only static travel data at peak and off-peak traffic intervals. The Yay Transit! application automatically selects each first available trip to the destination one can take when touring by transit, whereas *Commute GeoCalculator* takes a more general approach to travel scenarios by allowing the user to select peak and off-peak time intervals. In both applications, users who want to delay intermediate departures during a tour will create a new trip for each specific OD traversal along a commute.

Other open-source web GIS are more flexible in this regard, but they tend to be associated with a particular city or region. One example is the "RabbitTransit" application, built on Google Maps Transit services, allows users to view travel duration of each intermediate trip available for any chosen itinerary through select counties in Pennsylvania (CPTA 2021). This application operates similarly to a transit tour booking site, even offering its own express bus service. All origins and destinations are geocoded at predetermined locations, which is an amenity under future consideration in the project. While these kinds of location-based web applications often provide comprehensive transit-user experiences, none have been found by the author to provide a simultaneous view of travel cost by automobile and transit as part of an interactive survey. The future *Commute GeoCalculator* web application is designed to collect mode choice VGI, using multimodal linear referencing capabilities. Its purpose more closely resembles that of scientific web GIS for performing schedule-aware simulations and analysis, such as "Yay Transit!". However, the current project requires further development to fulfill its core purpose of empirical mode-choice data collection.

## 2.5. Summary of Insights Gained from Research

Foundational aspects of the project emerge as a result of the literature review. The investigation of travel costs and utility theory provide all of the components and methodology needed to construct the additive travel cost equations for driving and transit in the project. And, beforehand, there was no consideration for a future VGI web component in the next stage of the project. Discovery of the four methodological problems identified in critical literature led to the first design change in the user interface (UI) whereby the user is asked to indicate which mode option they would choose, based on the results of their route inquiry. Research on policy trends and regulations in transportation reveal a potential process gap in the federal funding of large projects, wherein reporting requirements have become more stringent with specific criteria mandated for GIS data and public participation. As a result, the project aspires to resolve disconnects between transportation stakeholders in meeting those criteria more effectively. Finally, the brief survey of related applications and their usage shed light on new market space in which combined elements of travel planning, navigation and surveying would be a good fit.

The additive cost approach found in the review of utility theory aligns well with the travel cost model issued by the NCRTPB. The deterministic elements articulated by Koppelman and Bhat can also be identified in the authoritative model: (1) attributes of the travel alternatives, (2) a characteristic variable of the traveler, and (3) interactions relating the traveler to their mode preference (Koppelman and Bhat 2006). The only element not included in travel cost utility by the regional authority is the non-deterministic utility bias due to excluded variables. In collaboration, MWCOG designed their set of travel cost utility equations. For transit travel:

$$\text{Total Cost (in dollars)} = pVT \times [pW \times (WT + W2T^*) + pWa \times (WaT + W2T^*) + pIV \times (IVT + IVT2^*)] + (Fare + Fare2^*) \quad (5)$$

**w**here pVT is equal to the value of time parameter for combined characteristics of the individual; pW is the walk Parameter; pWa equals the wait parameter; pIV is the in-vehicle parameter; WT is the walking time; WaT is equal to the waiting time; IVT equals the in-vehicle time; W2T* represents the walking time for transfer; Wa2T* is the waiting time for transfer; IVT2* equals the in-vehicle time for transfer; Fare is the fare cost; and Fare2 is the fare cost of transfer. For automobile travel:

Total Cost (in dollars) = pVT x [DT + KT] + pCM x DD + KC (6)

where pVT is equal to the value of time parameter; pCM equals the cost per mile parameter; KC is the parking cost; DT is the drive time; KT represents the parking penalty time; and DD is the drive distance.

Examination of the simple logit model reveals congruous elements in the deterministic portion of travel cost utility, and then a probabilistic error as the utility bias due to excluded variables. One important note regarding the treatment of the logit model exampled in this review is that the project takes an off ramp after solving the $U_x$ result for each mode. The project does not take the next conventional step, to address the relative probabilities that one mode will be chosen over the other. That step does not suit the purpose of the application. In the basic logit model, the set of weighted coefficients in $\boldsymbol{a_i}$ represent MWCOG parameters for walking (pW), waiting (pWa), in-vehicle time (pIV), and the value of time (pVT) per socio-economic characteristics of the individual, as well as the fuel cost per mile (pCM) for driving – all assigned to known costs $\mathbf{X_i}$ of a given travel mode: path length, waiting time, walking time, transfer time, parking time, driving distance, and driving time. This alignment in methodology establishes the modal cost equations used in this project and has become influential toward the prospect of a

long-range plan of expanding the application into other geographic regions. Chapter 4 discloses the additive cost equations used in this project, as well as each authoritative parameter applied.

The self-selection dilemma illuminates the fact that the network-routing application could also be a survey tool for building a database of results generated by participating users. Self-selection erodes the randomness of observations in travel modeling. In the project, future samples are treated as entirely random with commuter's choice responses invoked by performance metrics of the built environment by traffic analysis zone. Per the findings in self-selection research, this spatial scale which is smaller than a zip code and larger than a household is ideal for capturing correspondence of the built environment to the travel activity space.

Spatial autocorrelation is criticized as rendering observations less independent when nearby values tend to have similar values. The project does not use indicators of spatial autocorrelation, and the project's services promote the independence of observations through support of individual user sessions on a web application. The consumption of the service endpoints in a web application, however, eventually presents new opportunities for a wide range of spatial analyses on travel behavior in metropolitan Washington D.C. – including hot spot analysis and other indicators of spatial autocorrelation. The ability to capture the commuter's experiences geographically in a database enables future researchers to gather independent and random samples from the traveling population. Furthermore, the problem with trip inter-dependency and the critique of trip-based FSM brings forth the idea that individual OD trips could be chained together into a tour by the moderately savvy or well-informed user. Largely, this issue cannot be directly addressed within the initial release of the application, which is trip based. However, the OD routing tool does allocate transfer costs and at intermediate bus stops and rail stations, and the transit routing algorithm is multimodal – where the shortest cost path is

geolocated over pedestrian, bus, and rail modes. At a minimum, this accounts for some of the tour-based factors behind travel behavior.

Then, the MAUP, as the fourth identified problem, draws attention to the pervasiveness of spatial scale as an amplifier of other methodological issues. As the research consistently points to the value of these data at a disaggregate spatial scale, it was decided in the project that the travel events would remain distinct when collected in their tables with no geographic aggregation. However, the project's internal data processing of traffic metrics does invoke some level of ecological fallacy, a concept related to the MAUP. Travel times on linear segments are converted from travel speeds aggregated within standard US TAZ. These polygons are relatively small areas and are established by authoritative methods for estimating linear traffic characteristics therein. Nevertheless, this spatial translation is a simplification in the project's data model that infers traffic activity on linear facilities using areal units. Section 6.2 discusses this project limitation in more detail. Furthermore, the parking penalty parameter does combine labor force percentage at the county level with population density by census block group. This may create moderate inconsistencies with parking penalties. The outcome is that county-level employment percentages act as a scalar factor against block-group population densities to nearly satisfy the regional standard. Yet, the author finds that the proportional differences in parking costs are still sufficient to inform the user where parking congestion restrains automobile path routing. Despite all potential anomalies in travel and parking times, at a reasonable level of data abstraction, this initial release of *Commute GeoCalculator* services still delivers representational patterns of general travel conditions.

The reporting requirements of MAP-21 contribute other valuable insights for what is pertinent to the content of a potential public survey application that would use the project's

services. The working concept within the current project is uniquely designed to assist in fulfillment of these requirements. Decidedly, such a web application would not present as a census-like survey but rather as an interactive toolset that is providing immediate benefits to the user. From the perspective of the transportation authority, only what is necessary and constructive about the user's decision-making process should be included. Finally, the findings about ABM logit modeling, state dependence and expectation feedback only serve to enhance the interactive workflow design of commute routing tools with a survey component. Early in the project design phase, this context of dependence and expectation feedback introduced the prospect that the current project should eventually operate as a kind of intercept survey for gathering informed mode choice data on-the-fly, while the application user is commuting.

The investigation of related applications assisted with targeting a position for *Commute GeoCalculator* in the transportation management space. Reviewing the most popular traffic and navigation applications brought forth the realization that no mainstream web or mobile GIS is providing users with a simultaneous view of travel alternatives with their respective costs in a complete state. The author believes this to be a gap in the market for travel information systems that the current project can potentially fill. A few of the region-specific, open-source applications provided insights for the user-driven workflow and where survey questions should be presented. First and fore-most, the commuter must approve the sharing of their data before any survey questions are asked. Given approval, the commuter-user experience should then be a balance of "give and take" with the initiatives of the transportation manager role. One receives the desired commute information, then one can give something back to a brief survey as a contribution.

# Chapter 3 Data, Networks, and Databases

As noted in Section 1.4, a substantial amount of data processing occurs before the service user visits the REST endpoints and enters their parameters to plot commute events for automobile, pedestrian, bus, and rail modes. Data acquisition to initiate and maintain the application requires a focused effort, to gather specific data sets with most release dates coordinated within a predetermined 18-month timeframe. Construction of the data tier begins with the creation of SDE (spatial database engine) connections for two primary enterprise geodatabases for the project, named "staging" and "static", residing initially in the project's sandbox environment that is later described in Section 6.2.

In this initial project phase, spatial source data are staged in a local file geodatabase while non-spatial data are setup in the "staging" remote database; all of which undergo remove-and-replace manual loading. The *metrics* data model represents the cost data of each travel mode in the end result, composed of automobile and transit feature datasets housed within the "static" database. The *commuters* data model, planned for future development, will store users' mode choice responses to the dynamic cost returns in a future "modular" database. These databases are designed for perpetual growth of the service area, and the data management procedures for loading source data are for the time-being manual. However, once all source data are loaded to the "staging" database, ETL automation developed as part of this project handles nearly all data crunching – up to the last step requiring a manual build of two network datasets using the Esri Network Analyst extension. Network datasets are created from source features which include simple features (lines and points) and consist of edges and junctions that are transformations of the source features (Esri 2019). This workflow in Tier 1 is essential for refreshing the *metrics*

data model and making the data usable by the server object extension (SOE) in Tier 2. Figure 4 summarizes this end-to-end construction process.

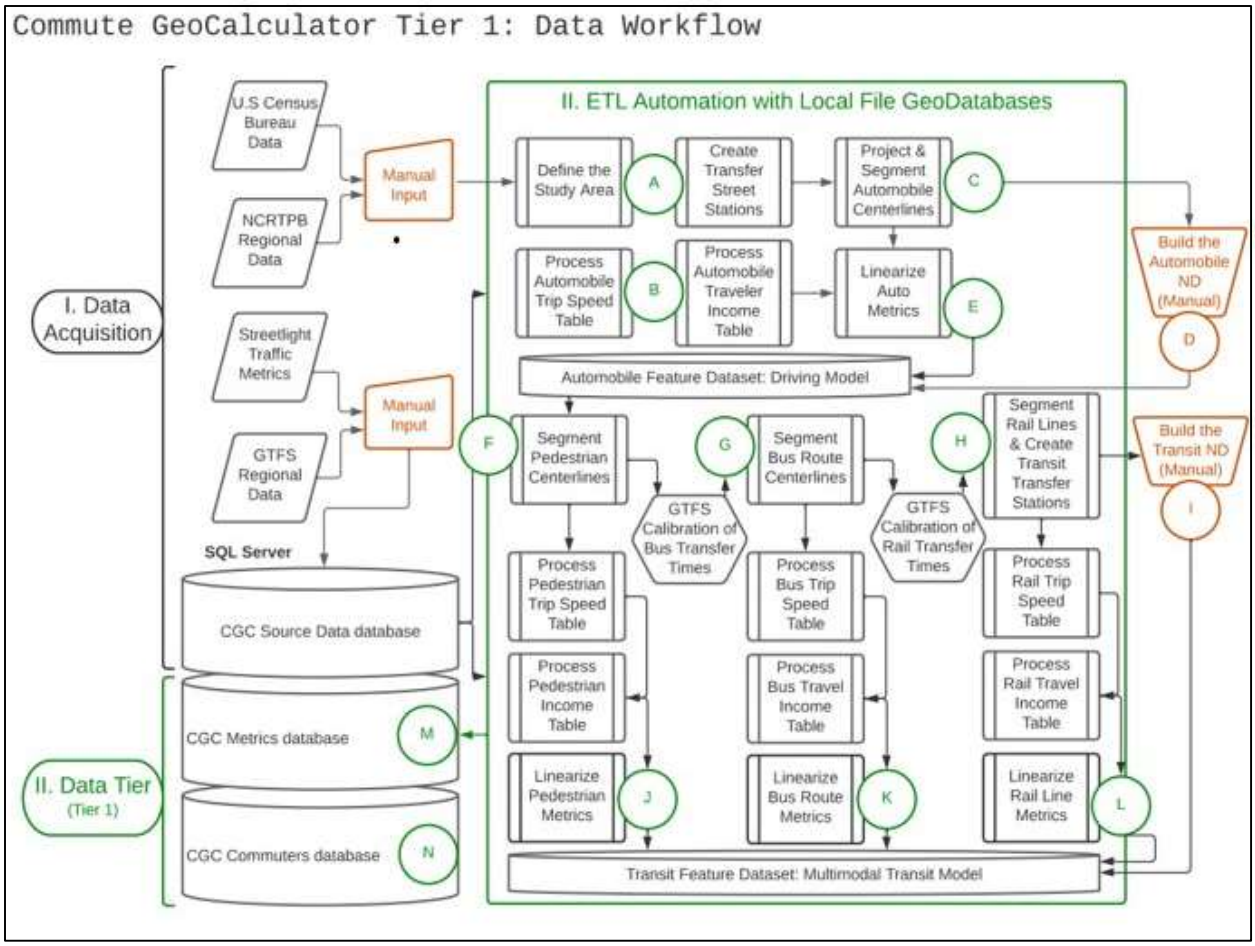



Figure 4. Tier 1 Workflow

## 3.1. Data Requirements

The primary objective in Tier 1 is to provide a data model that, for any given commute, enables attributed route traversals that represent the average travel costs of transit versus driving at different times of a weekday or weekend. The secondary objective is to provide a storage database for the mode choice data returned in the responses from application users. A standalone procedure is implemented to create two spatial tables for the anonymous commuter profile and the routes included in each commute. To achieve these objectives, the data model integrates eight

categories of required source data: 1) roadway, pedestrian, bus and rail centerlines; 2) bus and rail stop points; 3) bus and rail transfer wait times; 4) travel speeds for automobile, pedestrian, bus and rail modes; 5) traveler household income for automobile, pedestrian, bus and rail modes; 6) parking penalties based on employment densities in census block groups; 7) parking fees and transit fares estimated from transportation operators, and 8) authoritative coefficients for factoring fuel cost per mile, residual waiting and walking, as well as the traveler's overall value of time. The strategy in applying these source data governs the requirements and is hinged upon the routing function of each network dataset.

Transport facility centerlines must carry sufficient attributes in length, parking penalties, transfer wait times, and mode hierarchy to support routing that produces the most efficient path for each travel mode. These are input attributes required in each network dataset. By default, the Esri Network Analyst extension applies a shortest-path routing algorithm by length. From here, several options are available for adding travel impedance and preference to a network. Three such options are utilized in the network datasets – cost, restriction, and hierarchy attributes. Therein lies the requirement for the parking and transfer time costs, as well as the preference and hierarchical ranking for pedestrian, bus, and rail travel modes.

The other side of the data strategy concerns what happens after each network path is solved and rendered into geometry. Quite simply, the application requires total cost attributes along with each path geometry. In addition to length, parking penalties and transfer times, the total costs in minutes and dollars require travel speeds, parking fees, and transit fares, as well as the authoritative coefficients and traveler household income to calculate the overall value of time. These additional attributes are spatially extracted from processed feature classes using the path

geometry before the necessary cost variables are calculated. The most important output calculated in Tier 2 is the total travel time based on individual segment lengths and speeds.

## 3.2. Data Acquisition

Complete trip speed and traveler annual income are particularly difficult to acquire from open-source data by peak and non-peak travel times. For this reason, SL was called upon to provide these critical metrics in the study area, and this request was fulfilled via provisioning of a one-year academic license to their StreetLight Data InSight® platform. The traffic analysis tool by SL provides an array of selectable, reusable zones by which various forms of traffic measurement may be obtained, as depicted in Figure 5.

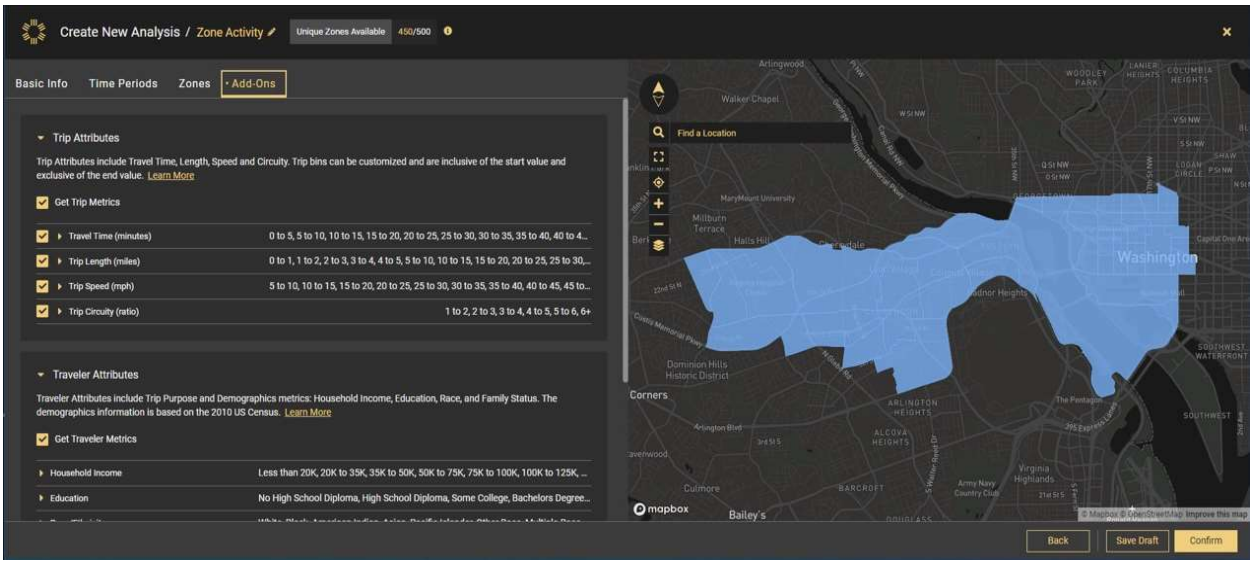



Figure 5. Setup, Analyses and Extraction of Traffic Metrics from SL Data InSight®

This software allowed the author to analyze and extract the required travel speeds and inferred traveler income within each traffic analysis zone from March to April 2019 and October to November 2019 data to support pre-pandemic annual average traffic flows. As briefly presented in Section 1.4, Table 1, these traffic-related data are organized into six day parts for each day type – weekday or weekend. These include all-day averages (12am – 12am), early AM

(12am – 6am), peak AM (6am – 10am), mid-day (10am – 3pm), peak PM (3pm – 7pm), and late PM (7pm – 12am) travel periods. OpenStreetMap is the underlying source for the road network, rail lines, and pedestrian paths to which all travel metrics are locked then loaded to TAZ polygons. For the project, SL metrics are downloaded as tables of zonal data with associated shapefiles, and non-pass-through traffic flows by standard US TAZ polygons are specified. This means that traffic detections via cellular GPS and LBS are limited to trips which begin or end in each selected zone. Yet, the SL measurements of average travel times follow the full extent of each trip, across multiple selected zones. Therefore, SL average travel times are useless to the project because the commuting user of the *Commute GeoCalculator* specifies his or her own trips. However, the non-pass-through option in SL Data InSight® is utilized because it isolates average travel speeds and inferred traveler income to each individual zone (StreetLight 2022). These metrics provisioned in TAZ polygons establish an objective spatial unit by which user-defined travel times and inferred incomes may be calculated in the application.

There is an additional matter to settle with the robustness and accuracy of the transit travel indexes. In 2020, an FHWA-sponsored study was conducted by the Virginia Department of Transportation (VDOT) to determine guidelines for using SL indexes in transportation planning. While their literature review points to effective yearly OD travel patterns from SL indices in the WMCOG area, the researchers conclude that simply relying on these indexes will not produce robust and accurate results. Through stratified random sampling, they found that integrating authoritative data sources with SL indices effectively addresses this issue (Hong, Cetin, and Ma 2020). For this reason, GTFS trips, stops, and stop-times tables were extracted from the study area in the Spring and Fall timeframes of year 2019. As reported by SL, this is sufficient to provide reasonable traffic averages that can represent the entire year. The results of this quality

control process are transfer locations, transfer times and the probabilistic error for travel times applied to the transit cost equation.

Table 2. Project Source Data

| Organization | Source/Dataset | Coefficient Weights | Trip Variables | Analysis Layers |
|---|---|---|---|---|
| MWCOG | MWCOG Model | ● | | |
| NCRTPB | NCRTPB Model | ● | | |
| Transportation Research Board | NCHRP Report 716 - Model D | ● | | |
| U.S. Census Bureau | 2020 Pct Employment | ● | | |
| U.S. Census Bureau | 2020 Population Density | ● | | |
| OpenMobility Data | 2019 GTFS Calibration Data | | ● | |
| StreetLight Data, Inc. | 2019 Transit Service Indexes | | ● | |
| StreetLight Data, Inc. | 2019 Road Trip Volumes | | ● | |
| StreetLight Data, Inc. | Traffic Analysis Zones | | | ● |
| U.S. Census Bureau | 2019 TIGER/Line Road and Rail | | | ● |
| NCRTPB Data Clearinghouse | 2022 Metro Rail Lines | | | ● |
| NCRTPB Data Clearinghouse | 2022 Metro Rail Stations | | | ● |
| NCRTPB Data Clearinghouse | 2022 Metro Bus Stops | | | ● |
| NCRTPB Data Clearinghouse | 2019 Metro Bus Lines | | | ● |
| NCRTPB Data Clearinghouse | 2022 Pedestrian Trail Network | | | ● |
| Esri | ArcGIS Network Analyst | | | ● |

The parking penalty values depend on the population density in each census block group multiplied by the percentage of the civilian labor force (Table 4.). The block group polygons are acquired from Living Atlas data and selected by spatial intersection with the service area. Employment density is acquired from the US Census 2019 quarterly workforce indicator (QWI) dataset to calculate parking time penalties. Parking fees and transit fares are estimated from the websites of transportation authorities and operators, respectively. By different methods, all of these travel metrics are assigned to US Census Bureau TIGER/Line features as well as NCRTPB authoritative line and point features. Thus, each of these linear data are part of the full

acquisition, summarized above in Table 2. Once these data are acquired and formatted into tables and feature classes, the next concern is geoprocessing workflows via Python scripting. Here, Esri Modelbuilder in ArcGIS Pro is leveraged to diagram key geoprocessing tools and to conveniently export lengthy, complex function calls in Python that would otherwise consume additional hours of development time (Zandbergen 2013).

## 3.3. Data Preparation

Tier-1 ETL automation integrates all source data into two models, each representing the form and function of driving or using transit. All data model inputs are projected into WGS 1984 Web Mercator Auxiliary Sphere before any geoprocessing steps are applied. These steps are explained in sequential detail through this section following the data processing strategy, stated in Section 3.1, as an approach hinged upon the routing function. In short, the source data are first transformed into linear segments carrying parking penalties, transfer wait times, and the fundamental structure that will support the routing functions needed in the automobile and multimodal transit network datasets. Then, these same input segments are used to prepare linear trip speeds and traveler incomes that are to be spatially extracted by the geometry outputs of the routing functions in Tier 2.

The first ETL task is preparing the census block groups for assignment of employment densities to centerlines. Census block employment density is the first zonal attribute that is attributed to the centerline streets layer, the US Census TIGER/Line features, commonly used to build street networks (Butler 2008). Parking penalties are assigned to the transfer streets according to Table 3, wherein employment density is the population density in each block group multiplied by the percentage of employment in the associated county, as shown in Table 4.

Table 3. Parking Penalty Time (NCRTPB 2020)

| Employment Density Range (Emp/Block Group) | Parking Penalty (Minutes) |
|---|---|
| 0 - 4,617 | 1 |
| 4,618 - 6,631 | 2 |
| 6,632 - 11,562 | 4 |
| 11,563 - 32,985 | 6 |
| 32,986 + | 8 |

Table 4. Percentage of the Civilian Labor Force in WMCOG (US Census 2020)

| County or District | Percent Civilian Labor Force in Population (2020) |
|---|---|
| Arlington County | 77.00% |
| District of Columbia, Washington | 70.20% |
| Fairfax County | 70.20% |
| Prince George's County | 70.90% |
| Montgomery County | 70.50% |

Figure 6 illustrates how census block groups are selected by these cyan-outlined polygons for a project service subarea, zone set 1. The next zone set of census blocks is added, which encompasses the remainder of the study area. TAZ polygons define the study area, and the incremental logic in the initial ETL procedures sets up the *metrics* data model for future growth. Here, zone set 1 is the initial subset of 50 TAZ polygons and zone set 2 is the collection of the 450 remaining TAZ polygons that together cover the central metropolitan Washington D.C area. In the next geographic expansion of *Commute GeoCalculator*, the current study area will be designated as zone set 1 – to then be merged with the next zone set 2 of surrounding traffic analysis polygons, and so on. Additional zone sets will not necessarily need to be contiguous, and the source of traffic data may change over the application's life cycle.

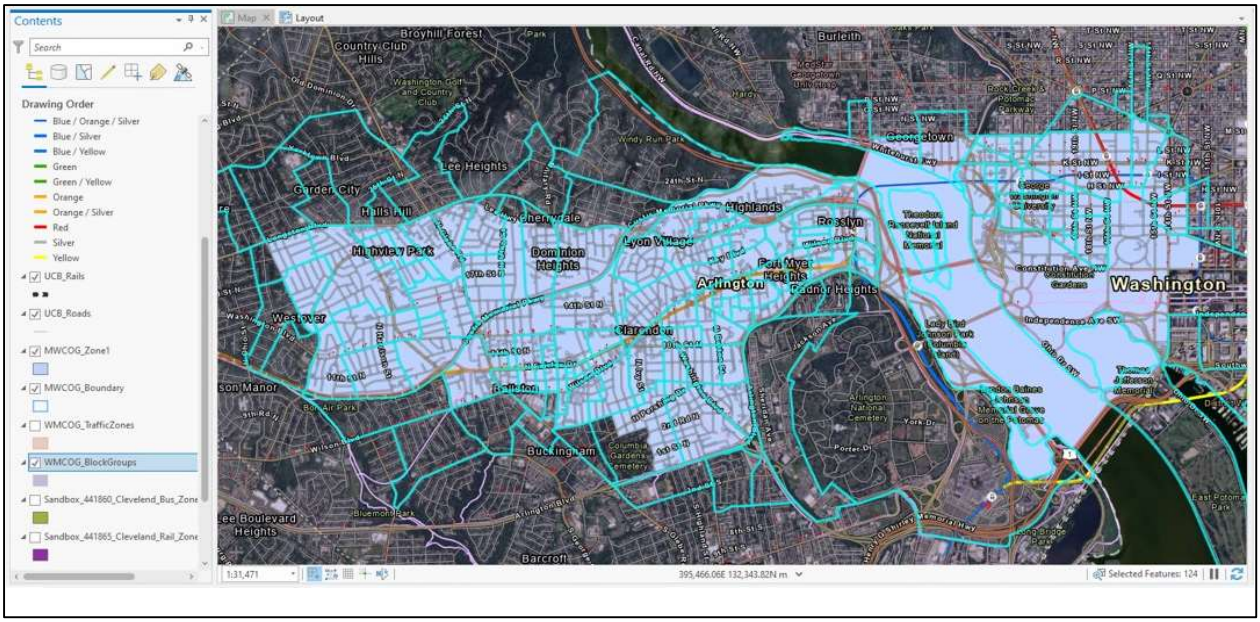



Figure 6. Export of TAZ Dataset to ArcGIS Pro® and Spatial Selection of Census Block Groups

For transfer street stations, road centerlines are clipped and assigned employment densities by census block group, so that parking cost and penalty time can be assessed. The basic geoprocessing steps for transfer streets are shown in Figure 7. The parking penalty values depend on the population density of US Census block groups and the percent of the civilian labor force in year 2020. The author did not find this percentage of employment statistic at the block group level, but instead captured this data at the county level. The census block group polygons are clipped to the study area and attributed with the employment density field based on Table 3. Employment density is the population density in each census block multiplied by the percent of employment in the containing county:

Employment Density = Population Density x Percent In Civilian Workforce (7)

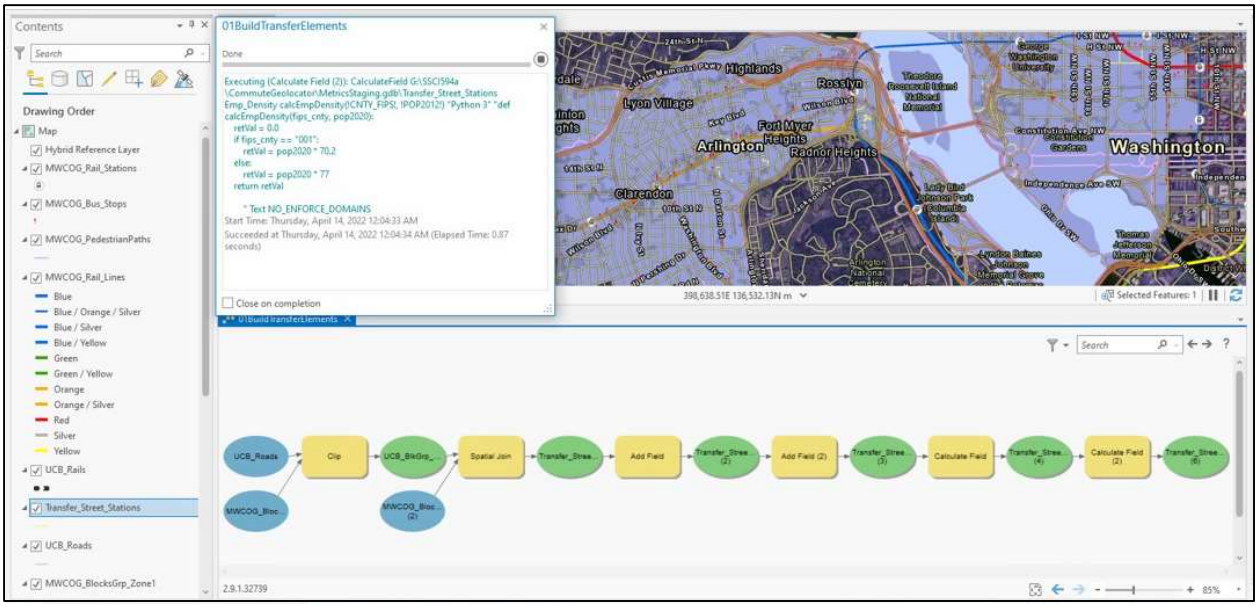



Figure 7. Road Centerlines Clipped and Assigned Census Block Group Employment Density

Next, SL metrics are written to each standardized TAZ, but in tabular fields that must be joined back to the originating polygons by a unique common zone name, as shown in Figure 8. This is done to enable spatial assignment of volumes and indexes to traversing roads, rails, walkways, and bus stops. For each day part, SL trip-speed and traveler-income ranges are provided in percentages of travelers per zone. To each road segment, ETL applies the total probability of average speeds per TAZ for each mode as exampled in Table 5.

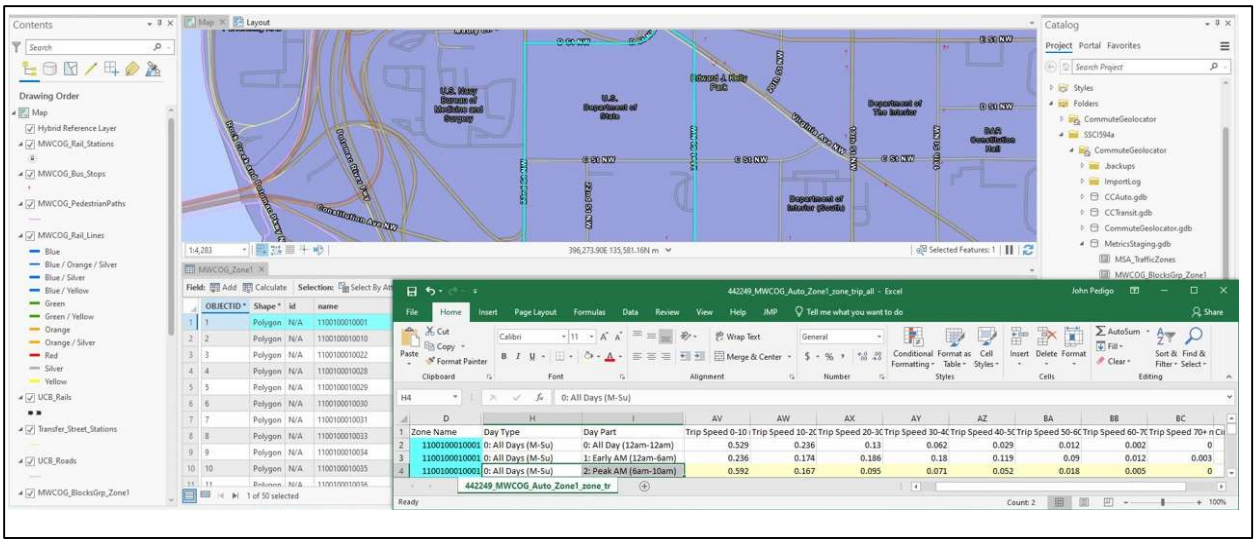



Figure 8. TAZ Traffic Metrics Applied to Roads, Rails, and Pedestrian Paths

The Law of Total Probability simply states that if there are *n* number of events in an experiment, then the sum of the probabilities of those *n* events is always equal to 1. StreetLight proportions average speed and traveler income events by measured range, within each TAZ. To calculate average travel times for each segment in a TAZ, the forementioned Newtonian expression applies. Each unique road segment in these polygons has a distinct distance to be traversed. Hence, the average travel time for each segment is: travel time = distance / trip speed.

Table 5. Total Probability of Average Speed per Traffic Analysis Zone – Example

| Peak AM Travel (1 Traffic Analysis Zone) | Probability (P) | Speed Interval (i) | P(i) = Average Speed |
|---|---|---|---|
| Trip_Speed_0_10_mph_percent | 0.623 | 5 | 3.115 |
| Trip_Speed_10_20_mph_percent | 0.161 | 15 | 2.415 |
| Trip_Speed_20_30_mph_percent | 0.090 | 25 | 2.250 |
| Trip_Speed_30_40_mph_percent | 0.065 | 35 | 2.275 |
| Trip_Speed_40_50_mph_percent | 0.041 | 45 | 1.845 |
| Trip_Speed_50_60_mph_percent | 0.017 | 55 | 0.935 |
| Trip_Speed_60_70_mph_percent | 0.003 | 65 | 0.195 |
| Trip_Speed_70_above_mph_percent | 0.000 | 75 | 0.000 |
| Total Probability of Average Speed: | | | 13.03 mph |

Given the research findings on the use of SL metrics, by researchers cited in Section 3.2 (Hong, Cetin, and Ma 2020), the project directly applies SL trip-based measurement of automobile traffic volumes, as they are based on actual counts taken by VDOT (StreetLight 2022). Then, transit indexes for bus and rail traffic receive calibration by GTFS data from local agencies in the base year, 2019, so that feature location and transfer times may be improved for accuracy. The GTFS “stops” file anchors the index calibration process by its geographic coordinates with arrival and departure times for bus and rail. In Python ETL code, tabular GTFS durations for both travel time and stop times are sequenced and linearized to road and rail lines by two separate algorithms. The first method addresses locational accuracy and completeness of

bus stops and rail stations, as well as the bus routes that provide connections to rail service lines. The second method builds upon these results, to calculate the average transfer times applicable to each stop and time range. The program loops through each individual bus and rail route, one stop at a time, to query and compare GTFS arrival and departure times between service routes by each day-part range (early AM, peak AM, mid-day, peak PM, etc..). Using this approach, the program is able to calculate the average transfer wait times at each stop on the project's temporal scale. Combined output of the two methods forms an intermediate lookup table of stop locations with average transfer wait times by day part, for bus and rail respectively. From the perspective of source GTFS data, the basic logic behind the calibration process is depicted in Figure 9.

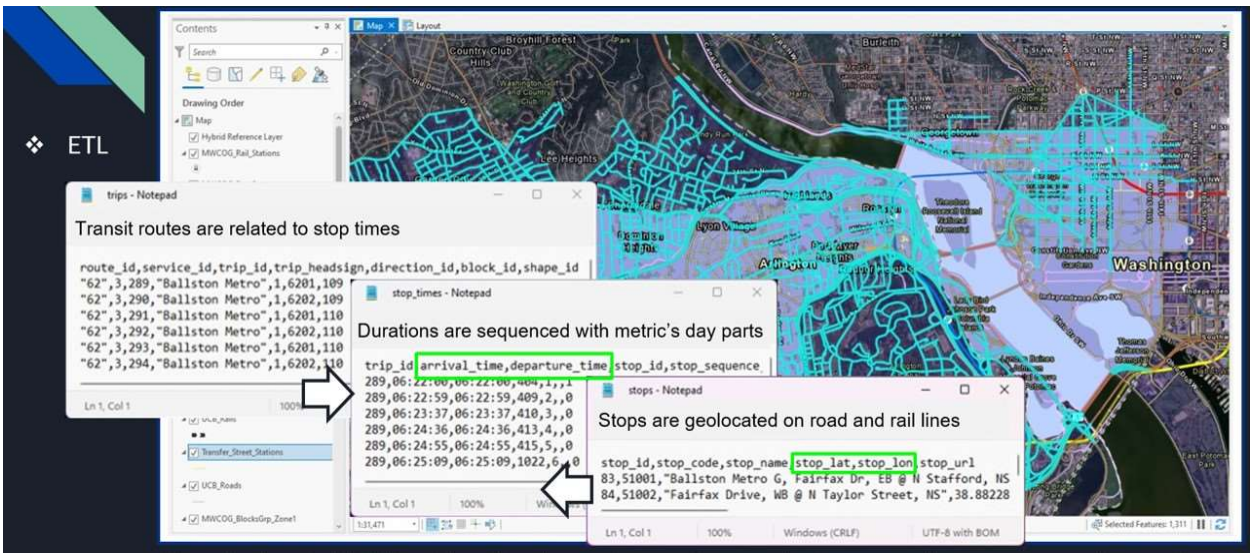


Figure 9. GTFS Calibration Logic

Each lookup table in the above process is used in the calculation of geolocated transfer times, including the all-day average which is applied as the cost evaluator for bus and rail routing in the multimodal network dataset. But these lookup tables serve an additional purpose outside of the transit network dataset and the routing parameters therein. They are used in the construction of attributed relationship classes that tie each stop to its linear segment of all transfer-time day

ranges, as well as all trip speeds and traveler incomes on the same temporal scale. The purpose of these relationship classes is performant and accurate retrieval of stop point and route segment data by the SOE.

Regarding transit fares, the GTFS providers provision website addresses where published rules on fares may be found. To the extent possible, these were manually processed from the transit operators in the cities of Alexandria, Arlington, and Woodbridge; counties of Fairfax, Montgomery, and Prince George; states of Virginia and Maryland, as well as WMATA metro transit authorities. Average fares are included in the bus and rail stop point feature classes, as per these providers. After the above steps are completed, route segmentation and the linearization of prepared data begins with highway centerlines because this is the predominant transport infrastructure in most urbanized areas. This is achieved using the Python arcpy intersect analysis tool on the US Census centerline feature class, itself, at a 0.003-mile tolerance (approximately 16 feet). The resulting road intersection point features are then clipped by the TAZ polygons comprising the study area. From here, the intersection points of the study area are used to split the centerline features by the same tolerance, and those linear results are then clipped to the study area as well. The last step in this subprocess is to spatially join the TAZ zone identifier to each clipped segment. Instead of simply trying to apply a clip tool directly between raw centerlines and TAZ polygons, the applied method allows for clean segmentation without errors that is sufficient for building each travel-mode network and linearizing associated trip speeds and traveler incomes.

The automobile network inputs include the segmented highway centerlines, providing facility length and street connectivity, and the linear transfer street stations which supply the parking penalties in minutes. In the forthcoming network dataset, each of these inputs will be

assigned to a connectivity group with routing connections set to the endpoints of each line segment. US Census TIGER/Line highway features are segmented at roadway centerline intersections (Her and Yu 2021), and so at this point there is no need to take further steps in ETL for these automobile network inputs. However, the last section of this chapter presents the final results of data preparation which includes all content that is internal and external to each network dataset.

Segmentation for the multimodal network dataset builds directly upon the results of highway segmentation with a few additional steps that vary among pedestrian paths, bus routes, and rail lines. During this procedure, a hierarchical structure is established in merging these linear features with highway system features, and a corresponding hierarchy is assigned to each transit mode. A preset rank number identifies each transit mode preference, and thus configures the routing function to predetermine the portion of trips made by each transit mode – the transit mode split. This is a simplification to the data model that will be replaced by user-defined preferences in a future release, but it does facilitate production of a rational shortest-cost path for transit.

The creation of the hierarchical structure in the multimodal transit network dataset starts with a spatial union of pedestrian paths with the clipped highway network. The resulting pedestrian network represents every linear facility where a traveler may walk. From here, the next mode layer applied to the structure is bus routes. A spatial intersect between the pedestrian network and bus routes is used to create the bus network. In turn, a spatial intersect between the bus network and rail lines is used to create the rail network. The outcome includes three distinct sets of network inputs, in which all point and line data are ideal for configuring the connectivity groups that tie each mode of transit together into a multimodal system (Esri 2019). This system

architecture is constructed by ETL during data preparation stage and sets up the build of the multimodal network dataset.

## 3.4. Network Datasets

Although Esri ArcGIS Pro 2.9 and Python 3 are used in data analysis and the construction of the entire ETL automation, each network dataset is manually built using ArcGIS Desktop 10.9 software. The network datasets are created with ArcDesktop software because this initial release of the logic tier depends upon ArcMap-based runtime services and the ArcObjects software development kit (SDK) for the .NET framework. To transition this linear referencing middleware over to an ArcGIS Pro build in an enterprise environment operating above ArcGIS Server 10.9.1, all SOE code will have to be updated to use the ArcGIS Pro Runtime API and ArcGIS Enterprise SDK in a future release. The reasoning behind this transitional design approach is based upon the ongoing pervasive use of the ArcDesktop platform across the transportation industry, particularly in the public sector. The author finds sufficient benefits to introducing this set of REST tools on a platform that is still widely used, so that subsequent publications may serve as reference for successfully migrating similar linear referencing tools.

In the project, the network datasets provide the vital function of routing. Configuring the automobile dataset is relatively simple compared to the multimodal transit dataset. In either case, the configuration components that provide realistic routing outcomes include the input vector data (points and lines), connectivity, and attributes. Once all data preparation is completed via ETL automation, specific input point and line source features are applied in the Network Analyst configuration wizard, as shown in Figures 10 and 11. It is important to reiterate that while all source data are processed in the Web Mercator Auxiliary Sphere, WGS 1984 datum, they are published to each respective data model as unprojected GCS in NAD 1983, and this applies to

each network dataset. Again, this is done to assist with uniform routing and to extend the option of any user-defined spatial reference at the service endpoints.

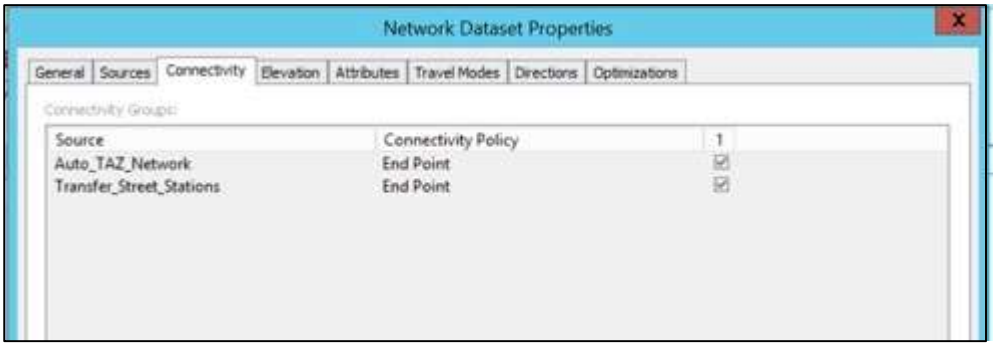


Figure 10. Connectivity in the Automobile Network Dataset

How network elements connect depends on which connectivity groups the elements are in. For example, two edges created from two distinct source feature classes can connect if they are in the same connectivity group. The edges will not connect unless they are joined by a junction that participates in both connectivity groups (Esri 2019). The automobile network and transfer street stations, as depicted in Figure 10, are set to connectivity by the endpoints of each edge. For the transit networks, separately defined connectivity groups keep the pedestrian, bus, and rail networks distinct yet connected at shared bus stops and rail stations. Each edge source is assigned to exactly one connectivity group, and each junction source can be assigned to one or more connectivity groups, as illustrated in Figure 11.

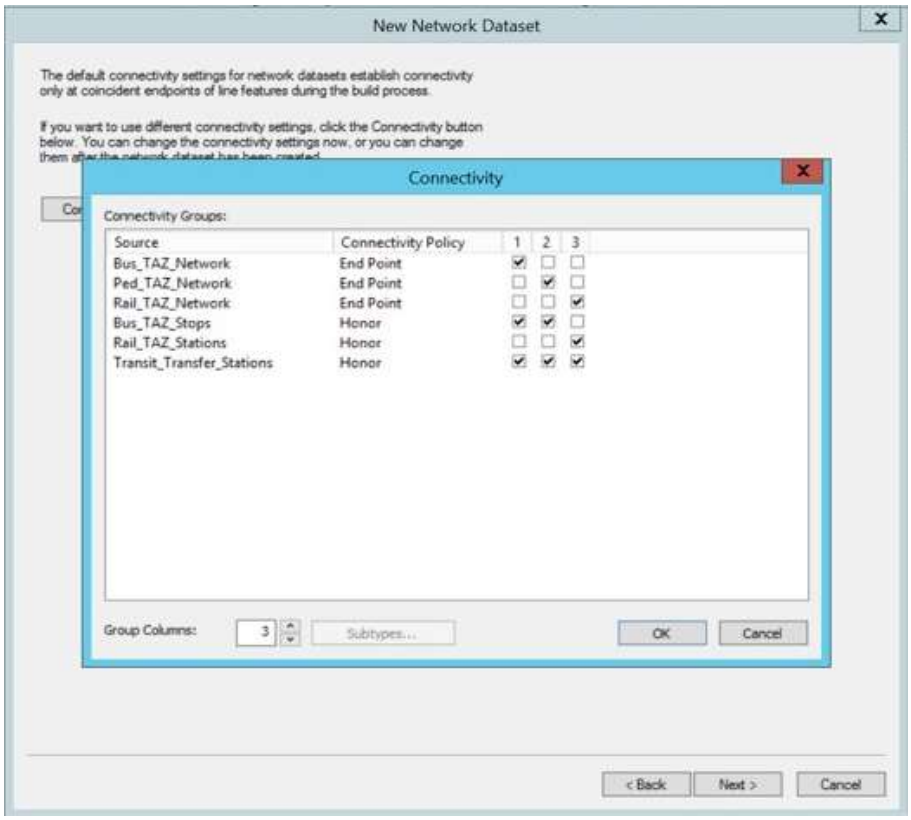


Figure 11. Multimodal Transit Network Connectivity

For performant treatment of network attributes – instead of assigning the dozens of metric fields directly to the network datasets – the wait times, trip speeds and traveler incomes are transposed from the rows of day types and day parts into fields of measurement within layers outside of the network datasets. The MWCOG authoritative model parameters and coefficient weights are also supplied outside of the network datasets, within the SOE code to satisfy the cost equations presented in the previous chapter, Section 2.5 (NCRTPB 2020). The parameters which are configured within each network dataset are as follows. As shown for driving in Figure 12, a default cost attribute is designated to the roadway length and a second cost attribute is assigned a field evaluator upon the “transfer streets stations” layer for the parking penalty time (KT). This is also observed as the impedance value for driving.

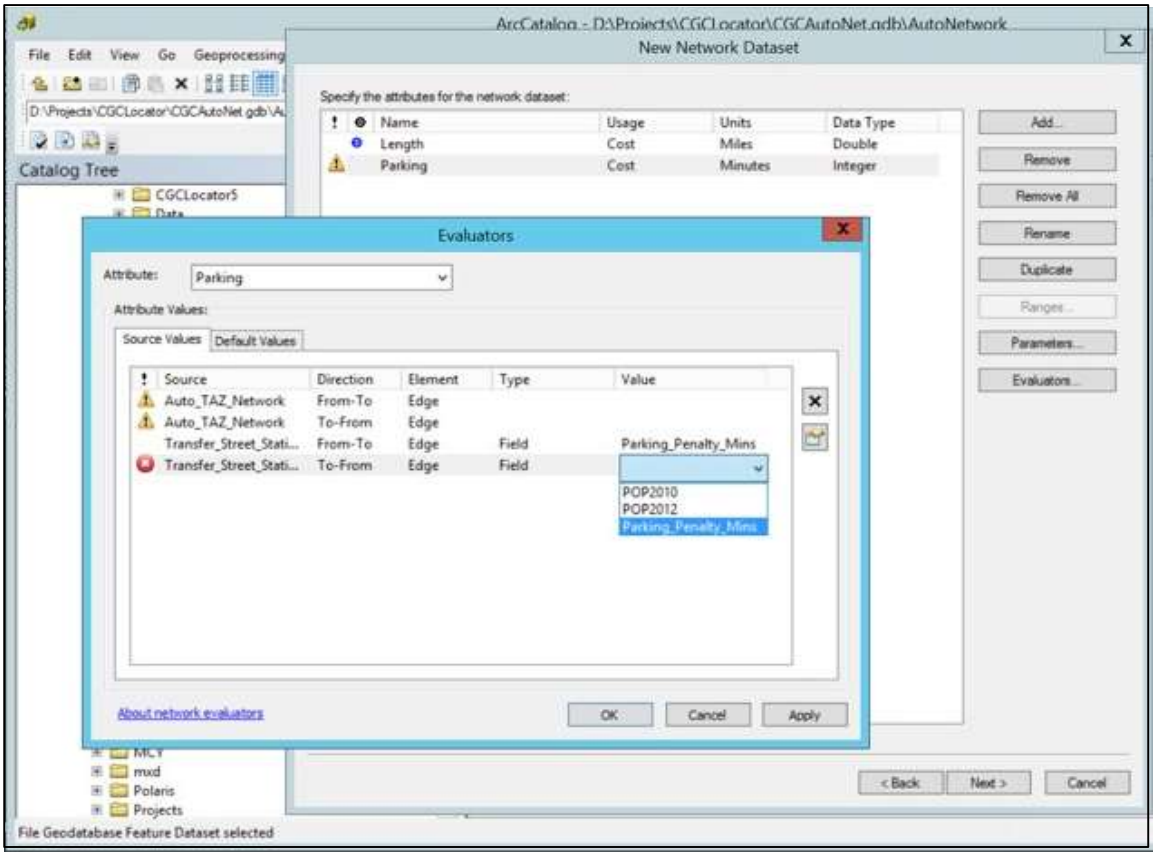



Figure 12. Cost Attributes in the Automobile Network Dataset

In the transit network dataset configuration, the default cost attribute is also assigned to the facility length and three additional cost attributes are configured for the different types of transfer times that impede travel. Bus and rail transfer times are set to field evaluators on the linear bus and rail networks, respectively. Then, the bus-to-rail transfer time is wired up to the "transit transfer stations" point layer, from GTFS data, representing all bus stops and rail stations on service lines that provide transfer between both transit modes. The common field that these cost attributes utilize is the average all-day wait time field, present in each of the above-mentioned layers. This configuration of attributes in the transit network dataset is depicted in Figure 13. The author finds that applying the multiple other wait time fields, by day type and day part, is not necessary for guiding the routing function in this initial release of the project. Instead, this additional parameterization of transfer wait times is performed after routing has solved the path

and all resulting costs are being assessed. It is this post-routing step that, in turn, satisfies the walking time for transfer (W2T*), waiting time for transfer (Wa2T*), and in-vehicle time for transfer (IVT2*) in the applied cost equation.

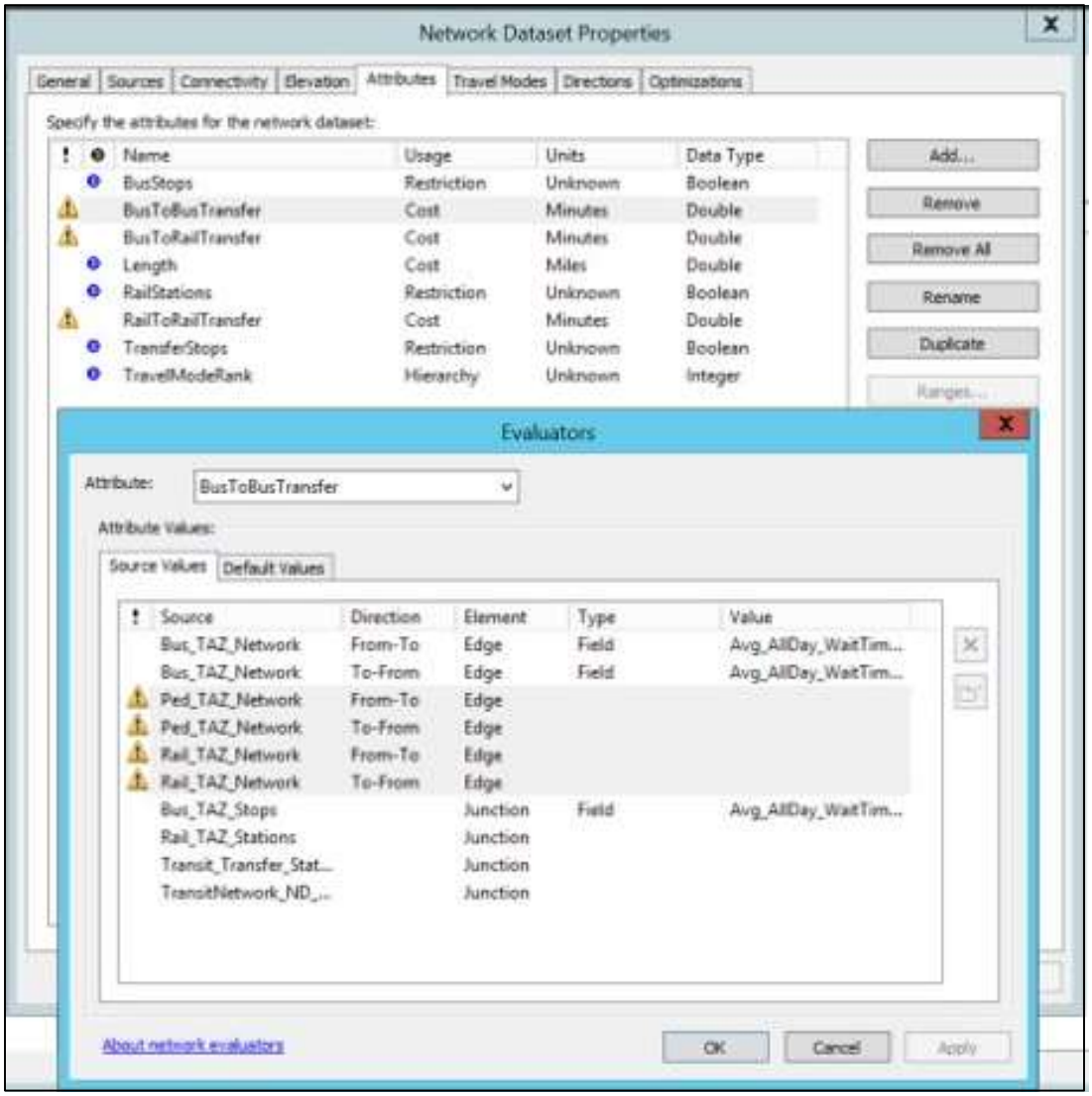



Figure 13. Cost Attributes in the Multimodal Transit Network Dataset

From here, the multimodal transit dataset receives two additional types of network attributes – restriction and hierarchy. There are three restrictions that specify the level of preference for each transit mode at bus stops, rail stops, and transfer stations connecting rail and bus lines. Here, rail stops are granted “high preference”, transfer stations are “medium

preference", and bus stops are set to "low preference". The single hierarchy network attribute compliments this configuration by ranking each transit network centerline in ascending order, from the pedestrian mode to rail travel.

Cost, restriction, and hierarchy attributes assist the Esri Network Analyst shortest cost-path algorithm in navigating across the connectivity of all three networks in a realistic manner. Of course, the assumptions built into these attributes simplify the travel model, but only to the extent that an economically rational path is found. The path geometry is then applied on the project's temporal scale to spatially extract transfer times at appropriate junctions, centerline trip speeds, and traveler incomes as needed. This design approach reduces the complexity of the network datasets and improves overall performance in computing total costs.

## 3.5. Databases

Previous sections describe the manual and automated procedures that construct the data tier of the application. The "staging" database serves as storage of all manually inputted tabular data, in a spatial format that is suitable for geoprocessing. The "static" database contains the end results in the two-part *metrics* data model organized into automobile and multimodal transit feature datasets. These separate feature datasets are containers of each respective model part, and they are required for each network dataset build. The routing configuration for each travel mode resides inside of each network dataset with cost fields that are pertinent to independently solving the shortest-cost path. The Network Analyst software extension creates edges and connecting junctions from the input data and configuration for each network dataset.

For the automobile mode, edges are created from the highway network feature class and junctions from the endpoints of each segment therein. For multimodal transit, edges are constructed from pedestrian, bus, and rail network feature classes, and the junctions are

generated according to the configuration of rail stations, bus stops, and the interconnecting transfer stations. All remaining cost variables, which depend on the solved path geometry, reside outside of the network dataset. The logical diagram in Figure 14 summarizes the complete data model, residing in the "static" database, as designed by the author.

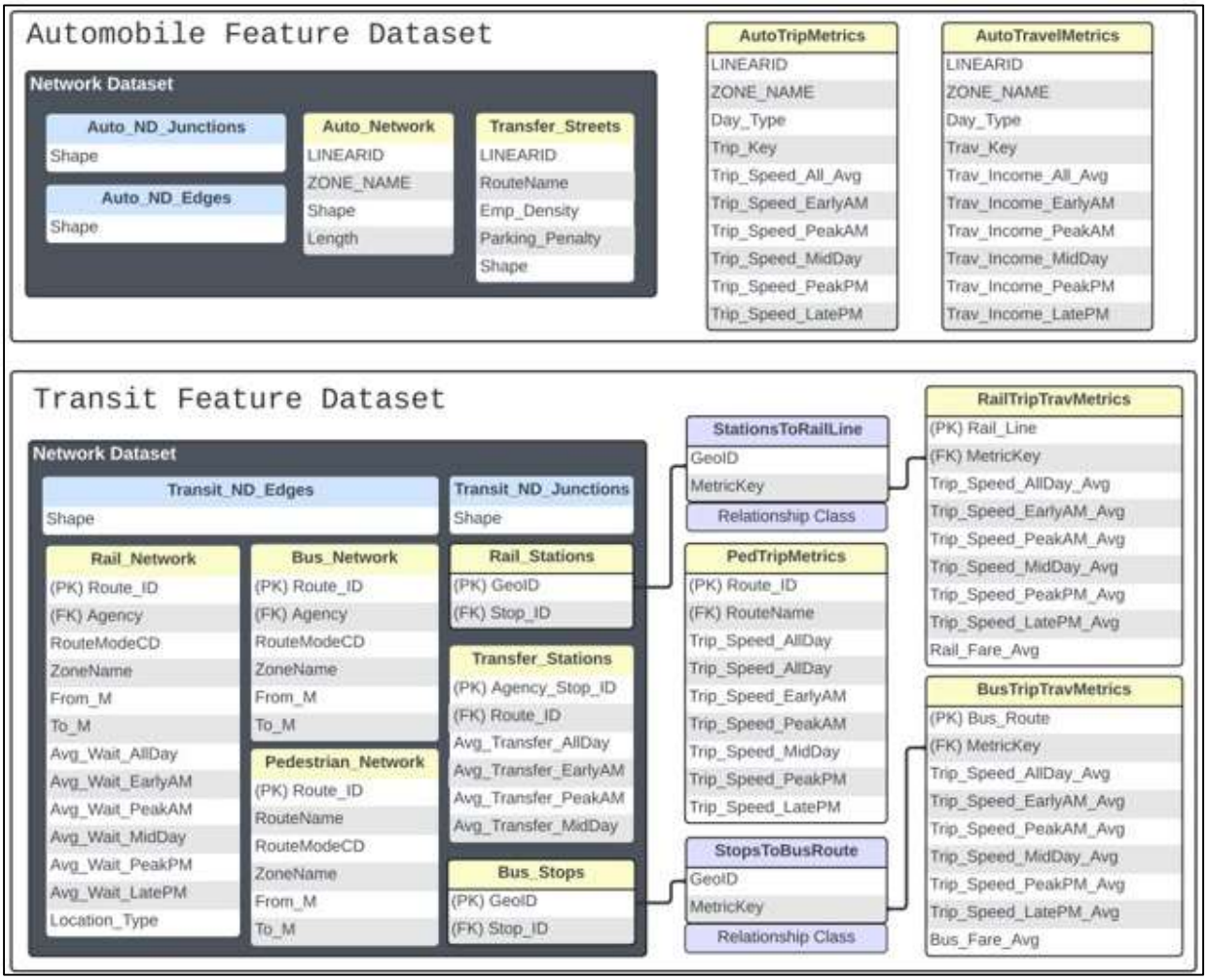



Figure 14. Tier 1: *Metrics* Data Model

Contents of feature datasets must be spatial, and non-spatial tables may not be implemented directly. For example, the bus and rail lookup tables forementioned in the previous section are entered as parameters into one of the tools provided by Esri in the "arcpy" Python library for building relationship classes. This arcpy tool, or function, is called

"TableToRelationshipClass_management()" and this step is performed using a non-spatial lookup table as input and a relationship class as output, for the transit feature dataset. The GTFS-based relationship classes are built within the feature datasets, but outside of the network datasets. Physical relationships can be implemented between feature classes applied in a network dataset with those used externally (Butler 2008). The geographical data that are pertinent to the shortest-cost path are implemented within each network dataset, and the metrics associated with the subsequent costs of traversing that path are installed outside of the network dataset. Two relationship classes link transit stops to the route segments they service, for both bus and rail modes, so that the SOE may quickly locate an intermodal transfer and the costs associated with that transfer.

There is a significant difference in relationship ontology between calibrating pre- and post-routing elements of travel data. In the project, the automobile data model does not exemplify this difference as well as the transit data model because only the latter involves physical relationships. Positional accuracy and transfer wait times of connected transit services are all pre-routing concerns. The logic by which GTFS-based relationships are applied to calibrate these transit pre-routing elements is described in Section 3.3 and Figure 10. From a commuter's perspective, once the most efficient path is drawn for transit and driving, the post-routing concern becomes: "Then how much is this route going to cost?" This is where physical relationships come into the picture, outside of the network dataset and for multi-modal transit.

Attributed relationship classes, with many-to-many cardinality, provide a high-performance and high-accuracy alternative to spatial querying of complex transit networks. Data anomalies caused by combining disparate data sources are circumvented by establishing a coordinate-based unique identifier for each point event (Jetlund and Neuhäuser 2022), in this

case transit stops, and then spatially relating those points to linear trip metrics using distinct natural keys created from the metrics data itself. That is, a unique point "GeoID" is created by concatenating latitude and longitude values in comma-delimited format. The GeoID is then spatially related via background geoprocessing to a distinct temporal collection of linear trip speeds and traveler incomes by a surrogate key field, called the "MetricKey". This relational structure is created over the entire extent of the link-point multimodal network, to assist the SOE with correctly interpreting traversals among the three transit networks.

# Chapter 4 Application Development

Today's busy users expect performance when exerting focus upon the content and functionality of a web application or web service. In this project, back-end geoprocessing in the data tier supports higher performance at each REST service endpoint. This overall development approach provides for a "thin" web client that can potentially operate on a wide range of devices. Furthermore, with respect to scalability of this service-oriented middleware, the decision to leverage higher-level service components of cloud computing instead of out-of-the-box Esri geoprocessing services has enabled support for elastic and on-demand provisioning of networked resources (Issarny et al. 2011). The *Commute GeoCalculator* has a maturity path leading to creation of a scalable and computationally intensive, but thin, web client. This chapter explains the application development steps for a two-tier performant MWAAS toolset, comprising the project's current web GIS, while taking a brief look at how a third-tier web user interface fits into the long-range plan.

In Tier 1, the ETL workflow (or automation) is developed in Python 3, but manual data acquisition was required in order to streamline code development time. Data transformation comprises most of the scripting work, yet the data load modules do handle the build of relationship class structures between the spatial tables to complete both driving and transit data models within two local file geodatabases. An additional load module was added to cleanly overwrite these model results in the remote Microsoft (MS) SQL Server geodatabase. Data results of Tier-1 ETL are registered through an SDE connection to ArcGIS Data Store. The primary map service that facilitates the SOE recognizes this registration as the data source. For the overall project infrastructure, Figure 15 is a summarized view of the current state with added elements of the future state.

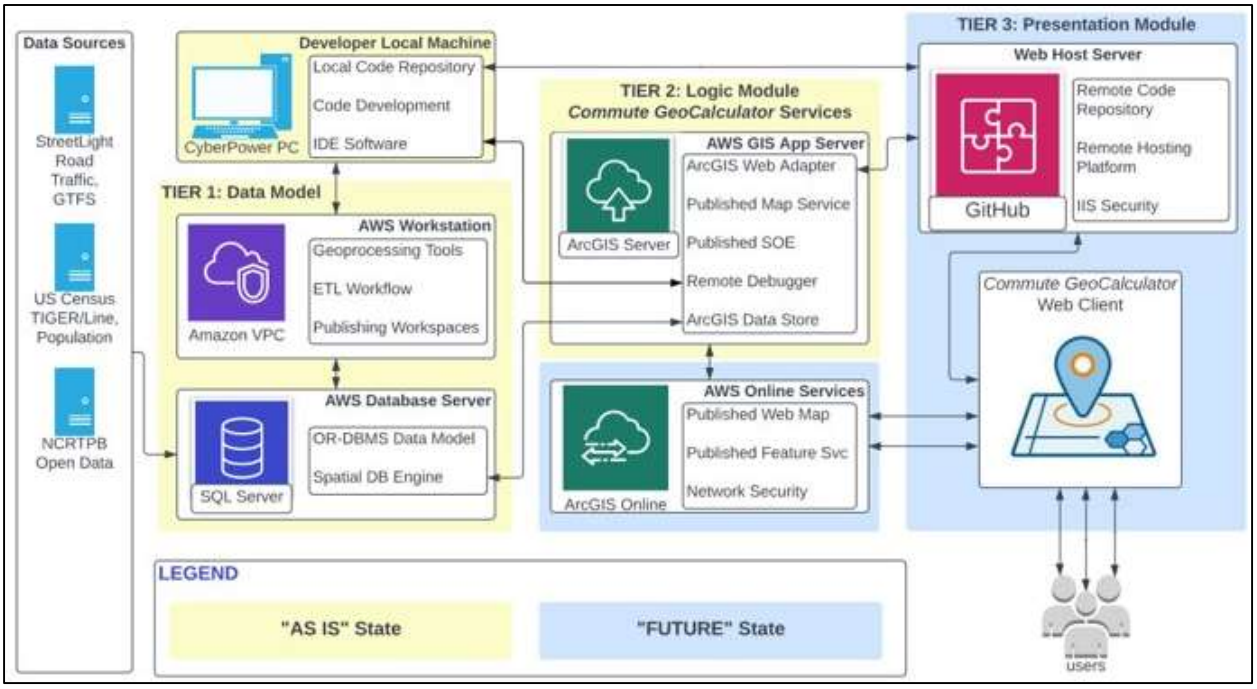



Figure 15. *Commute GeoCalculator* Infrastructure Diagram

## 4.1. Configuring the Application Environments

Tier 1 and Tier 2, as shown in the general diagram of Figure 15, are implemented in two separate locations, one for initial development and demonstration purposes and the other for future development and production deployment. For initial development, a sandbox environment was provisioned by information technology partners at the Texas Department of Transportation. For ongoing development and production, dual environments are planned for construction under contractual agreement with Amazon Web Services. All environments leverage AWS Cloud Builder for the creation of the "desktop" server and the ArcGIS Server "application" host, both configured as EC2 M4.2xlarge machine instance (AMI), with 4 CPU Cores, 16 - 32 GB of random-access memory (RAM), and at least 50 GB of disk storage space to handle the backend ETL workflow. Additionally, each location receives a relational database service (RDS) instance, running SQL Server 2019, for storing and managing all ingress source data as well as "staging" and "static" geodatabases. To complete this environment, the only additional step required of the

author was to install the PyCharm 2020 community-edition IDE on the "desktop" ArcGIS host machine so that pre-deployment updates to the ETL could be completed.

Multiple setup steps were required to complete the initial development environment, starting with the desktop installation of ArcGIS Pro 2.9.1 with an advanced Network Analyst license and procurement of an Amazon Web Services (AWS) cloud server instance with MS Windows Server and Esri ArcGIS Enterprise 10.7.1 platform components. Specifically, ArcGIS Server and Web Adapter services were installed on one Amazon EC2 instance, as the application host server. Then, the ArcGIS Data Store service was installed alongside the MS SQL Server2016 database management system on a separate EC2 database server. Two SDE connections were then created for storing source data in a "staging" database and published results in a "static" database, as mentioned in Chapter 3.

On the ArcGIS Server host machine, the MS Remote Debugger 12.0 service was also installed to provide a service runtime connection between the SOE and the MS Visual Studio 2017 integrated development environment (IDE) on the author's local desktop PC for code debugging. Several additional steps were needed on this local desktop IDE in order to allow all networked resources required by the SOE to function properly. These steps included desktop installation of the MS Visual C++ 2015–2019 Redistributable (x86 and x64) files, as well as the .NET Framework 4.5, ArcObjects SDK for .NET, and ArcGIS Runtime SDK. These are the core C# code modules with acquired libraries and assemblies for the project.

## 4.2. Developing the Route Analysis Layers

A critical function of the SOE is the linear routing capability that it invokes, particularly where it must solve a plotted path across separate but connected transit networks. For example, the route solver may determine that a commuter's optimal transit route traverses across a

sidewalk to a bus line that leads to a metro train line, then exiting at another pedestrian path leading up to the destination point. Connected polyline and point data in the multimodal transit network dataset makes this traversal possible. The same is generally true of the point and centerline data configured in the automobile network dataset, but there is only one travel network connected to one cost layer in this case.

A route analysis layer is created from the edge grid layer, the network dataset, then exposed during publishing of the map service to enable the Network Analyst extension to perform the on-demand routing functions. In the map document, each network dataset is removed after the route analysis layer is created, and all other layers and tables in the data model remain during publishing. All contents are published to ArcGIS Server as a map service, then the ".soe" build file is uploaded and appended to the map service in ArcGIS Server Manager. The SOE exploits the capabilities of Network Analyst using the route analysis layer, then spatially extracts cost attributes in both automobile and transit data models. While the underlying map service gains access to the physical data in the "static" geodatabase via the SDE connections, the core SOE modules filter and compute the travel costs on this data through a sequence of interfacing map server objects provided in the ArcGIS Runtime and ArcObjects SDK. The following subsections describe the ensuing logic applied to each data model, utilizing the analysis layers as well as the additional interface capabilities, to manifest realistic representations of travel based on user input.

### *4.2.1. Automobile Route Analysis Layer*

With roadway length as the default cost attribute, the route solver algorithm for automobile travel will redirect the output traversal toward the shortest path at any junction surrounded by conflicting cost attributes (Esri 2019). For the automobile route analysis layer, the only cost

attribute that can present such a conflict is the parking penalty time in the transfer street stations layer. Roadway traffic is not applied for determining path routing in this initial release, although it is very common in transportation network analyses. The main reason behind this decision is that average travel speed as a static traffic metric does not constitute a usable independent cost variable for routing. Average travel speed is implemented only as a route-dependent cost variable. The parking penalty is applied as the independent cost variable for routing because its values are not expected to change as frequently as roadway traffic flows.

The transfer street station features closely follow the automobile network centerlines and are connected by junctions at segment endpoints. This network attribute provides suitable impedance for modeling general driving conditions in this initial stage of the project. The route path is the output product of the route analysis layer, and once it has been solved, the SOE applies its linear shape to extract route-dependent cost values based on specific criteria. Since the parking penalty is also a route-dependent cost variable and occurs at the end of an automobile trip, its value is gleaned from the last transfer street segment intersecting the trip path by a 55-foot tolerance. Tour-based travel is planned for a future phase of the project, wherein parking penalties will be extracted at every intermediate destination point defined by the service user. This future capability forms the basis for the layer name, “transfer street stations”, as the traveler would be expected to transfer to a different mode after parking, namely walking or biking.

### *4.2.2. Transit Route Analysis Layer*

The facility length is also defined as the default cost attribute for the multimodal transit route analysis layer. And, in comparison to the automobile counterpart, travel speed is not applied as an input for routing. However, in contrast, this shortest-path route solver has substantially more work to perform in order to successfully traverse three interconnected transit

networks with differing criteria for impedance. Following the discussion of network datasets in Chapter 3, the elements of the transit route analysis layer include: pedestrian, bus, and rail networks; bus stops, transfer stops, and rail stations, all spatially integrated by connectivity groups and configured with cost, restriction, and hierarchy attributes.

The ensuing steps of route analysis read and compare the cost, restriction, and hierarchy of surrounding approaches to determine the most favorable direction toward the destination point. As shown in Section 3.4, Figure 13, the preference restriction is assigned to each transit stop type. Stepping onto a bus or changing busses at a bus stop is set to the "low preference" restriction. But walking onto a bus at a transfer stop means that bus route services one or more rail stations, and this junction has a restriction of "medium preference". The restriction set to rail stations is "high preference" and the route solver will navigate in favor of rail service until the rail stations are geographically out of range from the shortest path or the transfer cost becomes too high compared to surrounding options.

The transfer cost attribute is assigned to the following transfer cost variables: "BusToBusTransfer", "BusToRailTransfer", and "RailToRailTransfer". In each case, the field evaluator is assigned to all-day average transfer wait times from each transit stop. It's important to note that the multimodal transit network applies one and only one edge between any two transit stops. This design cannot physically represent the multiple bus routes or train routes that traverse each edge. So, how does the route solver know where one transit vehicle route ends and another one begins? How does it know where a transfer is optimal and where it is not? Both answers are provided by the transfer cost variables. The key factor, here, is that the lowest average transfer time is traversed at each bus stop, transfer stop, and rail station. This data construct is the product of GTFS calibration in Tier 1.

## 4.3. Developing the Server Object Extension

Tier 2 is composed of spatially enabled web services published through the ArcGIS REST API via the SOE, for activating routing capabilities and consuming the data. These services are written in code modules using the C# programming language on the MS .NET Framework. The ArcMap-based runtime API and the ArcObjects SDK for the .NET framework build and package these custom modules into a file that is uploaded to the primary map service. Esri documentation states that ArcGIS Server 10.9.1 does have ArcMap runtime services, on which most .NET SOE development patterns back to version 10.5.1 are fully supported (Esri 2022). The cited documentation, here, also provides steps for migrating ArcMap-based map services or image services to use the ArcGIS Pro runtime. For quick reference, one of the early steps in this process is transitioning the .NET ArcObjects-based SOE to be built with the ArcGIS Enterprise SDK.

Without the burden of having to perform data preparation tasks, the SOE needs only to expose routed variables for immediate computation. In turn, this simplifies the required tasks in a web client, essentially to sequencing calls to the service endpoints and providing the user interface (UI) with an effective user experience (UX). However, developing SOE middleware to perform its required steps should not be dismissed as easy or trivial. At a minimum, the effort requires some experience with object-oriented coding in an advanced IDE. It is best practice to maintain a template SOE solution with core modules updated to the latest code libraries and reference assemblies for the development platform. This strategy allows the developer to keep existing projects up-to-date and portable to supported web platforms, while new projects can be initiated without having to start from scratch every time. The author leverages this strategy in the current project and a brief description of the template solution, given the namespace title of

"LRSLocator", follows in this section. The project's SOE template code, referenced in the ensuing discussion, is provided in the Appendix.

In addition to the setup steps detailed at the end of Section 4.1, the development workstation requires Internet Information Services (IIS) installed, and ideally the latest version. Once the Visual C# .NET web application project is created in the workstation's "inetpub\wwwroot" directory, a few core ".cs" modules stored with the solution file must be created with several key functions. To begin, under the "properties" subdirectory of the "LRSLocator" project, an "AssemblyInfo.cs" module provides general information about the solution that may optionally be set as visible to standard Component Object Model (COM) elements on the web. More importantly, the file which exposes the project's functions to COM elements is the first module to reside in the main directory of the project, called "ComReleaser.cs". Its job is to manage the release and disposal of "LRSLocator" content as binary software components.

Next in the main directory is the "RESTContext.cs" module which declares all entities to be shared over the ArcGIS REST API. These include global solution variables for basic inputs and outputs (IO), as well as interface objects that represent all map server elements, network datasets, feature classes, relationship classes, and field names used in the project. To handle incoming requests to a specific REST resource or operation, a simple interface module called "IRESTHandler.cs" answers requests with return values in either JavaScript Object Notation (JSON), string, or byte format. To further assist the solution with encoding responses and handling errors in the JSON format, a "JSONHelper.cs" module is also added.

With the above web interface modules in place, the functional framework of *Commute GeoCalculator* is constructed in the much larger "LRSLocator.cs" file, the main module. This is

where the project's specific attributes and behaviors are declared and executed, wherein nearly everything is treated as an interface object or a REST operation. The globally unique identifier (GUID) is a 16-byte unique label for the project, which must be generated and then specified under the namespace and above the "ServerObjectExtension" declaration. From here, every variable and method forming the backbone of the project are written as private objects within a public class that inherits specific interfaces, all nested within the "LRSLocator" namespace.

All entities of the "RESTContext.cs" module are referenced in the main module, and every ArcObjects interface member used in the solution is declared. Furthermore, in this main module the schema for each REST page is declared, along with functions that control IO properties and the interface with any element in the registered "static" geodatabase. Here, the SOE communicates with the database server, essentially taking the map service as an input argument. Because the project utilizes routing capabilities of the Network Analyst software package, the main module applies very specific conditions within the geodatabase interface to distinguish between M-aware feature classes, relationship classes, tables, network datasets, and the analysis layer parts therein.

Finally, the implementation code modules leverage all attributes, behaviors, and interfaces of the main module to deliver the advanced functions of the middleware that meet project objectives. Regarding the use of routing capabilities, "RouteFromInputPoint.cs" mimics the actions of a desktop (or ArcGIS Pro) user who directly invokes the functions of the Network Analyst extension. This is also the module that writes out each solved path to any reachable machine on the current domain. The implementation code that applies the cost equations for driving and transit calls these functions of "RouteFromInputPoint.cs". Any number of implementation modules may be created in the SOE template to fulfill project requirements.

## 4.4. Computation of Travel Costs

The other vital function of the SOE is computation of the travel costs, using the linear shapes generated during routing. As stated in Section 4.2.1., the SOE applies the solved path to extract route-dependent cost values based on specific criteria. This section delves into these criteria to explain how usable variables are gleaned from each data model and applied to the travel cost equations. In this procedure, parameters (p) are coefficients specified by the regional transportation authority, while all other variables change value per each commute. All time and distance data, including those applied to travel costs, are expressed in minutes and miles at a precision of 0.1 minutes and 0.001 miles. The equation output provides a cost value associated with each travel mode path – transit and automobile. Both cost equation sets include inferred traveler income extracted at the trip origin, then factored into the "value of time" parameter. In Table 6, the monetary value of time is translated from minutes of total travel time, based on household income.

Table 6. Value of Time by Purpose of Travel and Income (NCRTPB 2020)

| Household Income | Midpoint of Household Income | Hourly Rate per Worker | Time Valuation (Minutes per Dollar) | |
|---|---|---|---|---|
| | | | Work Trips (75% Value of Time) | Non-work (50% Value of Time) |
| $ 0 - $50,000 | $25,000 | $9.23 | 8.7 | 13 |
| $50,001 - $100,000 | $75,000 | $27.70 | 2.9 | 4.3 |
| $100,001 - $150,000 | $125,000 | $46.17 | 1.7 | 2.6 |
| $150,001 + | $175,000 | $64.64 | 1.2 | 1.9 |

4.4.1. *Automobile Travel Costs*

The overall approach to cost computation is to apply authoritative parameters and estimates to additive equation sets derived from research, while using spatially extracted attributes as the variables. Link drive time, below, is the average driving time over one segment of the highway network, based on the input mean speed from traffic metrics divided by the length of segment within a given TAZ. Link mean speed, $\mu S_L$, is provided by the automobile trip speed table by day type (weekday or weekend) and day part (early AM, peak AM, mid-day, peak PM, and late PM). The service user specifies one day type and one day part per each request for a commute calculation. Spatial intersection tools from the ArcObjects interface language make it possible to take path-specific measurements from the layers in the automobile data model. The total cost in minutes, $T_A$, is the summation of all link travel times along the routed path, plus the parking penalty which is taken from the "transfer streets stations" layer at the destination point. The cost equation set for driving is as follows:

$$\text{Total Cost (in minutes)}, T_A = \sum_{d=1}^{n} (D_L / \mu S_L)_d + KT \quad (8)$$

where $(D_L / \mu S_L)_d$ is the link drive time composed of $D_L$, the link distance, and $\mu S_L$, the link mean speed. KT is equal to the parking penalty time, and n is the number of road links.

$$\text{Total Cost (in dollars)}, C_A = (pVT \times T_A) + (pCM \times D_T) + KC \quad (9)$$

where pVT represents the value of time parameter, as per Table 6; pCM is equal to the cost per mile parameter; $D_T$ is the total drive distance; and KC is given as the estimated parking fee. The total cost in dollars, $C_A$, is the income-driven value of time parameter multiple by $T_A$ and then added to the product of fuel cost per mile and total distance, plus the estimated parking fee. Roadway fuel cost (for the pCM parameter) is derived from the national-level model applied in

the NCHRP Report 716 at $0.21 per mile, then adjusted by the current inflation rate. The parking fee, KC, (in dollars) is extracted from the same layer at the same user-defined destination point where the parking penalty (in minutes) is assessed. In compliance with the regional authority, the parking cost is estimated within the "transfer streets stations" layer by the following equation:

Estimated Parking Fee, KC = 2.1724 x ln(employment density) – 15.533 (10)

Specifically, the shown natural logarithm function is calculated by the Python math library, as math.log(***x,*** [base *e*]), where ***x*** is the block group employment density.

4.4.2. *Multi-modal Transit Travel Costs*

For the travel cost computation of multi-modal transit, the overall approach to use authoritative estimates in research-based equation sets is fundamentally the same as described for automobile driving costs. Similar to the auto driving cost equations, the attributes of travel time for multi-modal transit are calculated over the facility links traversing each TAZ in which trip metrics are linearized by day type and day part. The interface and format of inputs and outputs at the REST endpoint are the same as used for the driving mode. What is quite different is the algorithm that captures and processes the larger number of variables used as inputs to solve the transit equation set. On one hand, pedestrian trip speed is simply read over all links that intersect the route path, which is the way that driving trip speed is collected. On the other hand, bus and rail modes require careful tracking of the specific service line utilized, from the first point of entry to the last. The cost equation set for transit is as follows:

$$\text{Total Cost (in minutes), } T_M = [\sum_{p=1}^{w} (P_L / \mu S_L)_p + WaT] + [\sum_{b=1}^{x} (B_L / \mu S_L)_b + W2T_b] + [\sum_{r=1}^{y} (R_L / \mu S_L)_r + W2T_r] + Wa2T^* \quad (11)$$

where $(P_L / \mu S_L)_p$ is link walk time; $(B_L / \mu S_L)_b$ is link bus time; $(R_L / \mu S_L)_r$ is link rail time. Here, each numerator and denominator replicates the quotient of link distance over mean speed used in the automobile cost equation. WaT represents the wait time, $W2T_b$ the bus transfer time, $W2T_r$ the rail transfer time, and Wa2T* the intermodal transfer wait time between bus and rail.

$$\text{Total Cost (in dollars)}, C_M = [pVT \times T_M] + (F + F2T^*) \quad (12)$$

**w**here pVT is the value of time parameter; F is equal to the fare cost of the initial bus or rail route, and F2T* is the fare cost of any ensuing transfer. Transit fares (F, F2T*) and intermodal transfer wait times (Wa2T*) between bus and rail are gathered and processed within separate server objects. That is, these variables are aggregated outside of the dictionaries containing values for the link travel time attributes. Fare costs are applied to the linearized metrics based upon the average fares of each transit operator, and then exposed after network paths are solved. And the parameters are constants assigned to the region according to NCRTPB specifications. Finally, the sum of intermodal transfer times are added to the transit equation (11) for the time-based cost, just before it is solved. Once equation (11) is fully executed, then the sum all fares are added to the product of total time expenditure ($T_M$) and the value of time parameter (pVT) to solve equation (12). Section 5.2 further explains how these cost variables are extracted and processed in the multimodal transit model.

# Chapter 5 Results

The proprietary traffic metrics and open-source content used to build the multi-network data models and enterprise web services in this project were initially unusable for such an application. Source travel speed and traveler income were provided as percentages over a set interval of movements detected within each traffic analysis zone. In order to model general travel conditions, this range of values had to be dissolved and linearized into the average trip speed and traveler income along facilities in each zone. The location of parking penalties for driving and transfer times for transit are not values that are readily available on an open-source data portal, nor from any online content provider. Rail and bus polyline layers from the regional authority do not contain distinct linear features for each individual route traversing the system, much less the route segments with average transfer times.

At the source, multiple rail and bus head signs are grouped into one route identifier field for every line segment and transit stop table. GTFS data were required to fully itemize and accurately position each transit stop, yet GTFS stop identifiers are not common with NCRTPB identifiers for bus and rail stops. Furthermore, all linear data provisioned by the regional authorities and the US Census Bureau contain no M-values in their shape fields that would enable routing, and a significant portion of features are disjoint at intersections. With all of the challenges in the data resolved in Tier-1 geoprocessing, Tier-2 results are dual routable network datasets that model driving and transit paths by estimated parking times, transfer wait times, distance, and preset hierarchical preferences. Each path geometry is provided in well-known text (WKT) format on the REST page. Also, every path is written to the workstation server in a feature layer collection that can expose each individual route analysis layer to most map viewers, including ArcGIS Pro.

The results on each REST page also include all user-entry parameters and cost outputs, in decimal units of minutes and dollars. Depictions of these results are presented in Sections 5.1 and 5.2. To commence the cost calculation, the only inputs needed are beginning and ending latitude and longitude in decimal degrees, and the pre-defined string literal for identifying the time of day (day part), weekday or weekend (day type). Over one hundred distinct coordinate pairs were tested at the REST page in the study area; ninety-two of these cases returned valid values. The following sections describe the full context of actual results, through an investigation of one of the successful travel cost samples.

Table 7. Time of Day Input Parameter

| Day Part Description | Weekday Entry Value | Weekend Entry Value |
|---|---|---|
| All-Day Average | 1:AllDayAvg:Weekday | 7:AllDayAvg:Weekend |
| Early AM (12am – 6am) | 2:EarlyAM:Weekday | 8:EarlyAM:Weekend |
| Peak AM (6am – 10am) | 3:PeakAM:Weekday | 9:PeakAM:Weekend |
| Mid-Day (10am – 3pm) | 4:MidDay:Weekday | 10:MidDay:Weekend |
| Peak PM (3pm – 7pm) | 5:PeakPM:Weekday | 11:PeakPM:Weekend |
| Late PM (7pm – 12am) | 6:LatePM:Weekday | 12:LatePM:Weekend |

## 5.1. Automobile Cost Model and Service

The rules of the road for basic routing on a highway network are relatively straight-forward when there is only one centerline file with no modeling of turns, elevation, service areas, or complex rules for impedance. The only impedance built into the project's highway network is the parking penalty that is based on census employment density. This routing cost for driving and street connectivity can and will redirect the route solver away from what may appear to be the shortest distance between origin and destination. The forementioned trip sample for driving is visualized in Section 5.3. Figure 16 shows the attribute values on the REST page.

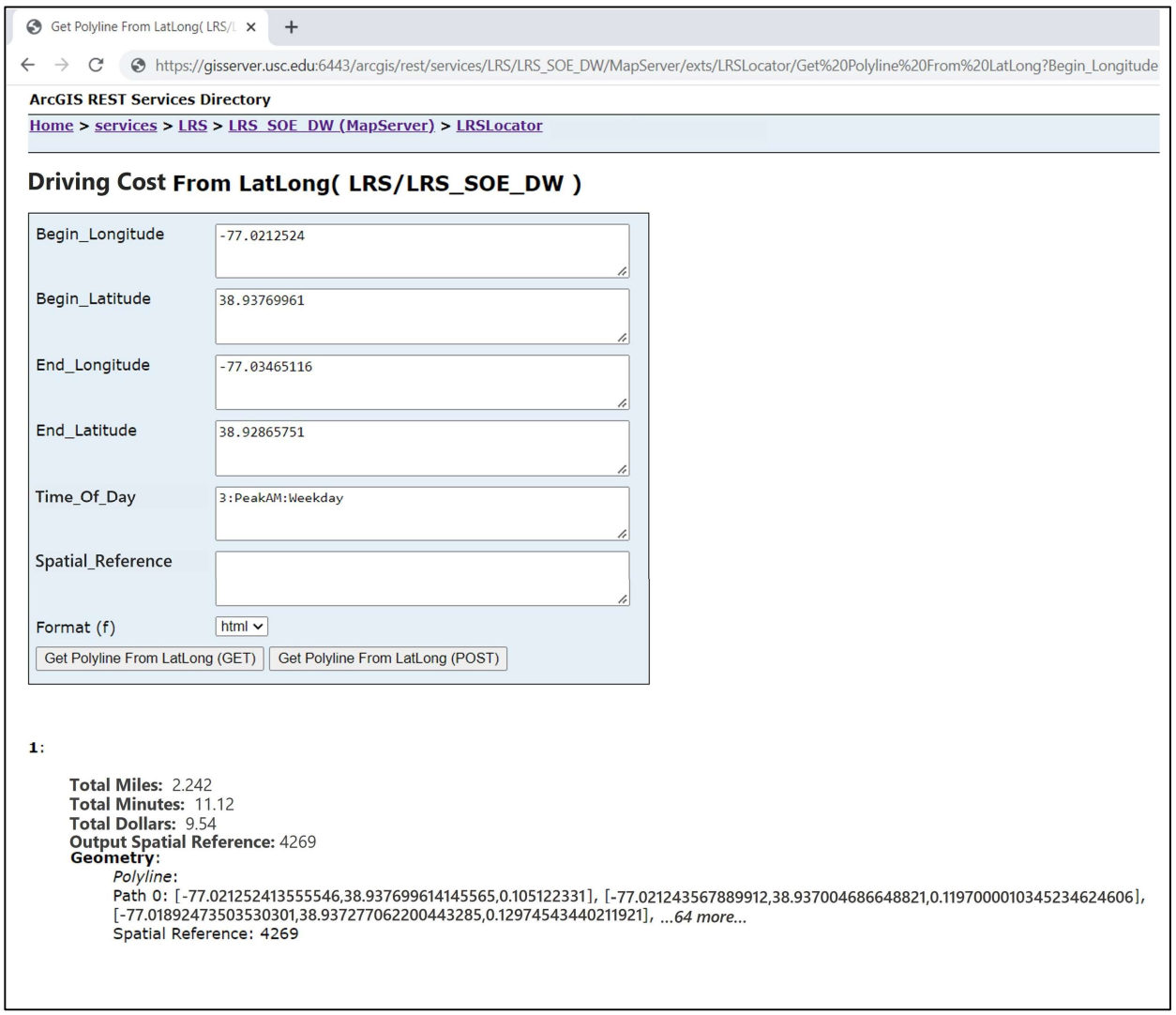



Figure 16. Trip Sample - Automobile Travel Cost Return

With segment endpoints of the transfer street stations created directly from the highway centerlines, the default edge connectivity enables the network route solver to redirect paths cleanly in the direction of lower parking penalties. After routing, a sequence of methods in the SOE use the geometry of the solved path to capture all segments which carry the penalty values. From this isolated subset, the destination point feature is then applied in a spatial intersect to extract one distinct record containing the whole number of minutes required on average to park. The same path geometry is also used to extract each link drive time that is plugged into the cost equation with given coefficients to calculate the totals shown in Figure 16.

## 5.2. Multimodal Transit Cost Model and Service

For the same trip sample, the multimodal transit costs at the REST page are depicted in Figure 17. Instead of segment endpoints used as the connecting junctions between these networks by default, bus stops, transfer stops, and rail stations serve this purpose in the transit route analysis layer. As pointed out in the previous chapter, the modal split is preconfigured by the elements in the transit network dataset.

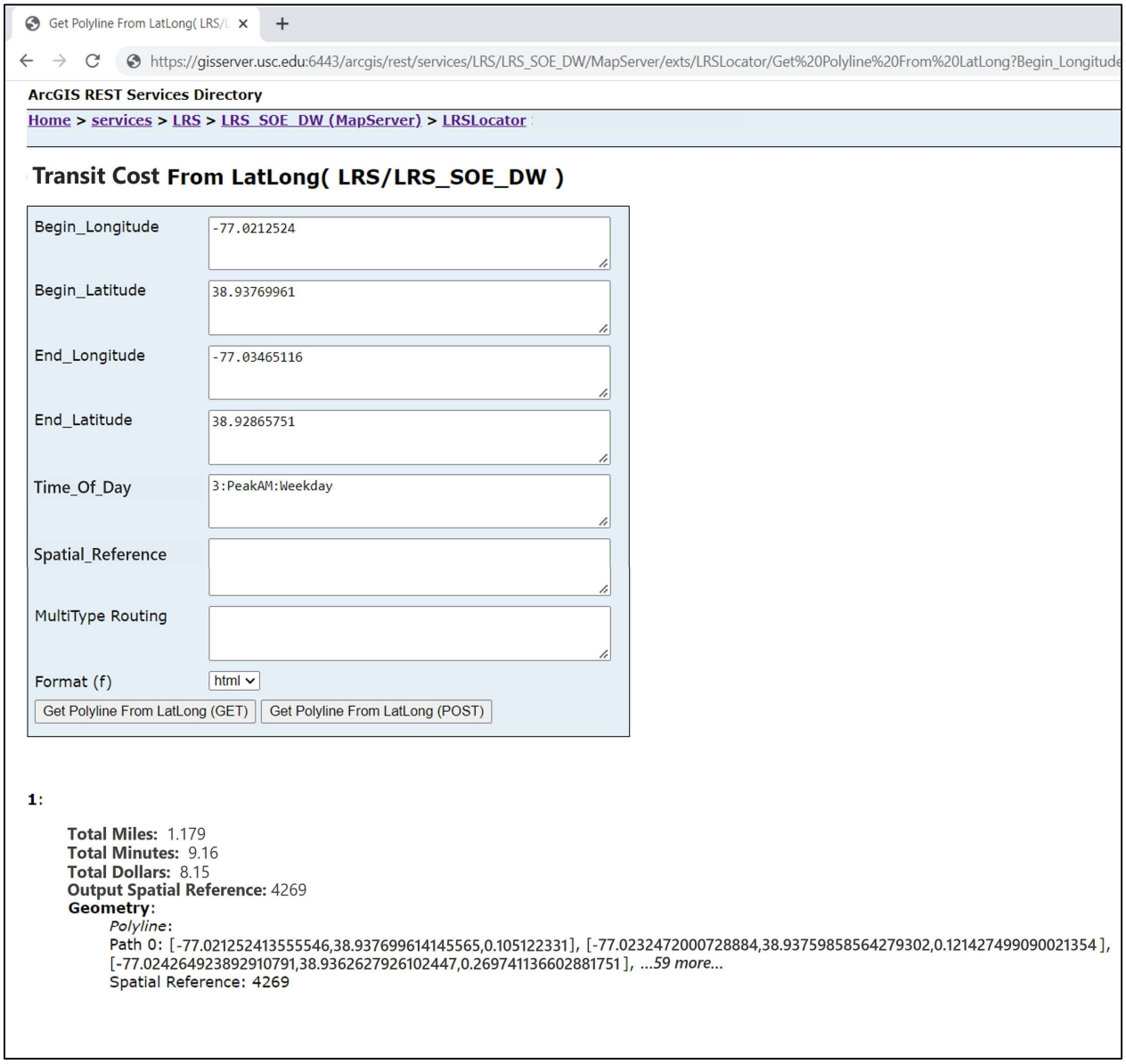



Figure 17. Trip Sample – Multimodal Transit Travel Cost Return

In most user-defined scenarios, the subsequent route analysis layer begins its routing task by reading restriction and hierarchy values in the pedestrian network features that surround the trip origin point. Of course, it is possible that an origin point may be specified within five feet of a bus stop, transfer stop, or rail station, in which case the cost, restriction, and hierarchy of the respective network would be processed first. But in the most prominent case, it is important to understand that the pedestrian network does not have an independent cost attribute for routing. Transfer wait time is the independent cost variable in the transit route analysis layer and walking simply does not impose a transfer waiting period in any practical situation. However, the pedestrian hierarchal ranking is the lowest of all networks, at a value of "3", and thus the route solver expeditiously navigates away from walking to the nearest junction having the most favorable attribute values and position on the shortest path toward the destination. Taking a bus ranks as "2", and riding a train is ranks highest at a value of "1".

In Chapter 3, Figure 14 illustrates the relationships that bind arrival and departure times to every route for bus and rail modes. These relationships are important to Tier-1 preparation of transfer times within the attributes of transit stop points and links. The transit route tracking algorithm in the SOE collects the route identifier from first service line encountered, and then checks whether it persists across every subsequent stop which has a non-zero transfer time (an opportunity to transfer). If the next link in the direction of travel does not have the *expected* route identifier in its attribute table, then the transfer time at the associated stop is added to a dictionary object in code, along with record of the link that is providing the trip speed at the location. Also, the next route identifier is assigned to the route tracking variable, and the process repeats along the solved route path for bus routes and rail service lines exclusively. As a result, trip speeds and

transfer times are organized by link and travel mode, for orderly insertion as variables into the multimodal cost equations.

## 5.3. Routed Automobile and Transit Travel Costs

Through the SOE, both automobile and transit cost models have their own API for computing individual trip samples with only lat-long coordinate pairs and the time-of-day parameter as required inputs. Each of these REST endpoints (pages) contain an optional input parameter for the EPSG number of the desired spatial reference. By default, SOE computation is performed on the model data in unprojected geographic coordinates on NAD 1983, even though the data undergo ETL projected in WGS 1984 Web Mercator Auxiliary Sphere with shapes preserved. Given the consistency in geoprocessing these data under a world projection well-suited for a web platform, the conversion back to geographic coordinates during SOE publishing allows the object libraries to apply geometric features to most any projection specified by a web client. It's a technique utilized in multi-level linear referencing systems (MLRS) that apply a network data construct known as a *linear datum* to handle practically any projection on the data (Butler 2008).

In the previous section, the trip sample exemplifies how much more complex the multimodal transit network is compared to the basic highway network. The multimodal transit network inherently requires more rules for analysis. For example, the route solver will continue charting along an assumed bus route on the transit route analysis layer where a bus stop carries a transfer time of zero. If there happens to be another bus route intersecting that bus stop point, it is ignored because GTFS data hold no record of both bus lines servicing that particular stop. Figure 18 shows the geometry paths for the scenario submitted at each REST page in Figure 16 and 17.

This particular case involves a transfer from rail to a bus route via a rail station well before the destination point, that is otherwise closer to a different rail station.

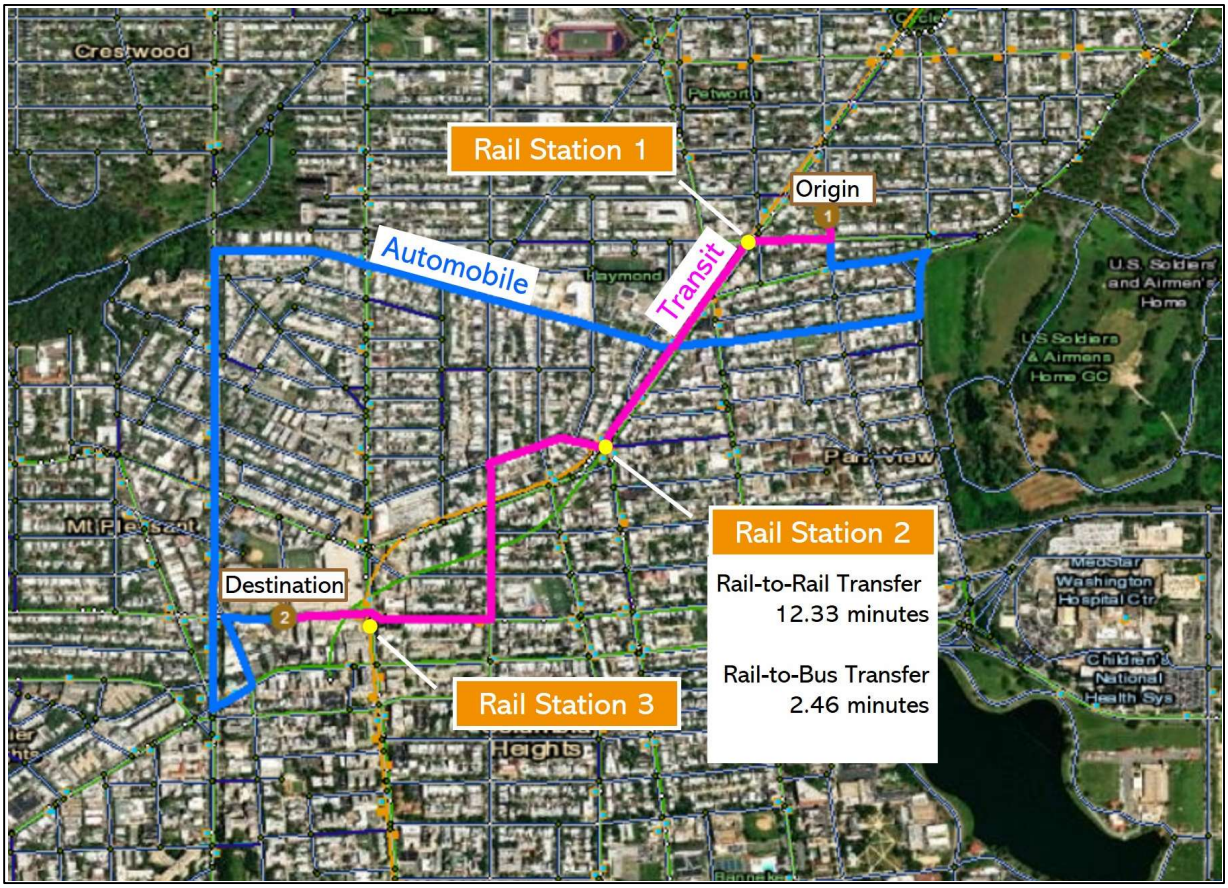



Figure 18. Transit and Automobile Cost Path Outputs in ArcGIS Pro®

Here, the path solution accepts a lower transfer cost from the "BusToRailTransfer" value that is read from the transfer stop at the exact location of "Rail Station 2". In theory, this also means that the bus transfer times along the remaining path to the destination are lower than the impedance measured from taking the route through "Rail Station 3". The thin orange line is the local rail line. By comparison, another noteworthy outcome in the above depicted example is the path solved for automobile travel. It is forementioned that the parking penalty is applied as the independent cost variable for automobile routing because its values are not expected to change very frequently. There is another reason that parking time is applied to routing, which is

illustrated in Figure 18. Longer parking times indicate higher employment density by census block group, which suggests more heavily congested activity centers (NCRTPB 2020). Thus, the automobile route solver negotiates around such areas of higher parking penalties, with respect to the shortest physical path. Recall that route length is the default cost variable for both travel modes, and the additional elements assigned to each route analysis layer refine the results.

# Chapter 6 Conclusions

A rigorous research and development effort has driven completion of this initial phase of the project, with authoritative source data provided by open data portals and traffic content from the SL Data InSight® platform. The initial analysis and data capture of traffic metrics was made possible through academic licensing by SL Data, Inc. SL content and analysis software is available to the project under a free one-year academic license with 500 traffic analysis zones, the approximate extent of central metropolitan Washington D.C. Instead of committing the entire project to open-source content and technologies, the decision was made to opt for long-term support and scalability made more practical by proprietary providers. This was deemed necessary for the future expansion of the service area into broader geographies, and for the implementation of a web client and more advanced functions.

## 6.1. Application Utility

The middleware provisioned in the project is designed as a tool for exposing probable travel cost paths, given general conditions over the built environment on which the data model is implemented. The system is functional over heterogenous urban development patterns in terms of street connectivity, transit accessibility, system mobility, service frequency, and the various costs of traversing the system at peak and off-peak traffic intervals. Tier-1 background geoprocessing scripts provide a repeatable process for integrating disparate geographic data in modeling multimodal travel. Subsequently, with these data now usable in an accessible model set, the *Commute GeoCalculator* travel-cost web services are well suited for hosting in a standard enterprise-level IT environment or research lab. From this solution, multiple possible use case scenarios exist for the scientific research community as well as for the transportation industry.

Transportation web and mobile applications that are purposed for quantifying multimodal travel would readily benefit from consumption of the project's REST services. As forementioned in Section 2.4, Google Maps utilizes live traffic feeds with routing functions in some service areas, but each selected travel mode may only be routed on separate map views. There is no combined view of all selected travel modes on one map UI, which is a limitation for the traveling user of Google Maps. Because the project web services leverage performant methods for operation on enterprise-level infrastructure, it is conceivable that a competitor application could utilize the project's middleware to address this gap in map UI design and to geographically extend coverage of estimated travel costs with updateable traffic metrics. Public-facing routing applications present the most immediate possibility for general usage in commercial industry.

A second possible use case that would serve both industry and scientific research is the installation of *Commute GeoCalculator* services in high-volume data generation processes. A scripted process could submit a predefined list of random and independent coordinate pairs with day parts (time ranges), as OD inputs at the service endpoints, to then write the returned trip cost information to a database. Transit agencies that are interested in analyzing the network performance of their route plans would benefit from this utilization. This use case also applies to data preparation for conventional inferential statistics, often seen in spatial analyses. Albeit the linearized data in the project's model are geared more toward network analysis, methods for spatial analysis of linear traffic patterns are found in transportation research (Haixiang, Yang, and Yonghui 2010; Bhat and Zhao 2002). The author finds that such methods would also be applicable to spatial analyses of multimodal travel cost patterns, whereby *Commute GeoCalculator* services would serve as a viable resource for the preparation of these data.

Thirdly, by applying the project's services in a web client to invoke mode-choice responses directly from the commuting public, the researcher or transportation manager could create a powerful VGI application. A practical scenario, aforementioned in the project overview and Section 2.3, is a VGI deployment that allows the general public to readily compare the cost-based utilities of their own individual commute patterns and options in order to provide input as participating stakeholders in local transportation improvements and TDM strategies. Perhaps an even more effective deployment would be a travel routing application that is open indefinitely to the commuting public for general use. Considerably more VGI content could be collected, but not without some incentive to travelers for participating.

## 6.2. Limitations and Costs of Development

While the *Commute GeoCalculator* services are designed for scalability, the initial release is a proof of concept with significant usage limitations. For the purposes of the project as a thesis work, the initial users shall be authorized students, faculty, and staff of the Spatial Sciences Institute at the University of Southern California. At each REST page, users may replicate the steps which a web client would apply, by entering OD points using XY coordinates in decimal degrees. To accommodate, a scripted utility is provided that randomly extracts and organizes lat-long coordinates from the endpoints of highway segments in the study area, so that initial users have the means to test the application. On May 20, 2023, these functions and documentation are planned for release at https://github.com/modomotiv/sandboxpage.github.io/index.html, where the future release of the *Commute GeoCalculator* website is also scheduled for release at a later date.

In the current project, the computation of travel costs for multimodal transit has a few notable simplifications that limit usage. The first of these is explained in Section 5.2., the

preconfigured modal split, which makes the transit cost service too inflexible for use in a travel web application. This rigidity in the network dataset configuration stems from the use of average transfer times at each transit stop and along each bus and rail route line. With points and lines as simple features, it is quite difficult to build a complete network representation of all available transfer times for all bus and rail routes. Therefore, a higher order structure of point and line features is part of future development plans, wherein GTFS calibration can more effectively articulate among individual transfer times and their specific operator routes.

The most limiting simplification imposed on the transit data model, which led to the use of average transfer times, is the lack of travel direction with respect to bus and rail modes. Even though GTFS data include a directional indicator of arrival and departure times with positional offset, the higher orders of point and line structure that are required for directional transit segments (Butler 2008) are not built into the current model. The primary reason behind this design decision is the generalization imposed by assigning the traffic indices from TAZ polygons to each contained transit segment. The direction of travel is lost in the organization of traffic metrics during this process. Nevertheless, the US standard TAZ features, which are typically larger than a city block but smaller than a census block group, do provide a valid characteristic unit of measurement for general traffic conditions. Hence, a low to moderate level of ecological fallacy is accepted in the construction of the current model, and future advancement of the model will address this issue. Whether it is a change in content provider or a different data analysis to extract these metrics, the traffic on linear facilities cannot continue to be inferred from polygon data if the application is to advance toward meeting its long-term objectives.

One final noteworthy curtailment in the transit cost model involves bus and rail fares, which are weekday and weekend averages for each transit operator with no accounting for

dynamic pricing. Fluctuations in demand, particularly for local train routes (NCRTPB 2020), drive significant intra-day changes on fares. Instead of tracking operator rules for dynamic pricing, fares are manually extracted from each operator's reported website then averaged for one day of ridership and hard coded into each route. At service runtime, fares are enforced one time in the cost calculation for bus and rail, independently. For the computation of total transit cost on individual commutes, this omission of dynamic pricing can be impactful to the accuracy of results. To a lesser extent, variable pricing on parking fees may also impose inaccuracies in the total driving costs of some commutes. Yet such parking data may prove to be challenging to acquire. Therefore, more comprehensive treatment of transit operator rules is the focus of future improvements to the assessment of direct charges from transportation service providers.

Initial project expenses were incurred by the acquisition of Windows Server and ArcGIS Enterprise software packages that run in an authorized AWS sandbox environment provisioned by GIS technology partners at the Texas Department of Transportation, plus the application fee for the one-year academic license from SL Data, Inc. Fortunately, this license application was accepted, thereby avoiding the substantial cost of purchasing the 500-zone traffic content used in the project. Minor expenses included presentation images procured from Almay, Inc., and some minimal leave of absence from currently contracted work to finish the project. One of the near-term future costs of the project will be the subscription fee to host all application functions herein, plus the planned web client, from a secured GitHub repository. Hosting fees for a startup website are relatively inexpensive in the short run, as long as the user demand is low. A more substantial cost in labor is expected for the near-future SOE migration away from ArcMap runtime services and ArcObjects SDK, onto the ArcGIS Pro runtime API and Enterprise SDK.

## 6.3. Future Improvements

Numerous important enhancements are planned for future release phases of the project, including steps to address the key limitations discussed in Section 6.2. However, the foremost initiative in the future improvement plan is a web client in a third tier. The overarching concept is that through a transactional web user interface, the user receives requested commute cost information while given the opportunity to share minimal and anonymous mode choice feedback about each travel query. Through the UI elements of a presentation tier – choosing OD locations, identifying routes and landmarks, sequencing multiple destinations, and specifying the time schedule for travel – substantial spatial cognition (Montello and Sas 2006) is required of the user. The first user requirement is spatially intuitive plotting of all OD points on the map UI, including any intermediate stops that would constitute a tour. A critical feature, here, will be street address geocoding, so that the end user is not confined to only plotting OD locations by XY coordinates on the map UI. And, with these enhancements, the addition of the bicycle mode will be an important feature as well.

The second requirement, here, is a collaborative and brief exchange of information through the end user workflow. This dialog commences only after the user has plotted their points to create the shortest-distance transit and driving paths. Time and cost results are forthcoming as soon as the user provides information about the time and purpose of travel, as well as household income on a strictly voluntary basis. There are disaggregate mode choice factors which are not included in the equations, such as traveler's comfort and sense of security or safety. These qualitative factors are based on users' experiences and will be shareable in the web client.

Given the requested cost values, the user may opt in to anonymously share VGI about which mode would be selected for their trip or tour and the reasons behind it. Alternatively, the

user may decline to do so and continue to use the system without any part of their session being saved. At the end of this workflow, the web client will call a feature service that is hosted in the logical tier to capture and timestamp any volunteered results in the *commuter* geodatabase. These results include the user's anonymous profile, travel preferences, cost attributes, path geometry, as well as which alternative was chosen and, again, the reasons behind the choice.

The initial web user interface will not necessarily be designed to operate on mobile devices in the next planned release, but mobile capability is crucial for the application's intended maturity path. In the first rollout, the web application will have an internal public user account that only tracks each session, not the individual user. Limited functions will be available for retrieving the traveler's volunteered results in the *commuter* geodatabase. The extent of preliminary capabilities will permit a transport manager role to access and download mode choice data in Excel tables and a corresponding shapefile.

In conclusion, the *Commute GeoCalculator* program is designed for automated data sampling and comparison of multimodal travel costs, with the development potential to become an empirical data sampling tool for travel mode choice. A fundamental aspect of travel behavior is mode choice. If travel mode choice can be sampled randomly and independently from the field directly with respect to the built environment, then new opportunities arise for improving studies of travel behavior in the space of flows (Pieri and Nelson 1999). Through scientific application of such technology, perhaps more effective interventions in transportation are possible. Regardless of the project's current constraints, the results serve as a viable proof of concept that web GIS services can be created from available resources to present an alternative approach toward researching travel behavior.

# References


Antipova, A., F. Wang, and C. Wilmont. 2011. "Urban Land Uses, Socio-Demographic Attributes and Commuting: A Multilevel Modeling Approach." *Applied Geography* 31, no. 3 (2011): 1010-1018.

Bai, T., X. Li, and Z. Sun. 2017. "Effects of Cost Adjustment on Travel Mode Choice: Analysis and Comparison of Different Logit Models." *Transportation Research Procedia* 25 (2017): 2649-2659.

Bhat, C. R. and H. Zhao, 2002. "The Spatial Analysis of Activity Stop Generation." *Transportation Research Part B: Methodological* 36, no. 6 (2002): 557-575.

Bhat, C. R. and N. Eluru. 2009. "A Copula-Based Approach to Accommodate Residential Self-Selection Effects in Travel Behavior Modelling." *Transportation Research Part B: Methodological* 43, no. 7 (2009): 749-765.

Bottai, M., N. Salvati, and N. Orsini. 2006. "Multilevel Models for Analyzing People's Daily Movement Behavior." *Journal of Geographic Systems* 8, no. 1 (2006): 97-108.

Butler, A. J. 2008. *Designing Geodatabases for Transportation*. 1st edition, 380 New York Street, Redlands, CA 92373: Esri Press.

Case, K. E. and R. C. Fair. 2004. *Principles of Microeconomics*. 7th edition, Upper Saddle River, NJ 07458: Pearson Prentice Hall.

Calthorpe, P. 1993. *The Next American Metropolis: Ecology, Community, and the American Dream*. New York: Princeton Architectural Press.

Centers for Disease Control and Prevention. 2021. "Leading Causes of Death: Mortality in the United States, 2020." National Center for Health Statistics, December 1, 2021. Accessed September 20, 2022. https://www.cdc.gov/nchs/fastats/leading-causes-of-death.htm.

Centers for Disease Control and Prevention. 2022. "Adult Obesity Facts." Overweight & Obesity Data and Statistics. Last modified May 17, 2022. Accessed March 3, 2022. http://www.cdc.gov/obesity/data/adult.html.

Central Pennsylvania Transportation Authority (CPTA). 2021. RabbitTransit – for Adams, Columbia, Cumberland, Franklin, Montour, Northumberland, Perry, Snyder, Union and York Counties. Accessed March 19, 2022. https://www.rabbittransit.org/.

Chaix, B., J. Merlo, S. V. Subramanian, J. Lynch., and P. Chauvin. 2005. "Comparison of a Spatial Perspective with the Multilevel Analytical Approach in Neighborhood Studies: The Case of Mental and Behavioral Disorders due to Psychoactive Substance Use." *American Journal of Epidemiology* 162, no. 2 (2005): 171-182.

Duany, A., and E. Plater-Zyberk. 1992. "The Second Coming of the American Small Town." *Wilson Quarterly* 16, no. 1 (1992): 3-51.

Duncan, J. C. and K. Jones. 2000. "Using Multilevel Models to Model Heterogeneity: Potential and Pitfalls." *Geographical Analysis* 32, no. 4 (2000): 279–305.

Energy Information Administration. 2007. "Annual Energy Outlook 2007 with Projections to 2030." Report #DOE/EIA-0383. Tables 47, 58. Modified March 3, 2022. Accessed March 12, 2022. http://www.eia.doe.gov/oiaf/aeo/index.html.

Esri. 2022. "Introduction to Extending Services." ArcGIS Server Documentation Support. Accessed December 11, 2022. https://enterprise.arcgis.com/en/server/latest/develop/windows/about-extending-services.htm.

Esri. 2019. "What is a network dataset?" ArcGIS Desktop Documentation Support. Accessed November 17, 2022. https://desktop.arcgis.com/en/arcmap/10.7/extensions/network-analyst/what-is-a-network-dataset.htm.

Fu, P. 2020. *Getting to Know Web GIS*. 4th edition, 380 New York Street, Redlands, CA 92373: Esri Press.

Giuffrida, N., M. Le Pira, G. Inturri, and M. Ignaccolo. 2019. "Mapping with Stakeholders: An Overview of Public Participatory GIS and VGI in Transport Decision-Making" *ISPRS International Journal of Geo-Information* 8, no. 4 (2019): 198.

Goodchild, M. F. 2011. "Scale in GIS: An Overview." *Geomorphology* 130, no. 1-2 (2011): 5-9.

Grazi, F., J. C. Van de Bergh, and J. Van Ommeren. 2008. "An Empirical Analysis of Urban Form, Transport, and Global Warming." *The Energy Journal* 29, no. 4 (2008): 97–122.

Haixiang, Y., Q. Yang, and S. Yonghui. 2010. "A Spatial Analysis Approach for Describing Spatial Pattern of Urban Traffic State." *13th International IEEE Annual Conference on Intelligent Transportation Systems*, Madeira Island, Portugal (September 2010): 557-562.

Hanson, S. 2001. Review of "The Boston Renaissance: Race, Space, and Economic Change in an American Metropolis", by B. Bluestone & M. H. Stevenson. *Annals of the Association of American Geographers* 91, no. 3 (2001): 579–81.

Hasnine, S. and K. N. Habib. 2018. "What about the Dynamics in Daily Travel Mode Choices? A Dynamic Discrete Choice Approach for Tour-Based Mode Choice Modelling." *Transport Policy* 71 (2018): 70-80.

Her, Y. G. and Z. Yu. 2021. "Mapping the US Census Data Using the TIGER/Line Shapefiles: AE557/AE557, 05/2021." *EDIS* 2021, no. 3 (2021).

Higgins, C. D. and P. S. Kanaroglou. 2016. “A Latent Class Method for Classifying and Evaluating the Performance of Station Area Transit-Oriented Development in the Toronto Region.” *Journal of Transport Geography* 52 (2016): 61-72.

Hoehner, C. M., C. E. Barlow, P. Allen, and M. Schootman. 2012. "Commuting Distance, Cardiorespiratory Fitness, and Metabolic Risk." *American Journal of Preventive Medicine* 42, no. 6 (2012): 571-578.

Hong, J., Q. Shen, and L. Zhang. 2014. “How do Built-Environment Factors Affect Travel Behavior? A Spatial Analysis at Different Geographic Scales.” *Transportation* 41 (2014): 419–440.

Hong, Y., M. Cetin, and Q. Ma. 2020. “Guidelines for Using StreetLight Data for Planning Tasks.” *Virginia Transportation Research Council*. FHWA/VTRC 20-R23 Report (2020): 1–120.

Issarny, V., N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M. A. Gerosa, and A. B. Hamida. 2011. “Service-Oriented Middleware for the Future Internet: State of the Art and Research Directions.” *Journal of Internet Services and Applications* 2, no. 1 (2011): 23–45.

Jetlund, K. and B. Neuhäuser. 2022. *Geographic Information Systems for Transportation*. In: Kresse, W., Danko, D. (eds) Springer Handbook of Geographic Information. Springer Handbooks. Springer, SN - 978-3-030-53125-6 (2022): 707–727.

Kirk, R., J. Frittelli, L. Luther, W. Mallett, and D.R. Peterman. 2012. “Surface Transportation Funding and Programs Under MAP-21.” (P.L. 112-141) Congress Research Service. September 27, 2012. Accessed March 20, 2022. https://sgp.fas.org/crs/misc/R42762.pdf.

Koppelman, F.S. and C. Bhat. 2006. “A Self Instructing Course in Mode Choice Modeling: Multinomial and Nested Logit Models.” US DOT Federal Transit Administration. Modified June30, 2006. Accessed February 25, 2022. http://www.ce.utexas.edu/prof/bhat/COURSES/LM_Draft_060131Final-060630.pdf.

Krizek, K. J. 2003. “Neighborhood Services, Trip purpose, and Tour-Based Travel” *Transportation* 30, no. 4 (2003): 387–410.

Kuby, M., A. Barranda, and C. Upchurch. 2004. "Factors Influencing Light-Rail Station Boardings in the United States." *Transportation Research Part A: Policy and Practice* 38, no. 3 (2004): 223-247.

Kushner, J. A. “City Life in the Age of High Technology.” *The Urban Lawyer* 37, no. 4 (2005): 893–905.

LeSage, J. P. 1997. “Regression Analysis of Spatial Data.” *Journal of Regional Analysis and Policy* 27, no. 1100-2016-89650 (1997): 83–94.

Litman, T. 2010. “Why Manage Transportation Demand?” Victoria Transport Policy Institute. Modified July 18, 2017. Accessed February 27, 2022. http://www.vtpi.org/tdm/tdm51.htm.

Miller, H. J. 1999. “Potential Contributions of Spatial Analysis to Geographic Information Systems for Transportation (GIS-T).” *Geographical Analysis* 31, no. 4 (1999): 373-399.

Mitch, W. 2021. “Northeast Corridor Map.” Northeast Corridor Commission, Established 2010. Accessed February 26, 2022. https://nec-commission.com/commission/.

Montello, D. R. and C. Sas. 2006. “Human Factors of Wayfinding in Navigation.” In W. Karwowski, Ed. *International Encyclopedia of Ergonomics and Human Factors*, 2nd Ed. CRC Press, Taylor & Francis. London, England (2006): 2003 – 2008.

Morang, M. and P. Stevens. 2013. “Yay, Transit! – Using GTFS Data in ArcGIS Network Analyst.” Environmental Sciences Research Institute. 2013. Accessed March 19, 2022. http://www.transit.melindamorang.com/index.html.

NCRTPB (National Capital Region Transportation Planning Board). 2020. User’s Guide for the MWCOG/NCRTPB Travel Demand Forecasting Model, Version 2.3, Build 78. National Capital Region Transportation Planning Board. Vol. 1, April 14, 2020. Accessed November 21, 2022. https://www.mwcog.org/documents/2020/04/22/tpb-version-2378-travel-model-documentation-travel-forecasts-travel-modeling/.

Pieri, G. L. and D. Nelson. 1999. “1997-1998 Survey of Potential American Applications for Emerging Diesel Multiple Unit Technologies.” *Transportation Research Record: Journal of the Transportation Research Board* 1677, no. 1 (1999): 61-72.

Sari, R. F., A. F. Rochim, E. Tangkudung, A. Tan, and T. Marciano. 2017. “Location-Based Mobile Application Software Development: Review of Waze and Other Apps.” *Advanced Science Letters* 23, no. 3 (2017): 2028-2032.

Shunk, G. A. and R. J. Bouchard. 1970. "An Application of Marginal Utility to Travel Mode Choice." *Highway Research Record* 322 (1970): 30-39.

Tallis, F. 2014. “Evaluating Transit and Driving Disaggregated Commutes through GTFS in ArcGIS.” MS thesis, University of Southern California (2014): 76-77.

US Census Bureau. 2019. TIGER/Line Shapefiles. 6 May 2019. Accessed March 18, 2022. https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html.

Vance, C. and R. Hedel. 2007. “The Impact of Urban Form on Automobile Travel: Disentangling Causation from Correlation.” *Transportation* 34, no. 5 (2007): 575–588.

Zandbergen, P. A. 2013. *Python Scripting for ArcGIS*. 1st edition, 380 New York Street, Redlands, CA 92373: Esri Press.

# Appendix
# *Commute GeoCalculator* SOE Template

*Module 1*: AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

[assembly: AssemblyTitle("LRSLocator")]
[assembly: AssemblyDescription("LRSLocator for 10.4.1")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("LRSLocator")]
[assembly: AssemblyCopyright("Copyright ©  2023")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
[assembly: ComVisible(false)]
// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("dd8a0333-fa32-4d30-9e27-31b2d0b0a89c")]
[assembly: AssemblyVersion("2.0.73.0")]
[assembly: AssemblyFileVersion("2.0.73.0")]
[assembly: ESRI.ArcGIS.SOESupport.AddInPackage("LRSLocator", "6469e3e3-890a-
4e56-8992-5b5b5fac3d32",
   Author = "314882",
   Company = "",
   Date = "1/7/2023 1:32:30 AM",
   Description = "",
   TargetProduct = "Server",
   TargetVersion = "10.4",
   Version = "1.9")]
```

*Module 2*: ComReleaser.cs

```
using System;
using System.Collections;
using System.Runtime.InteropServices;

namespace LRSLocator
{
   [Serializable]
   public class ComReleaser : IDisposable
   {
      // Fields
      private ArrayList _array = ArrayList.Synchronized(new ArrayList());

      // Methods
      public void Dispose()
      {
         this.Dispose(true);
         GC.SuppressFinalize(this);
      }

      protected virtual void Dispose(bool disposing)
      {
         int count = this._array.Count;
         for (int i = 0; i < count; i++)
         {
            if ((this._array[i] != null) && Marshal.IsComObject(this._array[i]))
            {
               while (Marshal.ReleaseComObject(this._array[i]) > 0)
               {

               }
            }
         }

         if (disposing)
         {
            this._array = null;
         }
      }

      ~ComReleaser()
      {
         this.Dispose(false);
      }
```

*Module 2*: ComReleaser.cs (*continued*)

```
        public void ManageLifetime(object o)
        {
            this._array.Add(o);
        }

        public static void ReleaseCOMObject(object o)
        {
            if ((o != null) && Marshal.IsComObject(o))
            {
                while (Marshal.ReleaseComObject(o) > 0)
                {

                }
            }
        }
    }
}
```

*Module 3 Summarized*: RESTContext.cs

```
using System.Collections.Specialized;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.SOESupport;
using System.Collections.Generic;

namespace LRSLocator
{

    public class RESTContext
    {
        // REST request
        public NameValueCollection BoundVariables;
        public JsonObject OperationInput; // operations only
        public string OutputFormat;
        public string RequestProperties; // in JSON format

        // REST response
        public string ResponseProperties;

        // SOE properties
        public ServerLogger Logger;
        public IMapServer3 MapServer;
```

*Module 3 Summarized*: RESTContext.cs (*continued*)

```
// Auto Layer Properties
public IFeatureClass HighwayFeatureClass;
public IFeatureClass LrsTransferStreetsFeatureClass;
public IFeatureClass LrsAutoTripMetricFeatureClass;
public IFeatureClass LrsAutoTravMetricFeatureClass;

// Transit Layer Properties
public IFeatureClass RailwayFeatureClass;
public IFeatureClass RailStationFeatureClass;
public IFeatureClass LrsRailTripMetricFeatureClass;
public IFeatureClass LrsRailTravMetricFeatureClass;
public IFeatureClass BuslineFeatureClass;
public IFeatureClass BusStopFeatureClass;
public IFeatureClass TransferStationFeatureClass;
public IFeatureClass LrsBusTripMetricFeatureClass;
public IFeatureClass LrsBusTravMetricFeatureClass;
public IFeatureClass WalkwayFeatureClass;
public IFeatureClass LrsWalkTripMetricFeatureClass;
public IFeatureClass LrsWalkTravMetricFeatureClass;

// Auto Field Properties – Weekend Metrics Fields (WE) Excluded
public string HighwayFeatureClass_RIDField;
public string HighwayFeatureClass_FDFOField;
public string HighwayFeatureClass_TDFOField;
public string LrsTransferStreetsFeatureClass_RIDField;
public string LrsTransferStreetsFeatureClass_ParkPenaltyField;
public string LrsAutoTripMetricFeatureClass_RIDField;
public string LrsAutoTripMetricFeatureClass_FDFOField;
public string LrsAutoTripMetricFeatureClass_TDFOField;
public string LrsAutoTripMetricFeatureClass_WDAllDayAASField;
public string LrsAutoTripMetricFeatureClass_WDEarlyAMAASField;
public string LrsAutoTripMetricFeatureClass_WDPeakAMAASField;
public string LrsAutoTripMetricFeatureClass_WDMidDayAASField;
public string LrsAutoTripMetricFeatureClass_WDPeakPMAASField;
public string LrsAutoTripMetricFeatureClass_WDLatePMAASField;
public string LrsAutoTravMetricFeatureClass_RIDField;
public string LrsAutoTravMetricFeatureClass_FDFOField;
public string LrsAutoTravMetricFeatureClass_TDFOField;
public string LrsAutoTravMetricFeatureClass_WDAllDayAAIField;
public string LrsAutoTravMetricFeatureClass_WDEarlyAMAAIField;
public string LrsAutoTravMetricFeatureClass_WDPeakAMAAIField;
public string LrsAutoTravMetricFeatureClass_WDMidDayAAIField;
public string LrsAutoTravMetricFeatureClass_WDPeakPMAAIField;
public string LrsAutoTravMetricFeatureClass_WDLatePMAAIField;
```

*Module 3 Summarized*: RESTContext.cs (*continued*)

```
// Transit Field Properties – Weekend Metrics Fields (WE) Excluded
public string RailwayFeatureClass_RIDField;
public string RailwayFeatureClass_FDFOField;
public string RailwayFeatureClass_TDFOField;
public string RailStationFeatureClass_GeoID;
public string RailStationFeatureClass_RIDField;
public string RailStationFeatureClass_WDAllDayWaitField;
public string RailStationFeatureClass_WDEarlyAMWaitField;
public string RailStationFeatureClass_WDPeakAMWaitField;
public string RailStationFeatureClass_WDMidDayWaitField;
public string RailStationFeatureClass_WDPeakPMWaitField;
public string RailStationFeatureClass_WDLatePMWaitField;
public string LrsRailTripMetricFeatureClass_MetricKey;
public string LrsRailTripMetricFeatureClass_RIDField;
public string LrsRailTripMetricFeatureClass_FDFOField;
public string LrsRailTripMetricFeatureClass_TDFOField;
public string LrsRailTripMetricFeatureClass_WDAllDayAASField;
public string LrsRailTripMetricFeatureClass_WDEarlyAMAASField;
public string LrsRailTripMetricFeatureClass_WDPeakAMAASField;
public string LrsRailTripMetricFeatureClass_WDMidDayAASField;
public string LrsRailTripMetricFeatureClass_WDPeakPMAASField;
public string LrsRailTripMetricFeatureClass_WDLatePMAASField;
public string LrsRailTravMetricFeatureClass_RIDField;
public string LrsRailTravMetricFeatureClass_FDFOField;
public string LrsRailTravMetricFeatureClass_TDFOField;
public string LrsRailTravMetricFeatureClass_WDAllDayAAIField;
public string LrsRailTravMetricFeatureClass_WDEarlyAMAAIField;
public string LrsRailTravMetricFeatureClass_WDPeakAMAAIField;
public string LrsRailTravMetricFeatureClass_WDMidDayAAIField;
public string LrsRailTravMetricFeatureClass_WDPeakPMAAIField;
public string LrsRailTravMetricFeatureClass_WDLatePMAAIField;
public string BuslineFeatureClass_RIDField;
public string BuslineFeatureClass_FDFOField;
public string BuslineFeatureClass_TDFOField;
public string BuslineFeatureClass_WDAllDayWaitField;
public string BuslineFeatureClass_WDEarlyAMWaitField;
public string BuslineFeatureClass_WDPeakAMWaitField;
public string BuslineFeatureClass_WDMidDayWaitField;
public string BuslineFeatureClass_WDPeakPMWaitField;
public string BuslineFeatureClass_WDLatePMWaitField;
```

*Module 3 Summarized*: RESTContext.cs (*continued*)

```
public string BusStopFeatureClass_GeoID;
public string BusStopFeatureClass_RIDField;
public string TransferStationFeatureClass_RIDField;
public string TransferStationFeatureClass_WDAllDayWaitField;
public string TransferStationFeatureClass_WDEarlyAMWaitField;
public string TransferStationFeatureClass_WDPeakAMWaitField;
public string TransferStationFeatureClass_WDMidDayWaitField;
public string TransferStationFeatureClass_WDPeakPMWaitField;
public string TransferStationFeatureClass_WDLatePMWaitField;
public string LrsBusTripMetricFeatureClass_MetricKey;
public string LrsBusTripMetricFeatureClass_RIDField;
public string LrsBusTripMetricFeatureClass_FDFOField;
public string LrsBusTripMetricFeatureClass_TDFOField;
public string LrsBusTripMetricFeatureClass_WDAllDayAASField;
public string LrsBusTripMetricFeatureClass_WDEarlyAMAASField;
public string LrsBusTripMetricFeatureClass_WDPeakAMAASField;
public string LrsBusTripMetricFeatureClass_WDMidDayAASField;
public string LrsBusTripMetricFeatureClass_WDPeakPMAASField;
public string LrsBusTripMetricFeatureClass_WDLatePMAASField;
public string LrsBusTravMetricFeatureClass_RIDField;
public string LrsBusTravMetricFeatureClass_FDFOField;
public string LrsBusTravMetricFeatureClass_TDFOField;
public string LrsBusTravMetricFeatureClass_WDAllDayAAIField;
public string LrsBusTravMetricFeatureClass_WDEarlyAMAAIField;
public string LrsBusTravMetricFeatureClass_WDPeakAMAAIField;
public string LrsBusTravMetricFeatureClass_WDMidDayAAIField;
public string LrsBusTravMetricFeatureClass_WDPeakPMAAIField;
public string LrsBusTravMetricFeatureClass_WDLatePMAAIField;
public string WalkwayFeatureClass_RIDField;
public string WalkwayFeatureClass_FDFOField;
public string WalkwayFeatureClass_TDFOField;
public string LrsWalkTripMetricFeatureClass_RIDField;
public string LrsWalkTripMetricFeatureClass_FDFOField;
public string LrsWalkTripMetricFeatureClass_TDFOField;
public string LrsWalkTripMetricFeatureClass_WDAllDayAASField;
public string LrsWalkTripMetricFeatureClass_WDEarlyAMAASField;
public string LrsWalkTripMetricFeatureClass_WDPeakAMAASField;
public string LrsWalkTripMetricFeatureClass_WDMidDayAASField;
public string LrsWalkTripMetricFeatureClass_WDPeakPMAASField;
public string LrsWalkTripMetricFeatureClass_WDLatePMAASField;
public string LrsWalkTravMetricFeatureClass_RIDField;
public string LrsWalkTravMetricFeatureClass_FDFOField;
public string LrsWalkTravMetricFeatureClass_TDFOField;
```

*Module 3 Summarized*: RESTContext.cs (*continued*)

```
        public double SearchTolerance;
        public IFeatureClass NARoutes;

        // Relationship Class Fields
        public string RailStopToTripLnRelationshipClass_GeoID;
        public string RailStopToTripLnRelationshipClass_MetricKey;
        public string BusStopToTripLnRelationshipClass_GeoID;
        public string BusStopToTripLnRelationshipClass_MetricKey;

        //--- Auto Network Dataset Elements ---------------
        public IFeatureClass lrsAutoNetworkPath;
        public IFeatureClass NetworkCGCAutoJunctions;
        public INetworkDataset NetworkCGCAuto;
        public string NetworkCGC_AutoNDName;

        //--- Transit Network Dataset Elements -------------
        public IFeatureClass lrsTransitNetworkPath;
        public IFeatureClass NetworkCGCTransitJunctions;
        public INetworkDataset NetworkCGCTransit;
        public string NetworkCGC_TransitNDName;

        //-- Relationship Classes --
        public IRelationshipClass RailStopToTripLnRelationshipClass;
        public IRelationshipClass BusStopToTripLnRelationshipClass;

        //-- map server elements ------------------------
        public IFeatureWorkspace LrsFeatureWorkspace;
        public List<string> FtrsNames;
        public int NbrOfnonMfts;
        public int NbrFtrs;
        public int AllLayerCount;
        public int HasMcount;
        public List<string> ListHasMFtrs;
        public List<ITable> SaTables;
        public List<IRelationshipClass2> RelClasses;
        public IMapLayerInfos MapLayerInfos

        {
    get { return MapServer.GetServerInfo(MapServer.DefaultMapName).MapLayerInfos; }

        }

    }

}
```

*Module 4*: IRESTHandler.cs

```
namespace LRSLocator

{
   /// <summary>
   /// Interface that handles incoming Rest requests
   /// </summary>
   interface IRESTHandler
   {
      /// <summary>
      /// Handles a request to a specific REST resource or operation.
      /// The return value can be JsonObject, string, or byte[].
      /// </summary>
      object HandleRequest(RESTContext context);
   }
}
```

*Module 5*: JSONHelper.cs

```
using System.Collections.Generic;
using System.Text;
using ESRI.ArcGIS.SOESupport;

namespace LRSLocator
{
   /// <summary>
   /// Json Helper Methods
   /// </summary>
   public static class JSONHelper
   {
      public static JsonObject BuildErrorObject(int code, string message, List<string>
details = null)
      {
         JsonObject errorObj = new JsonObject();
         errorObj.AddLong("code", code);
         errorObj.AddString("message", message);
         if (details == null)
         {
            errorObj.AddArray("details", new object[0]);
         }
         else
         {
            errorObj.AddArray("details", details.ToArray());
         }
```

*Module 5*: JSONHelper.cs (*continued*)

```
            JsonObject outer = new JsonObject();
            outer.AddJsonObject("error", errorObj);
            return outer;
        }

        public static byte[] BuildErrorObjectAsBytes(int code, string message, List<string>
    details = null)
        {
            return EncodeResponse(BuildErrorObject(code, message, details));
        }

        public static byte[] EncodeResponse(object response)
        {
            // Handle various output data types
            string strRetval = null;
            if (response is byte[])
            {
                return (byte[])response;
            }
            else if (response is JsonObject)
            {
                strRetval = ((JsonObject)response).ToJson();
            }
            else if (response != null)
            {
                strRetval = response.ToString();
            }
            else
            {
                strRetval = "{}";
            }
            return Encoding.UTF8.GetBytes(strRetval);
        }
    }
}
```

*Module 6 Summarized*: LRSLocator.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections.Specialized;
using System.Runtime.InteropServices;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Server;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.SOESupport;
using ESRI.ArcGIS.DataSourcesGDB;

// Auto and Transit - Commute GeoCalculator

namespace LRSLocator
{
   [ComVisible(true)]
   [Guid("3F3E38DE-3351-4807-B1DB-9897B42DEA90")]
   [ClassInterface(ClassInterfaceType.None)]
   [ServerObjectExtension("MapServer",//use "MapServer" if SOE extends a Map
                                      service and "ImageServer" if it extends an Image service.
      AllCapabilities = "",
      DefaultCapabilities = "",
      Description = "CGC LRS Locator",
      DisplayName = "LRSLocator",
      Properties = "",
      SupportsREST = true,
      SupportsSOAP = false)]
   public class LRSLocator : IServerObjectExtension, IObjectConstruct,
IRESTRequestHandler
   {
      private string soe_name;
      private IPropertySet configProps;
      private IServerObjectHelper serverObjectHelper;
      private ServerLogger logger;
      private IRESTRequestHandler reqHandler;
      // Auto Field Names -----------------------------------------------------------
      private string _highwayFeatureClass_RIDFieldName = "LINEARID";
      private string _highwayFeatureClass_FDFOFieldName = "FROM_M";
      private string _highwayFeatureClass_TDFOFieldName = "TO_M";
      private string _lrsTransferStreets_RIDFieldName = "LINEARID";
      private string _lrsTransferStreets_ParkPenaltyFieldName=“Parking_Penalty_Mins";
      private string _lrsAutoTripMetrics_RIDFieldName = "LINEARID";
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
private string _lrsAutoTripMetrics_FDFOFieldName = "FROM_M";
private string _lrsAutoTripMetrics_TDFOFieldName = "TO_M";
private string _lrsAutoTripMetrics_AllDayAvgSpeed_WkDay =
   "WkDay_AllDay_AAS";
private string _lrsAutoTripMetrics_EarlyAMAvgSpeed_WkDay =
   "WkDay_EarlyAM_AAS";
private string _lrsAutoTripMetrics_PeakAMAvgSpeed_WkDay =
   "WkDay_PeakAM_AAS";
private string _lrsAutoTripMetrics_MidDayAvgSpeed_WkDay =
   "WkDay_MidDay_AAS";
private string _lrsAutoTripMetrics_PeakPMAvgSpeed_WkDay =
   "WkDay_PeakPM_AAS";
private string _lrsAutoTripMetrics_LatePMAvgSpeed_WkDay =
   "WkDay_LatePM_AAS";
private string _lrsAutoTravMetrics_RIDFieldName = "LINEARID";
private string _lrsAutoTravMetrics_FDFOFieldName = "FROM_M";
private string _lrsAutoTravMetrics_TDFOFieldName = "TO_M";
// private string _lrsAutoTravMetrics_ZoneName = "ZONE_NAME";
private string _lrsAutoTravMetrics_AllDayAvgInc_WkDay =
   "WkDay_AllDay_AAI";
private string _lrsAutoTravMetrics_EarlyAMAvgInc_WkDay =
   "WkDay_EarlyAM_AAI";
private string _lrsAutoTravMetrics_PeakAMAvgInc_WkDay =
   "WkDay_PeakAM_AAI";
private string _lrsAutoTravMetrics_MidDayAvgInc_WkDay =
   "WkDay_MidDay_AAI";
private string _lrsAutoTravMetrics_PeakPMAvgInc_WkDay =
   "WkDay_PeakPM_AAI";
private string _lrsAutoTravMetrics_LatePMAvgInc_WkDay =
   "WkDay_LatePM_AAI";
// ------------------------------------------------------------------------
// Transit Field Names -----------------------------------------------------
private string _railwayFeatureClass_RIDFieldName = "LINEARID";
private string _railwayFeatureClass_FDFOFieldName = "FROM_M";
private string _railwayFeatureClass_TDFOFieldName = "TO_M";
private string _railstationFeatureClass_GeoID = "GeoID";
private string _railstationFeatureClass_RIDFieldName = "Station_Name";
private string _railstationFeatureClass_AllDayWait_WkDay =
   "Avg_AllDay_WaitTime_WD";
private string _railstationFeatureClass_EarlyAMWait_WkDay =
   "Avg_EarlyAM_WaitTime_WD";
private string _railstationFeatureClass_PeakAMWait_WkDay =
   "Avg_PeakAM_WaitTime_WD";
private string _railstationFeatureClass_MidDayWait_WkDay =
   "Avg_MidDay_WaitTime_WD";
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
private string _railstationFeatureClass_PeakPMWait_WkDay =
   "Avg_PeakPM_WaitTime_WD";
private string _railstationFeatureClass_LatePMWait_WkDay =
   "Avg_LatePM_WaitTime_WD";
private string _lrsRailTripMetrics_MetricKey = "MetricKey";
private string _lrsRailTripMetrics_RIDFieldName = "LINEARID";
private string _lrsRailTripMetrics_FDFOFieldName = "FROM_M";
private string _lrsRailTripMetrics_TDFOFieldName = "TO_M";
private string _lrsRailTripMetrics_AllDayAvgSpeed_WkDay =
   "WkDay_AllDay_AAS";
private string _lrsRailTripMetrics_EarlyAMAvgSpeed_WkDay =
   "WkDay_EarlyAM_AAS";
private string _lrsRailTripMetrics_PeakAMAvgSpeed_WkDay =
   "WkDay_PeakAM_AAS";
private string _lrsRailTripMetrics_MidDayAvgSpeed_WkDay =
   "WkDay_MidDay_AAS";
private string _lrsRailTripMetrics_PeakPMAvgSpeed_WkDay =
   "WkDay_PeakPM_AAS";
private string _lrsRailTripMetrics_LatePMAvgSpeed_WkDay =
   "WkDay_LatePM_AAS";
private string _lrsRailTravMetrics_RIDFieldName = "LINEARID";
private string _lrsRailTravMetrics_FDFOFieldName = "FROM_M";
private string _lrsRailTravMetrics_TDFOFieldName = "TO_M";
private string _lrsRailTravMetrics_AllDayAvgInc_WkDay =
   "WkDay_AllDay_AAI";
private string _lrsRailTravMetrics_EarlyAMAvgInc_WkDay =
   "WkDay_EarlyAM_AAI";
private string _lrsRailTravMetrics_PeakAMAvgInc_WkDay =
   "WkDay_PeakAM_AAI";
private string _lrsRailTravMetrics_MidDayAvgInc_WkDay =
   "WkDay_MidDay_AAI";
private string _lrsRailTravMetrics_PeakPMAvgInc_WkDay =
   "WkDay_PeakPM_AAI";
private string _lrsRailTravMetrics_LatePMAvgInc_WkDay =
   "WkDay_LatePM_AAI";
private string _buslineFeatureClass_RIDFieldName = "BusRouteID1";
private string _buslineFeatureClass_FDFOFieldName = "FROM_M";
private string _buslineFeatureClass_TDFOFieldName = "TO_M";
private string _buslineFeatureClass_AllDayWait_WkDay =
   "Avg_AllDay_WaitTime_WD";
private string _buslineFeatureClass_EarlyAMWait_WkDay =
   "Avg_EarlyAM_WaitTime_WD";
private string _buslineFeatureClass_PeakAMWait_WkDay =
   "Avg_PeakAM_WaitTime_WD";
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
private string _buslineFeatureClass_MidDayWait_WkDay =
   "Avg_MidDay_WaitTime_WD";
private string _buslineFeatureClass_PeakPMWait_WkDay =
   "Avg_PeakPM_WaitTime_WD";
private string _buslineFeatureClass_LatePMWait_WkDay =
   "Avg_LatePM_WaitTime_WD";
private string _busstopFeatureClass_GeoID = "GeoID";
private string _busstopFeatureClass_RIDFieldName = "BusRouteID1";
private string _transferstationFeatureClass_RIDFieldName = "BusRouteID1";
private string _transferstationFeatureClass_AllDayWait_WkDay =
   "Avg_AllDay_WaitTime_WD";
private string _transferstationFeatureClass_EarlyAMWait_WkDay =
   "Avg_EarlyAM_WaitTime_WD";
private string _transferstationFeatureClass_PeakAMWait_WkDay =
   "Avg_PeakAM_WaitTime_WD";
private string _transferstationFeatureClass_MidDayWait_WkDay =
   "Avg_MidDay_WaitTime_WD";
private string _transferstationFeatureClass_PeakPMWait_WkDay =
   "Avg_PeakPM_WaitTime_WD";
private string _transferstationFeatureClass_LatePMWait_WkDay =
   "Avg_LatePM_WaitTime_WD";
private string _lrsBusTripMetrics_MetricKey = "MetricKey";
private string _lrsBusTripMetrics_RIDFieldName = "BusRouteID1";
private string _lrsBusTripMetrics_FDFOFieldName = "FROM_M";
private string _lrsBusTripMetrics_TDFOFieldName = "TO_M";
private string _lrsBusTripMetrics_AllDayAvgSpeed_WkDay =
   "WkDay_AllDay_AAS";
private string _lrsBusTripMetrics_EarlyAMAvgSpeed_WkDay =
   "WkDay_EarlyAM_AAS";
private string _lrsBusTripMetrics_PeakAMAvgSpeed_WkDay =
   "WkDay_PeakAM_AAS";
private string _lrsBusTripMetrics_MidDayAvgSpeed_WkDay =
   "WkDay_MidDay_AAS";
private string _lrsBusTripMetrics_PeakPMAvgSpeed_WkDay =
   "WkDay_PeakPM_AAS";
private string _lrsBusTripMetrics_LatePMAvgSpeed_WkDay =
   "WkDay_LatePM_AAS";
private string _lrsBusTravMetrics_RIDFieldName = "BusRouteID1";
private string _lrsBusTravMetrics_FDFOFieldName = "FROM_M";
private string _lrsBusTravMetrics_TDFOFieldName = "TO_M";
private string _lrsBusTravMetrics_AllDayAvgInc_WkDay =
   "WkDay_AllDay_AAI";
private string _lrsBusTravMetrics_EarlyAMAvgInc_WkDay =
   "WkDay_EarlyAM_AAI";
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
private string _lrsBusTravMetrics_PeakAMAvgInc_WkDay =
   "WkDay_PeakAM_AAI";
private string _lrsBusTravMetrics_MidDayAvgInc_WkDay =
   "WkDay_MidDay_AAI";
private string _lrsBusTravMetrics_PeakPMAvgInc_WkDay =
   "WkDay_PeakPM_AAI";
private string _lrsBusTravMetrics_LatePMAvgInc_WkDay =
   "WkDay_LatePM_AAI";
private string _walkwayFeatureClass_RIDFieldName = "RouteID";
private string _walkwayFeatureClass_FDFOFieldName = "FROM_M";
private string _walkwayFeatureClass_TDFOFieldName = "TO_M";
private string _lrsWalkTripMetrics_RIDFieldName = "RouteID";
private string _lrsWalkTripMetrics_FDFOFieldName = "FROM_M";
private string _lrsWalkTripMetrics_TDFOFieldName = "TO_M";
private string _lrsWalkTripMetrics_AllDayAvgSpeed_WkDay =
   "WkDay_AllDay_AAS";
private string _lrsWalkTripMetrics_EarlyAMAvgSpeed_WkDay =
   "WkDay_EarlyAM_AAS";
private string _lrsWalkTripMetrics_PeakAMAvgSpeed_WkDay =
   "WkDay_PeakAM_AAS";
private string _lrsWalkTripMetrics_MidDayAvgSpeed_WkDay =
   "WkDay_MidDay_AAS";
private string _lrsWalkTripMetrics_PeakPMAvgSpeed_WkDay =
   "WkDay_PeakPM_AAS";
private string _lrsWalkTripMetrics_LatePMAvgSpeed_WkDay =
   "WkDay_LatePM_AAS";
private string _lrsWalkTravMetrics_RIDFieldName = "RouteID";
private string _lrsWalkTravMetrics_FDFOFieldName = "FROM_M";
private string _lrsWalkTravMetrics_TDFOFieldName = "TO_M";
private string _lrsWalkTravMetrics_AllDayAvgInc_WkDay =
   "WkDay_AllDay_AAI";
private string _lrsWalkTravMetrics_EarlyAMAvgInc_WkDay =
   "WkDay_EarlyAM_AAI";
private string _lrsWalkTravMetrics_PeakAMAvgInc_WkDay =
   "WkDay_PeakAM_AAI";
private string _lrsWalkTravMetrics_MidDayAvgInc_WkDay =
   "WkDay_MidDay_AAI";
private string _lrsWalkTravMetrics_PeakPMAvgInc_WkDay =
   "WkDay_PeakPM_AAI";
private string _lrsWalkTravMetrics_LatePMAvgInc_WkDay =
   "WkDay_LatePM_AAI";
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
//---- Relationship Class Field Names -----------------------------------------
private string _busStopToTripLnRelationshipClass_GeoID = "GeoID";
private string _busStopToTripLnRelationshipClass_MetricKey = "MetricKey";
private string _railStopToTripLnRelationshipClass_GeoID = "GeoID";
private string _railStopToTripLnRelationshipClass_MetricKey = "MetricKey";
// --- Network Dataset Names -------------------------------------------------
private string _cgcAutoND_NetworkName = "AutoNetwork_ND";
private string _cgcTransitND_NetworkName = "TransitNetwork_ND";
private double _searchTolerance = 0.00015;  // version 2.0, this is approx. 55 feet

private IFeatureClass _naRoutes = null;

//-- auto data model feature classes ----------------------------
private IFeatureClass _highwayFeatureClass = null;
private IFeatureClass _lrsAutoTripMetrics = null;
private IFeatureClass _lrsAutoTravMetrics = null;
private IFeatureClass _lrsTransferStreets = null;
private IFeatureClass _lrsAutoNetworkPath = null;
private IFeatureClass _cgcAutoNetJunctions = null;

//-- transit data model feature classes ----------------------------
private IFeatureClass _railwayFeatureClass = null;
private IFeatureClass _railstationFeatureClass = null;
private IFeatureClass _lrsRailTripMetrics = null;
private IFeatureClass _lrsRailTravMetrics = null;
private IFeatureClass _buslineFeatureClass = null;
private IFeatureClass _busstopFeatureClass = null;
private IFeatureClass _transferstationFeatureClass = null;
private IFeatureClass _lrsBusTripMetrics = null;
private IFeatureClass _lrsBusTravMetrics = null;
private IFeatureClass _walkwayFeatureClass = null;
private IFeatureClass _lrsWalkTripMetrics = null;
private IFeatureClass _lrsWalkTravMetrics = null;
private IFeatureClass _lrsTransitNetworkPath = null;
private IFeatureClass _cgcTransitNetJunctions = null;

//-- map server objects -----------------------------------------
private IMapServer3 _mapserver = null;
private IMapServerDataAccess _dataAccess;
private IFeatureWorkspace _gdWorkspace = null;
private INetworkDataset _cgcAutoND = null;
private INetworkDataset _cgcTransitND = null;
private IRelationshipClass _busStopToTripLnRelationshipClass = null;
private IRelationshipClass _railStopToTripLnRelationshipClass = null;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
private List<string> _ftrsNames = null;
private int _nbrFtrs = 0;
private int _nbrOfnonMfts = 0;
private int _allLayerCount = 0;
private int _hasMcount = 0;
private List<string> _listHasMFtrs = null;

private int _saTblCount = 0;
private List<string> _saTblNames = null;
private IStandaloneTableInfos _tableInfos = null;
private List<ITable> _saTables = null;
private List<IRelationshipClass2> _relCls = null;

public LRSLocator()
{
   soe_name = this.GetType().Name;
   logger = new ServerLogger();
   reqHandler = new SoeRestImpl(soe_name, CreateRestSchema()) as
   IRESTRequestHandler;
}

#region IServerObjectExtension Members

public void Init(IServerObjectHelper pSOH)
{
   serverObjectHelper = pSOH;
   IMapServer3 mapServer = serverObjectHelper.ServerObject as IMapServer3;
   IMapServerObjects3 mapServerObjects = mapServer as IMapServerObjects3;

   IMapServerDataAccess dataAccess = (IMapServerDataAccess)mapServer;
   this._mapserver = mapServer;
   this._dataAccess = dataAccess;
   initiateFeatures(mapServer);
}

public void Shutdown()
{
   soe_name = null;
   serverObjectHelper = null;
   logger = null;
}

#endregion
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
        #region IObjectConstruct Members
        public void Construct(IPropertySet props)
        {
            configProps = props;
        }
        #endregion
        #region IRESTRequestHandler Members
        public string GetSchema()
        {
            return reqHandler.GetSchema();
        }

        public byte[] HandleRESTRequest(string Capabilities, string resourceName, string
    operationName, string operationInput, string outputFormat, string requestProperties, out
    string responseProperties)
        {
            return reqHandler.HandleRESTRequest(Capabilities, resourceName,
    operationName, operationInput, outputFormat, requestProperties, out
    responseProperties);
        }
        #endregion
        private RestResource CreateRestSchema()
        {
            RestResource rootRes = new RestResource(soe_name, false, RootResHandler);

            RestOperation getAutoCostFromLatLongOperation = new
    RestOperation("Driving Cost From LatLong",
                          new string[] { "Begin_Longitude", "Begin_Latitude",
    "End_Longitude", "End_Latitude", "Time_Of_Day", "Spatial_Reference" },
                          new string[] { "json" },
                          HandleOp_getAutoCostFromLatLongHandlerOperation);

            RestOperation getTransitCostFromLatLongOperation = new
    RestOperation("Transit Cost From LatLong",
                          new string[] { "Begin_Longitude", "Begin_Latitude",
    "End_Longitude", "End_Latitude", "Time_Of_Day", "Spatial_Reference" },
                          new string[] { "json" },
                          HandleOp_getTransitCostFromLatLongHandlerOperation);

            rootRes.operations.Add(getAutoCostFromLatLongOperation);
            rootRes.operations.Add(getTransitCostFromLatLongOperation);
            return rootRes;
        }
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
        private byte[]
HandleOp_getAutoCostFromLatLongHandlerOperation(NameValueCollection
boundVariables, JsonObject operationInput, string outputFormat, string
requestProperties, out string responseProperties)

        {
            return HandleOperation(new GetAutoCostFromLatLongHandler(),
boundVariables, operationInput, outputFormat, requestProperties, out
responseProperties);
        }

        private byte[]
HandleOp_getTransitCostFromLatLongHandlerOperation(NameValueCollection
boundVariables, JsonObject operationInput, string outputFormat, string
requestProperties, out string responseProperties)
        {
            return HandleOperation(new GetTransitCostFromLatLongHandler(),
boundVariables, operationInput, outputFormat, requestProperties, out
responseProperties);
        }
        //-----------------------------------------------------------------------------------------
        private byte[] RootResHandler(NameValueCollection boundVariables, string
outputFormat, string requestProperties, out string responseProperties)
        {
            responseProperties = null;

            JsonObject result = new JsonObject();
            return Encoding.UTF8.GetBytes(result.ToJson());
        }

        /// A generic internal handler for all REST operations.
        /// The REST operation delegate methods should call this method to benefit
        /// from uniform request processing, response formatting, and exception handling.

        private byte[] HandleOperation(IRESTHandler handler,
                                       NameValueCollection boundVariables,
                                       JsonObject operationInput,
                                       string outputFormat,
                                       string requestProperties,
                                       out string responseProperties)
        {
            RESTContext context = CreateContext(boundVariables, operationInput,
outputFormat, requestProperties);
            return HandleHelper(handler, context, out responseProperties);
        }
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
        /// A generic internal handler for all REST resources.
        /// The REST resource delegate methods should call this method to benefit
        /// from uniform request processing, response formatting, and exception handling.

        private byte[] HandleResource(IRESTHandler handler,
                                      NameValueCollection boundVariables,
                                      string outputFormat,
                                      string requestProperties,
                                      out string responseProperties)
        {
          RESTContext context = CreateContext(boundVariables, null, outputFormat,
    requestProperties);
          return HandleHelper(handler, context, out responseProperties);
        }

        private RESTContext CreateContext(NameValueCollection boundVariables,
                                      JsonObject operationInput,
                                      string outputFormat,
                                      string requestProperties)
        {
          RESTContext context = new RESTContext();
          context.BoundVariables = boundVariables;
          context.OperationInput = operationInput;
          context.OutputFormat = outputFormat;
          context.RequestProperties = requestProperties;
          context.NARoutes = this._naRoutes;

          context.SearchTolerance = _searchTolerance;
          context.LrsFeatureWorkspace = this._gdWorkspace;

          //-------- Auto network analysis layer ----------------------------------------------------
          context.NetworkCGCAuto = this._cgcAutoND;
          //-------- Auto network dataset (edges) ---------------------------------------------------
          context.NetworkCGC_AutoNDName = this._cgcAutoND_NetworkName;
          //---------- Auto network junctions ---------------------------------------------------------
          context.NetworkCGCAutoJunctions = this._cgcAutoNetJunctions;
          //--------- Auto solved network path -------------------------------------------------------
          context.lrsAutoNetworkPath = this._lrsAutoNetworkPath;
          //-----------------------------------------------------------------------------------------
          //-------- Transit network analysis layer -------------------------------------------------
          context.NetworkCGCTransit = this._cgcTransitND;
          //-------- Transit network dataset (edges) ------------------------------------------------
          context.NetworkCGC_TransitNDName = this._cgcTransitND_NetworkName;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
//---------- Transit network junctions ---------------------------------------------------
context.NetworkCGCTransitJunctions = this._cgcTransitNetJunctions;
//--------- Transit solved network path ------------------------------------------------
context.lrsTransitNetworkPath = this._lrsTransitNetworkPath;
//-------------------------------------------------------------------------------------------
//-- Relationship Classes ---------------------------------------------------------------
context.BusStopToTripLnRelationshipClass =
this._busStopToTripLnRelationshipClass;
context.RailStopToTripLnRelationshipClass =
this._railStopToTripLnRelationshipClass;

// -- Auto Network Feature Classes
context.HighwayFeatureClass = this._highwayFeatureClass;
context.LrsAutoTripMetricFeatureClass = this._lrsAutoTripMetrics;
context.LrsAutoTravMetricFeatureClass = this._lrsAutoTravMetrics;
context.LrsTransferStreetsFeatureClass = this._lrsTransferStreets;

// -- Auto Network layer fields
context.HighwayFeatureClass_RIDField =
this._highwayFeatureClass_RIDFieldName;
context.HighwayFeatureClass_FDFOField =
this._highwayFeatureClass_FDFOFieldName;
context.HighwayFeatureClass_TDFOField =
this._highwayFeatureClass_TDFOFieldName;

// -- Transfer Streets Parking Penalty fields
context.LrsTransferStreetsFeatureClass_RIDField =
this._lrsTransferStreets_RIDFieldName;
context.LrsTransferStreetsFeatureClass_ParkPenaltyField =
this._lrsTransferStreets_ParkPenaltyFieldName;

// -- Auto Trip Speed fields – Weekend Fields (WE) Excluded
context.LrsAutoTripMetricFeatureClass_RIDField =
this._lrsAutoTripMetrics_RIDFieldName;
context.LrsAutoTripMetricFeatureClass_FDFOField =
this._lrsAutoTripMetrics_FDFOFieldName;
context.LrsAutoTripMetricFeatureClass_TDFOField =
this._lrsAutoTripMetrics_TDFOFieldName;
context.LrsAutoTripMetricFeatureClass_WDAllDayAASField =
this._lrsAutoTripMetrics_AllDayAvgSpeed_WkDay;
context.LrsAutoTripMetricFeatureClass_WDEarlyAMAASField =
this._lrsAutoTripMetrics_EarlyAMAvgSpeed_WkDay;
context.LrsAutoTripMetricFeatureClass_WDPeakAMAASField =
this._lrsAutoTripMetrics_PeakAMAvgSpeed_WkDay;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
context.LrsAutoTripMetricFeatureClass_WDMidDayAASField =
this._lrsAutoTripMetrics_MidDayAvgSpeed_WkDay;
context.LrsAutoTripMetricFeatureClass_WDPeakPMAASField =
this._lrsAutoTripMetrics_PeakPMAvgSpeed_WkDay;
context.LrsAutoTripMetricFeatureClass_WDLatePMAASField =
this._lrsAutoTripMetrics_LatePMAvgSpeed_WkDay;

// -- Auto Traveller Income fields – Weekend Fields (WE) Excluded
context.LrsAutoTravMetricFeatureClass_RIDField =
this._lrsAutoTravMetrics_RIDFieldName;
context.LrsAutoTravMetricFeatureClass_FDFOField =
this._lrsAutoTravMetrics_FDFOFieldName;
context.LrsAutoTravMetricFeatureClass_TDFOField =
this._lrsAutoTravMetrics_TDFOFieldName;
context.LrsAutoTravMetricFeatureClass_WDAllDayAAIField =
this._lrsAutoTravMetrics_AllDayAvgInc_WkDay;
context.LrsAutoTravMetricFeatureClass_WDEarlyAMAAIField =
this._lrsAutoTravMetrics_EarlyAMAvgInc_WkDay;
context.LrsAutoTravMetricFeatureClass_WDPeakAMAAIField =
this._lrsAutoTravMetrics_PeakAMAvgInc_WkDay;
context.LrsAutoTravMetricFeatureClass_WDMidDayAAIField =
this._lrsAutoTravMetrics_MidDayAvgInc_WkDay;
context.LrsAutoTravMetricFeatureClass_WDPeakPMAAIField =
this._lrsAutoTravMetrics_PeakPMAvgInc_WkDay;
context.LrsAutoTravMetricFeatureClass_WDLatePMAAIField =
this._lrsAutoTravMetrics_LatePMAvgInc_WkDay;

// -- Transit Network Feature Classes
context.RailwayFeatureClass = this._railwayFeatureClass;
context.RailStationFeatureClass = this._railstationFeatureClass;
context.LrsRailTripMetricFeatureClass = this._lrsRailTripMetrics;
context.LrsRailTravMetricFeatureClass = this._lrsRailTravMetrics;
context.BuslineFeatureClass = this._buslineFeatureClass;
context.BusStopFeatureClass = this._busstopFeatureClass;
context.TransferStationFeatureClass = this._transferstationFeatureClass;
context.LrsBusTripMetricFeatureClass = this._lrsBusTripMetrics;
context.LrsBusTravMetricFeatureClass = this._lrsBusTravMetrics;
context.WalkwayFeatureClass = this._walkwayFeatureClass;
context.LrsWalkTripMetricFeatureClass = this._lrsWalkTripMetrics;
context.LrsWalkTravMetricFeatureClass = this._lrsWalkTravMetrics;

// -- Transit Network layer fields
context.RailwayFeatureClass_RIDField =
this._railwayFeatureClass_RIDFieldName;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
context.RailwayFeatureClass_FDFOField =
this._railwayFeatureClass_FDFOFieldName;
context.RailwayFeatureClass_TDFOField =
this._railwayFeatureClass_TDFOFieldName;
context.BuslineFeatureClass_RIDField =
this._buslineFeatureClass_RIDFieldName;
context.BuslineFeatureClass_FDFOField =
this._buslineFeatureClass_FDFOFieldName;
context.BuslineFeatureClass_TDFOField =
this._buslineFeatureClass_TDFOFieldName;
context.WalkwayFeatureClass_RIDField =
this._walkwayFeatureClass_RIDFieldName;
context.WalkwayFeatureClass_FDFOField =
this._walkwayFeatureClass_FDFOFieldName;
context.WalkwayFeatureClass_TDFOField =
this._walkwayFeatureClass_TDFOFieldName;

// -- Transit Transfer Wait Time fields – Weekend Fields (WE) Excluded
context.RailStationFeatureClass_WDAllDayWaitField =
this._railstationFeatureClass_AllDayWait_WkDay;
context.RailStationFeatureClass_WDEarlyAMWaitField =
this._railstationFeatureClass_EarlyAMWait_WkDay;
context.RailStationFeatureClass_WDPeakAMWaitField =
this._railstationFeatureClass_PeakAMWait_WkDay;
context.RailStationFeatureClass_WDMidDayWaitField =
this._railstationFeatureClass_MidDayWait_WkDay;
context.RailStationFeatureClass_WDPeakPMWaitField =
this._railstationFeatureClass_PeakPMWait_WkDay;
context.RailStationFeatureClass_WDLatePMWaitField =
this._railstationFeatureClass_LatePMWait_WkDay;
context.BuslineFeatureClass_WDAllDayWaitField =
this._buslineFeatureClass_AllDayWait_WkDay;
context.BuslineFeatureClass_WDEarlyAMWaitField =
this._buslineFeatureClass_EarlyAMWait_WkDay;
context.BuslineFeatureClass_WDPeakAMWaitField =
this._buslineFeatureClass_PeakAMWait_WkDay;
context.BuslineFeatureClass_WDMidDayWaitField =
this._buslineFeatureClass_MidDayWait_WkDay;
context.BuslineFeatureClass_WDPeakPMWaitField =
this._buslineFeatureClass_PeakPMWait_WkDay;
context.BuslineFeatureClass_WDLatePMWaitField =
this._buslineFeatureClass_LatePMWait_WkDay;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
context.TransferStationFeatureClass_RIDField =
this._transferstationFeatureClass_RIDFieldName;
context.TransferStationFeatureClass_WDAllDayWaitField =
this._transferstationFeatureClass_AllDayWait_WkDay;
context.TransferStationFeatureClass_WDEarlyAMWaitField =
this._transferstationFeatureClass_EarlyAMWait_WkDay;
context.TransferStationFeatureClass_WDPeakAMWaitField =
this._transferstationFeatureClass_PeakAMWait_WkDay;
context.TransferStationFeatureClass_WDMidDayWaitField =
this._transferstationFeatureClass_MidDayWait_WkDay;
context.TransferStationFeatureClass_WDPeakPMWaitField =
this._transferstationFeatureClass_PeakPMWait_WkDay;
context.TransferStationFeatureClass_WDLatePMWaitField =
this._transferstationFeatureClass_LatePMWait_WkDay;

// == Transit Trip Speed fields – Weekend Fields (WE) Excluded
context.RailStationFeatureClass_GeoID = this._railstationFeatureClass_GeoID;
context.RailStationFeatureClass_RIDField =
this._railstationFeatureClass_RIDFieldName;
context.LrsRailTripMetricFeatureClass_MetricKey =
this._lrsRailTripMetrics_MetricKey;
context.LrsRailTripMetricFeatureClass_RIDField =
this._lrsRailTripMetrics_RIDFieldName;
context.LrsRailTripMetricFeatureClass_FDFOField =
this._lrsRailTripMetrics_FDFOFieldName;
context.LrsRailTripMetricFeatureClass_TDFOField =
this._lrsRailTripMetrics_TDFOFieldName;
context.LrsRailTripMetricFeatureClass_WDAllDayAASField =
this._lrsRailTripMetrics_AllDayAvgSpeed_WkDay;
context.LrsRailTripMetricFeatureClass_WDEarlyAMAASField =
this._lrsRailTripMetrics_EarlyAMAvgSpeed_WkDay;
context.LrsRailTripMetricFeatureClass_WDPeakAMAASField =
this._lrsRailTripMetrics_PeakAMAvgSpeed_WkDay;
context.LrsRailTripMetricFeatureClass_WDMidDayAASField =
this._lrsRailTripMetrics_MidDayAvgSpeed_WkDay;
context.LrsRailTripMetricFeatureClass_WDPeakPMAASField =
this._lrsRailTripMetrics_PeakPMAvgSpeed_WkDay;
context.LrsRailTripMetricFeatureClass_WDLatePMAASField =
this._lrsRailTripMetrics_LatePMAvgSpeed_WkDay;

context.BusStopFeatureClass_GeoID = this._busstopFeatureClass_GeoID;
context.BusStopFeatureClass_RIDField =
this._busstopFeatureClass_RIDFieldName;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
context.LrsBusTripMetricFeatureClass_MetricKey =
this._lrsBusTripMetrics_MetricKey;
context.LrsBusTripMetricFeatureClass_RIDField =
this._lrsBusTripMetrics_RIDFieldName;
context.LrsBusTripMetricFeatureClass_FDFOField =
this._lrsBusTripMetrics_FDFOFieldName;
context.LrsBusTripMetricFeatureClass_TDFOField =
this._lrsBusTripMetrics_TDFOFieldName;
context.LrsBusTripMetricFeatureClass_WDAllDayAASField =
this._lrsBusTripMetrics_AllDayAvgSpeed_WkDay;
context.LrsBusTripMetricFeatureClass_WDEarlyAMAASField =
this._lrsBusTripMetrics_EarlyAMAvgSpeed_WkDay;
context.LrsBusTripMetricFeatureClass_WDPeakAMAASField =
this._lrsBusTripMetrics_PeakAMAvgSpeed_WkDay;
context.LrsBusTripMetricFeatureClass_WDMidDayAASField =
this._lrsBusTripMetrics_MidDayAvgSpeed_WkDay;
context.LrsBusTripMetricFeatureClass_WDPeakPMAASField =
this._lrsBusTripMetrics_PeakPMAvgSpeed_WkDay;
context.LrsBusTripMetricFeatureClass_WDLatePMAASField =
this._lrsBusTripMetrics_LatePMAvgSpeed_WkDay;
context.LrsWalkTripMetricFeatureClass_RIDField =
this._lrsWalkTripMetrics_RIDFieldName;
context.LrsWalkTripMetricFeatureClass_FDFOField =
this._lrsWalkTripMetrics_FDFOFieldName;
context.LrsWalkTripMetricFeatureClass_TDFOField =
this._lrsWalkTripMetrics_TDFOFieldName;
context.LrsWalkTripMetricFeatureClass_WDAllDayAASField =
this._lrsWalkTripMetrics_AllDayAvgSpeed_WkDay;
context.LrsWalkTripMetricFeatureClass_WDEarlyAMAASField =
this._lrsWalkTripMetrics_EarlyAMAvgSpeed_WkDay;
context.LrsWalkTripMetricFeatureClass_WDPeakAMAASField =
this._lrsWalkTripMetrics_PeakAMAvgSpeed_WkDay;
context.LrsWalkTripMetricFeatureClass_WDMidDayAASField =
this._lrsWalkTripMetrics_MidDayAvgSpeed_WkDay;
context.LrsWalkTripMetricFeatureClass_WDPeakPMAASField =
this._lrsWalkTripMetrics_PeakPMAvgSpeed_WkDay;
context.LrsWalkTripMetricFeatureClass_WDLatePMAASField =
this._lrsWalkTripMetrics_LatePMAvgSpeed_WkDay;

// -- Trnasit Traveler Income fields – Weekend Fields (WE) Excluded
context.LrsRailTravMetricFeatureClass_RIDField =
this._lrsRailTravMetrics_RIDFieldName;
context.LrsRailTravMetricFeatureClass_FDFOField =
this._lrsRailTravMetrics_FDFOFieldName;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
context.LrsRailTravMetricFeatureClass_TDFOField =
this._lrsRailTravMetrics_TDFOFieldName;
context.LrsRailTravMetricFeatureClass_WDAllDayAAIField =
this._lrsRailTravMetrics_AllDayAvgInc_WkDay;
context.LrsRailTravMetricFeatureClass_WDEarlyAMAAIField =
this._lrsRailTravMetrics_EarlyAMAvgInc_WkDay;
context.LrsRailTravMetricFeatureClass_WDPeakAMAAIField =
this._lrsRailTravMetrics_PeakAMAvgInc_WkDay;
context.LrsRailTravMetricFeatureClass_WDMidDayAAIField =
this._lrsRailTravMetrics_MidDayAvgInc_WkDay;
context.LrsRailTravMetricFeatureClass_WDPeakPMAAIField =
this._lrsRailTravMetrics_PeakPMAvgInc_WkDay;
context.LrsRailTravMetricFeatureClass_WDLatePMAAIField =
this._lrsRailTravMetrics_LatePMAvgInc_WkDay;
context.LrsBusTravMetricFeatureClass_RIDField =
this._lrsBusTravMetrics_RIDFieldName;
context.LrsBusTravMetricFeatureClass_FDFOField =
this._lrsBusTravMetrics_FDFOFieldName;
context.LrsBusTravMetricFeatureClass_TDFOField =
this._lrsBusTravMetrics_TDFOFieldName;
context.LrsBusTravMetricFeatureClass_WDAllDayAAIField =
this._lrsBusTravMetrics_AllDayAvgInc_WkDay;
context.LrsBusTravMetricFeatureClass_WDEarlyAMAAIField =
this._lrsBusTravMetrics_EarlyAMAvgInc_WkDay;
context.LrsBusTravMetricFeatureClass_WDPeakAMAAIField =
this._lrsBusTravMetrics_PeakAMAvgInc_WkDay;
context.LrsBusTravMetricFeatureClass_WDMidDayAAIField =
this._lrsBusTravMetrics_MidDayAvgInc_WkDay;
context.LrsBusTravMetricFeatureClass_WDPeakPMAAIField =
this._lrsBusTravMetrics_PeakPMAvgInc_WkDay;
context.LrsBusTravMetricFeatureClass_WDLatePMAAIField =
this._lrsBusTravMetrics_LatePMAvgInc_WkDay;
context.LrsWalkTravMetricFeatureClass_RIDField =
this._lrsWalkTravMetrics_RIDFieldName;
context.LrsWalkTravMetricFeatureClass_FDFOField =
this._lrsWalkTravMetrics_FDFOFieldName;
context.LrsWalkTravMetricFeatureClass_TDFOField =
this._lrsWalkTravMetrics_TDFOFieldName;
context.LrsWalkTravMetricFeatureClass_WDAllDayAAIField =
this._lrsWalkTravMetrics_AllDayAvgInc_WkDay;
context.LrsWalkTravMetricFeatureClass_WDEarlyAMAAIField =
this._lrsWalkTravMetrics_EarlyAMAvgInc_WkDay;
context.LrsWalkTravMetricFeatureClass_WDPeakAMAAIField =
this._lrsWalkTravMetrics_PeakAMAvgInc_WkDay;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
            context.LrsWalkTravMetricFeatureClass_WDMidDayAAIField =
            this._lrsWalkTravMetrics_MidDayAvgInc_WkDay;
            context.LrsWalkTravMetricFeatureClass_WDPeakPMAAIField =
            this._lrsWalkTravMetrics_PeakPMAvgInc_WkDay;
            context.LrsWalkTravMetricFeatureClass_WDLatePMAAIField =
            this._lrsWalkTravMetrics_LatePMAvgInc_WkDay;

            // -- Relationship Class Fields
            context.RailStopToTripLnRelationshipClass_GeoID =
            this._railStopToTripLnRelationshipClass_GeoID;
            context.RailStopToTripLnRelationshipClass_MetricKey =
            this._railStopToTripLnRelationshipClass_MetricKey;
            context.BusStopToTripLnRelationshipClass_GeoID =
            this._busStopToTripLnRelationshipClass_GeoID;
            context.BusStopToTripLnRelationshipClass_MetricKey =
            this._busStopToTripLnRelationshipClass_MetricKey;

            context.NbrOfnonMfts = this._nbrOfnonMfts;
            context.NbrFtrs = this._nbrFtrs;
            context.FtrsNames = this._ftrsNames;
            context.AllLayerCount = this._allLayerCount;
            context.HasMcount = this._hasMcount;
            context.ListHasMFtrs = this._listHasMFtrs;
            context.SaTables = this._saTables;
            context.RelClasses = this._relCls;

            context.MapServer = this._mapserver;

            return context;
        }

        /// A generic internal handler for all REST resources and operations.
        private byte[] HandleHelper(IRESTHandler handler, RESTContext context, out
    string responseProperties)
        {
            object response = null;
            try
            {
                response = handler.HandleRequest(context);
                responseProperties = context.ResponseProperties;
            }
            catch (Exception e)
            {
                response = null; // JsonBuilder.BuildErrorObject(500, e.Message);
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
                responseProperties = null;
                Console.WriteLine(e.Message);
                // throw e;
            }
            return JSONHelper.EncodeResponse(response);
        }

        private void initiateFeatures(IMapServer3 mapServer)
        {
            IMapServerDataAccess dataAccess = (IMapServerDataAccess)mapServer;
            IMapServerInfo msInfo =
            mapServer.GetServerInfo(mapServer.DefaultMapName);

            IMapLayerInfos layerInfos = msInfo.MapLayerInfos;
            this._mapserver = mapServer;
            this._dataAccess = dataAccess;

            //--Checkpoint ---------------------------------------------------
            int layerCount = layerInfos.Count;
            int featureLyrCount = 0;
            List<string> fcNames = new List<string>();
            int nonMFtrsCount = 0;
            int hasMCount = 0;
            int allLayersCount = layerCount;
            List<string> saTblNames = new List<string>();
            List<string> listOfMftrs = new List<string>();
            //---------------------------------------------------------------

            for (int j = 0; j < layerCount; j++)
            {
                IMapLayerInfo layerInfo = layerInfos.get_Element(j);

                if (layerInfo.IsFeatureLayer)
                {
                    featureLyrCount++;

                    IFeatureClass featureClass =
(IFeatureClass)dataAccess.GetDataSource(mapServer.DefaultMapName, layerInfo.ID);
                    IGeometryDef geometryDef =
featureClass.Fields.get_Field(featureClass.FindField(featureClass.ShapeFieldName)).Ge
ometryDef;

                    fcNames.Add(featureClass.AliasName);
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
            if (featureClass != null)
            {
               if (geometryDef.HasM)
               {
                  hasMCount++;
                  listOfMftrs.Add(featureClass.AliasName);

                  if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.Auto_TAZ_Network")  //1
                  {
                     // Use this as Route FeatureClass
                     this._highwayFeatureClass = featureClass;
                  }
                  else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.AutoTripMetrics") //2
                  {
                     this._lrsAutoTripMetrics = featureClass;
                  }
                  else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.AutoTravMetrics") //3
                  {
                     this._lrsAutoTravMetrics = featureClass;
                  }
                  else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.Rail_TAZ_Network") //4
                  {
                     this._railwayFeatureClass = featureClass;
                  }
                  else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.RailTripMetrics") //5
                  {
                     this._lrsRailTripMetrics = featureClass;
                  }
                  else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.RailTravellerMetrics") //6
                  {
                     this._lrsRailTravMetrics = featureClass;
                  }
                  else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.Bus_TAZ_Network") //7
                  {
                     this._buslineFeatureClass = featureClass;
                  }
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
                    else if (featureClass.AliasName ==
            "SANDBOX_STATIC.GIS.BusTripMetrics") //8
                    {
                       this._lrsBusTripMetrics = featureClass;
                    }
                    else if (featureClass.AliasName ==
            "SANDBOX_STATIC.GIS.BusTravellerMetrics") //9
                    {
                       this._lrsBusTravMetrics = featureClass;
                    }
                    else if (featureClass.AliasName ==
            "SANDBOX_STATIC.GIS.Ped_TAZ_Network") //10
                    {
                       this._walkwayFeatureClass = featureClass;
                    }
                    else if (featureClass.AliasName ==
            "SANDBOX_STATIC.GIS.PedTripMetrics") //11
                    {
                       this._lrsWalkTripMetrics = featureClass;
                    }
                    else if (featureClass.AliasName ==
            "SANDBOX_STATIC.GIS.PedTravellerMetrics") //12
                    {
                       this._lrsWalkTravMetrics = featureClass;
                    }
                    else
                    {
                       this._naRoutes = featureClass;
                    }
                    //System.Diagnostics.Debugger.Break();
                 }
                 else
                 {
                    if ((featureClass.AliasName != "Stops") || (featureClass.AliasName !=
         "Barriers") || (featureClass.AliasName != "PolylineBarriers") || (featureClass.AliasName
         != "PolygonBarriers"))
                    {
                       if (featureClass.AliasName ==
            "SANDBOX_STATIC.GIS.Transfer_Street_Stations") //13
                       {
                          nonMFtrsCount++;
                          this._lrsTransferStreets = featureClass;
                       }
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
                    if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.Rail_TAZ_Stations") //14
                    {
                       nonMFtrsCount++;
                       this._railstationFeatureClass = featureClass;
                    }
                    if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.Bus_TAZ_Stops") //15
                    {
                       nonMFtrsCount++;
                       this._busstopFeatureClass = featureClass;
                    }
                    if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.Transit_Transfer_Stations") //16
                    {
                       nonMFtrsCount++;
                       this._transferstationFeatureClass = featureClass;
                    }
                    else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.AutoNetwork_ND_Junctions") //17
                    {
                       nonMFtrsCount++;
                       this._cgcAutoNetJunctions = featureClass;
                    }
                    else if (featureClass.AliasName ==
"SANDBOX_STATIC.GIS.TransitNetwork_ND_Junctions") //18
                    {
                       nonMFtrsCount++;
                       this._cgcTransitNetJunctions = featureClass;
                    }
                    else
                    {
                       this._naRoutes = featureClass;
                    }
                 }
              }

           }// end if Feature
        }
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
            if (layerInfo.Name == "RouteAuto")
            {
               INetworkDataset nwkAutoDataset =
(INetworkDataset)dataAccess.GetDataSource(mapServer.DefaultMapName,
layerInfo.ID);
               this._cgcAutoND = nwkAutoDataset;
            }
            else if (layerInfo.Name == "RouteTransit")
            {
               INetworkDataset nwkTransitDataset =
(INetworkDataset)dataAccess.GetDataSource(mapServer.DefaultMapName,
layerInfo.ID);
               this._cgcTransitND = nwkTransitDataset;
            }
            this._nbrFtrs = featureLyrCount;
            this._ftrsNames = fcNames;
            this._nbrOfnonMfts = nonMFtrsCount;
            this._hasMcount = hasMCount;
            this._allLayerCount = allLayersCount;
            this._listHasMFtrs = listOfMftrs;
         }
         IMapServerObjects3 msObj = (IMapServerObjects3)mapServer;
         //get map server info
         IMapServerInfo3 msInfo3 = (IMapServerInfo3)msInfo;
         IStandaloneTableInfos tableInfos = msInfo3.StandaloneTableInfos;
         this._tableInfos = tableInfos;

         //get any standalone table info collection
         List<ITable> saTables = new List<ITable>();
         List<IRelationshipClass2> relClasses = new List<IRelationshipClass2>();
         ITable table = null;
         IRelationshipClass2 relCl = null;
         if (tableInfos != null)
         {
            int tableCount = tableInfos.Count;
            int? tableID = null;
            for (int j = 0; j < tableCount; j++)
            {
               IStandaloneTableInfo tableInfo = tableInfos.get_Element(j);
               //tableInfo.Name = "";
               saTblNames.Add(tableInfo.Name);
               tableID = tableInfo.ID;
```

*Module 6 Summarized*: LRSLocator.cs (*continued*)

```
                    if (tableID != null)
                    {
                        table = msObj.get_StandaloneTable(mapServer.DefaultMapName,
Convert.ToInt16(tableID));
                        saTables.Add(table);
                        relCl = (IRelationshipClass2)table;
                        relClasses.Add(relCl);
                    }
                }
            }
            this._saTables = saTables;
            int standAloneTablesCount = tableInfos.Count;
            this._saTblCount = tableInfos.Count; ;
            this._saTblNames = saTblNames;
            this._relCls = relClasses;
        }
    }
}
```

*Module 7*: RouteFromInputPoint.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.Serialization;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.SOESupport;
using ESRI.ArcGIS.SystemUI;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Server;
using ESRI.ArcGIS.Output;
using ESRI.ArcGIS.GISClient;
using ESRI.ArcGIS.DataSourcesFile;
using ESRI.ArcGIS.DataSourcesGDB;
using ESRI.ArcGIS.DataSourcesOleDB;
using ESRI.ArcGIS.DataSourcesRaster;
using ESRI.ArcGIS.GeoDatabaseDistributed;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geoprocessing;
using ESRI.ArcGIS.NetworkAnalyst;

namespace LRSLocator
{
   class RouteFromInputPoints
   {
      [DataMember(Order = 0, Name = "network_path")]
      public IFeature network_path { get; set; }

      public void SimpleRouteSetupSolveAndSaveWorkflow(INetworkDataset
netDataset, List<IPoint> inputPoints, RESTContext _context)
      {
         IFeatureWorkspace imFtWorkspace =
(IFeatureWorkspace)CreateInMemoryWorkspace();
         IFeatureClass wayPointFC = CreateFeatureClassFromPoints(imFtWorkspace,
inputPoints);

         //var networkDataset = _context.networkTxdot,  as the input parameter for the
class method, getting the object from the RESTContext/Context Object Model-COM.
         var networkDataset = netDataset;
```

*Module 7*: RouteFromInputPoint.cs (*continued*)

```
        var deNetworkDataset = ((IDatasetComponent)networkDataset).DataElement as
    IDENetworkDataset;

        // Set up your solver
        var routeSolver = new NARouteSolverClass() as INASolver;
        INASolverSettings naSolverSettings = routeSolver as INASolverSettings;

        // Set up your context by creating it, then binding it to a network dataset.
        var context = routeSolver.CreateContext(deNetworkDataset, "Path from points
    Context") as INAContext;
        var contextEdit = context as INAContextEdit;
        IGPMessages gpMessages = new GPMessagesClass();
        contextEdit.Bind(networkDataset, gpMessages);

        // Load new stops using the input feature class and a NAClassLoader.
        var inputStopsFClass = wayPointFC;
        var cursor = inputStopsFClass.Search(null, false) as ICursor;
        var classLoader = new NAClassLoaderClass() as INAClassLoader;
        classLoader.NAClass = context.NAClasses.get_ItemByName("Stops") as
    INAClass;
        classLoader.Locator = context.Locator;
        int rowsInCursor = 0;
        int rowsLocated = 0;
        classLoader.Load(cursor, null, ref rowsInCursor, ref rowsLocated);

        // Solve the route using current settings
        // And check the GPMessages after a successful solve to see if there are any
    warning or informational messages.

        try
        {
           bool IsPartialSolution = routeSolver.Solve(context, gpMessages, null);
        }
        catch (Exception e)
        {
           string ex = e.ToString();
           Console.WriteLine(e.ToString());
        }

        // Get the FeatureClass containing the route results.
        // Iterate over the route class features' attribute values to examine the results.
        var routesClass = context.NAClasses.get_ItemByName("Routes") as
    IFeatureClass;
```

*Module 7*: RouteFromInputPoint.cs (*continued*)

```
            IFeature resultFC = null;
            IFeatureCursor pFeatureCursor = routesClass.Search(null, false);
            IFeature pFeature;
            int num1 = 0;
            while ((pFeature = pFeatureCursor.NextFeature()) != null)
            {
               resultFC = pFeature;
               num1++;
            }

            this.network_path = resultFC;

            //--SAVE THE NETWORK PATH TO LOCAL DRIVE FOR VISUAL
            INTERPRETATION OF THE RESULT----
            string outputFilePath = @"\\<full machine
name>\\D$\Projects\CGCLocator\Output\AutoRte.lyr";
            //------------------------------------------------------------------------------
            try
            {
               INALayer3 naLayer = routeSolver.CreateLayer(context) as INALayer3;
               ILayerFile layerfile = new LayerFileClass();
               layerfile.New(outputFilePath);
               layerfile.ReplaceContents(naLayer as ILayer);
               layerfile.Save();
               layerfile.Close();
            }
            catch (Exception e)
            {
               string ex = e.ToString();
               Console.WriteLine(e.Message);
            }
         }
```

*Module 7*: RouteFromInputPoint.cs (*continued*)

```
        private IWorkspace CreateInMemoryWorkspace()
        {
            try
            {
                // Create an in-memory workspace factory.
                IWorkspaceFactory workspaceFactory = new InMemoryWorkspaceFactory()
as IWorkspaceFactory;

                // Create a new in-memory workspace. This returns a name object.
                IWorkspaceName workspaceName = workspaceFactory.Create(string.Empty,
"MyWorkspace", null, 0);
                IName name = (IName)workspaceName;

                // Open the workspace through the name object.
                IWorkspace workspace = (IWorkspace)name.Open();

                return workspace;
            }

            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                throw e;
            }
        }
        //------------------------------------------------------------------------------------------------
        public static IFeatureClass CreateFeatureClassFromPoints(IFeatureWorkspace
featWorkspace, List<IPoint> inputPts)
        {
            String fcName = "StopPoints";

            IFieldsEdit ptFieldsEdit = new FieldsClass();
            IFieldEdit ptField = new FieldClass();

            IPoint pt = inputPts[0];
            ISpatialReference ptSR = pt.SpatialReference;


            ptField = new FieldClass();
            ptField.Type_2 = esriFieldType.esriFieldTypeOID;
            ptField.Name_2 = "OBJECTID";
            ptField.AliasName_2 = "OBJECTID";
            ptFieldsEdit.AddField(ptField);
```

*Module 7*: RouteFromInputPoint.cs (*continued*)

```
            IGeometryDefEdit ptGeomDef;
            ptGeomDef = new GeometryDefClass();
            ptGeomDef.GeometryType_2 = esriGeometryType.esriGeometryPoint;
            ptGeomDef.SpatialReference_2 = ptSR;
            ptGeomDef.HasZ_2 = false;

            ptField = new FieldClass();
            ptField.Name_2 = "SHAPE";
            ptField.AliasName_2 = "SHAPE";
            ptField.Type_2 = esriFieldType.esriFieldTypeGeometry;
            ptField.GeometryDef_2 = ptGeomDef;
            ptFieldsEdit.AddField(ptField);

            ptField = new FieldClass();
            ptField.Name_2 = "stopOrder";
            ptField.AliasName_2 = "stopOrder";
            ptField.Type_2 = esriFieldType.esriFieldTypeInteger;
            ptFieldsEdit.AddField(ptField);

            IFeatureClass ptFC = featWorkspace.CreateFeatureClass(fcName, ptFieldsEdit,
        null, null, esriFeatureType.esriFTSimple, "SHAPE", "");

            // Add data to Feature class setting the "stopOrder" field with position order of
        respecively to the position in the list
            int position = 0;
            int ptIdx;
            foreach (IPoint pnt in inputPts)
            {
               position++;
               IFeature stopFeature = ptFC.CreateFeature();
               stopFeature.Shape = pnt;
               ptIdx = stopFeature.Fields.FindField("stopOrder");
               stopFeature.set_Value(ptIdx, position);
               stopFeature.Store();
            }
            return ptFC;
         }
      }
   }
```