# A FEDERATED SIMULATION APPROACH TO MODELING PORT AND ROADWAY OPERATIONS

A Thesis
Presented to
The Academic Faculty

by

Thomas A. Wall

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science in the
School of Civil and Environmental Engineering

Georgia Institute of Technology
May 2010

# A FEDERATED SIMULATION APPROACH TO MODELING PORT AND ROADWAY OPERATIONS

Approved by:

Dr. Michael P. Hunter, Advisor
School of Civil and Environmental Engineering
*Georgia Institute of Technology*

Dr. Michael O. Rogers
School of Civil and Environmental Engineering
*Georgia Institute of Technology*

Dr. Michael D. Meyer
School of Civil and Environmental Engineering
*Georgia Institute of Technology*

Date Approved: April 05, 2010

# ACKNOWLEDGEMENTS

I would like to thank my adviser, Dr. Michael P. Hunter for the knowledge and support he has provided throughout this study, and for his thorough review of this thesis. I would like to thank Dr. Michael O. Rodgers, the principal investigator for the Port of Savannah research project for the opportunity to be a part of this study, and for his review of this thesis. I would to thank Dr. Michael D. Meyer for his review of this thesis and for his moral support throughout this study.

I am indebted to Matthew Roe for contributing his time and deep knowledge of computer programming and databasing systems to support study. I would like to thank Christopher Puglisi for his initial efforts in developing the basic federation method built upon in this study, and especially the method of time and management. I would like to thank Lakshmi Peesapati for developing the initial version of the Arena© port model used in this study, and Franklin Gbologah for his continued support for that model throughout the study. Lastly, I would like to thank Dwayne Henclewood for his informative discussions of VISSIM© modeling methods and his general support, both moral and technical, throughout this study.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

This research develops a computer simulation method for federating an Arena© port operations model and a VISSIM© roadway network operations model. The development of this method is inspired by the High Level Architecture (HLA) standard for federating simulations, and incorporates several elements of the HLA principles into its design. The federated simulation model is then tested using a time-lag experiment to demonstrate the presence of feedback loops between federated model components wherein changes to input parameters of one model during runtime can be shown to affect the operational performance of the other model. This experiment also demonstrates how several initial transient phase and steady state operating characteristics of the federated system can be determined from the federation output data.

The results indicate that the method developed in this study is capable of capturing the dynamic interaction of two models in federated simulation. It is shown that feedback loops can exist between two models in federated simulation. Most notably, the federation output shows that increased traffic volume in the roadway network model influences the accumulation of containers in the port terminal queue of the port model. The federation output also shows that increased container volume leaving the port terminal model affects both port and road truck utilization, as well as the total number of port trucks in the roadway network model.

Challenges and future directions for research in federating transportation-related simulations are also presented.

# CHAPTER 1

# INTRODUCTION

Advances in computer simulation methods over the past decades have allowed engineers to construct meaningful models of increasingly complex systems. These simulation modeling techniques enable engineers to better understand and analyze real-world systems in a risk-free environment, to test assumptions and to preview possible outcomes [1]. In doing so, simulation modeling provides an analytically based decision tool to better inform engineers and planners prior to implementing designs in the real-world environment. Transportation engineering, where infrastructure systems and components can be costly to design and implement, has especially benefitted since the advent of computer-based simulation. The simulation of traffic movement enables engineers and planners to model potentially multi-million dollar systems and arrive at meaningful conclusions prior to any real-world design implementation. Similarly, computer simulation has enabled the modeling of complex logistics systems such as railway networks, air traffic operations, and freight facilities. This, in turn, allows system managers to analyze system performance and identify areas where operational efficiency can be enhanced.

As modeling techniques have become increasingly capable, so too has the level of system complexity they are able to model. However, the focus of work in computer simulation – particularly in engineering applications – has primarily centered on individual models or individual modeling software packages. This effectively limits the breadth and complexity of simulation modeling endeavors, especially for transportation

systems. Furthermore, it inhibits the ability to explore the interaction between transportation systems in a modeling environment.

The objective of this study is to develop and test a technique to effectively combine, or federate, a seaport operation logistics model and a traffic network simulation model, each constructed using different simulation software packages. For now, a federated simulation can be broadly defined as "a composable set of interacting simulations." [2] The model federation is then tested to determine if a feedback loop can be established between the two models and to determine time-based characteristics of that interaction.

### 1.1 Problem and Motivation

The projected expansion of the Port of Savannah over the coming decades will increase the container traffic moving through the port, as well as on the surrounding roadway network. The current throughput capacity of the Garden City Terminal, at the Port of Savannah, is 2.62 million twenty-foot equivalent units (TEUs) per year. Standard shipping containers are typically 40 feet in length, or two TEUs. By 2018, the Georgia Port Authority projects that that terminal's throughput capacity will grow to 6 million TEUs per year [3].

The Port of Savannah employs trucks to transport containers both within the port, and between the port and nearby intermodal facilities (i.e., distribution centers and rail facilities). Container transportation by truck within the port exclusively utilizes the port's private roads and facilities. However, truck transport of containers to nearby intermodal

facilities utilizes the public roadway network surrounding the port, and is therefore mixed with public traffic.

A 100% increase in the Port of Savannah container traffic within the next decade will necessitate an increase in the number of trucks required to carry the additional containers to and from nearby intermodal facilities. Therefore, the question arises, how will this related increase in truck traffic generated by the port impact the surrounding roadway network? Similarly, it is unknown how the performance of the surrounding roadway network will impact the operational efficiency of the Port of Savannah, given the context of this growth in freight volume.

## 1.2 Modeling Approach Overview

The operations of a port and surrounding roadways are typically modeled in disparate simulation software packages that provide no dynamic feedback between the two models. The modeling challenge is to effectively combine these two models into a larger federated simulation that encompasses both systems. Puglisi (2008) proposed an initial effort to federate a port simulation and traffic simulation [4]. This study will therefore build upon Puglisi's work to develop a fully federated, large scale platform for modeling both roadway transportation and port operations systems.

The following will introduce and briefly discuss these two computer modeling software packages. A more complete discussion of the specific models used in this study is presented in Chapter 3 – Methodology.

### 1.2.1 Port of Savannah and Arena©

To simulate the operations of the Port of Savannah, a model was created using Rockwell Automation Arena© 12.0. Arena© is a discrete event-based, object-oriented simulation software package that implements the SIMAN language, a general purpose simulation language [5, 6]. An object-oriented simulation is one in which one "considers the software in terms of objects and how those objects interact with each other." [6] When considering the port's operations, one can reasonably treat containers and trucks as objects that interact with one another through various processes in a series of sequential events. These events, detailing not only with an object's movements through the system, but also with the underlying operation of the system itself, are carried out in sequential order according to an event calendar. The event calendar is essentially a very detailed timeline that specifies the sequence of events that need to occur as the simulation steps forward through time. As the events for the current time step are completed, future events are scheduled on the event calendar for future time steps, and the simulation clock is advanced [7]. By this method, the simulation clock advances "to the next scheduled event, regardless of the amount of time between events." [6] This is an important distinction from the commonly utilized traffic simulation models in that that Arena© does not employ a continuous simulation clock, but advances through time from one event to the next. Therefore, whereas one time step in Arena© may be a tenth of a second, other time steps may be several seconds in length; this simply depends on the time-spacing of events on the event calendar.

The major components of this Arena© port model, capturing seaport terminal, storage, inspections and railroad operations were developed by Lakshmi Peesapati and

Franklin Gbologah as part of a companion Georgia Tech Research Institute research effort [8]. Other components of the port model capturing roadway, trucking and intermodal facility operations were constructed specifically for this federated modeling effort. These components will be identified and discussed in greater detail in later sections of this document.

### 1.2.2 Roadway Network and VISSIM©

A model of the roadway network surrounding the Port of Savannah was created using PTV-VISSIM© 5.10. VISSIM© is a microscopic, behavior based traffic simulation model [9]. It differs from Arena© in that it employs a continuous simulation clock that institutes a consistent, continuous time-step advancement through simulation time [4, 9].

Networks in VISSIM© are constructed by means of links (representing sections of roadway) and connectors (connecting vehicle movements between links, typically at intersections).

## 1.3 Study Overview

The remainder of this study is broken into the following sections. First, a discussion of general computer simulation, transportation-specific simulation and federated simulation approaches will be presented to give context to this study and provide a base of knowledge for the methodological approach. Second, each of the study methodology and federated model elements will be presented in detail. Third, after acquainting the reader with the finished model, the design of experiment will be

presented. This experiment will test the presence of feedback loops between each model and determine time-related characteristics of how the federation readjusts to steady-state operation following changes in operational parameters of each federate during runtime. Fourth, the results of the experiment will be presented. Fifth, a discussion of these results will be presented, accompanied by a discussion of potential sources of error in the experiment. Lastly, conclusions from this study and future research needs will be presented.

# CHAPTER 2

# LITERATURE REVIEW & BACKGROUND

To provide context for this study, this section will introduce simulation, as well as the motivation and methods for federating simulations. This will include a discussion of the evolution of simulation, distributed simulations and parallel simulations, and the high-level architecture (HLA). It will then discuss simulations in a transportation-specific context and the state of current research in that area.

## 2.1 Computer Simulation

Chung (2004) describes simulation modeling and analysis, in its most basic form, as "the process of creating and experimenting with a computerized mathematical model of a physical system." [10] These physical systems typically fall within one of three categories: manufacturing systems (e.g., warehousing, machining, assembly, materials handling and production facilities), service systems (e.g., medical, retail, food service, information technology, and customer service facilities), and transportation systems (e.g., traffic operations, airport operations, port operations, rail and transit, and distribution logistics) [10].

Ni (2006) suggests that there are four primary reasons for "generating an electronic version of the real world." [1] Namely, it provides: (1) a tool for learning and understanding the physical world and its phenomena, (2) a basis on which it is risk-free to experiment and test assumptions, (3) a means to predict by allowing preview of possible

outcomes, and (4) a decision tool to show the effects by means of visualization [1]. These four motivating factors are similar to those identified in [5, 7, 10, 11].

In the evolution of computer simulation since its advent in the 1960s, Pegden (2005) describes the shift from time-driven modeling to event-driven, and now process-driven modeling [12]. One consequence of this is that the prevalence of event and process driven modeling approaches has motivated a more object-oriented approach to modeling – that is, the interaction among physical objects with other objects and processes in a system.

Another distinction – one of special importance to this study – is the treatment of time in the simulated model world, namely event-based versus continuous or time-stepped simulations. Simulation time refers to the representation of time within the model, or the simulated world. Within the simulation model, simulation time is controlled and measured by the simulation clock [5]. Conversely, real time refers to the actual, real-world time outside of the simulation. Because of the speed of computer processors and current simulation modeling software, simulation time can occur at several times the speed of real time. In some cases, the execution of several simulation hours of a complex system model can be completed in mere minutes, or even seconds, of real time [13].

As mentioned above, Arena© is an event-driven simulation software package. This means that events on an event calendar are executed for the current time step, future events are scheduled on the event calendar, and the simulation clock advances the current simulation time to the next event on the calendar. Because simulation time is broken into intervals according to the time-spacing of events on the event calendar, the advancement

of simulation time, via the simulation clock, may not be continuous. For example, if one event occurs at time t = 10 seconds and the next event on the event calendar occurs at time t = 15 seconds, the simulation clock effectively "skips" from t = 10 seconds to t = 15 seconds. As there are no events occurring in the intervening five seconds of simulation time, there is no need – in the context event-driven simulation – to step through each of the five intermediate seconds of simulation time during which no events occur.

In the realm of traffic simulation, the underlying nature of event-based simulation is a significant limitation in modeling vehicle queuing habits and the continuous nature of traffic flow [6]. For this reason, VISSIM© and other commercially available traffic simulators utilize a time-stepped approach to simulate the continuous nature of time necessary for traffic simulation. Time-stepped simulation subdivides the time between the beginning and end of a simulation into equally sized time step intervals. As the model advances continuously from one equal interval to the next, the entire state of the simulation is recomputed [14]. The resolution of a continuous time-stepped model is defined by the length of each equal time-step. In traffic simulation, these time-step intervals are commonly on the order of seconds or tenths of seconds.

## 2.2 Transportation-Specific Simulation

Transportation and traffic flow simulation modeling are typically broken into three categories according to the levels of model detail: macroscopic, mesoscopic and microscopic [1, 13]. The spectrum from macroscopic to microscopic modeling largely reflects the evolution of computer technology and its ability to process large amounts of data.

Macroscopic models are time-driven simulation processes that treat traffic as a compressible fluid. Through the law of mass conservation, that which enters a system should be equal to that which exists, plus any storage within the system [1]. So, by inputting traffic density, speed and volume characteristics, system characteristics and traffic phenomena can be calculated during each time interval using fundamental traffic flow theory and differential calculus [13].

Mesoscopic traffic models treat traffic as "discrete particles without mass and personality," rather than as a continuous fluid [1]. While this does give some greater level of detail (and requires greater computing power than macroscopic simulation), mesoscopic traffic simulation operates using some pre-defined set of local rules to govern traffic flow. One such example would be maximum-speed constraints on traffic flow [1].

Microscopic traffic simulation diverges from macroscopic and mesoscopic traffic simulation in that it incorporates behavior-based vehicle following algorithms to model the movements of individual vehicles. Behavior-based vehicle following simulation implies that the behavior of each individual vehicle is based on its interaction with the vehicle immediately ahead [13], the lane configuration, traffic composition and traffic signals [9]. Because of the large volume of calculations entailed for each successive time step of continuous microscopic simulation, this method reflects the most recent evolution of traffic simulation and the need for greater computing power than for macroscopic and mesoscopic simulation. VISSIM© 5.10 is a microscopic traffic simulation modeling program.

## 2.3 Methods for Federating Simulators

This sub-section will introduce the concept of federated simulation and discuss motivations and methods for federating distributed and parallel simulations. Although this study does not utilize distributed or parallel computing techniques, methods of federating such simulations provide an excellent framework for federating multiple, disparate simulation models on a single-processor computing platform. The primary federation method discussed is high-level architecture (HLA). This sub-section will also discuss transportation-specific motivations for implementing federated simulations and the current state of research in this area.

### 2.3.1 Distributed and Parallel Simulation

One of the primary limitations of computer simulation is the time required to execute a simulation. While advances in computing and information processing technologies have significantly reduced simulation execution times, there has been a push to investigate other techniques to reduce simulation time. Central to this effort has been the development of parallel simulation and distributed simulation. These methods have significantly evolved since their initial investigation by Chandy and Misra (1979) in their seminal case study of distributed simulation [15, 16].

Fujimoto (2000) distinguishes parallel and distributed simulation as "technologies that enable a simulation program to execute on a computing system containing multiple processors, such as personal computers, interconnected by a communication network." [11] Parallel simulation refers generally to simulation conduced on a computing platform with multiple processors in close physical proximity. These processors are connected by

a central, customized switch and can have either shared or distributed system memory. Distributed simulation refers to simulation conducted on a computing platform with multiple processors distributed among multiple workstations, each with its own memory. These workstations can be located within close physical proximity to one another or great physical distance, and are connected by local area networks (LANs) or wide area networks (WANs) [11].

In addition to the significant reductions in simulation execution time by subdividing simulation computation across multiple processors, Fujimoto (2000) identifies three additional benefits of parallel and distributed simulation [11]: (1) geographical distribution (i.e., physical space constraints may be overcome as computers may be distributed across multiple, different locations), (2) integrating simulators that execute on machines from different manufacturers, and (3) fault tolerance (i.e., should one processor in the system fail, its workload can be shifted to the remaining processors).

### 2.3.2 Motivation for Federated Simulation

Thus far, the discussion of parallel and distributed simulations has been with the implication that multiple simulations, utilizing either similar or disparate software, can be conducted across multiple processors for the purposes of increasing simulation speed. However, there is another important motivation for such simulations. Kewly et al (2008) notes that "large single models that enable analysis of systems of systems are not effective because no single modeling effort can account for all of the [system's] complexities." [17] The suggestion is that "subsystem [simulation] models that

effectively analyze different domains" be constructed and then combined in a federated simulation to analyze the larger system of federated simulations [17].

However, different from the parallel and distributed computing methods outlined above, this type of federation could combine disparate simulation modeling software packages on either multiple or single processor computing platforms. This insight is particularly relevant as this study will federate two disparate simulators on a single processor computing platform.

### 2.3.3 Federated Simulation Implementation

Synchronization and simulation time management, data management and efficient information exchange, and architecture independence [18] are central to the methods of parallel and distributed simulation. The management of these functions can be conducted through the federation of simulations.

Multiple methods have emerged to standardize federated simulations. These methods include parallel discrete event simulation (PDES), Distributed Interactive Simulation (DIS), Aggregate Level Simulation Protocol (ALSP) and the High Level Architecture (HLA) [19], to name a few.

Most notably, High Level Architecture (HLA) was developed by the US Department of Defense (DoD) Modeling and Simulation Coordination Office (formerly the Defense Modeling Simulation Office) [20] to establish a common and broadly applicable technical architecture for simulation federation by merging DIS and ALSP into a single architecture [11]. Dahman (1997) notes that the motivation for developing the HLA is "based on the premise that no one simulation can solve all the…functional

needs for modeling and simulation." [2] To that end, the primary goal of the HLA is to provide a common architecture standard to "facilitate the interoperability among simulations and promote reuse of simulations and their components." [21]

Specific to the HLA, Dahman and Lutz (1998) define a federation as "a named set of simulations interacting via the services of the HLA Runtime Infrastructure (RTI) and in accordance with a common object model and a common HLA rule set to achieve some desired purpose." [21] Put more plainly, individual simulations, called federates [22], are aggregated together to form a larger, more cohesive system simulation called a federation. Under the HLA, these federates interface according to the structural basis of (1) the HLA Rules, (2) the HLA Infrastructure Specification, (3) the Object Model Template (OMT) [2, 21-23].

Broadly, the HLA rules define the underlying design principles used in the HLA [11]. Chief among those design principles are ensuring proper communication and synchronization during runtime [22].

The HLA Interface Specification defines how the federates should communicate amongst themselves, as well as interface with the runtime infrastructure (RTI). Tan (2005) describes the RTI as "a supporting software which provides services that the federates use to coordinate their operations and data exchange during a federation execution." [22] Dahmann (1997) defines six classes of services [2]: (1) federation management, to support basic functions of the federation, (2) declaration management, to define the data provided by federates during execution, (3) object management, to manage creation, deletion and identification services at the object level (4) ownership management, to manage the dynamic ownership of objects and attributes during

execution, (5) time management, to support synchronization of data exchange during runtime simulation, and (6) data distribution management, to facilitate the efficient routing of data among federates.

The Object Model Template (OMT) is intended to provide a standard by which shareable elements, or objects, of the simulations should be defined. The primary goal of the OMT is to facilitate reusability and interoperability of simulation models [21]. The two types of object models specified in the OMT are the HLA Federation Object Model (FOM) and the HLA Simulation Object Model (SOM). For simplification, the FOM describes objects and elements that are common across the federation. The SOM defines objects and elements of individual federates, in terms of the types of information that they can provide. Dahmann (1998) notes that the SOM is distinct from information internal to each federate; the SOM instead defines the information available from

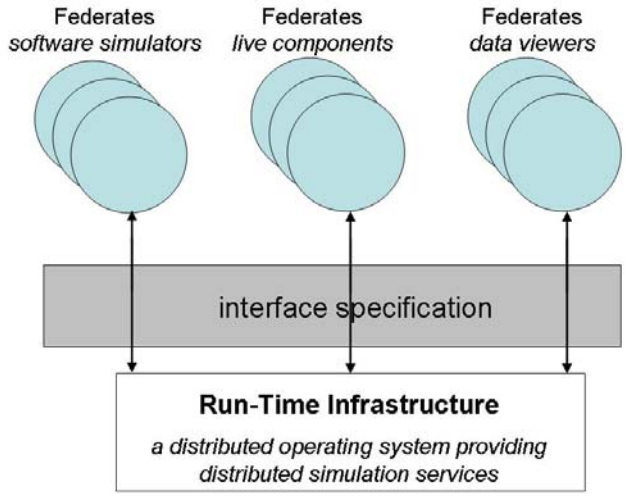



**Figure 1. Components of an HLA in federation**
**(Figure Credit: Parallel and Distributed Simulation Systems [11])**

federates for the purpose of assessing a federate's appropriateness for incorporation into future federated simulations [21]. Figure 1 shows the relationship of components in an HLA federation.

As noted, the HLA is applicable to both federations utilizing parallel and distributed multi-processor computing methods, and to federations of multiple disparate models using a single-processor computer.

### 2.3.4 Current Research in Transportation Simulation Federation

In the field of transportation, current research in federating simulations is somewhat limited in scope and extremely focused. The primary objective of most recent studies is to increase simulation speed [13, 14, 24, 25] by breaking one single monolithic model into several distributed sub-models and distributing these federates across multiple processors. Notably, Klein et al (1998) investigates and tests an HLA-based prototype distributed simulation of an urban traffic model [26]. Other applications have been to increase simulation speed by employing distributed traffic simulation techniques for real-time traffic analysis [13]. Federating disparate simulation software packages has not been a focus in current transportation simulation research.

### 2.3.5 Relevance of this Study Given Current Research

As noted, most recent research in federating traffic simulators has been to break one monolithic simulation model into several distributed sub-models. Nonetheless, the statement by Kewly et al (2008) that large single models cannot account for all of the complexities when modeling a system of systems is extremely relevant given the context

of multimodal transportation [17]. Accurately modeling the Port of Savannah operations and those of the surrounding roadway network introduces such complexity and specialization that it is beyond the capabilities of a single, monolithic simulation model. Port operations are most effectively modeled using a discrete, event-based logistics simulator. As noted, such a simulator is ineffective at modeling the nature of traffic flow, and so a continuous, time-step traffic microscopic simulator is a more appropriate application for the roadway system surrounding the port.

The challenge is then to integrate these two disparate system models into a larger, federated system model; and to ensure that the larger model captures not only the interaction of the two smaller system sub-models, but also the operation of the larger system as a whole. The HLA structure provides an excellent backdrop to federate multiple sub-models for this effort.

Because of the nascent nature of research in federating disparate transportation simulators, it would be overly ambitious for initial investigations to employ parallel or distributed multi-processor simulation. As such, this study employs a method utilizing a single-processor workstation computer as the computing platform. Also, this study is not intended to be fully HLA compliant. Still, the principal concepts of the HLA are employed, such as standardization of object data types and the use of an RTI to govern synchronization, time management and data management.

The future of the HLA in transportation simulation holds much promise for the reuse and interoperability of simulation models. It is the intention that this study will provide the basis for future work in federating disparate transportation models with greater or full HLA compliance.

### 2.3.6 Previous Efforts in Federating Port and Transportation Models

Initial efforts to create a federated port and transportation simulation were undertaken by Puglisi (2008) using the Port of Savannah terminal model developed by Peesapati and Gbologah [4]. In his study, Puglisi used Microsoft Excel© spreadsheet software as a federation database. He then built the federation RTI utilizing the Visual Basic for Applications© capabilities built in to the Microsoft Excel© federation database. Puglisi then tested and validated his federation using a simplified roadway network model federate before incorporating the more expansive roadway network model surrounding the Port of Savannah.

The federation effort of this study builds on three cornerstone elements of the Puglisi study: (1) the method of time management, (2) the database-oriented method of data and object management, and (3) the custom-built Arena© Vehicle Input and Container Input entity creation blocks (these will be discussed in Chapter 3).

To manage time, Puglisi effectively allowed the VISSIM© model federate and Arena© model federate to sequentially step forward through simulation time. The RTI first allowed the VISSIM© model to advance by one uniform time step (1 second) before pausing the VISSIM© simulation clock. The RTI then allowed the Arena© model federate to advance through as many simulation events as was necessary for its simulation clock to catch up to, or match, the time of the VISSIM© simulation clock, before pausing. The process was then repeated for the next time-step by advancing the VISSIM© model federate, and so on. As will be discussed later, this study utilizes the same time management methodology.

Object and data management in the Puglisi study was done using multiple worksheets in a Microsoft Excel© spreadsheet. When transactions of containers and vehicles occurred between federates, the details of those transactions were logged in the spreadsheet. This study utilizes the same methodology of tracking and recording object and federation data only when transactions between federates occur.

The final major contribution of the Puglisi study was the construction of custom vehicle and container input blocks in the Arena© model. Arena© has no standard module that allows for the real-time creation of an entity (e.g., a vehicle or container) during simulation execution. To circumvent this deficiency, Puglisi utilized the Template Development capability of Arena© and custom-built an ad hoc logical process that allows entities to be created in real-time during simulation by the RTI. These blocks were slightly modified and then widely used in this study. The exact construction of these blocks will be discussed in Section 3.3.1.1.4.

# CHAPTER 3

# METHODOLOGY

This section will describe the methodology behind the federated simulation used in this study. The first subsection will present a conceptual overview of the system being modeled. It will describe the physical system and then describe the federation – the roles of each component and the ways in which they interact. The subsections that follow will provide a deeper description or each individual model component. These will include the Port of Savannah model, constructed in Arena©; the roadway network, constructed in VISSIM©; the RTI, constructed using Microsoft Visual Studio©; and the database system constructed using Microsoft Access©.

## 3.1 Port System Operational Overview

The physical system being modeled is the Port of Savannah, located in Savannah, GA, and the surrounding state and local roadway network. Operations of the port system are distributed across several geographically spaced components, including the Garden City Terminal (GCT), three major distribution centers, and the Interstate 16 (I-16) junction. Each of these port system components are interconnected by the state highway and surface roadway network. Figure 2 shows an aerial photograph depicting the geographical layout and orientation of the port facilities that are incorporated into this study's model. Note that in the case of the three distribution centers and the I-16 junction, many components of the Port of Savannah system are separated from one another by several miles.

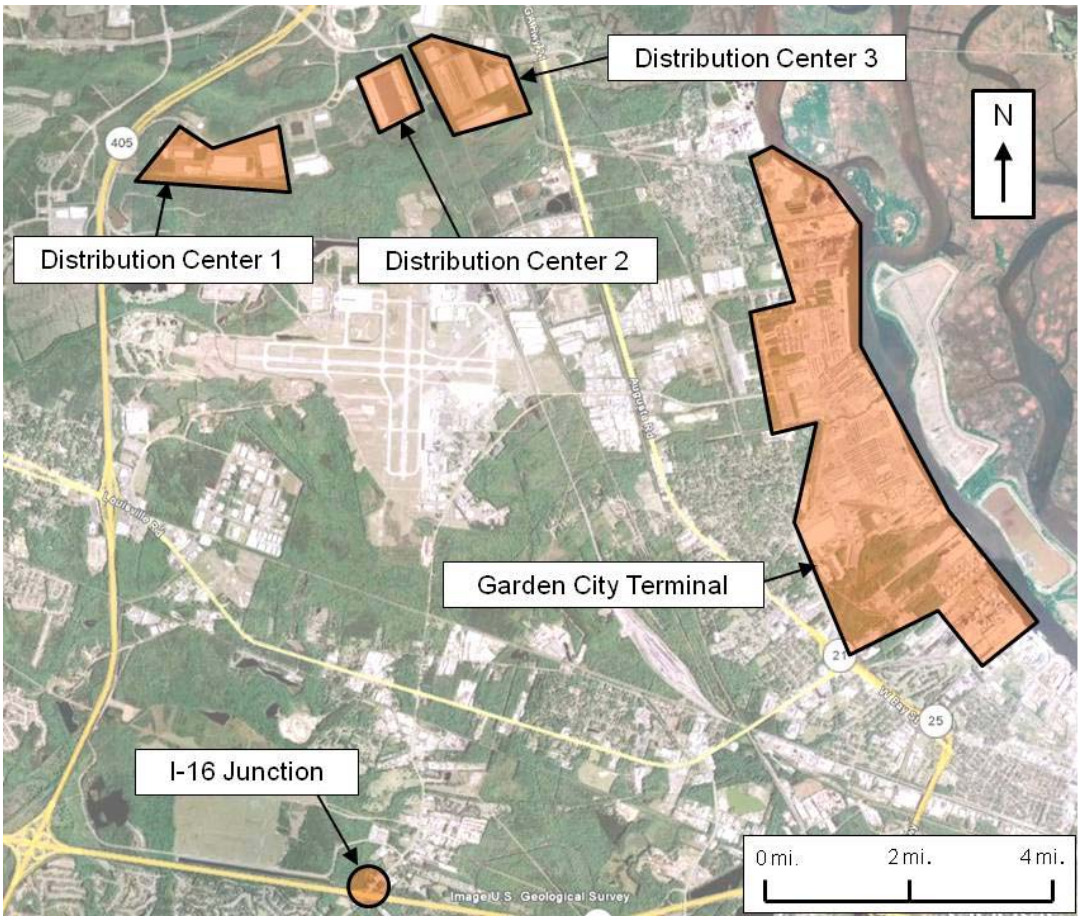



**Figure 2. Port of Savannah Geographical Orientation**
**(Figure Credit: Background image from Google Earth [27])**

### 3.1.1 Port Container Origins and Destinations

The primary focus of the port model is to reasonably reflect the flow of containers into and out of the Port of Savannah by truck. There are five container origins/destinations in the port model: (1) distribution center 1, (2) distribution center 2, (3) distribution center 3, (4) the GCT, and (5) the I-16 Junction.

Containers are generated at two of these origin locations: (1) the GCT, where they are offloaded from container ships, or (2) Interstate 16 (I-16), where long-distance trucking carriers enter the model from the interstate highway system. Similarly,

containers are deleted from the model at these two locations: (1) the GCT where they are loaded onto ships, and (2) the I-16 Junction where they leave the local roadway network to enter the interstate highway system.

Distribution Centers 1, 2, and 3 serve as intermediate destinations for containers. Containers are neither created nor deleted at the distribution centers. Instead, they are offloaded from trucks, processed for some interval of time, and reloaded on to outgoing trucks in route to their next destinations.

When containers are generated at the GCT, they can have one of four destinations: one of the three distribution centers or the I-16 junction. Similarly, when containers are generated at the I-16 Junction from long-distance trucking, they can have one of four destinations: one of the three distribution centers or the GCT.

### 3.1.2 Port and Roadway Trucks

As noted, the transport of containers among Port of Savannah facilities is done by truck. There are two truck categories: (1) port trucks and (2) privately owned, long-distance roadway trucks (or simply, "roadway trucks"). Port trucks exclusively move containers among the four port facilities (i.e., the GCT and Distribution Centers 1, 2, and 3). Port trucks are initially generated at the GCT during simulation initialization. The quantity of port trucks generated is specified by the user. As these port trucks only carry containers among the four port destinations, they remain in circulation and are never removed from the federated system during simulation.

Private long-distance roadway trucks exclusively transport containers between one of the four port facilities and the I-16 junction, where trucks and containers exit the

simulation. An initial, user-specified quantity of long-distance roadway trucks is also generated at the GCT during simulation initialization. Throughout simulation execution, empty roadway trucks and roadway trucks carrying containers are also randomly generated at the I-16 Junction to simulate arrival from the interstate highway system. During simulation execution, roadway trucks are removed from the simulation at the I-16 junction where they enter the interstate highway system.

### 3.1.3 Port Roadway Network

The major roadways included in this study are GA Highway 21, South Coastal Highway, Dean Forest Rd./Bourne Ave. and Jimmy de Loach Parkway. Figure 3 shows these roads in relation to the port, distribution centers, and the I-16 junction. All traffic

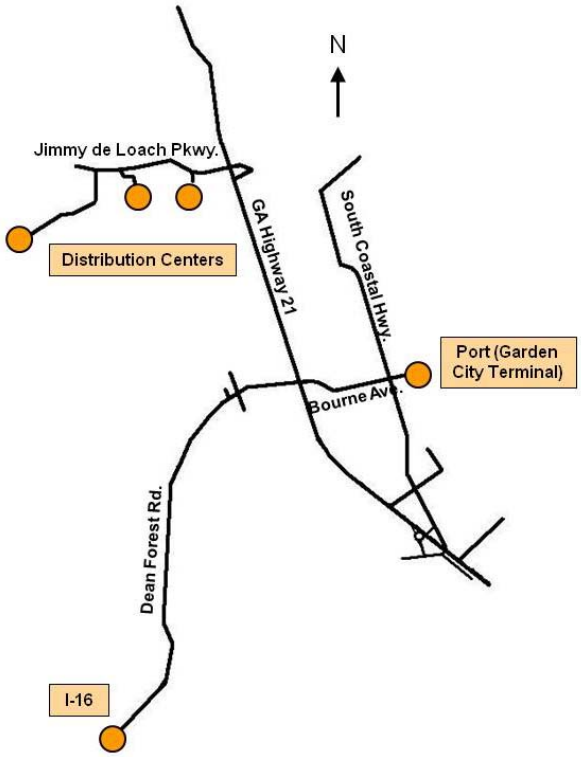



**Figure 3. Port of Savannah Roadway Network**

flow between the three distribution centers and the port is routed along Bourne Ave, GA Hwy 21 and Jimmy de Loach Pkwy. All traffic flow between the port and the long distance trucking/I-16 junction is routed along Bourne Ave. and Dean Forest Rd. All roads contain non-port related "background" traffic.

### 3.2 Federation Model Components Overview

As stated, the focus of this federation is the movement of container objects to, from, and within each of the four port locations and the I-16 junction. The utilization of trucks to facilitate this movement is of particular interest.

As has been stated in Section 2.1, the Arena© software package is best suited to model the movement and interaction of simulated objects in industrial systems and processes. Because of its object-oriented approach, Arena© can very reasonably simulate the system operations of the Port of Savannah. Therefore, Arena© is used to simulate the operations of the five major port system components: Distribution Centers 1, 2, and 3, the GCT, and the I-16 Junction

However, for reasons described in Section 2.1, the object-oriented, event-based nature of Arena© cannot reasonably approximate the complex movements and interactions of vehicles in a roadway environment. For this reason, VISSIM© – a continuous time-step, behavior based vehicle following simulator – is used to simulate the operations of the local highway and surface roadway network surrounding the Port of Savannah.

The Arena© model of the Port of Savannah system components used in this study is actually an aggregation of several smaller models, called submodels, each of which

simulates one of the five port system components (distribution centers, GCT and I-16 junction). This aggregation of submodels, which comprises the entire port system, is referred to as the "port model." Individually, each of these port system components within the aggregate port model is referred to as a submodel, identified specifically by the name of the component which they simulate (e.g., "Distribution Center 1 submodel").

The next four subsections will provide an overview of the simulated operation of the port and roadway network systems. Subsection 3.2.1 introduces relevant terminology used throughout this study and document. Subsection 3.2.2 provides an operational overview of each port submodel. Subsection 3.2.3 provides a brief operational overview of the roadway network model. Subsection 3.2.4 provides a brief overview of the RTI developed for this federation. Subsection 3.2.5 provides a brief overview of the database system developed for this federation. Lastly, Subsection 3.2.6 brings these four federation elements together to give an overview of their interactions, providing illustrative examples of the simulated movement of trucks and containers through the federation.

### 3.2.1 Study Terminology

To maintain clarity and consistency throughout this document, specific terminology is used, and therefore must be defined. This section briefly introduces terms and text formatting elements relevant to this study and document.

**Simulation –** This refers to the general method of "using computer models to imitate, or simulate, the operations of…real world facilities or processes." [5] In the context of this study, Arena© and VISSIM© are simulation software packages.

**Model** or **Computer Model –** This refers to the specific instances of the simulation software, or models, built to imitate the real world system or process. In this study, the roadway network model built in VISSIM© is an example of a computer model, as is the port model built in Arena©.

**Federation –** This refers to the integration, or combination, of several specific models in a system simulation using an RTI. This encompasses both the models themselves, as well as any other software or programs required to combine the models. In this study, the federation is comprised of the port model, the roadway network model, the RTI and the federation database.

**Federate –** This term is used interchangeably with the term "model," as a federate simply refers to a model in the context of a federation. In this study, the term federate is generally paired with the model name (e.g., port model federate) and is used only when a model is being discussed in the context of the federation.

**Container –** When used alone, the term "container" refers to a physical, real-world freight container whose movement throughout the Port of Savannah system is being simulated.

**Truck –** When used alone, the term "truck" refers to a physical, real-world freight truck whose movement and carriage of containers throughout the Port of Savannah system is being simulated.

**Object –** An object is a generalized term that refers to the simulated instance of a real-world item with associated data, attributes and methods. In this study, the term "object" is used only in conjunction with the terms "container" or "truck" (e.g., container object, or truck object) to identify a simulated container or truck within the federation. Because the term "object" is generalized to the context of the federation – that is, it refers to simulated truck and containers located anywhere within the federation – truck objects, for example, can be instantiated in both the port model federate and the roadway network model federate.

**Entity –** An entity refers to the specific instance of an object located in the Arena© model. Like the term "object," the term "entity" is generally used in conjunction with either "truck" or "container" to denote what it represents (e.g., truck entity, container entity).

**Vehicle –** In the Arena© model, the term vehicle is used somewhat interchangeably with the term truck. This is because at this point in the development of the port model, the only vehicles interacting with containers are trucks. In future versions, vehicles may be a generalized term that can refer to trucks, forklifts, trains, etc. In the VISSIM© model, the

term vehicle refers to both trucks (not including port or roadway trucks described previously) and cars from the general public circulating throughout the network.

**Block –** This term generally refers to a logical construct, or logic, in Arena© that interacts with entities. For example, a **Queue** block is a logical construct that holds entities in a queue. Similarly, a **Delay** block is a logical construct, or block, that delays an entity's passage through that block for some user-defined time interval.

**Process –** For the purposes of this study and only in the context of Arena©, the term "process" will be used to refer to a collection or series of several logic blocks with which entities interact.

**Submodel –** A submodel refers to a collection of processes in Arena© that imitate the functionality of a real-world system. For example, in this study, the port model is composed of several submodels. Each distribution center is a submodel. The I-16 Junction is a submodel. The GCT is several submodels that have been aggregated to simulate the operation of the GCT (e.g., a dockside operations submodel, a customs processing submodel, etc.). Only one of these aggregated GCT submodels– the GCT Gate submodel – interacts with the federation. Accordingly, only the structure of the GCT Gate submodel is discussed.

**Arena© Global Variable** – A global variable is a readable/writeable numerical variable within Arena© that is accessible to all submodels in an Arena© model. Global variables

are not specific to any single entity, block or process, however each variable is unique from other variables. In this study, unique sets of global variables have been created and set aside for use by specific processes and submodels. This avoids the erroneous duplicate use or re-writing of global variables by more than one process or submodel.

**Attribute –** In Arena©, an attribute is "a common characteristic of all entities, but with a specific value that can differ from one entity to another." [5]. For example, all truck entities are assigned an attributed called 'Vehicle_ID,' however the value of this attribute is unique to each entity. Also, although attributes are assigned to truck and container objects when each entity is created in Arena©, the unique attribute values assigned to that entity follow that truck and container object as it moves throughout the federation and between federates.

As stated, specific syntax has also been incorporated. All Arena© block types are written in bold print (e.g., **Queue** block). Names of Arena© blocks are written in single quotations. For example, if a **Queue** block represents the instance of the "Port_RoadVeh_Queue," it is written as: **Queue** block 'Port_RoadVeh_Queue.' Similarly, entity attribute names are written in single quotations (e.g., 'Vehicle_ID,' 'Vehicle_Type').

### 3.2.2 Arena© Federate

In language consistent with the HLA, the Arena© port model is one of the federates being combined into the federated simulation. Figure 4 shows a spatial

representation for this relationship in terms of the port system's geographical layout. The dashed blue line represents the boundary between the port and roadway federates. Everything contained within the dashed blue line is part of the Arena© port model federate. Everything outside of the dashed blue line is part of the VISSIM© roadway model federate, which is discussed in the Subsection 3.2.3. There are five transaction points between the two federates, one at each distribution center, one at the GCT (port) terminal and one at the I-16 junction.

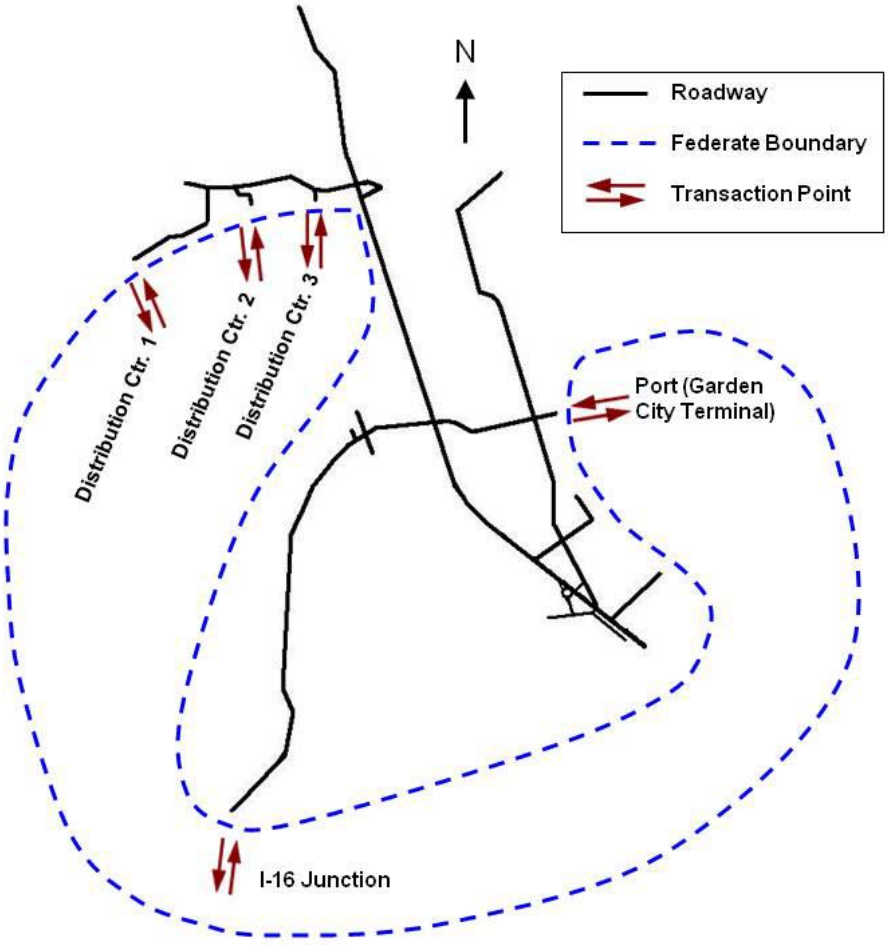



**Figure 4. Spatial Relationship between VISSIM© and Arena© Federates**

3.2.2.1 Arena© Port Model Federate – Truck Object Generation and Deletion

Port and roadway truck objects are created at one of two locations in the federation: the GCT Gate submodel and the I-16 Junction submodel. Port trucks are never removed during simulation execution, but instead circulate throughout the federation. Road trucks, however, are removed from the federation at one location: the I-16 Junction submodel. This is because the I-16 Junction simulates a truck object's entry into the interstate highway system, which is outside of the realm of this federation. Truck objects are neither created nor deleted at the Distribution Center submodels.

During the initialization of the port model federate, initial quantities of port and roadway truck entities are created at the GCT Gate submodel. The exact quantities of these truck entities are specified by the user prior to simulation execution. Because port truck objects circulate within the federation during simulation execution, this initial quantity of port trucks neither grows nor diminishes during simulation execution. The initial quantity of roadway trucks generated at the GCT Gate submodel during model initialization serve to transport container objects from the GCT Gate submodel to the I-16 Junction submodel.

Because the initial quantity of simulated roadway trucks depart the federation during its execution as roadway truck objects are deleted upon arrival at the I-16 Junction submodel, it is necessary to replenish the supply of roadway truck objects available for transport. Therefore, roadway truck objects are created (representing the arrival of a truck from the freeway system) at the I-16 Junction during simulation execution. As will be discussed later, both empty roadway truck objects and roadway truck objects carrying container objects are created at the I-16 Junction.

3.2.2.2 Arena© Port Model Federate – Container Object Generation and Deletion

Container objects are created at one of two locations in the federation: the GCT submodel and the I-16 Junction submodel. Note that container objects are not created at the GCT Gate submodel, but are created within the aggregated GCT submodels and then assigned to the GCT Gate submodel, which interacts directly with the federation. Container objects are neither created nor deleted at the Distribution Center submodels.

The creation of container objects at the GCT submodel simulates the arrival and offloading of containers from oceangoing freight ships. Within the aggregated GCT submodels, the arrival rate of oceangoing ships and the quantity of containers onboard those ships are user specified. For example, ship interarrival time could be set to 2 hours, and the quantity of containers objects onboard could be set to 250. Once the GCT submodel generates the oceangoing ship and simulates the offloading of all onboard container objects, the container objects are then processed among the aggregated GCT submodels before finally arriving at the GCT Gate submodel.

Container objects are also created at the I-16 Junction submodel to simulate the arrival of containers to the port system from the interstate highway system. A container object is only created with an associated truck object to simulate a container arriving on a truck. The creation rate of container objects in the I-16 Junction submodel is a user-specified parameter that is set prior to simulation execution.

3.2.2.3 Arena© Port Model Federate – Truck and Container Object Attributes

When truck and container objects are generated, these objects are assigned several attribute values. As stated in Section 3.2.1, attributes can be a common class of

characteristics shared among all objects or entities, but hold unique values when assigned to specific objects or entities. Table 1 shows the seven attribute class names and the object types to which those attribute names can be assigned.

**Table 1. Attribute Names and Object Assignments**

| Attribute Name | Object Assignment |
|---|---|
| Vehicle ID | Truck |
| Vehicle Type | Truck |
| Reroute Destination | Truck |
| Container ID | Container |
| Destination ID | Container |
| Destination ID2 | Container |
| Origin ID | Container |

Immediately upon creation in either the GCT Gate submodel or the I-16 Junction submodel, all truck objects are assigned attribute values for the 'Vehicle ID' and 'Vehicle Type' attributes. 'Vehicle ID' values are unique to each truck object and are never repeated during simulation execution. That is, if a truck object is deleted during simulation, its 'Vehicle ID' attribute value is not assigned to subsequent truck objects. 'Vehicle Type' attribute values are also assigned to truck objects immediately upon creation. A 'Vehicle Type' attribute value equal to 1 denotes a roadway truck and a 'Vehicle Type' attribute value equal to 2 denotes a port truck.

The 'Reroute Destination' attribute class is used to route or reroute empty truck objects from port model locations (e.g., I-16 Junction, distribution centers, etc.) to other destinations in the port model. This attribute only applies to empty trucks as the destination of trucks carrying containers is determined by the 'Destination ID' or 'Destination ID2" attributes of the container. Use of the 'Reroute Destination' attribute

class will be discussed further in section 3.3.1.1.3. Attribute values for the 'Reroute Destination' attribute class is assigned to truck objects, as needed, upon departure from port submodels. Therefore, the 'Reroute Destination' attribute value for a truck object may change during the course of simulation as empty truck objects are rerouted to other locations in the port model.

Container objects are similarly assigned attribute values immediately upon creation in both the GCT aggregated submodels (when offloading from ships is simulated) and the I-16 Junction submodel. 'Container ID' values are unique to each container object and are never repeated during simulation execution. That is, if a container object is deleted during simulation, its 'Container ID' attribute value is not assigned to subsequent container objects.

The 'Destination ID' and 'Destination ID2' attribute values are also assigned to container objects immediately upon creation in both the GCT submodels and the I-16 Junction submodel. The value of a container's 'Destination ID' attribute denotes the first destination to which a container object will be routed. The value of a container's 'Destination ID2' attribute denotes the second destination to which a container object will be routed. An example of this routing is detailed in Section 3.2.6. Table 2 shows the 'Destination ID' values and the associated port model destinations/interaction points that

**Table 2. Federation Destination ID Numbers**

| Interaction Point | Destination ID |
|---|---|
| Distribution Center 1 | 1 |
| Distribution Center 2 | 2 |
| Distribution Center 3 | 3 |
| I-16 Junction | 6 |
| GCT | 7 |

these values represent. For the purposes of this study, only two destinations (one intermediate and one final destination) were assigned to each container; however, the model is capable of being expanded to allow for processing containers with more than two destinations.

Container objects are also assigned an attribute value for the 'Origin ID' attribute class during simulation. 'Origin ID' attribute values correspond to the same values for locations associated with the 'Destination ID' attribute values shown in Table 2. Every time a container object is passed from an Arena© port submodel to the roadway network model federate, it is assigned the 'Origin ID' attribute value associated with the port submodel that it is leaving. If a container object has an intermediate destination and a final destination, its 'Origin ID' attribute value will be reassigned when the container object leaves the intermediate destination. Therefore, a container object may have multiple 'Origin ID' values associated with it over the course of simulation, but its original destination is tracked in the federation database. It is noted that the federate maintains a complete history of the container (and truck) history allowing for detailed post-analysis of a complete trip within the federation, to be discussed in detail in Sections 3.3.2 and 3.3.3.

It is important to note one characteristic of 'Destination ID' and 'Destination ID2' attributes as they are assigned to container objects. Although all container objects are assigned both attribute values, the final destination value ('Destination ID2') is not always utilized by the federation for some number of container objects. This occurs for two cases in the federation.

The first instance is when the first destination value ('Destination ID') assigned to container/truck object pairs entering the roadway network model at the GCT Gate submodel is equal to 6, the I-16 Junction. Doing so routes container objects directly to the I-16 Junction, without an intermediate destination, where they are deleted from the simulation immediately upon arrival. This deletion simulates their entry into the interstate highway system. Therefore, there is no need for a second (i.e., final) destination as the container's first destination is a terminating location (in this case, the interstate).

The second instance is when the first destination value ('Destination ID') assigned to container/truck object pairs entering the roadway network model at the I-16 Junction submodel is equal to 7. Doing so routes container objects directly to the GCT Gate, without an intermediate destination. Again, there is no need for a second (i.e., final) destination as the container's first destination is a terminating location (in this case, the GCT).

In all other instances, container objects are assigned 'Destination ID' attribute values equal to 1, 2, or 3 (denoting an intermediate destination at one of the three distribution centers) and a 'Destination ID2' attribute value equal to either 6 or 7 (denoting a final destination at either the I-16 Junction or the GCT, respectively).

#### 3.2.2.4 Arena© Port Model Federate – Submodel Object Processing

While objects are created at the GCT submodel and the I-16 Junction submodel, all submodels process both truck and container objects in some way. This subsection will

provide a brief overview of the ways in which truck and container objects are processed at each of the five port model submodels.

While container objects are created by the aggregated GCT submodels, their processing, in the context of ground transportation pertaining to this federation, is accomplished at the GCT Gate submodel. At this submodel, outgoing container objects (those leaving the port model for the roadway network model) are paired with available truck objects for release to the roadway network model where they are transported to a distribution center or the freeway, i.e. I-16 Junction.

Truck and container object pairs entering the GCT gate submodel are first separated and then processed. The container objects are routed within the aggregated GCT submodel and assigned to simulated outgoing freight ships. The truck objects are routed within the GCT Gate submodel to a queue to await assignment with a new outgoing container object. Port truck objects will wait in this queue indefinitely, regardless of the queue size, until pairing with an outgoing container object occurs. Roadway truck objects, however, may be released to the roadway network model without containers (destined for the I-16 Junction submodel) if it is found that there is a sufficient supply of roadway truck objects in the GCT Gate submodel. That is, surplus roadway trucks are sent to the I-16 Junction submodel where they are deleted from the simulation. All of these container/truck object transactions with the roadway network model are logged by the RTI and federation database. This is discussed in greater detail in Sections 3.3.2 and 3.3.3.

Little object processing is undertaken by the I-16 Junction submodel. The primary function of the I-16 Junction submodel is simply to create empty truck objects

and paired truck and container objects to be released to the roadway network model, bound for one of the port destinations. The I-16 Junction submodel also receives truck and paired truck/container objects that are leaving the roadway network model at the I-16 junction. All of these container/truck object transactions with the roadway network model federate are logged by the RTI and federation database. As stated, this is discussed in greater detail in Sections 3.3.2 and 3.3.3.

The primary function of the distribution center submodels is to process container and truck objects. As container or truck objects are neither created nor deleted in the distribution center submodels, the distribution centers can be viewed as intermediate destinations for containers in route to their final destinations. Upon arrival at a distribution center, truck and container object pairs are separated. The container objects are then processed for some user-defined time delay interval. Container objects are then sent to a queue where they await pairing with available outgoing truck objects to be transported to their final destinations. Separate container queues are maintained in relation to a containers final destination and the type of truck necessary to carry that container. For example, container objects destined for the I-16 Junction submodel (to be carried by a roadway truck) are sent to a separate queue than container objects destined for the GCT Gate submodel (to be carried by a port truck).

Similarly, incoming truck objects are processed for some user-defined time delay interval. Truck objects are then sent to a truck queue to be paired with outgoing container objects. Separate truck queues are maintained for port truck objects and roadway truck objects.

All distribution center submodels also have the capability to reroute surplus port and roadway truck objects. If the queue of roadway truck objects at a distribution center is sufficient, empty roadway truck objects are assigned a 'Reroute Destination' attribute value corresponding to either the GCT Gate submodel or the I-16 Junction (depending on the need for additional roadway vehicles at the GCT) and rereleased to the roadway network model. Similarly, if the queue of port truck objects at a distribution center is sufficient, empty port truck objects are assigned a 'Reroute Destination' attribute value corresponding to the GCT Gate submodel and rereleased to the roadway network model. Upon arriving at the GCT Gate submodel, these empty surplus port truck objects enter the queue of port truck available for pairing with outgoing container objects.

### 3.2.3 VISSIM© Federate

As stated, Arena© is limited in its capability to reasonably simulate the movement of vehicles on a roadway network. That is, given the multitude of variables that affect the travel of vehicles in a roadway environment, Arena© cannot reasonably simulate the delay experienced by containers that are being transported on a roadway network.

A traditional monolithic Arena© model would maintain the container/truck objects within the model and simply route them from one submodel directly to another submodel (e.g., directly from the GCT Gate submodel to a distribution center submodel). A user-specified time delay value representing transport time would be assigned to the container object. However, this delay assignment in Arena© would not accurately capture the dynamic nature of the traffic environment and the delays associated with

fluctuations in traffic volume, congestion and traffic signalization. To overcome this limitation of the Arena© modeling environment, the federation instead "passes" the truck and container object pairs from submodel in the Arena© port model to the VISSIM© model of the roadway network.

For example, a truck exits the GCT Gate submodel of the Arena© model and enters the roadway network at the origin link associated with the GCT. After completing its route through the VISSIM© roadway network to the destination link associated with one of the distribution center submodels, the container/truck object pair is then "passed" back to the Arena© model. More specifically, the container/truck combination is recreated in the specific distribution center submodel of the Arena© port model. The difference in simulation time between when the container/truck combo leaves the Arena© port model at the GCT and when it reenters the Arena© port model at Distribution Center 2 represents the transport time.

There are several types of delay that truck objects could experience while traveling the simulated roadway network model. Several intersections incorporate simulated traffic control devices. Therefore, some control delay could be attributed to these signals. Also, the roadway network model is populated with "background" traffic vehicles to represent non-port traffic associated with the general public. The volume of background traffic, and therefore the level of network congestion, could influence the travel of port and roadway truck objects through the roadway network model.

When a container/truck object pair is passed from the port model to the roadway network model, the object carries with it the two destination-related attribute values: 'Destination ID' and 'Destination ID2.' These attribute values are utilized by the

roadway network model's routing decisions to correctly route the container/truck object pair to the relevant destination. Similarly, empty truck objects released to the roadway network model for rerouting carry with them a value for the 'Reroute Destination' attribute. In this case, the roadway network model's routing decisions use this attribute value to appropriately route the truck object to its destination. Section 3.3.4 discusses the development and structure of the roadway network model in greater detail.

### 3.2.4 Runtime Infrastructure (RTI)

The "passing" of container and truck objects between the two federates is facilitated by the runtime infrastructure (RTI). In this application, the three primary purposes of the RTI are time management, object management, and data management.

Time management is concerned with the synchronization of simulation time between the two federates. As Arena© is a discrete, event-based simulator and VISSIM© is a continuous, time-step simulator, effective time management is fundamentally critical to the success of the federation. Also, most performance measures for port and roadway operations are time-based metrics. Therefore, time management is also important to collect accurate and meaningful measures of system performance.

Object management in the RTI ensures that simulation objects are correctly passed between federates. For example, when a truck exits the port model federate and enters the roadway model federate, the RTI ensures that all object attributes of the truck created in the roadway federate correctly match those of the corresponding truck that has just exited from the port federate. The object management role of the RTI also ensures

that truck and container objects are neither erroneously duplicated between the two federates, nor deleted when passing between federates.

In this federation, data management primarily refers to the recordkeeping of all transactions between the two federates by bringing together elements of both time and object management. A time-stamped record is created to log each transaction; that is, instance of an object crossing one of the transaction points (shown in Figure 4) between federates. Recordkeeping will be further discussed in subsection 3.2.3.

The RTI for this study was constructed in the Visual Basic© (VB) programming language using Microsoft Visual Studio 2005©. The primary motivation for using VB in this study is that both VISSIM© and Arena© have built-in component object model (COM) interfaces that utilize the VB programming language. These interfaces allow the objects, methods and properties of each simulator to be dynamically accessed, assigned and run from within other applications (e.g., Visual Studio©) [28].

### 3.2.5 Federation Database

As stated, the role of the RTI is primarily one of time, object and data management. Accordingly, the RTI does not store any of the data or records associated with these management tasks. Instead, the data is recorded to a federation database consisting of time-stamped entries for each container and/or truck transaction between federates.

In this study, the database software used is Microsoft Access©. This choice was motivated by the inherent interoperability between Microsoft Access©, the Microsoft© based RTI, and VB-based federate COM interfaces.

### 3.2.6 Federation Structure, Interaction and Examples

The above subsections have provided an operational overview of the Arena© port model federate and its respective submodels, as well as the VISSIM© roadway network model federate. This subsection provides a brief discussion of the interaction of these model components. This subsection also provides several illustrative examples of truck and container objects moving within the federation.

The interaction among the VISSIM© and Arena© model federates, the federation RTI, and the federation database is shown below in Figure 5. The RTI facilitates the movement of data between port and roadway network model federates. When data is collected from one model federate, it is recorded in the federation database and then disseminated to the other model federate involved in the transaction. For example, when a truck leaves the port terminal for the roadway, all object and data information

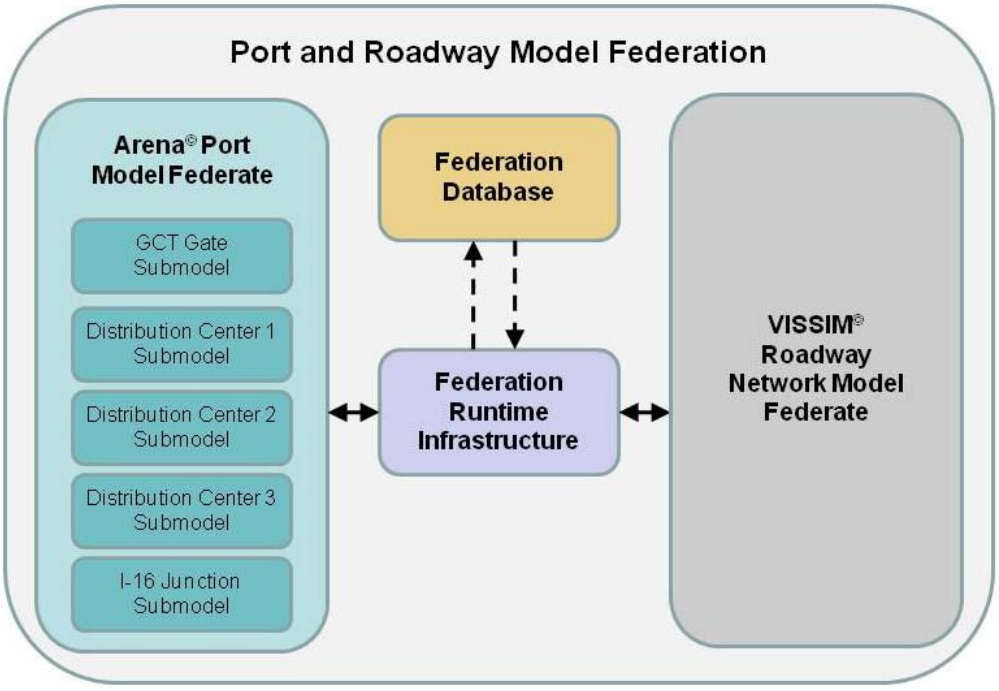



**Figure 5. Port and Roadway Model Federation Structure**

pertaining to that truck is collected by the RTI from the Arena© port model federate and written to the federation database. The RTI then disseminates that collected information to the VISSIM© model federate, which recreates that truck in the simulated roadway network.

3.2.5.1 Example – Movement of a Container Object Generated at the GCT

This subsection provides an example of the simulated movement of a container object from its generation at the GCT to its deletion at the I-16 Junction.

A container object is generated at the aggregated GCT submodel (the model created by Peesapati and Gbologah) and assigned unique attribute values. Among these attributes are 'Container ID,' 'Destination ID,' and 'Destination ID2.' For this example, the 'Container ID' value will be equal to 123, the 'Destination ID' value will be equal to 2 (denoting the container object's first destination is Distribution Center 2) and the 'Destination ID2' value will be equal to 6 (denoting the container object's second and final destination is the I-16 Junction). The container object is sent to the GCT Gate submodel for pairing with a truck object and release to the roadway network model federate.

Upon entering the GCT Gate submodel, the container can enter one of two queues: one for container objects assigned directly to the I-16 Junction (to be paired with roadway truck objects), or one for container objects assigned to an intermediate destination at a distribution center (to be paired with port truck objects). In this example, the container object enters the queue associated with the local distribution centers to await being paired with an outgoing port truck object.

When a port truck object becomes available, the truck and container objects are paired together. Once paired together, all attributes associated with both the container object and the truck object are written to a unique set of Arena© global variables (making those values accessible to the RTI via the COM interface) that are specifically associated with port truck objects and container objects exiting the GCT Gate submodel. Once these attribute values are written to the unique set of global variables, the values are collected by the RTI. The RTI then writes these values to the federation database, including a transaction time-stamp equal to the current simulation time. The truck/container object pair is then deleted from the Arena© port model federate and recreated as a truck object in the VISSIM© roadway network model federate on the link associated with the GCT Gate submodel. This truck object created in the roadway network model federate carries with it a unique identifier value that relates the truck object with the corresponding truck and container object attribute values that have been logged in the federation database.

The roadway network model federate utilizes the 'Destination ID' attribute value equal to 2 to route the truck object through the roadway network model to the destination link associated with Distribution Center 2. Upon arrival at the end of the roadway link associated with Distribution Center 2, the roadway network model federate notifies the RTI of the truck arrival. The RTI then uses the truck object's unique identifier value to recover from the federation database all relevant attribute values for both the truck and container objects in the pair. The RTI then writes these attribute values to a unique set of Arena© global variables (different from the set used above when exiting the GCT Gate submodel) that are specifically associated with truck and container objects entering the Distribution Center 2 submodel.

The RTI logs the arrival of the truck and container objects as a new record in the federation database and then removes the truck object from the roadway network model federate. The RTI then signals the Arena© submodel for Distribution Center 2 to create a container object and a truck object and to reassigns the container and truck objects their attribute variables being held in the global variable set. The container and truck objects are then separated.

Once separated, the port truck object is first processed for some time-delay interval to simulate unloading and then routed, within the submodel, to a process that decides if there are a sufficient number of port trucks available in the Distribution Center 2 submodel. If there is not a sufficient number, the port truck object is sent to a queue to await pairing with outgoing containers bound for the GCT Gate submodel. If there are a sufficient number of port truck objects in the Distribution Center 2 queue, the surplus port truck object is assigned a 'Reroute Destination' attribute value equal to 7 and released to the roadway to be routed back to the GCT Gate submodel.

The separated container object is first processed for some time-delay interval to simulate unloading. Because the container object's 'Destination ID2' value is equal to 6 (denoting the I-16 Junction as its final destination) the container object is then routed within the Distribution Center 2 submodel to a queue for outgoing container objects bound for the I-16 Junction. The container objects waits in the queue until a roadway truck object becomes available, at which point it is paired with the outgoing roadway truck object.

Once paired with the roadway truck object, all attributes associated with both the container object and the truck object are written to a unique set of Arena© global

variables that are specifically associated with roadway truck objects and container objects exiting the Distribution Center 2 submodel. Note that these global variables are different from those identified above and are unique to this location and scenario. Once these attribute values are written to the unique set of global variables, the values are collected by the RTI. The RTI then writes these values as a time-stamped record to the federation database. The truck/container object pair is then deleted from the Arena© port model federate and recreated as a truck object in the VISSIM© roadway network model federate on the outgoing link associated with the Distribution Center 2 submodel. Again, this truck object created in the roadway network model federate carries with it a unique identifier value that relates the truck object with the corresponding truck and container object attribute values that have been logged in the federation database.

The roadway network model federate then utilizes the 'Destination ID2' attribute value equal to 6 to route the truck object through the roadway network model to the destination link associated with the I-16 Junction submodel. Upon arrival at the destination roadway link associated with the I-16 Junction, the roadway network model federate notifies the RTI of the truck arrival. The RTI then uses the truck object's unique identifier value to recover from the federation database all relevant attribute values for both the truck and container objects in the pair. These attribute values are written to a unique set of Arena© global variables (again, different from the set used above) that are specifically associated with truck and container objects exiting the roadway network model federate at the I-16 Junction submodel.

The RTI logs the arrival of the container and truck object pair at the I-16 Junction as a time-stamped record in the federation database and then removes the truck object

from the roadway network model federate. The RTI then signals the Arena© I-16 Junction submodel to create a container object and a truck object. The I-16 Junction then reassigns these container and truck objects their associated attribute variables currently being held in the unique global variable set. The container and truck objects are then separated and deleted from the simulation.

3.2.5.2 Example – Movement of a Container Object Generated at the I-16 Junction

This subsection provides an example of the simulated movement of a container object from its generation at the I-16 Junction submodel to its arrival at the GCT Gate submodel.

A container object is generated at the I-16 Junction submodel and assigned unique attribute values. Among these attributes are 'Container ID,' 'Destination ID' and 'Destination ID2.' For this example, the 'Container ID' value will be equal to 456, the 'Destination ID' value will be equal to 7 (denoting the container object's first destination is the GCT). Although the I-16 Junction submodel will assign the container object an attribute value for 'Destination ID2,' this value is irrelevant in this example as the container object is being directly routed to the GCT Gate submodel without any intermediate destination.

Simultaneously, a roadway truck object is generated at the I-16 Junction submodel and is paired with the container object. Once paired together, all attributes associated with both the container object and the roadway truck object are written to a unique set of Arena© global variables that are specifically associated with roadway truck objects and container objects entering the roadway network model federate from the I-16

Junction submodel. Once these attribute values are written to the unique set of global variables, the values are collected by the RTI. The RTI then writes these values to the federation database. The truck/container object pair is then deleted from the Arena© port model federate and recreated as a truck object in the VISSIM© roadway network model federate on the link associated with the I-16 Junction submodel. This truck object created in the roadway network model federate carries with it a unique identifier value that relates the roadway truck object with the corresponding truck and container object attribute values that have been logged in the federation database.

The roadway network model federate utilizes the 'Destination ID' attribute value equal to 7 to route the truck object through the roadway network model directly to the destination link associated with GCT Gate submodel. Upon arrival at GCT Gate, the roadway network model federate notifies the RTI of the truck object's arrival. The RTI then uses the truck object's unique identifier value to recover from the federation database all relevant attribute values for both the truck and container objects in the pair. These attribute values are written to a unique set of Arena© global variables that are specifically associated with truck and container objects entering the GCT Gate submodel.

The RTI logs the arrival of the truck and container objects as a time-stamped record in the federation database and then removes the truck object from the roadway network model federate. The RTI then signals the Arena© GCT Gate submodel to create a container object and a truck object. The GCT Gate submodel then reassigns these container and truck objects their attribute variables that are currently being held in the unique global variable set. The container and truck objects are then separated.

Once separated, the roadway truck object is processed for some time-delay interval to simulate unloading. The truck object is then routed to a process within the submodel that decides if there are a sufficient number of roadway truck objects available in the GCT Gate submodel. If there is not a sufficient number, the roadway truck object is sent to a queue to await pairing with outgoing containers bound directly for the I-16 Junction submodel. If there are a sufficient number of roadway truck objects in the GCT Gate submodel roadway queue, the surplus roadway truck object is assigned a 'Reroute Destination' attribute value equal to 6 and released to the roadway to be routed to the I-16 Junction submodel where it will be deleted from the simulation.

The separated container object is processed for some time-delay interval to simulate unloading. The container object is then directed within the aggregated GCT submodel to a submodel that simulates the container object's loading onto an outgoing freight ship and then deletes the container from the simulation.

### 3.3 Description of Federation Model Components

This section will provide a detailed description of the construction and operation of each federation component. This discussion starts with a description of the Arena© port model federate, and its respective submodels, where containers and trucks are generated, processed, and deleted. Next, the VB-based RTI that manages time, data and objects, and facilitates the passage of simulation objects between federates is discussed. Then the VISSIM© roadway network model is discussed. Finally, the Access© federation database is discussed.

### 3.3.1 Arena© Port Model Federate

Within the port model, there are five submodels: the GCT/GCT Gate, Distribution Centers 1, 2, and 3, and the I-16 Junction. The following will provide a detailed discussion of both the GCT Gate submodel and the I-16 Junction submodel, and a general discussion of the standardized Distribution Center submodel. Recall that the term "entity" is used to refer specifically to truck and container objects interacting with the logical blocks and processes within the Arena© port model federate.

#### 3.3.1.1 Garden City Terminal (GCT) Gate Submodel

To simulate the operations of the entire GCT (e.g., customs, container inspection, bonded storage, forklift and crane resource utilization, truck transfer, etc.) several Arena© submodels were constructed and aggregated together. This comprehensive Port of Savannah model was constructed as part of a previous study of the Port of Savannah operations by Lakshmi Peesapati and Franklin Gbologah for the Georgia Tech Research Institute.

For the purposes of this study, the larger GCT operations port model serves to simulate the arrival and handling of containers at the Port of Savannah from oceangoing container vessels and landside railroad networks. However, this study focuses on the interaction of the port with the surrounding roadway network. Therefore, the structure and operation of these aggregated submodels will not be discussed.

Instead, this discussion of the aggregated GCT submodel specifically focuses on the "GCT Gate" submodel, constructed specifically for this study, which simulates operations at the terminal's roadway gate, or point of interaction with the roadway

network. There are three primary logical functions in the GCT Gate submodel: (1) one associated with outgoing trucks (i.e., exiting the GCT to the roadway), (2) one associated with incoming trucks (i.e., entering the GCT from the roadway), and (3) one to release, or reroute, unneeded empty trucks to the roadway. Figure 6 shows the relationship between these three functions. Elements shown in teal represent logic processes for the outgoing function, elements shown in purple represent logic processes for the incoming function, and the element in grey represents logic processes for the rerouting function.

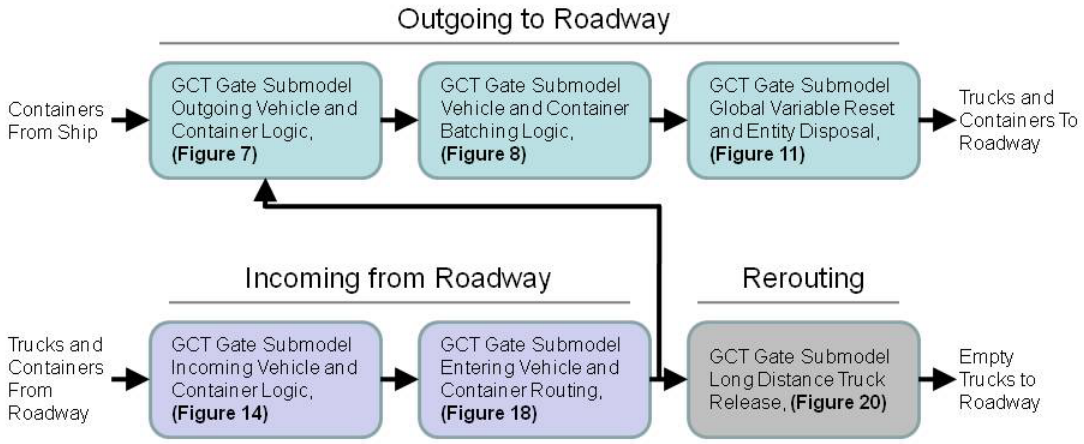



**Figure 6. GCT Gate Submodel Logical Series Overview**

The outgoing process series consists of three general processes. In the first process, containers ready for transport enter the GCT Gate submodel from the port submodels built by Peesapati and Gbologah to represent containers that have recently arrived on freight ships. In the second process, these containers are paired, or "batched," with available trucks waiting in a queue for release to the roadway. This second process is where exiting truck/container batched pairs are detected by the RTI and are "passed" to, or recreated in the roadway network model federate. The third process then resets the

previous process for the next truck/container pair and disposes of (i.e., deletes) the current batched pair that has just been passed to the roadway.

The incoming logic process series consists of two basic processes. In the first process, trucks and containers entering the port from the roadway are created in the GCT Gate submodel and assigned their appropriate attribute values. In the second process, containers are processed and routed for appropriate batching with outgoing trucks for future destinations. This second process also routes trucks to appropriate queues. Port trucks are automatically sent to the submodel's port vehicle queue (denoted by the arrow connecting to the first process in the outgoing logical process series). However, if the roadway vehicle queue has excessive numbers of available trucks, the unneeded trucks are sent to a logical process to be released back to the roadway network model federate, destined for the I-16 Junction.

The rerouting logic simply takes the unneeded or excess long-distance road trucks and passes, or releases, them back to the roadway network model federate, destined for the I-16 Junction.

*3.3.1.1.1 Garden City Terminal Outgoing Operations Logic*

Figure 7 shows the logical process used to generate and route trucks and containers in the GCT gate submodel. Container entities that have been offloaded from ships, routed through the GCT freight handling submodels (constructed in the Peesapati and Gbologah study) and sent to the GCT Gate submodel arrive at the red **Station** block 'Receiving From Truck Transfer.' Upon arrival at the 'Receiving From Truck Transfer'

block, container entities will have already been assigned attribute values for ‘Container ID,’ ‘Destination ID’ and ‘Destination ID2.’

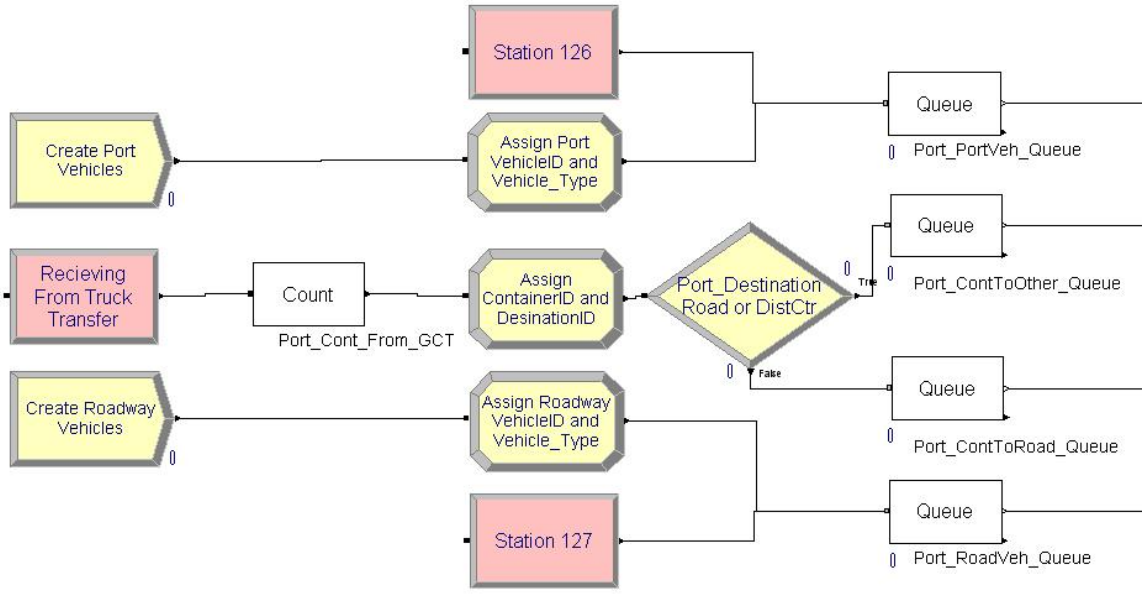



**Figure 7. GCT Gate Submodel Vehicle and Container Generation**

Container entities then proceed to a **Count** block ‘Port_Cont_From_GCT’ which increments a counter internal to Arena© that is associated with containers entities arriving to the GCT Gate submodel from the aggregated GCT submodel. Container entities then proceed to the **Assign** block ‘Assign Origin ID’ where they are assigned an ‘Origin ID’ attribute value. Recall that the GCT ‘Origin ID’ value is equal to 7. Container entities then proceed to the **Decide** block ‘Port Destination Road or DistCtr.’ As there are two types of trucks – port trucks and long-distance roadway trucks, this decision block directs the container entities to the correct queue to await batching with the relevant truck type. The **Decide** block ‘Port Destination Road or DistCtr’ evaluates each container based on its ‘Destination ID’ attribute value. If the container has a local

destination, i.e., if the if/then decision statement ['Destination ID' ≠ 6], then the entity is routed to the **Queue** block 'Port_ContToOther_Queue' to await batching with a port truck for transport to one of the distribution centers. If the statement is false and the destination is long distance trucking (i.e., Destination ID = 6), container entities are sent to the **Queue** block 'Port_ContToRoad_Queue.'

The GCT Gate submodel also contains the logic that creates port and long distance truck objects during simulation initialization. The **Create** block 'Create Port Vehicles' creates the initial supply of port truck entities during simulation initialization.

Port truck entities then proceed to the **Assign** block 'Assign Port VehicleID and Vehicle_Type' where they are assigned 'Vehicle ID' and 'Vehicle_Type' attribute values. Port trucks all are vehicle type 2. Also, port truck Vehicle ID values are assigned sequentially beginning at 200. Once attributes are assigned, the port truck entities proceed to the **Queue** block 'Port_PortVeh_Queue' to await batching with containers. As port truck objects circulate through the federation, port trucks that return to the port from the roadway are routed to the **Station** block 'Port Veh From Road.' As these port trucks already have attribute values associated with them, these port trucks proceed directly to the **Queue** block 'Port_PortVeh_Queue' to await batching with outgoing container entities.

The **Create** block 'Create Roadway Vehicles' creates the initial supply of roadway truck entities during simulation initialization. Roadway truck entities then proceed to the **Assign** block 'Assign Roadway VehicleID and Vehicle_Type' where they are assigned a 'Vehicle ID' and 'Vehicle_Type' attribute values. Roadway trucks all are vehicle type 1. Also, roadway truck Vehicle ID values are assigned sequentially

beginning at 1000. Roadway truck entities then proceed to the **Queue** block 'Port_RoadVeh_Queue' to await batching with outgoing containers.

As roadway trucks arrive at the GCT Gate submodel from the roadway model federate, these arriving entities are eventually (following simulated unloading, etc.) routed to the **Station** block 'Road Veh From Road.' The roadway truck entities then proceed to the **Queue** block 'Port_RoadVeh_Queue' to await batching with outgoing container entities bound for the I-16 Junction.

The next set of logic blocks in outgoing operations logic series is concerned with ensuring that trucks and containers are correctly paired, or "batched." This logic is shown in Figure 8, and is a continuation of the logic shown in Figure 7. In Figure 8, the top set of logic is for port truck and container batching, while the bottom set of logic is

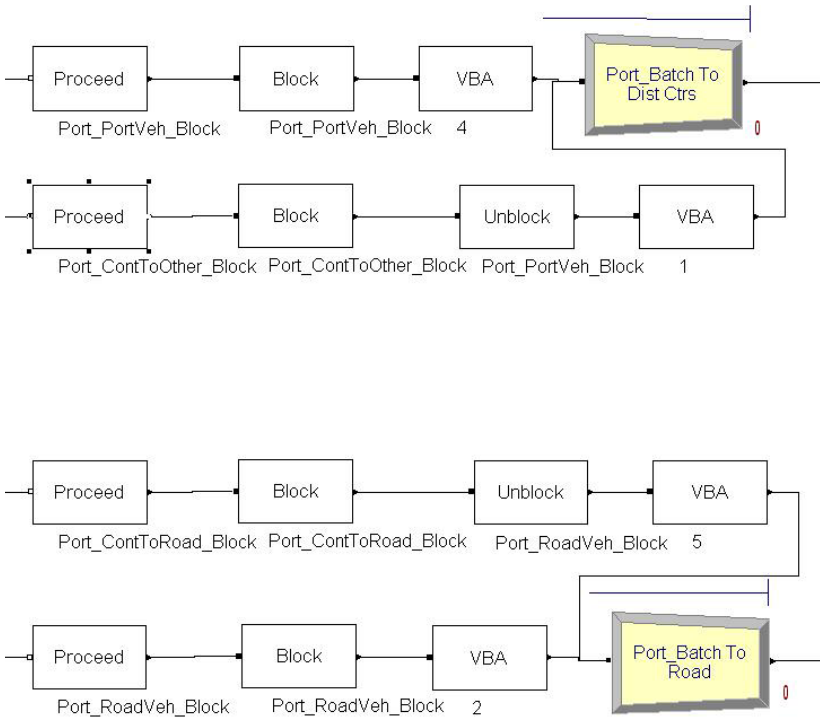



**Figure 8. GCT Gate Submodel Container and Vehicle Batching Logic**

for long-distance roadway truck and container batching. As both series of logic operate in the same way, only the top set of logic for port trucks will be described.

To ensure that containers and trucks are properly batched in pairs of two, the container must arrive first at the yellow **Batch** block. To accomplish this, a series of blockages are sequentially put in place and removed, causing a container entity to arrive first. The **Proceed** blocks are analogous to a physical gate or blockage. The **Proceed** block labeled 'Port_PortVeh_Block' has an initial blockage value of 1 (i.e., it is "closed"), restricting passage of port truck entities from the previous queue 'Port_PortVeh_Queue' (shown in Figure 7). However, the **Proceed** block labeled 'Port_ContToOther_Block' has an initial blockage value of zero – or no blockage – and can be considered "open." Therefore, a port container entity from the previous queue, 'Port_ContToOther_Queue,' (see Figure 7) may freely pass through the 'Port_ContToOther_Block' **Proceed** block. It then passes through the **Block** block labeled ''Port_ContToOther_Block.' This creates a blockage equal to 1 – effectively closing the gate – at the **Proceed** block immediately upstream. The container entity then passes through the **Unblock** block labeled 'Port_PortVeh_Block,' which lifts the blockage that is restricting truck entities from passing through the **Proceed** block labeled 'Port_PortVeh_Block.' The container entity then passes through the **VBA 1** block and proceeds to the **Batch** block 'Port_Batch To Dist Ctrs' to await the arrival of a truck entity.

As the blockage at the **Proceed** block 'Port_PortVeh_Block' was lifted by the container entity passing through the corresponding **Unblock** block, the truck entity waiting in the queue (Figure 7) is now free to pass. The truck entity then immediately

passes through the **Block** block labeled 'Port_PortVeh_Block,' resetting the blockage at the **Proceed** block immediately upstream. The truck entity then passes through the **VBA 4** block and proceeds to the **Batch** block 'Port_Batch To Dist Ctrs.'

The batch size at the **Batch** block has been set to 2, ensuring that only one container entity is paired with one truck entity. This number could be adjusted to simulate larger trucks capable of transporting more than one container simultaneously. However, doing so would require that the initial blockages at the **Block** and **Unblock** blocks be adjusted to allow more than one container to pass through for every one truck.

In order for the truck and container entity attributes to be accessible by the RTI via the Visual Basic© COM interface (and thus pass to the roadway model federate), they must first be written to global variables. Attributes (e.g., 'Container ID' and 'Vehicle_Type') hold values specific to each entity, whereas global variables hold values that are general to the entire simulation and are accessible through the COM interface. In this case, when a container entity passes through the **VBA 1** block, a Visual Basic for Applications© (VBA) command is triggered. Note that the code for the VBA blocks throughout the port model are internal to Arena© and are separate from the RTI commands. The VBA commands associated with the **VBA 1** block are shown in Figure 9.

In Arena©, global variables are identified and accessed according to unique number identifiers associated with each variable, not by their variable names. Similarly, attributes are referred to, within Arena© VBA commands, according to unique number identifiers associated with an attribute class. In the commands shown in Figure 9, for example '5011' corresponds to a unique global variable specifically used to hold

```
Private Sub VBA_Block_1_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
ActEnt = s.ActiveEntity
s.VariableArrayValue(5011) = s.EntityAttribute(ActEnt, 1001)
s.VariableArrayValue(5014) = s.EntityAttribute(ActEnt, 4004)
s.VariableArrayValue(5019) = s.EntityAttribute(ActEnt, 5005)
s.VariableArrayValue(5111) = s.EntityAttribute(ActEnt, 7007)
End Sub
```

**Figure 9. Visual Basic Commands for VBA 1 Block**

Container ID numbers of outgoing container entities. Similarly, 'ActEnt' refers to the entity actively passing through the associated VBA block at that moment, and the number '1001' refers to the attribute class 'Container ID.' While this attribute class is common to all container entities, its value is unique to each individual container entity. This command line sets the global variable with identifier number '5011' equal to the 'Container ID' attribute value, '1001,' for the active entity. A more complete discussion of how these global variable values are accessed by the RTI from the Arena© model will be discussed in Section 3.3.1. The code for the **VBA 4** block concerning truck entities is shown in Figure 10. Also, Table 3 shows the attribute class name associated with each

```
Private Sub VBA_Block_4_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
ActEnt = s.ActiveEntity
s.VariableArrayValue(5012) = s.EntityAttribute(ActEnt, 2002)
s.VariableArrayValue(5013) = s.EntityAttribute(ActEnt, 3003)
End Sub
```

**Figure 10. Visual Basic Commands for VBA 4 Block**

attribute class number. Appendix B has a complete list of all variable names and associated unique variable identifier numbers. Note that each of the **VBA** blocks shown in Figure 8 writes the attribute values of the active entities to separate sets of global variables that are uniquely associated with each **VBA** block.

**Table 3. Attribute Names and Class Identifier Numbers**

| Attribute Name | Attribute Class Number |
|---|---|
| Container ID | 1001 |
| Vehicle ID | 2002 |
| Vehicle Type | 3003 |
| Destination ID | 4004 |
| Destination ID2 | 5005 |
| Reroute Destination | 6006 |
| Origin ID | 7007 |
| Dispersion Destination | 8008 |

Once batched at the **Batch** block, the container/truck entity pairs then proceed to a series of logic blocks that reset the blockages to their original states and global variable sets to zero-values. This final series of logic is shown in Figure 11. The **Delay** block

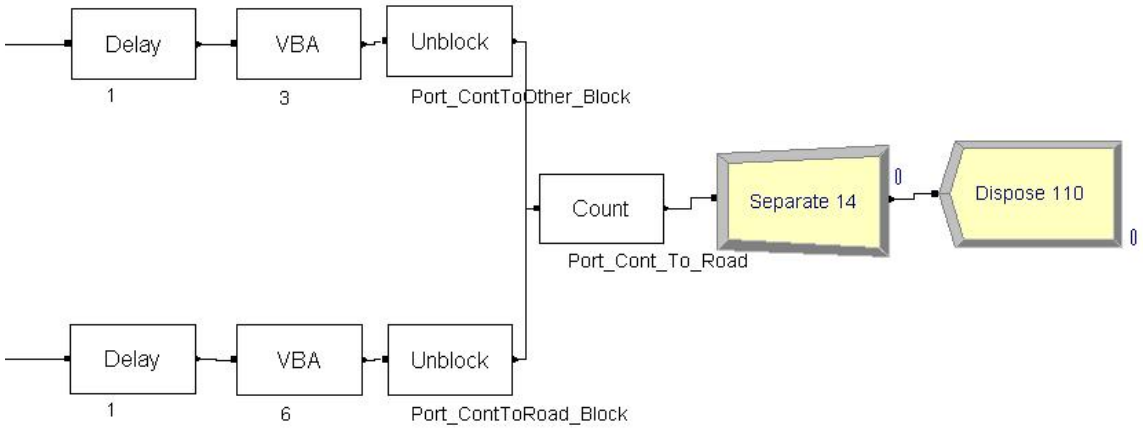



**Figure 11. Port GCT Submodel Global Variable Reset and Entity Disposal**

delays the batched entities for 1 second. During this delay, the RTI has one full time step during which to collect the variable values associated with the pair of batched container and truck entities before the global variables containing these values are reset to zero. The batched entities pair then passes through the **VBA 3** block. This executes a series of commands that reset the global variables for exiting port trucks and containers to zero. Note that these are the two unique sets of global variables to which the active entity'sattribute values were assigned by the **VBA 1** and **VBA 4** blocks in Figure 8. The resetting VBA commands for the **VBA 3** block are shown in Figure 12. The commands for the **VBA 6** block are similar to those shown in Figure 12, but reference two different unique sets of global variable array values.

```
Private Sub VBA_Block_3_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
s.VariableArrayValue(5011) = 0
s.VariableArrayValue(5012) = 0
s.VariableArrayValue(5013) = 0
s.VariableArrayValue(5014) = 0
s.VariableArrayValue(5019) = 0
s.VariableArrayValue(5111) = 0
End Sub
```

**Figure 12. Visual Basic Commands for VBA 3 Block**

The batched entity pair then passes through the **Unblock** block 'Port_ContToOther_Block.' This lifts the blockage at the upstream **Proceed** block 'Port_ContToOther_Block', shown in Figure 8, by assigning it a blockage value of zero.

This allows the entire batching process just described to occur again, having reset all global variable values to zero and blockages to their original state.

The batched pair of container and truck entities then passes through the **Count** block 'Port_Cont_to_Road' which increments a counter internal to Arena© that is associated with the combined number of containers entities departing the GCT Gate submodel. The batched pair of container and truck entities then passes through the **Separate** block 'Separate 14' and then to the **Dispose** block 'Dispose 110' where they are removed from the Arena© model. The **Separate** block is only necessary as Arena© does not allow entities to be disposed of while still batched.

As stated, when a truck entity (which arrives second to the 'Batch' block) passes through the **VBA 4** block its 'Vehicle ID' attribute value is written to the associated global variable and a 1 second delay is assigned. During this time, the RTI first collects all variable values for the batched pair of entities. It then writes those values to the federation database. A truck object with the same attributes is then created in VISSIM© roadway network model (see Section 3.3.2.2.1). Thus, the truck/container pair is effectively "passed" to the VISSIM© federate. Exactly how this is handled by the RTI and federation database will be discussed in Sections 3.3.2 and 3.3.3 respectively.

During simulation, a dynamically updated visual representation is displayed in the GCT Gate submodel window. This graphical display shows the current values of the global variables as assigned by the VBA blocks. The display is updated when an entity passes through one of the corresponding VBA blocks. This display is shown in Figure 13. Note that the 'Destination ID' attribute class is associated only with containers. This is because during simulation, individual containers have a fixed destination path through

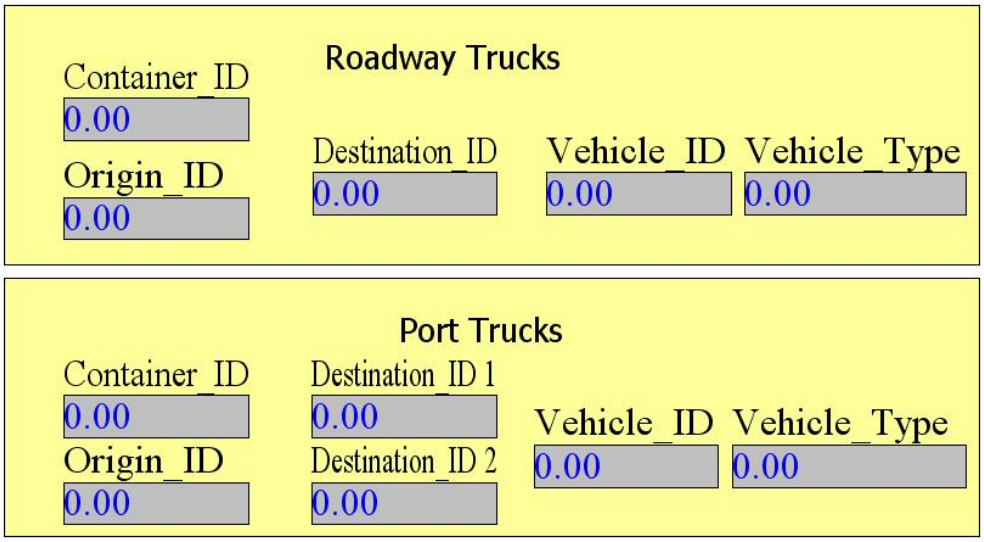



**Figure 13. GCT Gate Submodel Outgoing Variable Graphics**

the system whereas port trucks and roadway trucks do not. Also note in the 'Roadway Trucks' display in Figure 13 that there is only one 'Destination ID' variable. Recall from Figure 7 that containers at the GCT with an immediate destination (Destination ID) equal to 6 are assigned to long-distance roadway trucks and that the only destination for long distance trucks is the I-16 Junction. Conversely, containers assigned to port trucks will have an intermediate destination (Destination ID) at one of the three distribution centers, and a final destination (Destination ID2) of either the GCT or the I-16 Junction for long-distance trucking.

*3.3.1.1.2 Garden City Terminal Incoming Operations Model*

The previous section described the Arena© model logic underlying how outgoing container and truck entities are created and batched, and how their attributes are written to global variables for access by the RTI. This section will describe the logic for creating and assigning attributes to incoming truck and container entities in the GCT Gate

submodel. The logic series associated with incoming container and truck entity creation in the GCT Gate submodel is shown in Figure 14.

When a truck in the VISSIM© roadway network model federate approaches the end of the destination link associated with the GCT Gate entrance, it triggers a series of commands in the RTI (this triggering is discussed in Section 3.3.2.2). The RTI first collects the attribute values, which have been stored in the federation database, associated with the specific truck and container pair entering the GCT submodel. It then writes those attribute values to a unique set of global variables in the Arena© port model that are

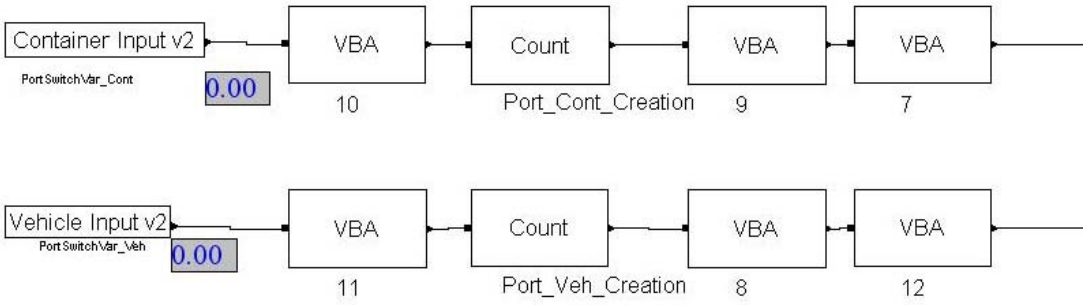



**Figure 14. GCT Gate Submodel Incoming Vehicle and Container Logic**

specifically associated with truck and container objects entering the GCT Gate submodel. The RTI then momentarily changes a "switch variable" in the GCT gate submodel from 0 to 1 to trigger container and/or truck entity creations in the submodel. There are separate switch variables for entering containers and trucks as some trucks may be not be carrying a container when entering the GCT gate submodel. A non-zero switch variable value triggers a container and/or truck creation in the GCT gate submodel at the 'Container Input v2' and 'Vehicle Input v2' blocks, respectively. These **Input** blocks were custom

made using the Arena© Template Developer. Their construction will be discussed in greater depth in Section 3.3.1.1.4, but for now they can be considered analogous to a "triggerable" **Create** block.

When a container entity is created at the 'Container Input v2' block, it then passes through the **VBA 10** block. This executes a VBA command that immediately resets the associated switch variable value to zero to prevent erroneous duplicate container creations. The VBA command for the **VBA 10** block is shown in Figure 15. The container entity then passes through the **Count** block 'Port_Cont_Creation' which increments an internal Arena© counter. This internal counter serves only to collect model statistics for later evaluation. The container entity then passes through the **VBA 9** block. Prior to arriving at this block, the container entity has no associated attribute values; it is effectively a "blank" entity. However, recall that just prior to the entity's

```
Private Sub VBA_Block_10_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
s.VariableArrayValue(8011) = 0
End Sub
```

**Figure 15. Visual Basic Commands for VBA 10 Block**

creation, the RTI collected that specific entering container entity's attribute values from the federation database. The RTI then wrote those values to a specific set of global Arena© variables set aside for container entities entering at the GCT Gate submodel. Therefore, when an entity passes through the **VBA 9** block, it triggers a series of VBA

```
Private Sub VBA_Block_9_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
ActEnt = s.ActiveEntity
s.EntityAttribute(ActEnt, 1001) = s.VariableArrayValue(1011)
s.EntityAttribute(ActEnt, 4004) = s.VariableArrayValue(1014)
s.EntityAttribute(ActEnt, 5005) = s.VariableArrayValue(1015)
s.EntityAttribute(ActEnt, 7007) = s.VariableArrayValue(1016)
End Sub
```

**Figure 16. Visual Basic Commands for VBA 9 Block**

commands that assign the values currently held in the global variable set to the active container entity as attribute values. Note that the attribute variable classes are consistent with Table 2. The VBA commands for the **VBA 9** block are shown in Figure 16. The container entity then passes through the **VBA 7** block which triggers a series of VBA commands that resets to zero all of the global variables associated with entering container attribute values. Figure 17 shows the Visual Basic commands associated with the **VBA 7** block.

```
Private Sub VBA_Block_7_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
s.VariableArrayValue(1011) = 0
s.VariableArrayValue(1014) = 0
s.VariableArrayValue(1015) = 0
s.VariableArrayValue(1016) = 0
End Sub
```

**Figure 17. Visual Basic Commands for VBA 7 Block**

The series of logic that has been described to create container entities at the 'Container Input v2' block and assign attribute values is the same logical construction used to assign attribute values to truck entities created at the 'Vehicle Input v2' block shown in Figure 14. The only difference is that a separate set of unique global variables is used to hold truck attribute values collected by the RTI from the federation database for the GCT Gate submodel.

Once entering container and truck entities have been assigned attribute values, the entities then proceed to a series of logical blocks that directs them from the GCT Gate submodel to other locations within the larger, aggregated GCT model. The series of logic for this entity routing is shown in Figure 18.

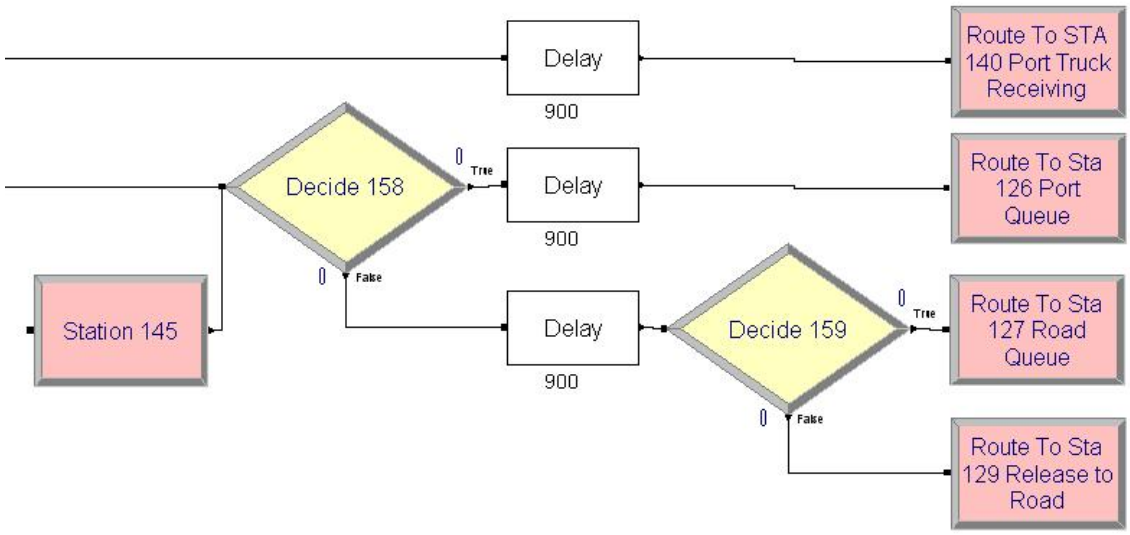



**Figure 18. GCT Gate Submodel Entering Container and Vehicle Routing**

The container entity then proceeds directly to the **Delay** block '900'. This block delays the container entity's passage for some user- defined time interval (in this instance

it is set to 900 seconds, or 15 minutes). This delay is optional and serves to simulate the delay time associated with unloading a container from its truck. The container entity then proceeds to the **Route** block 'Route To STA 140 Port Truck Receiving.' This routes the container entity to a truck receiving submodel that is part of the larger, aggregated GCT submodel. This truck receiving submodel processes entities for loading on to outgoing freight ships. For the purposes of this federated simulation, this can be considered as a terminal destination for containers arriving at the GCT.

Arriving truck entities first proceed to the **Decide** block 'Decide 158' where the truck's 'Vehicle_Type' attribute value is evaluated according to the if/then decision statement [Vehicle_Type = 2]. If the statement is found to be true (i.e., the truck is a port truck type) then the port truck entity proceeds to the **Delay** block '900' which delays the port truck entity's passage for some user- defined time interval (in this instance it is set to 900 seconds, or 15 minutes). Again, this delay is optional and serves to simulate the delay time associated with unloading a container from its truck. The port truck entity then proceeds to the **Route** block 'Route to Sta 126 Port Queue.' This **Route** block sends the entity to the **Station** block 'Port Veh From Road,' shown in Figure 7, which sends the entity to the queue of port trucks available to be batched with outgoing containers.

However, if the decision is evaluated at the **Decide** block 'Decide 158' is found to be false (i.e., the truck is a road truck) it proceeds to the **Delay** block '900' which delays the road truck entity's passage for some user-defined time interval (in this instance it is set to 900 seconds, or 15 minutes). Again, this delay is optional and serves to simulate the delay time associated with unloading a container from its truck. The road truck entity then proceeds to the **Decide** block 'Decide159.' This **Decide** block effectively evaluates

direction according to the length of roadway truck queue available for batching with outgoing containers (this queue is shown in Figure 7). If the queue contains an insufficient number of trucks, the **Decide** block directs the truck entity to the **Route** block 'Route to Sta 127 Road Queue.' If the queue contains a sufficient number of trucks, the **Decide** block directs truck entities to the **Route** block 'Route to Sta 129 Release to Road.' This **Route** block routes these empty, unneeded long-distance truck entities to a series logic that then passes them back to the roadway network model, bound for the I-16 Junction. This so called 'Release to Road' logic will be discussed in the next section.

The 'Decide 159' block evaluates the statement [NQ(Port_RoadVeh_Queue) <=X], where 'NQ(Port_RoadVeh_Queue)' represents the number of trucks in the referenced **Queue** block – 'Port_RoadVeh_Queue' (see Figure 7). The 'X' represents some user input criterion for acceptable minimum queue length. By examining the port road truck object queue length, this logic ensures that queue does not grow to an unrealistic size and maintains a balance of road trucks within the federation system.

Note that Figure 18 also shows a **Station** block 'Station145' upstream from the **Decide** block 'Decide 158.' This block receives truck entities from the Vehicle Diffusion module which recreates truck objects that have been erroneously deleted in the VISSIM© roadway model federate. The construction and reason for the Vehicle Diffusion module is discussed later in subsection 3.3.1.4. For now, this **Station** block can simply be considered an alternate entry point for truck entities entering the GCT Gate submodel.

The logic described in this section for container and truck entities entering the GCT Gate submodel also has a dynamically updated graphical display to show the entity

attribute values currently being held in the unique global variable set for incoming trucks and containers. This graphical display is shown in Figure 19.

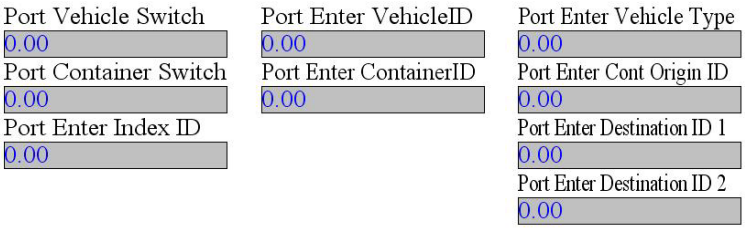



**Figure 19. GCT Gate Submodel Entering Variable Graphic**

*3.3.1.1.3 Garden City Terminal Empty Long-Distance Truck Release*

If there is a sufficient number of long-distance road trucks in the associated queue, the **Decide** block 'Decide 159' (shown in Figure 18) instead routes road trucks to a series of logic blocks that releases the empty (unbatched) truck entities to the roadway network model, bound for the I-16 Junction. Figure 20 shows the series of logical blocks

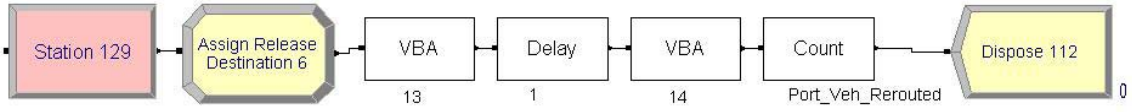



**Figure 20. GCT Gate Submodel Long-Distance Truck Release**

that transfers the empty road truck entities from the GCT Gate submodel to the roadway network model federate.

To be released, the roadway truck entity first arrives at the **Station** block 'Station 129.' It then proceeds to the 'Assign Release Destination 6' assign block. Here the truck

entity is assigned a new attribute called 'Reroute_Destination' with an attribute value equal to 6. This attribute is similar to the 'Destination ID' attribute assigned to containers. It will be used to correctly route the empty truck through the VISSIM© roadway network to the I-16 Junction destination. The method for routing trucks through the roadway network model will be discussed in Section 3.3.4.2.

The truck entity then proceeds to the **VBA 13** block. Similar to VBA blocks 1, 2, 4 and 5 described above, this VBA block executes a series of VBA commands that writes the active entity's attribute values ('Vehicle ID', 'Vehicle_Type' and 'Reroute_Destination') to a set of unique global variables specifically associated with empty road trucks being released by the GCT Gate submodel. Figure 21 shows the VBA commands associated with the **VBA 13** block. The truck entity then proceeds to the **Delay** block where it is delayed for 1 second during which the RTI has one complete time step to collect the exiting truck entity's attribute values that are being held in the

```
Private Sub VBA_Block_13_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
ActEnt = s.ActiveEntity
s.VariableArrayValue(9210) = s.EntityAttribute(ActEnt, 2002)
s.VariableArrayValue(9211) = s.EntityAttribute(ActEnt, 3003)
s.VariableArrayValue(9212) = s.EntityAttribute(ActEnt, 6006)
End Sub
```

**Figure 21. Visual Basic Commands for VBA Block 13**

associated unique global variable set. As seen before, when a truck entity passes through the **VBA 13** block and its 'Vehicle ID' attribute value is written to the associated global

variable, the RTI collects all attribute values being held in the global variable set and writes those values to the federation database. The RTI then creates a truck with the same attributes in the VISSIM© roadway network model.

The truck entity next proceeds to the **VBA 14** block where a series of VBA commands are executed that resets the global variables specific to exiting empty road trucks to zero. Figure 22 shows the VBA commands associated with the **VBA 14** block. The entity then proceeds to the **Count** block 'Port_Veh_Rerouted' which increments a

```
Private Sub VBA_Block_14_Fire()
Dim s As SIMAN
Dim m As Model
Set m = Arena.ActiveModel
Set s = m.SIMAN
ActEnt = s.ActiveEntity
s.VariableArrayValue(9210) = 0
s.VariableArrayValue(9211) = 0
s.VariableArrayValue(9212) = 0
End Sub
```

**Figure 22. Visual Basic Commands for VBA 14 Block**

counter variable internal to Arena© representing the number of road vehicles that are rerouted by the GCT Gate submodel during simulation. The entity is then disposed of at the 'Dispose 112' block. Also as before, a dynamically updating graphic is incorporated to display the active entity attribute values currently being stored in the set of unique global variables assigned to exiting truck entities. Figure 23 shows this global variable display graphic.

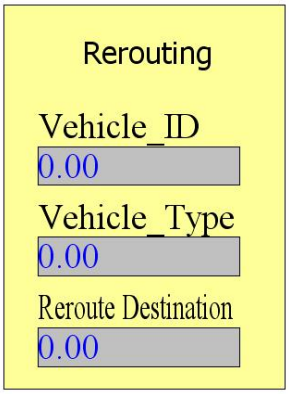



**Figure 23. GCT Gate Submodel Truck Release Variable Graphic**

*3.3.1.1.4 Vehicle and Container Input Block Template Development*

This section will describe the construction of the 'Container Input v2' and 'Vehicle Input v2' **Input** blocks that were created using the Arena© Template Developer. As the 'Container Input v2' and 'Vehicle Input v2' blocks are identical, differing only in name and the entity type generated, the following description of the 'Vehicle Input v2' block can be applied to both blocks. Figure 24 shows the underlying logic for the 'Vehicle Input v2' block. Puglisi (2008) outlines the original design of an earlier version of the Container and Vehicle **Input** blocks [4].

Notice in Figure 24 that there are two parallel series of logical process blocks. The top series, containing three blocks will be referred to as the "switch variable series" and the bottoms series will be referred to as the "vehicle entity series."

The **Create** block in the bottom, "vehicle entity series logic," is set to create 1 truck entity at time zero. This truck entity then proceeds to the **Queue** block. Recall that the **Proceed** block serves as a gate that is either blocked or unblocked. The initial blockage of this **Proceed** block is set to 1, preventing the truck entity from proceeding forward from its current location in the **Queue** block.

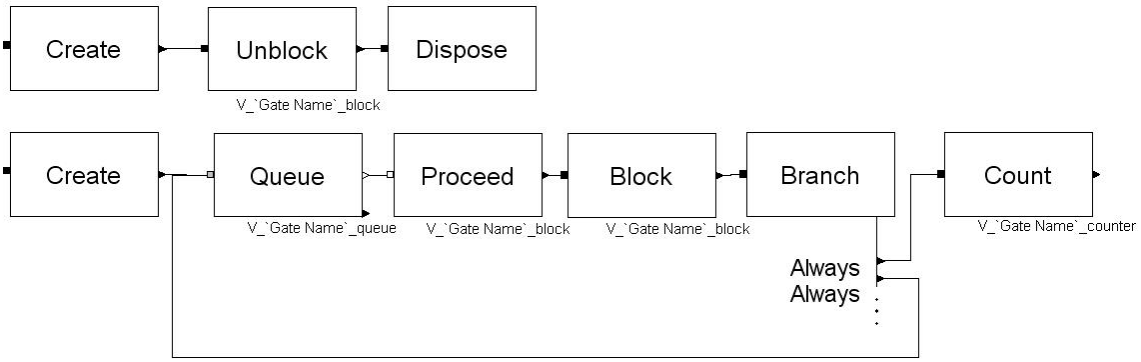



**Figure 24. Vehicle Input v2 Block Template Logic**

Meanwhile, the **Create** block in the top, "switch variable series logic," is set to create a generic entity at a constant time interval. For this model, creations are set to occur at 1 second intervals, or once for every time step of the federation. The batch size of each creation is 1 entity, and the first entity is created at time zero. These generic entities then pass through the **Unblock** block. This **Unblock** block is associated with the blockage at the **Proceed** block in the "vehicle entity series logic," discussed above. However, the number of blockages to be removed by the **Unblock** block at the **Proceed**' block is set equal to a variable called 'Switch Variable.' This switch variable functions the same as that which was discussed above in section 3.3.1.1.2, concerning trucks and containers exiting the roadway network model federate and entering the GCT Gate submodel federate.

As the value of the switch variable is initially zero, it does not initially cause the **Unblock** block to remove any blockages from the **Proceed** block. However, recall that the RTI momentarily sets a switch variable equal to 1 to indicate that a truck is entering the GCT Gate submodel and that a truck entity should be created by the 'Vehicle Input v2' block. When this occurs, the generic entity that passes through the **Unblock** block

every second removes the blockage (i.e., the number of blockages to remove is the switch variable value, or 1) from the **Proceed** block in the vehicle entity series logic. This allows the single entity waiting in the **Queue** block to pass through the **Proceed** block. The entity then passes through the **Block** block, which resets the blockage at the previous **Proceed** block to prevent any additional entities from passing through. The entity then passes through the **Branch** block, which creates a duplicate of the entity. The original entity proceeds to the **Count** block, while the duplicate entity is sent back into the **Queue** block, where it waits for the next time the switch variable is momentarily set equal to 1, indicating that another vehicle is entering the port model at that location. The original entity then proceeds to the **Count** block, where a counter is incremented. This counter is internal to Arena© and counts the number of entities "created" by the 'Vehicle Input v2' **Input** block. The **Count** block is the last block in the 'Vehicle Input v2' **Input** block template. The entity next exits the 'Vehicle Input v2' **Input** block.

Recall from section 3.3.1.1.2 that immediately upon exiting the 'Vehicle Input v2' block, the entity passes through a VBA block (Figure 14) that resets the switch variable value to zero. This is critical to ensure that duplicate entities are not created by the 'Vehicle Input v2' block as the RTI does not reset this value for the next time step. Also, because the **Create** block in the switch variable series logic is set to create generic entities at 1 second intervals, it allows for only one blockage to be removed from the **Proceed** block every time step of the federation. If this interval were set to a smaller time value, it could cause erroneous duplicate entities to be created by the 'Vehicle Input v2' block. Similarly, if the interval were set to a larger time value, it could cause trucks entering the GCT Gate submodel from the roadway network model to be missed and not

created. Section 3.3.5 discusses the limiting model implications of this 1 second interval in greater detail.

The user interface window for the logical block constructed using the Template Developer is shown below in Figure 25. Fields shown in grey must contain information whereas fields shown in white are optional. 'Creation Size' refers to the batch size for the **Create** block in the vehicle entity series of logic. Recall from above that it is set to a value of 1 for this study.

Note in Figure 24 that many of the blocks contain 'Gate Name' in their label. This is because the text entered in the 'Gate Name' field of the interface window is inserted into the queue and blockage names used by the logic blocks to differentiate those queues and blockages from ones in other 'Vehicle Input v2' and 'Container Input v2' template blocks. Therefore, it is important that the text in the 'Gate Name' field be

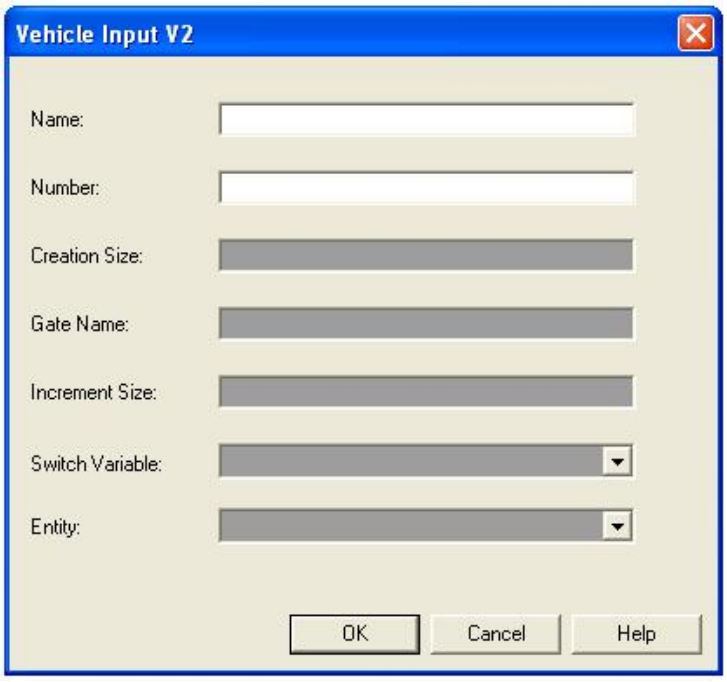



**Figure 25. Vehicle Input v2 Template Interface Window**

unique for every instance of input block. 'Increment Size' refers to the time interval for the 'Create' block in the switch variable series of logic. Recall from above that this was set to 1 second to correspond to the federation time step interval. The 'Switch Variable' field is where the name for the global variable corresponding to the specific truck or container creation switch variable is entered. The 'Entity' field in the interface window allows the user to specify what type of entity is created by the 'Vehicle Input v2' block.

#### 3.3.1.2 Distribution Center Submodel

Now that the complete GCT Gate submodel has been explained, this section will describe the logical structure of a Distribution Center submodel. As the basic structure of each of the three distribution centers is the same, this discussion can be applied to all distribution centers. For purposes of discussion, however, this section will specifically consider Distribution Center 1. Also, many of the fundamental logical process series that were described in the GCT Gate submodel have been reconfigured and reused in the Distribution Center submodels. Therefore, to avoid redundancy, this discussion will not be as detailed as the GCT Gate submodel discussion.

The Distribution Center submodel consists of two general functions. The first function is to receive, route, batch and rerelease containers and trucks. This function is called the Incoming/Outgoing function. The second function is to release, or reroute, unneeded empty trucks back to the roadway network model federate. This second function is called the Rerouting function. Figure 26 shows these two functions and the series of logical processes constituting both functions.

Within the first function there are four processes. The first process is triggered by the RTI to create truck and container entities and assign attribute values to those entities. This simulates an arrival at the Distribution Center submodel. The second process routes container and truck entities to their appropriate queues. Container entities continue to follow the solid arrow to the third process (which references Figure 29) which batches

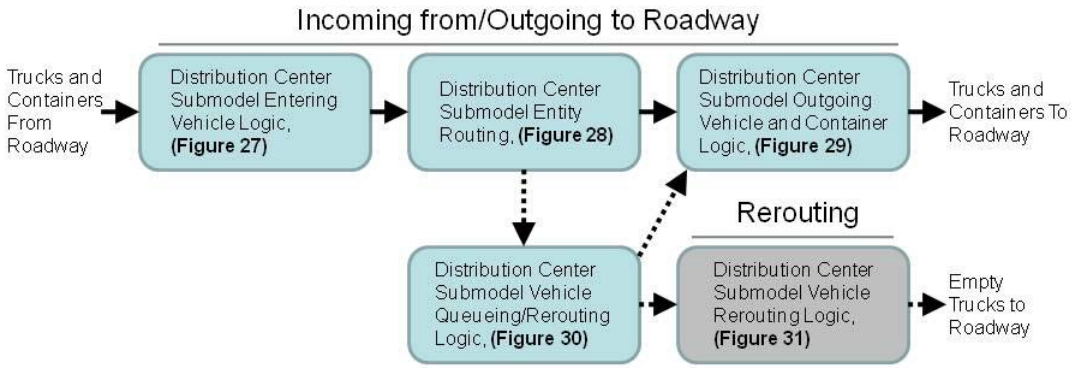



**Figure 26. Distribution Center Submodel Logical Series Overview**

them with outgoing truck entities that are waiting in queues. Truck entities follow the dashed arrows. The fourth process (which references Figure 30) determines whether there is a sufficient number of trucks in the Distribution Center submodel's queue. If there is not a sufficient number of trucks, truck entities are sent up to the third process for batching with outgoing container entities. If there is a sufficient number of trucks in the submodel truck queue, excess or unneeded trucks are sent to the logic process associated with the rerouting function.

The second function consists of one process that releases, or reroutes, unneeded empty trucks to the roadway network model federate. Unneeded port trucks are released

to the roadway destined for the port and unneeded long-distance road trucks are destined for the I-16 Junction.

*3.3.1.2.1 Distribution Center Entering Vehicle Creation*

When a truck in the roadway network model approaches the end of the destination link associated with Distribution Center 1, it triggers a series of commands in the RTI. The RTI first collects the attribute values from the federation database associated with the specific truck entering the Distribution Center 1 submodel. It then writes those attribute values to a set of unique global variables in the Arena© port model federate that are specifically associated with trucks and containers entering Distribution Center 1. The RTI then changes a "switch variable" in the distribution center submodel from 0 to 1 to trigger container and/or truck entity creations in the submodel. The logic blocks for entering truck and container entity creations at Distribution Center 1 are shown below in Figure 27. Note that the structure of these truck and container creation processes within the distribution center submodel are identical to those which were described in subsection 3.3.1.1.2 above for the GCT Gate submodel. The momentary change in switch variable

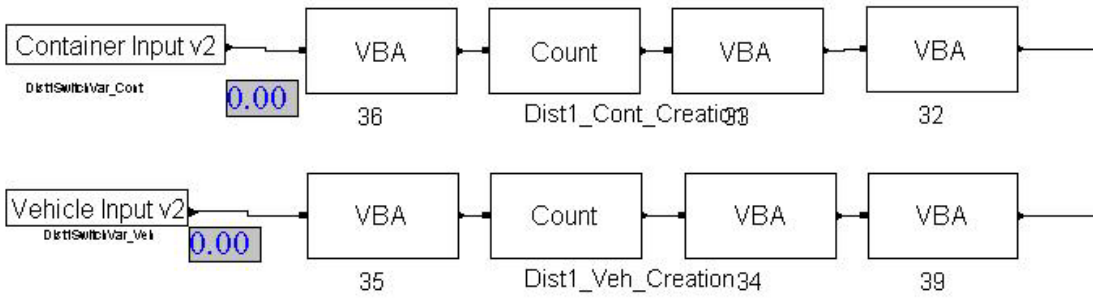



**Figure 27. Distribution Center Submodel Entering Vehicle Creation**

value from 0 to 1 triggers the 'Container Input v2' and 'Vehicle Input v2' blocks to create container and truck entities, respectively.

A created entity, for example a container, then proceeds to the first VBA block (in this case the **VBA 36** block) which resets the switch variable for the corresponding input block immediately from 1 to 0 to prevent duplicate entity creations. The container entity then proceeds to the **Coun**t block (in this case **Count** block 'Dist1_Cont_Creation') which increments a counter internal to Arena©. The entity then proceeds to the second VBA block (the **VBA 33** block) which assigns the attribute values to the entity that were collected by the RTI from the federation database and are currently being held in the unique set of global variables for the Distribution Center submodel. The container entity then proceeds to the third VBA block (the **VBA 32** block) which resets the unique global variable set values to zero. Again, these processes are nearly identical to those described for the GCT Gate submodel in section 3.3.1.1.2, differing only in that the global variables used are unique to the specific distribution center submodels.

*3.3.1.2.2 Distribution Center Submodel Vehicle and Container Routing*

After the entities have been created and assigned attribute values, they proceed to a series of logic blocks that direct the entities to the proper queues to await batching. This series of logic is shown in Figure 28. Container entities first proceed to the **Assign** block 'origin_id' which assigns an 'origin_id' attribute value to the container entity equal to that which is associated with the distribution center; in this case the new 'origin_id' attribute value is equal to 1. This ensures that the next time the container entity is batched with a truck and "passed" to the roadway model federate, its 'origin_id' attribute value reflects the distribution center it is currently leaving.

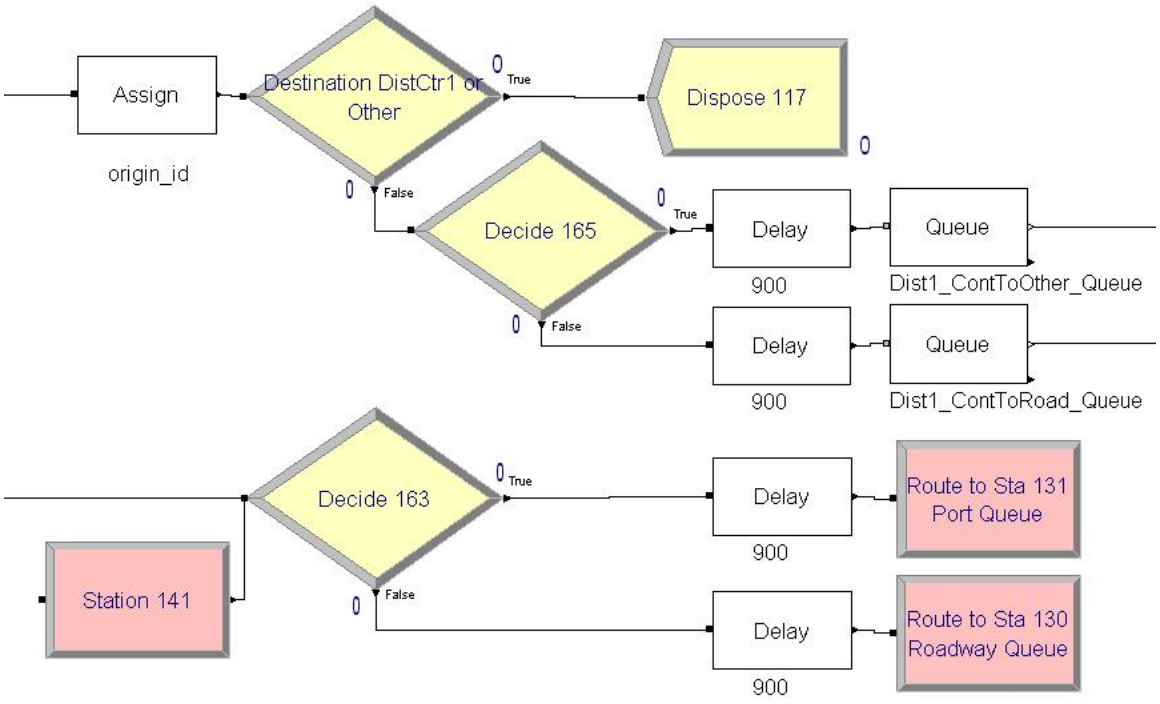



**Figure 28. Distribution Center Submodel Entity Routing**

The container entity then proceeds to the **Decide** block 'Destination DistCtr1 or Other.' This **Decide** block evaluates the container entity's 'Destination ID2' attribute value, to determine if the container's second (and in this case, final) destination is Distribution Center 1, or it has some other second destination (as the distribution centers are intended to only serve as intermediate destinations, this **Decide** block serves primarily as a fail-safe in the event that a container is erroneously assigned a final destination at one of the distribution centers). If the statement [Destination ID2 = 1] is true, the entity proceeds to the **Dispose** block 'Dispose 117.' If the statement is found to be false, the entity proceeds to the **Decide** block 'Decide 165.' This **Decide** block evaluates the entity's 'Destination ID2' attribute value to determine if the second destination is long-distance trucking. If the statement [Destination ID2 ≠ 6] is true, meaning that the

container's next destination is the GCT or another distribution center, then the container entity proceeds to the **Delay** block '900,' where the container is delayed for some user-input time interval (in this case 900 seconds, or 15 minutes) to simulate unloading time. The container entity then proceeds to the **Queue** block 'Dist1_ContToOther_Queue' to await batching with a port truck. However, if the **Decide** block 'Decide 165' finds the statement [Destination ID2 ≠ 6] to be false, meaning that the container's next destination is the I-16 Junction, then the container entity proceeds to the **Delay** block '900.' Again, this block delays the container for some user-input time interval (in this case 900 seconds, or 15 minutes) to simulate unloading time. The container entity then proceeds to the **Queue** block 'Dist1_ContToRoad_Queue' to await batching with a road truck.

Truck entities first proceed to the **Decide** block 'Decide 163' where their 'Vehicle_Type' attribute value is evaluated to determine whether it is a long-distance road truck or a port truck. If the statement [Vehicle_Type = 2] is found to be true, meaning that the entity is a port truck, it then proceeds to the **Delay** block '900' for a user-input unloading delay time interval (in this case 900 seconds, or 15 minutes) to simulate unloading time. The truck entity then proceeds to the **Route** block 'Route to Sta 131 Port Queue' where it is routed to the port truck queuing/rerouting logic (this will be discussed in the next section).

Conversely, if the **Decide** block 'Decide 163' finds the statement (Vehicle_Type = 2) to be false, meaning that the entity is a long-distance road truck, it proceeds to the **Delay** block '900' for a user-input unloading delay time interval (in this case 900 seconds, or 15 minutes) to simulate unloading time. The truck entity then proceeds to the

**Route** block 'Route to Sta 130 Roadway Queue' where it is routed to the road truck queuing/rerouting logic.

Note that Figure 28 also shows a **Station** block 'Station 141' upstream from the **Decide** block 'Decide 163.' This block receives truck entities from the Vehicle Diffusion module which recreates truck objects that have been erroneously deleted in the VISSIM© roadway model federate. The construction and reason for the Vehicle Diffusion module is discussed later in subsection 3.3.1.4. For now, this **Station** block can simply be considered an alternate entry point for truck entities entering the GCT Gate submodel.

*3.3.1.2.3 Distribution Center Submodel Queuing/Rerouting*

When a truck entity leaves the **Route** block 'Route to Sta 131 Port Queue' (shown in Figure 28), it is received by the **Station** block 'Station 131,' shown in Figure 29. From the **Station** block 'Station 131,' the port truck entity next proceeds to the **Decide** block 'Route Directly to Port.' This **Decide** block evaluates the number of trucks present in Distribution Center 1's port truck **Queue** block, also shown in Figure 29. If the number

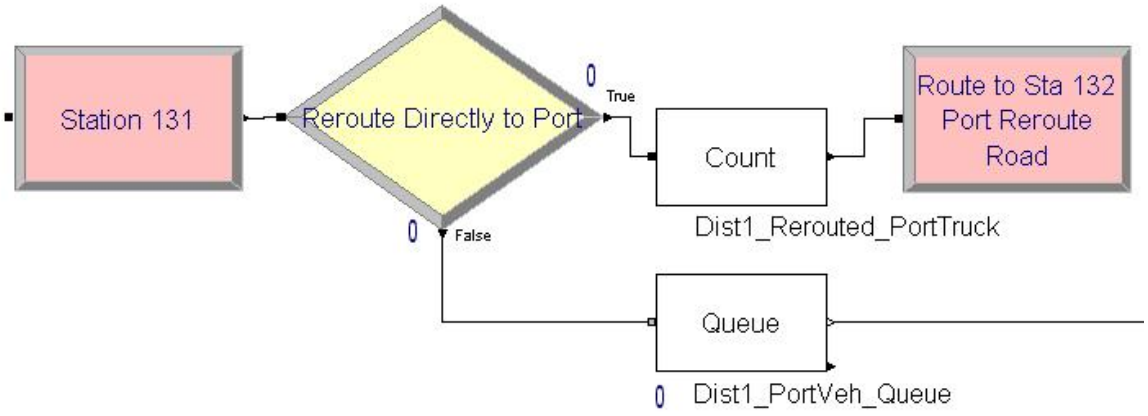



**Figure 29. Distribution Center Submodel Vehicle Queuing/Rerouting Logic**

of idle port trucks in the queue is sufficiently large, the truck entity at this **Decide** block proceeds to the **Count** block 'Dist1_Rerouted_PortTruck'(which increments a counter internal to Arena© denoting the number of port trucks that Distribution Center 1 has rerouted to the GCT during simulation) and then to the **Route** block 'Route to Stat 132 Port Reroute Road.' The **Route** block then sends the empty truck entity to a series of rerouting logic (this logic will be described in Section 3.3.1.2.5). The rerouting logic will release the empty truck to the roadway network model federate, bound for the port, where it will replenish the supply of port trucks in the GCT Gate submodel port truck queue. However, if the **Decide** block finds that the Distribution Center 1 port truck queue is sufficiently small, then the truck entity proceeds to the port truck **Queue** block to await batching with outgoing container entities. The statement that the **Decide** block 'Route Directly to Port' evaluates is [NQ(Dist1_PortVeh_Queue) >= X], where NQ(Dist1_PortVeh_Queue) is the number of trucks in the Distribution Center 1 submodel port truck queue and X is a user-defined value representing the minimum acceptable number of trucks for that queue.

Long-distance road truck entities that are sent from the **Route** block 'Route to Sta 130 Roadway Queue' shown in Figure 28 encounter a series of logic identical to that described above for port truck entities. The only difference is that the queue evaluated in the analogous **Decide** block is the Distribution Center 1 submodel road truck queue, not the port truck queue.

*3.3.1.2.4 Distribution Center 1 Submodel Outgoing Vehicle Logic*

If the truck and container entities are not rerouted directly to the port, but remain at Distribution Center 1, they next proceed to a series of logic that batches truck and container entities for release to the roadway network model federate. This logic is shown in Figure 30. Note that this logic is identical to that which was described for outgoing truck/container batched pairs in the GCT Gate submodel in Section 3.3.1.1.1. As such, a detailed description of the processes shown in Figure 30 will not be conducted

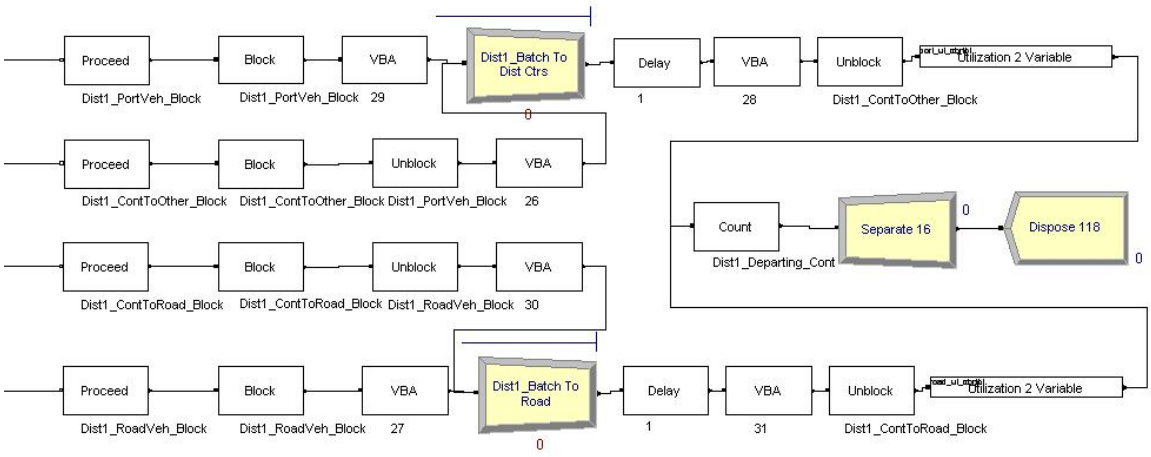



**Figure 30. Distribution Center 1 Submodel Outgoing Vehicle and Container Logic**

However, it should be noted that a different set of unique global variables are used by this submodel than were used by the GCT Gate submodel for the collection of entity attribute values by the RTI.

*3.3.1.2.5 Distribution Center 1 Submodel Vehicle Rerouting Logic*

If the Distribution Center 1 port and/or road truck queues are found to be sufficiently large (see logic in Figure 29, Section 3.3.1.2.3), truck entities are sent to a

series of logic blocks that reroutes these excess, empty trucks to the GCT Gate submodel by "passing" them to the roadway model federate bound for the GCT. This logic is shown in Figure 31. Truck entities are received at the **Station** block 'Station 132' and immediately proceed to the **Decide** block 'True if Veh Type is Port1.' This Decide block evaluates the truck entity's 'Vehicle_Type' attribute value. If the vehicle type is equal to 2, meaning that it is a port truck, the entity is proceeds to the **Assign** block 'Assign Dist 1 to Port.' If the vehicle type is not equal to 2, meaning that it is a long-distance road truck, the truck entity proceeds to the **Decide** block 'Decide 168.'

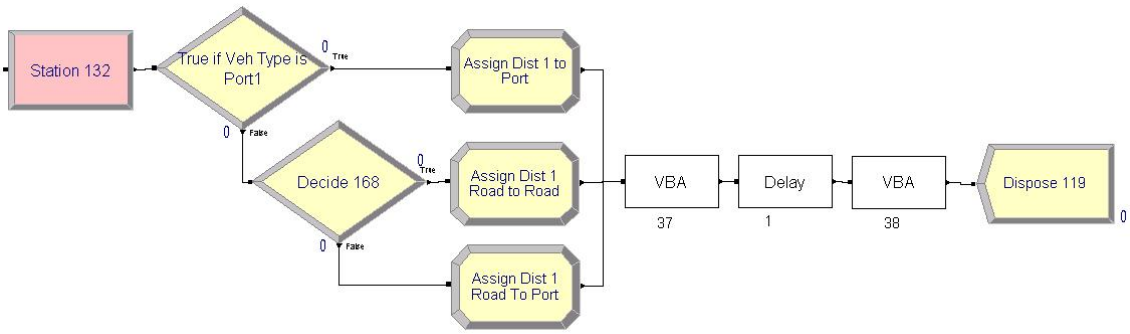



**Figure 31. Distribution Center 1 Submodel Vehicle Rerouting Logic**

Truck entities that proceed to the **Assign** block 'Assign Dist 1 to Port' are assigned an attribute called 'Reroute_Destination' with a value of 7. Note that the value 7 corresponds to the 'Destination ID' value associated with the GCT. All other truck entities proceed to the **Decide** block 'Decide 168,' which evaluates the length of the road truck queue in the GCT Gate submodel. If the queue is sufficient in length, meaning that the road truck is not needed back at the GCT, it proceeds to the **Assign** block 'Assign Dist 1 Road to Road' where it is assigned a 'Reroute_Destination' attribute value equal to

6 for routing to the I-16 Junction. If the number of road trucks in the GCT Gate submodel queue is found to be sufficiently small, the truck entity proceeds to the **Assign** block 'Assign Dist 1 to Port' where it is assigned a 'Reroute Destination' attribute value equal to 7 for routing to the GCT. The statement that the Decide block evaluates is [NQ(Port_RoadVeh_Queue) >= X], where NQ(Port_RoadVeh_Queue) is the number of trucks in the GCT Gate submodel road truck queue. The variable X is a user-defined value representing minimum acceptable road truck queue length for the GCT Gate submodel.

After being assigned a reroute destination attribute value at one of the three **Assign** blocks, the truck entities then proceed to the **VBA 37** block. This block functions similarly to the **VBA 13** block described in section 3.3.1.1.3, and shown in Figure 20, by writing the truck entity's attribute values to a unique set of global variables for collection by the RTI. When the **VBA 37** block executes its commands and the attribute values are written to the global variable set, it triggers a series of commands in the RTI that creates a truck in the VISSIM© roadway network model federate on the link associated with Distribution Center 1. The VBA commands executed by the **VBA 37** block are similar to those of the **VBA 13** block shown in Figure 20.

The truck entity then proceeds to the **Delay** block where it is delayed for 1 second to ensure that the RTI has full one time-step during which to collect the entity attributes values that were written to global variables by the previous VBA block. The truck entity then proceeds to the **VBA 38** block which executes a series of commands to reset to zero the value of the global variables written by the **VBA 37** block. The commands that this VBA block executes are similar to those of the **VBA 14** block shown in Figure 20 for the

GCT Gate submodel. The truck entity then proceeds to the **Dispose** block 'Dispose 119' where it is disposed of.

3.3.1.3 I-16 Junction Submodel

The I-16 Junction submodel serves as the terminus for long-distance road trucks in the Arena© port model federate. This is because it is the point at which road trucks enter or leave the local highway and surface road network from or for the interstate highway system. Therefore, it has two primary functions. First, the I-16 Junction submodel generates long-distance roadway truck traffic, both trucks carrying containers as well as empty trucks. Second, the submodel receives trucks leaving the roadway network model federate to enter I-16 Junction submodel, and therefore the interstate highway system. There are three functions in the I-16 Junction submodel: to simulate the

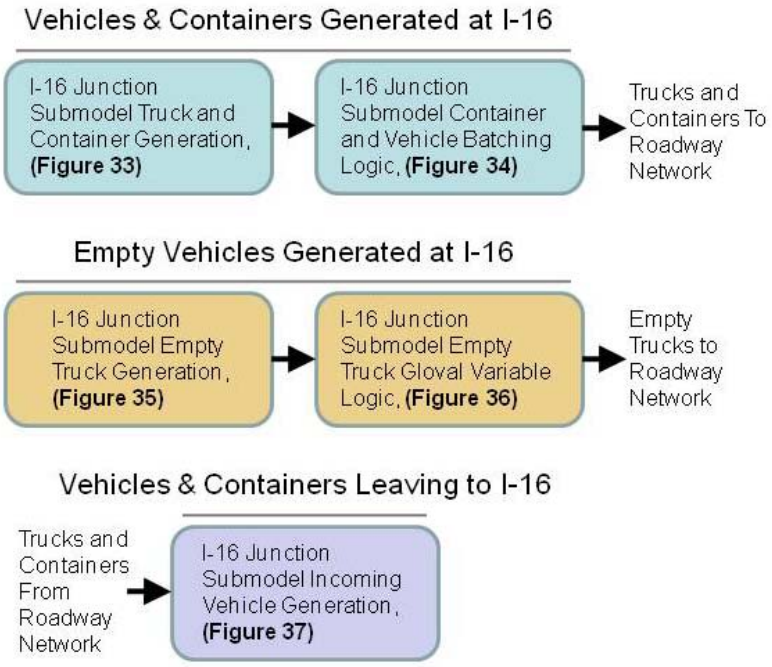



**Figure 32. I-16 Junction Submodel Logical Series Overview**

generation of trucks with containers from the interstate highway system, to simulate the generation of empty trucks from the interstate highway system, and to receive incoming trucks that are leaving the roadway network model federate to enter the interstate highway system. Figure 32 shows these three functions.

The first function, labeled "Vehicles & Containers Generated at I-16," generates outgoing truck and container entities and consists of two processes. Within this function, the first process generates container and truck entities and assigns attribute values to those entities. The second process batches the outgoing container and truck entities together. It then triggers the RTI to pass that entity to the roadway network model federate and then disposes of the batched pair.

The second function, labeled "Empty Vehicles Generated at I-16," generates outgoing empty trucks consists of two processes. Within this function, the first process generates truck entities and assigns attribute values to those entities. The second process triggers the RTI to pass the entity to the roadway network model federate and then disposes of the truck entity.

The third function, labeled "Vehicles & Containers Leaving to I-16" consists of only one process. When triggered by the RTI, this process creates container and truck entities to simulate receiving of incoming containers and trucks from the roadway network model federate. It then disposes of these entities as the I-16 Junction is a terminal destination within the federation.

*3.3.1.3.1 I-16 Junction Submodel Truck with Container Generation from Interstate Logic*

The first function of the I-16 Junction submodel is to generate road trucks carrying containers to simulate their arrival from the interstate highway system. Figure 33 shows the first set of logic blocks in this process. The **Create** block 'Create Road Combos' creates a generic entity using a random exponential distribution for inter-arrival time. Note that this **Create** block generates only one entity per truck/container arrival. Because this **Create** block is set to create entities according to an exponential interarrival rate of several seconds or minutes (the average rate is a user-defined value), it was found during preliminary testing of the model that entities were occasionally created with an interarrival time of less than one second (i.e., less than one individual time-step of the federation). Therefore, it was found that a series of logic blocks is necessary to ensure that only one entity is processed by the Vehicles & Containers Generated at I-16 function of this submodel per each one second time-step of simulation time. Therefore, the newly

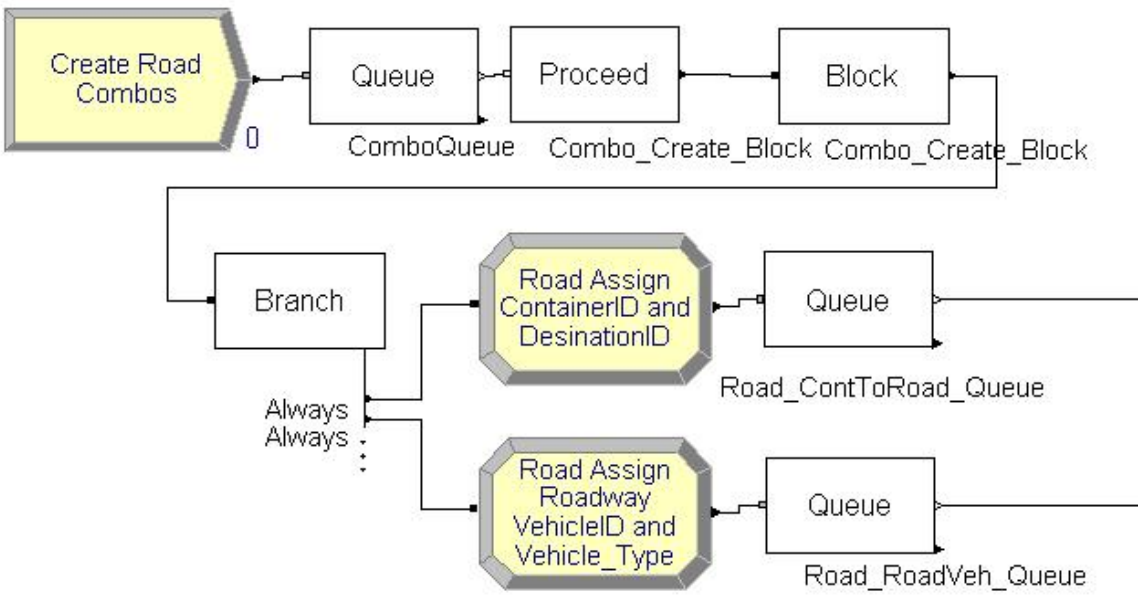



**Figure 33. I-16 Junction Submodel Truck-Container Generation**

created single generic entity then proceeds to a sequence of three blocks that prevent multiple entities entering the system during one single time-step: the **Queue** block 'ComboQueue,' the **Proceed** 'Combo_create_Block,' and the **Block** block 'Combo_Create_Block.' The entity first enters the **Queue** block. If the **Proceed** block downstream has a blockage of zero (recall from earlier that the **Proceed** block functions as a gate, and that a zero value indicates the gate is "open"), then the entity proceeds through the **Proceed** block and on to the **Block** block. As the entity passes through the **Block** block, it creates a blockage at the upstream **Proceed** block, preventing subsequent entities from passing.

The entity then proceeds to the **Branch** block, which serves to simultaneously duplicate the generic block into two generic entities, sending one to each of the two output branches. Creating one entity at a time and immediately duplicating that entity, as opposed to using two **Create** block – one to create truck entities and one to create container entities – ensures that the truck and corresponding container are created simultaneously and according to the same inter-arrival time.

The first entity then proceeds to the **Assign** block 'Road Assign ContainerID and DestinationID.' The first action of the **Assign** block is to assign the entity type to be a container entity. The second action of the **Assign** block is to assign a 'Container ID' attribute value. To ensure that all Container ID numbers are not duplicated, the **Assign** block first increments an internal global variable value called 'Container Counter' by assigning it a new value equal to ['Container Counter' + 1]. The **Assign** block then assign the entity's 'Container ID' attribute value to be equal to the global variable 'Container Counter,' which has just been incremented by one since the previous

container. The **Assign** block then assigns an 'Origin ID' attribute value equal to six (the Destination ID corresponding to the I-16 Junction, see Table 1) and 'Destination ID' and 'Destination ID2' attribute values. The 'Destination ID' and 'Destination ID2' attributes are assigned according to a user-input discrete distribution. An example for this would be to assign distribution center 'Destination ID' values each with a 30 percent occurrence and to assign a GCT 'Destination ID' value with the remaining 10 percent occurrence. The container entity then proceeds to its corresponding **Queue** block.

Entities from the second branch of the **Branch** block proceed to the **Assign** block 'Road Assign Roadway VehicleID and Vehicle_Type.' This block first assigns the entity type to be a truck entity. The **Assign** block then similarly increments a global variable called 'RoadVehicleIncrement' by a value of one. It then assigns the entity a 'Vehicle ID' attribute value equal to the global variable 'RoadVehicleIncrement.' Finally, the **Assign** block assigns the 'Vehicle_Type' attribute value equal to one, denoting a long-distance roadway truck. The truck entity then proceeds to its corresponding **Queue** block.

From this logic, the container and truck entities next proceed to a series of logical blocks identical to those described in Section 3.3.1.1 that properly batches the container and truck entities, writes their respective attribute values to unique global variables specific to the I-16 Junction submodel for collection by the RTI, and triggers the RTI to create the truck/container batched pair in the roadway network model federate. This logic is shown in Figure 34. The only variations between this logic and that described in Section 3.3.1.1, other than different blockage names and unique global variable sets, occurs once the entity leaves the **Unblock** block 'Road_ContToRoad_Block.' Recall

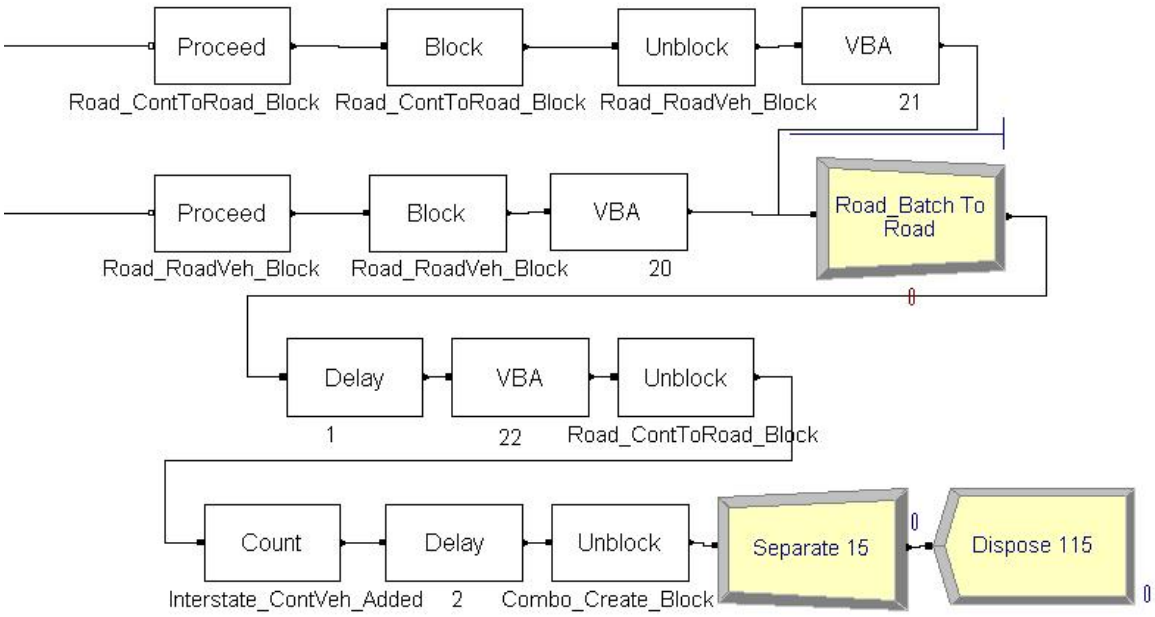



**Figure 34. I-16 Junction Submodel Container and Vehicle Batching Logic**

the discussion that the **Create** block shown in Figure 33, which utilized an exponential function for interarrival rates, occasionally created more than one entity within a single one-second time-step. In Figure 33, a blockage was created at the **Proceed** block 'Combo_Create_Block.' Therefore, once the truck/entity pair leaves the **Unblock** block shown in Figure 34, it first proceeds to the **Count** block 'Interstate_ContVeh_Added' which increments a counter internal to Arena©, and then to a **Delay** block. At the **Delay** block, the entity pair is delayed for two seconds to ensure that at least one time-step of the model elapses. Then the entity pair proceeds to the **Unblock** block 'Combo_Create_Block', which resets to zero the blockage at the **Proceed** block shown in Figure 33. This enables the next entity in the upstream **Queue** to proceed for branching and batching, ensuring that this occurs in a subsequent time step. This familiar logic will not be described in detail, but is shown in Figure 34.

*3.3.1.3.2 I-16 Junction Submodel Truck Generation From Interstate Logic*

The second function of the I-16 Junction submodel generates trucks without containers to simulate the arrival of empty trucks from the interstate highway system. Figure 35 shows the first set of logic pertaining to the generation of empty trucks. Truck entities are first created at the **Create** block 'Create Empty Road Veh.' As discussed above in subsection 3.3.1.3.1, the exponential interarrival of entities from the **Create** block occasionally creates two entities within a single time-step. Therefore, truck entities next proceed to a series of three blocks (a **Queue** block, a **Proceed** block, and a **Block** block) that ensure only one entity proceeds through the downstream logical processes per each time step. These three block function exactly the same as those shown in Figure 36 and described in subsection 3.3.1.3.1.

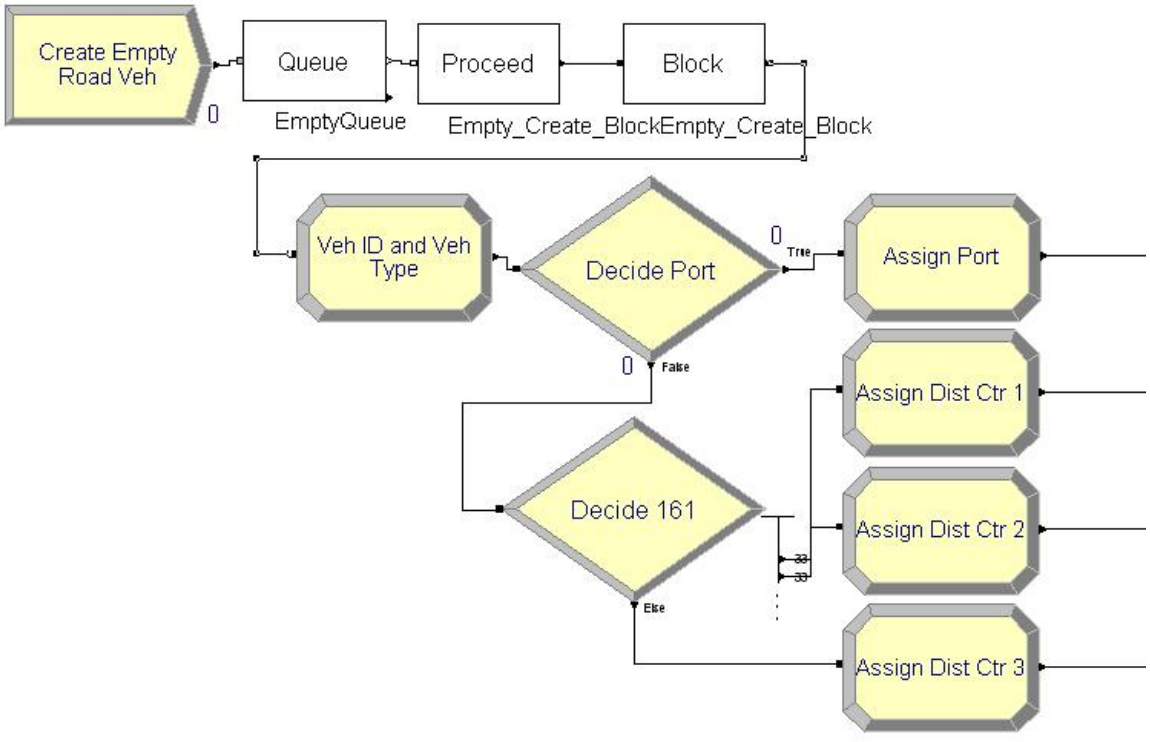



**Figure 35. I-16 Junction Submodel Empty Truck Generation**

Truck entities then proceed to the **Assign** block 'Veh ID and Veh Type.' Similar to the truck/container creation submodel component discussed immediately above in Section 3.3.1.3.1, this assign block first increments the same global variable 'RoadVehicleIncrement' by one. It then assigns the entity a 'Vehicle ID' attribute value equal to the newly incremented 'RoadVehicleIncrement' variable value. It then assigns a 'Vehicle_Type' attribute value equal to one, denoting a long-distance road truck.

The truck entity then proceeds to the **Decide** block 'Decide Port.' The routing decision at this Decide block is '2-Way by Chance' with a user-defined "true" probability value. For example, if the value is set to 10 percent, then there is a 10 percent probability that the entity will be directed towards the **Assign** block 'Assign Port.' Truck entities that proceed to the **Assign** block 'Assign Port' are assigned a 'Reroute_Destination' value equal to seven, the Destination ID corresponding to the GCT.

All other entities are directed towards the **Decide** block 'Decide 161.' The **Decide** block 'Decide 161' has three output branches and directs entities to each output point with a uniformly random one-third probability. From the **Decide** block 'Decide 161,' entities proceed to one of three **Assign** blocks: 'Assign Dist Ctr 1,' 'Assign Dist Ctr 2,' and 'Assign Dist Ctr 3.' These three **Assign** blocks assign 'Reroute_Destination' attribute values equal to one of the three corresponding distribution centers 'Destination ID' values, respectively.

The truck entity then proceeds to the final series of logic that writes its attribute values to a unique series of global variables specific to empty trucks exiting the I-16 Junction submodel. This logic is shown in Figure 36. The sequence of three blocks **VBA 24**, **Delay**, and **VBA 25** function identically to those in the empty truck rerouting logic

for the distribution center submodel to write exiting entity attribute values to a unique global variable set for collection by the RTI (detailed in Section 3.3.1.2.5 and shown in Figure 31). The **Count** block 'Interstate_EmptyVeh_Added' increments a counter internal to Arena© which counts the number of empty road trucks created at the interstate during simulation. Truck entities then proceed to a two second **Delay** block and the **Unblock** block 'Empty_Create_Block.' Similar to the discussion in subsection 3.3.1.3.1 and Figure 34, these blocks delay and then resets to zero the blockage at the upstream **Proceed** block 'Empty_Create_Block' (shown in Figure 35) which ensures that only one container is processed by the logic during each individual time step. Entities then proceed to the **Dispose** block 'Dispose 116' where they are deleted.

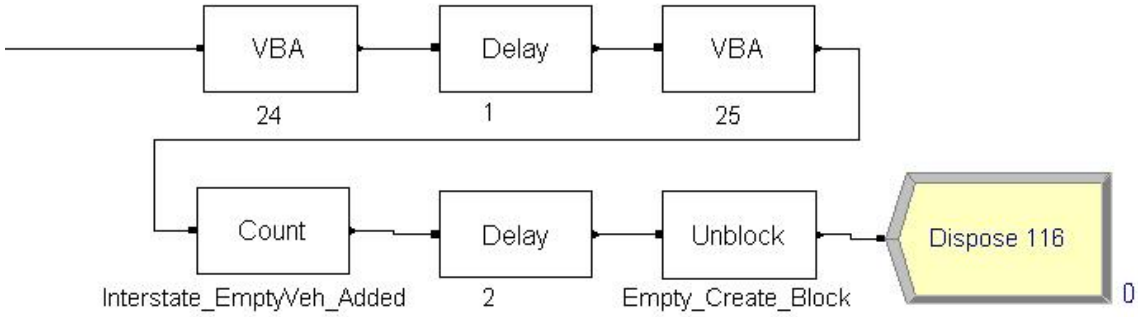



**Figure 36. I-16 Junction Submodel Empty Truck Global Variable Logic**

*3.3.1.3.3 I-16 Junction Submodel Truck Exiting to Interstate Logic*

The final component of the I-16 Junction submodel concerns receiving trucks and containers exiting from the roadway network submodel to the interstate highway system at the I-16 junction interaction point. This series of logic is shown in Figure 37. Note

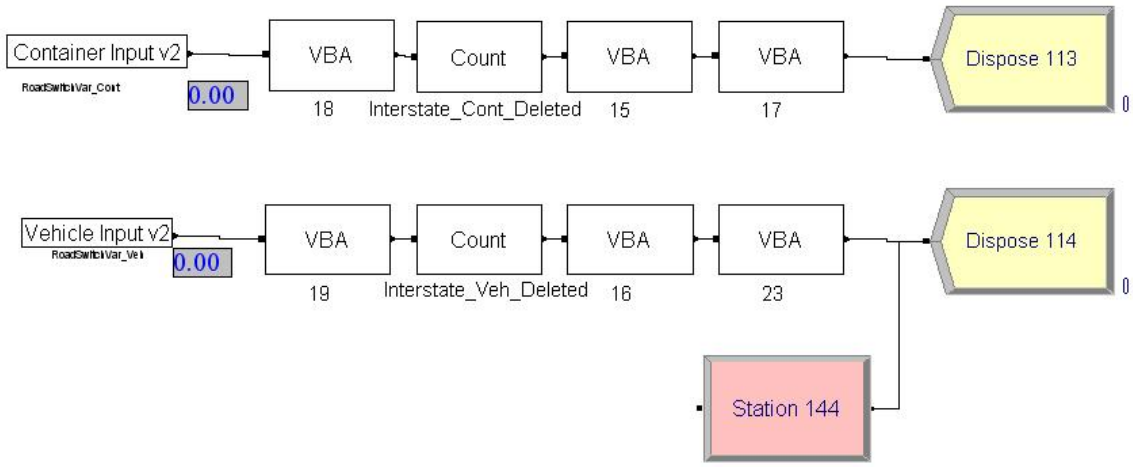



**Figure 37. I-16 Junction Submodel Incoming Vehicle Generation**

that this logic is nearly identical to that detailed in Section 3.3.1.1.2 and shown in Figure 14 concerning generation of incoming trucks and containers in the GCT Gate submodel. Accordingly, this logic will not to be discussed again in detail in this section.

The only variation between this logic and that in the GCT submodel shown in Figure 14 is the addition of **the Station** block 'Station 144.' The **Station** block 'Station 144' receives truck entities from the Vehicle Diffusion module which recreates truck objects that have been erroneously deleted in the VISSIM© roadway model federate. This will be discussed later in subsection 3.3.1.4.

Also, instead of routing these incoming truck and container entities to other submodels within the port model federate, they are disposed of at the end of the logic series as the interstate is a terminal destination.

3.3.1.4 Vehicle Diffusion Module

Occasionally during simulation runtime, VISSIM© deletes, or "diffuses," deadlocked vehicles. Vehicles can become deadlocked for a number of reasons, such as excessive lane-changing delay or passage beyond the "emergency stop position." Therefore, to prevent unrealistic delays that are more a reflection of a modeling limitation than real-world scenarios, VISSIM© intentionally diffuses these deadlocked vehicles (i.e., removes them from the network) after some user-defined deadlock interval has elapsed. Vehicle diffusion, as it as it occurs in the VISSIM© roadway network model federate, will be discussed in greater detail in section 3.3.4.4.1.

While vehicle diffusion accounts for only a small percentage of all trips in the VISSIM© model federate, it can have a significant impact where trucks are a reusable resource. In other words, if 100 port vehicles are created during initialization of the federation, and 10 of these vehicles become diffused over the course of several days of simulation time, the initial resource of 100 reusable port vehicles dwindles over the course of simulation. The purpose of the Vehicle Diffusion Module is to recreate both port and road trucks as they are diffused, during simulation runtime, to effectively replenish the port and road truck resources. Note that the Vehicle Diffusion Module will not recreate the truck in the roadway network, but instead recreates that truck in the Arena© port model federate at the port submodel to which it was destined.

During simulation runtime, the RTI first determines if a port or road truck has been diffused during the previous time-step (this will be discussed in section 3.3.2.2). If the RTI finds that a port or road truck has been diffused during the previous time step, it recreates that truck object in the Vehicle Diffusion Module using the logic shown below

in Figure 38. The RTI first collects all attribute information corresponding to the diffused truck from the federation database. It then writes that information to a unique set of variables specific to the Vehicle Diffusion Module. The RTI then resets the value of a unique switch variable from 0 to 1. This causes a truck entity to be created at the 'Vehicle Input v2' block. The truck entity first proceeds to the **VBA 68** block which resets the switch variable value to 0 to prevent erroneous duplicate entity creations at the 'Vehicle Input v2' block. The truck entity then proceeds to the **Count** block 'Dispersion_Counter' which increments a counter internal to Arena© that counts the number of vehicles diffused during simulation. The truck entity then proceeds to the **VBA 69** block which assigns that entity all of the relevant attribute values that are currently being held in the unique global variable set specific to the Vehicle Diffusion Module.

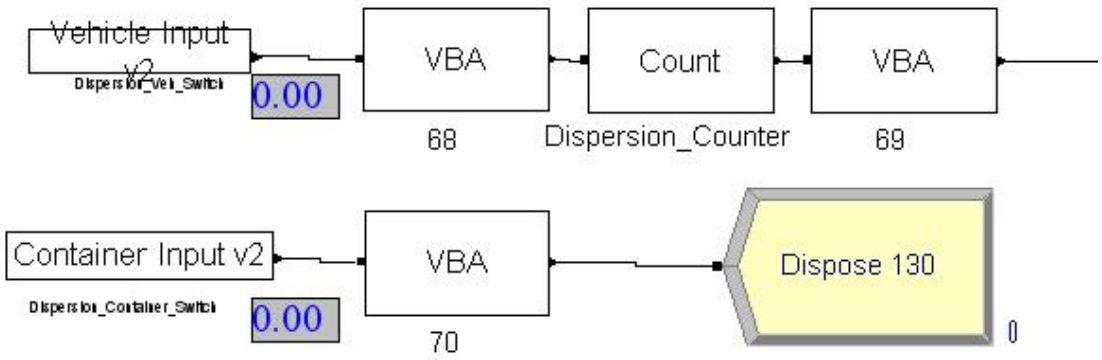



**Figure 38. Vehicle Diffusion Module Vehicle Generation Logic**

If the RTI has determined that the truck that was diffused during the previous time step was carrying a container, it resets a unique switch variable from 0 to 1 which causes a container entity to be created at the 'Container Input v2' block. The container entity

then proceeds to the **VBA 70** block, which executes two commands. First, it resets the switch variable value to 0 to prevent erroneous duplicate entity creations at the 'Container Input v2' block. Second, the block assigns the container entity a 'Vehicle_Type' attribute value equal to that currently being held in the unique set of global variables. Note that while the container does not actually have a certain vehicle type, by assigning it a 'Vehicle_Type' attribute value for the vehicle upon which it was being transported when it was diffused, we can ascertain if it was on a port or road truck. The container entity then proceeds to the **Dispose** block 'Dispose 130' where it is disposed of.

Two important notes should be made about the Vehicle Diffusion Module thus far. First, it is important that the global variables 'road_ut_cont' and 'port_ut_cont' are both incremented by minus one. Doing so ensures that there is not an inaccurate and erroneous gradual increase in the number of containers indicated to be in the roadway network over the course of simulation. Second, the Vehicle Diffusion Module is primarily concerned with maintaining the reusable road and port truck resources during simulation. As containers are not a reused resource during simulation (they travel to an intermediate and then final destination, after which they are deleted from the federation) they are simply disposed of once the aforementioned global variables are incremented by minus one. However, as will be discussed later, diffusion of both the truck and container objects are logged in the federation database.

Once truck entities leave the **VBA 69** block shown in Figure 38, they proceed to a series of logic that directs them to the submodel that they were originally destined for

when they were diffused in the roadway network model federate. This next series of logic is shown in Figure 39.

The truck entity first proceeds to the **Decide** block 'Decide 184', which evaluates the truck entity's 'Dispersion_Destination' attribute value. The RTI has simply set the 'Diffusion_Destination' value equal to either the truck entity's 'Reroute_Destination' attribute value (in the case of an empty truck being diffused) or the container entity's

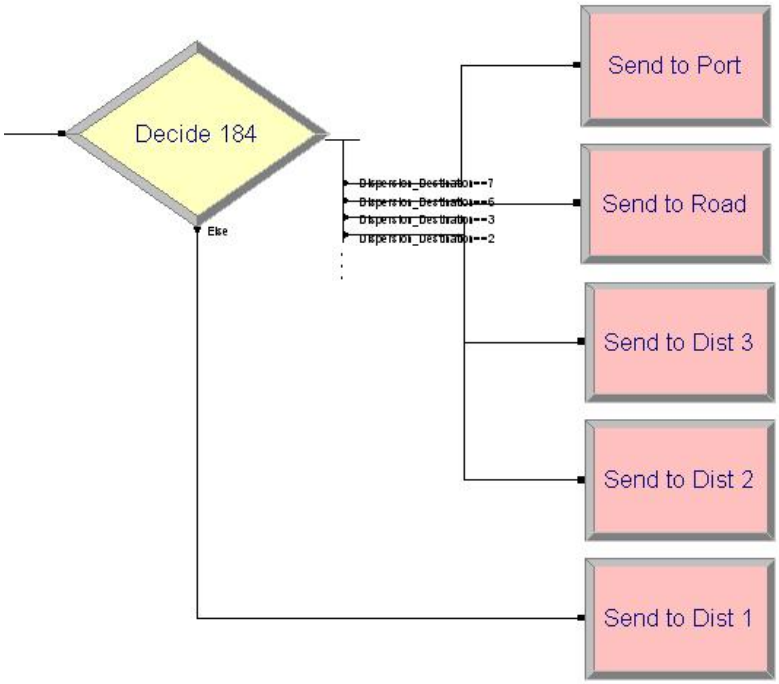



**Figure 39. Vehicle Diffusion Module Vehicle Routing Logic**

'Destination_ID' attribute value (in the case that the diffused truck was carrying a container). This allows the Vehicle Diffusion Module to direct the truck entity to the correct submodel. If the truck entity's 'Diffusion_Destination' value is found to be equal to 7 (indicating that it is a port truck originally destined for the GCT), then the truck

entity proceeds to the **Route** block 'Send to Port'. This routes the truck entity to the **Station** block 'Station 145' located in the GCT Gate submodel, and shown in Figure 18.

If the truck entity's 'Diffusion_Destination' value is found to be equal to 6 (indicating that it is a road truck originally destined for the I-16 Junction), then the entity proceeds to the **Route** block 'Send to Road.' This routes the road truck entity to the **Station** block 'Station 144' located in the I-16 Junction submodel, and shown in Figure 37. If the truck entity's 'Diffusion_Destination' value is found to be equal to 1, 2, or 3, then the entity proceeds to the **Route** block associated with Distribution Center 1, 2, or 3, respectively. For the example of a 'Diffusion_Destination' value equal to 1, the entity proceeds to the **Route** block 'Send to Dist 1.' This **Route** block routes the truck entity to **Station** block 'Station 141' located in the Distribution Center 1 submodel, and shown in Figure 28.

Once diffused trucks are routed to the appropriate submodel within the port model federate, they are processed as arriving trucks and directed to the appropriate queue to await batching with outgoing container objects. Therefore, the supply of both road and port trucks are continually replenished during runtime as truck objects are periodically diffused during simulation.

### 3.3.2 Visual Basic Runtime Infrastructure

The federation RTI was implemented in Microsoft Visual Studio 2005© using the Visual Basic© general purpose programming language. This section describes in detail the structure and command coding of the runtime infrastructure. Relevant portions

of the command code will be provided in the description below and the code in its entirety is contained in Appendix A.

The general structure and execution of the RTI occurs in five steps. Step one, the RTI advances the simulation clock for VISSIM© by one time step – in this study's model, the time step interval is 1 second. Step two, the RTI checks each transaction point (Figure 4) for trucks leaving the VISSIM© roadway network model to enter the Arena© port model. If there are trucks present at one or more of the interaction points, information is exchanged between federates, via the RTI, and the trucks are passed to the Arena© port model. Step three, the RTI advances the Arena© port model simulation clock to match that of the VISSIM© model. Because Arena© is event based, it may execute many events or no events in the 1 second during which it advances its simulation clock to match the time of the VISSIM© simulation clock. Step four, the RTI determines if any port or road trucks were diffused within the VISSIM© roadway network model during the previous time-step. If a vehicle was diffused during the previous time-step, the RTI signals the Arena© port model to recreate that port or road truck (and container if one was being carried on the diffused truck) in the Vehicle Diffusion Module. Step five, the RTI checks each transaction point (Figure 4) for a truck in one or more of the Arena© port submodels entering the roadway network model. Again, if a truck is present at one or more of the interaction points, information is exchanged between the federates, via the RTI, and the trucks are passed to the VISSIM© roadway network model.

The advancement of simulation time in the RTI is coded as a continuous loop, called the "simulation loop." Therefore, after the execution of the fifth step outlined above, the simulation loops to continue at step one for the next federation time step in the

simulation loop. This continuous looping from the fifth step back to the first continues until a user-defined time (terminating condition) is reached.

3.3.2.1 Time Management

Four integer variables are declared in the RTI concerning time: (1) 'Time,' (2) 'DurationTime,' (3) 'aTime,' and (4) 'fTime.' The variable 'Time' is used in the advancement of the federation simulation loop. The variable 'DurationTime' is used to define the overall duration of the simulation currently being executed. The variable 'fTime' is the VISSIM© simulation clock time. The variable 'aTime' is the Arena© simulation clock time. As the Arena© simulation clock essentially advances through as many events as is necessary to catch up to the previous time-step of the VISSIM© simulation clock, the VISSIM© time is the master simulation time, or federation time – hence 'fTime.'

Figure 40 shows the excerpted commands from the RTI code that advance time in the federation. The variables are first dimensioned as 'Long' integers. The simulation loop is then set to run from 'Time' equals 1 to 'Time' equals 'simulation.Period' or 'DurationTime.' The 'simulation.RunSingleStep()' command then advances VISSIM© one, one-second time-step. The variable 'fTime' is then set equal to the current VISSIM© simulation clock elapsed time. A 'While' loop then advances the Arena© simulation clock, or 'aTime,' as many event-based steps as are necessary for it to become equal to 'fTime.' Then, the loop continues for the next integer value of 'Time' and the process repeats. These are effectively steps 1 and 3 of the RTI process discussed above. Step 2, in which the RTI checks for trucks at the interaction points leaving the roadway

```
Dim Time As Long
Dim DurationTime As Long
Dim aTime As Long
Dim fTime As Long
Simulation.Period = DurationTime

For Time = 1 To simulation.Period
     simulation.RunSingleStep()
     fTime = vissim.Simulation.AttValue("ELAPSEDTIME")


     While Int(aTime) <= fTime
            ArenaModel.Step()
            aTime = ArenaLanguage.RunCurrentTime
     End While

Next
```

**Figure 40. RTI Time Advancement Code**

network model, occurs just prior to the 'While' statement. Steps 4, where the RTI checks for diffused vehicles, and step 5, where the RTI checks for trucks at the interaction points leaving the port model to enter the roadway network model, occur immediately following the 'End While' statement and prior to the 'Next' statement that causes the loop to repeat for the next integer value of 'Time.'

3.3.2.2 Object Management and Passing Objects Between Federates

As shown in Figure 4, there are five transaction points where trucks can pass between federates: Distribution Centers 1-3, the GCT Gate, and the I-16 Junction. The RTI code for incoming trucks (those entering one of the Arena© federate port submodels from the roadway network model federate) at each of these transaction points is nearly identical. Similarly, the RTI code for outgoing trucks (those exiting one of the Arena© federate port submodels for the roadway model federate) at each of these transaction points is also nearly identical. The differences in RTI coding among the five interactions

points are primarily that each interaction point has its own unique set of RTI local variables that directly correspond to one of the unique sets of Arena© global variables described in the previous sections.

As it is currently implemented in the RTI, the series of commands that check for trucks exiting the roadway network model federate and entering the port model federate occur before the series of commands that check for trucks exiting the port model federate. This ensures that no truck exiting the network in the first second of simulation time would be inadvertently missed. While it is extremely unlikely that any truck would enter, traverse the roadway network and then be removed from the roadway network all within the first second of simulation, this nonetheless constitutes a best-practice effort to ensure simulation integrity.

In an effort to logically describe the object-oriented movement of a truck/container entity through the model from origin to destination, the exchange of a truck being passed from the port model to the roadway network model will be discussed first. Then the exchange of a truck exiting the roadway network model being passed back to the port model will be discussed. As mentioned, these processes actually occur in the reverse order during execution.

*3.3.2.2.1 RTI Management of Port Model to Roadway Model Entity Exchange*

After the Arena© simulation clock is advanced so that its time matches that of the VISSIM© simulation clock, the RTI checks to see if there are any trucks exiting the port model and entering the roadway model. This is done sequentially, first for the GCT, then for Distribution Centers 1, 2, and 3, and then for the I-16 Junction. Figure 41 shows the

RTI code excerpt for port trucks exiting the port model at the GCT and entering the roadway model at that location. Note that these commands are only for port trucks, not road trucks. Recall from section 3.3.1.1.1 and Figure 8 that port trucks and roadway trucks are batched with containers using separate logic. Therefore, the RTI has a separate series of commands governing road trucks that occur immediately after those shown in Figure 41.

```
If ArenaLanguage.VariableArrayValue(5012) > 0 Then
      array5011 = ArenaLanguage.VariableArrayValue(5011) ''''''''container id''''''''
      array5012 = ArenaLanguage.VariableArrayValue(5012) ''''''''vehicle id''''''''
      array5013 = ArenaLanguage.VariableArrayValue(5013) ''''''''vehicle type''''''''
      array5014 = ArenaLanguage.VariableArrayValue(5014) ''''''''destination id 1''''
      array5019 = ArenaLanguage.VariableArrayValue(5019) ''''''''destination id 2''''
      array5111 = ArenaLanguage.VariableArrayValue(5111) ''''''''origin id''''''''''
      ArenaLanguage.VariableArrayValue(5012) = 0


      ContainerTempTable = ContainerAdapter.GetData
      If ContainerTempTable.Rows.Contains(array5011) = True Then
      ContainerAdapter.UpdateCont(array5011, array5012, array5111,  array5014,
                                              array5019, 9999, aTime, array5011)
      Else
      ContainerAdapter.Insert(array5011, array5012, array5111, array5014, array5019,
                                                                    9999, aTime)
      End If



      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array5012) = True Then
      VehicleAdapter.UpdateVeh(array5012, array5013, 0, 7, array5014, 9999, aTime,
                                                          array5011, array5012)
      Else
      VehicleAdapter.Insert(array5012, array5013, 0, 7, array5014, 9999, aTime,
                                                              array5011)
      End If



      vehicle = vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5014, 50, 50, 1, 0)
      VehIDPort1 = vehicle.ID
      IndexAdapter.Insert(VehIDPort1, array5011, array5012, aTime, 1)

      ContainerLogAdapter.Insert(array5011, array5012, array5111, array5014, array5019,
                                                            9999, aTime, VehIDPort1)
      VehicleLogAdapter.Insert(array5012, array5013, 0, 7, array5014, 9999, aTime,
                                                          array5011, VehIDPort1)
End If
```

**Figure 41. RTI Commands for Vehicles Exiting Port to Roadway Model**

Recall that in the GCT Gate submodel, a truck entity arrives at the batching station after a container entity has already arrived at the station. On its way from the **Queue** block to the **Batch** block (see Figure 8) the entity passes through the **VBA 4** block. Before, it was explained that this block executes a series of commands that writes the entity's attribute values to a set of global variables in Arena© for access by the RTI. In the first line of Figure 41, the statement 'ArenaLanguage.VariableArrayValue(5012)' corresponds to one of these unique global variables in the Arena© model for trucks exiting the GCT gate submodel. Specifically, the **VBA 4** block shown in Figure 8 writes the truck entity's 'Vehicle ID' attribute value to the '5012' global variable in Arena©, and this command line in Figure 41 accesses that global variable value (the 'variable array value,' or 'array value' of variable '5012') through the built-in Arena© COM interface.

Every time step of the federation (that is, every advancement of the 'Time' simulation loop), the RTI executes the 'If' command in the first line of Figure 41. If it is found that a 'Vehicle ID' entity attribute value has been written to the '5012' variable array within the last federation time step, or in other words that the variable array's value is greater than zero ('Vehicle ID' values are always positive, non-zero integers), then the series of commands embedded in the 'If-Then' statement are executed. If the **VBA 4** block has not received an entity and written its 'Vehicle ID' attribute value to the '5012' global variable array in the last time step, the RTI moves to the next 'If-Then' statement associated with the next federation transaction point, which is not shown in Figure 41 but will be discussed later. Effectively, a non-zero value of the '5012' variable array relating to the exiting truck's 'Vehicle ID' attribute value triggers the series of commands in the

RTI that pass the truck from the port model federate to the roadway network model federate.

Recall from Figure 8 that the **VBA 4** and **VBA 1** blocks both write container and truck entity attribute values to unique global variables for access of the RTI. Therefore, if the value of the '5012' variable array is found to have become greater than zero in the last time step, the RTI first collects the the exiting truck entity's attribute values by accessing these global variable arrays through the COM interface. In this instance, these variable arrays are 5011, 5012, 5013, 5014, 5019 and 5111. They are then set equal to local RTI variables 'array5011,' 'array5012,' 'array5013,' 'array5014,' 'array5019,' and 'array5111,' respectively. Note that the corresponding attribute names for each of these variables are commented in Figure 41 (enclosed in several series' of quotation marks at the end of the second through seventh command lines)

Next, the RTI immediately sets 'ArenaLanguage.VariableArrayValue(5012)' equal to zero. As a non-zero value for this variable triggers the RTI to collect entity attribute values and to generate a truck in the VISSIM© model, immediately resetting this to zero ensures that duplicate port trucks and containers are not created during the next time-step of the simulation loop.

The next series of commands, starting with 'ContainerTempTable,' accesses and updates the federation database, created in Microsoft Access©. The command 'ContainerTempTable = ContainerAdapter.GetData' retrieves information from the databases data table that holds container data. It then writes this data to a temporary table that is internal to the RTI called ContainerTempTable. This is done using 'TableAdapters,' an SQL based database query method that is built into the

Access©/Visual Studio© interface. This will be discussed in greater depth in Section 3.3.3.4.

The next 'If-Then' statement checks to see if there is a row in the database with the current container's 'Container ID' as its primary key. If this is found to be true, then the RTI updates that existing entry for the container given its 'Container ID' value (held in the local variable 'array5011') using the 'ContainerAdapter.UpdateCont' command. This command updates the database fields using the collected local RTI variables: array5011, array5012, etc. Note that this includes an entry for 'aTime' to timestamp the record in the database with the time at which the transaction occurred. If the statement is false, then the RTI inserts a new line for the current container using the 'ContainerAdapter.Insert' command, and related local RTI variables, to create an entry for the transaction.

The next series of commands repeats for trucks the process just described for containers. This starts with the 'VehicleTempTable = VehicleAdapter.GetData' command and includes the 'If-Then' statement.

The next series of commands creates the truck in the VISSIM© roadway network model federate. In the first line in the series, the instance 'vehicle' has been previously dimensioned as 'Vehicle,' the VISSIM© COM interface object for a network vehicle. This first line sets the generic VISSIM© vehicle object 'vehicle' equal to 'vissim.Net.Vehicles.AddVehicleAtLinkCoordinate,' providing in parenthesis the vehicle type (Destination ID, or array5014), the desired speed (50 mph), the desired link number (50), the desired lane number on that link (1) and its x-coordinate on that link (0, denoting the beginning of the link). The rationale for setting vehicle type equal to

'Destination ID' relates to the routing methods in the VISSIM© model and will be discussed in Section 3.3.4.2. Nonetheless, once this command is executed, a truck is created in VISSIM© roadway network model federate.

In VISSIM©, vehicle ID numbers corresponding to the vehicle objects generated on the roadway network are randomly assigned by VISSIM©. These vehicle ID values cannot be assigned by the user or any COM interface command. Therefore, it is necessary to collect the VISSIM©-assigned vehicle ID and include that in an indexing table in the database that relates the VISSIM© vehicle ID number with our federation 'Vehicle ID' and 'Container ID' values pertaining to that vehicle. For the purposes of discussion, the VISSIM©-generated vehicle ID will henceforth be referred to as the 'Index ID.'

To extract this VISSIM©-generated 'Index ID,' the command 'VehIDPort1 = vehicle.ID' collects the VISSIM©-assigned vehicle ID and sets it equal to an RTI local variable called 'VehIDPort1.' Then, the next command, 'IndexAdapter.Insert(VehIDPort1, array5011, array5012)', inserts a row entry into an indexing table in the database. Note that the first entry in the row (its primary key) is the VISSIM©-generated vehicle 'Index ID,' the second entry is 'array5011' (the corresponding 'Container ID'), the third entry is 'array5012' (the corresponding 'Vehicle ID'), the fourth entry is 'aTime' (the time at which the transaction occurred), and 1 (a secondary indexing value relevant to the vehicle diffusion commands that will be discussed later).

The final two command lines starting with 'ContainerLogAdapter.Insert' and 'VehicleLogAdapter.Insert' duplicate the container and vehicle database table entries

detailed above in two separate federation database data tables that log all transactions of containers and vehicles across federate boundaries.

Recall that this process is only for port vehicles and containers exiting from the GCT Gate submodel to the roadway network model federate. A similar set of commands are also executed when road vehicles exit the GCT gate submodel to the roadway network model federate. The only significant difference is that a different series of local RTI variables (array5015, 5016, VehIDPort2, etc.) are used.

Lastly, a third series of commands are executed for when empty long-distance road vehicles that are released to the roadway network model federate, bound for the I-16 Junction. These commands are identical to those described above with the exception that no container-related commands are present, and that a separate set of local RTI variables are used.

*3.3.2.2.2 RTI Management of Roadway Model to Port Model Entity Exchange*

The transaction between model federates for vehicles/containers leaving the port model and entering the roadway network model has just been described. This section will detail how the RTI receives vehicles from the roadway network model federate and passes them to the port model federate. Specifically, this section will examine the transaction between federates for vehicles entering the port model federate at the GCT. Because of its length, the RTI commands for this task are shown in two parts. Figure 40 shows the first half of these RTI commands.

The first command line in Figure 42 is an 'If-Then' statement that triggers the transaction of a vehicle between federates. When a vehicle arrives at the end of a

```
If PortDetector.AttValue("IMPULSE") = 1 Then

      array1010 = PortDetector.AttValue("VEHICLEID")
      vissim.Net.Vehicles.RemoveVehicle(array1010)
      ArenaLanguage.VariableArrayValue(1010) = array1010

      IndexTempTable = IndexAdapter.GetDataByIndexID(array1010)
      IndexTempRow = IndexTempTable.FindByindex_id(array1010)
      PortContID = IndexTempRow.container_id
      PortVehID = IndexTempRow.vehicle_id

      If PortContID = 0 Then
          VehicleTempTable = VehicleAdapter.GetDataByVehicleID(PortVehID)
          VehicleTempRow = VehicleTempTable.FindByvehicle_id(PortVehID)
          PortVehType = VehicleTempRow.vehicle_type
          PortVehOrigin = VehicleTempRow.origin_id

          ArenaLanguage.VariableArrayValue(1012) = PortVehID
          ArenaLanguage.VariableArrayValue(1013) = PortVehType
          ArenaLanguage.VariableArrayValue(8010) = 1

          IndexAdapter.DeleteIndex(array1010)
          VehicleAdapter.UpdateVeh(PortVehID, PortVehType, 0, PortVehOrigin, 0, 7,
          fTime, PortContID, PortVehID)
          VehicleLogAdapter.Insert(PortVehID, PortVehType, 0, PortVehOrigin, 0, 7,
          fTime, PortContID, array1010)
      Else
```

**Figure 42. RTI Commands for Vehicles Exiting Roadway to Port Model – Part 1**

destination link in the VISSIM© network model, it crosses a vehicle detector in the roadway. If a vehicle has been detected in the current time step, the "IMPULSE" value of the detector is equal to 1. Otherwise, the "IMPULSE" value of the detector is equal to zero when no vehicles are present over the detector [28]. The RTI checks the "IMPULSE" value of each detector associated with each destination link every federation time step. This is accomplished by evaluating an 'If-Then' statement for each location that contains a VISSIM© COM interface object for each detector. For the destination link associated with the GCT gate submodel, the 'If-Then' statement is 'If PortDetector.AttValue("IMPULSE") = 1 Then'

If a vehicle is detected at the detector associated with the GCT gate, a series of commands are first executed to determine the vehicle's identity. The first line sets the

local RTI variable ‘array1010’ equal to the VISSIM© COM object ‘PortDetector.AttValue(“VEHICLEID”), which extracts the vehicle’s VISSIM©-assigned vehicle ‘Index ID’ from the VISSIM© model federate. The next command removes that vehicle from the roadway network model. The next line in this series sets a unique Arena© global variable equal to the local RTI variable (‘array1010’) value corresponding to the ‘Index ID.’

The next two commands, starting with ‘IndexTempTable’ query the database to recover the federation ‘Vehicle ID’ and ‘Container ID’ attribute values for the current truck/container entities from the IndexTable data table. The local RTI variable ‘PortContID’ is set equal to the ‘Container ID’ attribute value and the local RTI variable ‘PortVehID’ is set equal to the ‘Vehicle ID’ attribute value.

Next, an ‘If-Then’ statement determines whether the entering vehicle is carrying a container or whether it is empty (i.e., has a Container ID = 0). If the statement ‘If PortContID = 0’ is found to be true, then a series of commands are executed to trigger only a vehicle creation in the GCT gate submodel. If the statement is false, then a series of commands are executed to create both a container and vehicle entity in the GCT Gate submodel (these latter commands are shown in Figure 43).

If the vehicle that arrives at the destination link detector is not carrying a container, then the first series of commands query the database data table containing vehicle data. The query extracts the ‘Vehicle_Type’ and ‘Origin ID’ attribute values associated with the vehicle and temporarily assigns them to the local RTI variables ‘PortVehType’ and ‘PortVehOrigin,’ respectively. The next series of commands uses the Arena© COM interface to write these local RTI variables to specific global variables in

the GCT gate submodel. These variable arrays are '1012' and '1013.' These global variables will be accessed by the **VBA 8** block, shown in Figure 14 to assign the associated attribute values to the newly created truck entity in the GCT Gate submodel.

The RTI then sets the Arena© global variable '8010' equal to one. This is the global "switch" variable uniquely associated with the 'Vehicle Input v2' block that triggers a vehicle entity creation in the GCT gate submodel (this was discussed in Section 3.3.1.2.1).

The last three command lines shown in Figure 40 update the federation database tables. The first line deletes the table row in the indexing table associated with the current vehicle's VISSIM©-assigned vehicle 'Index ID.' The next line updates the data table associated with active vehicle data to reflect that the vehicle is currently located at the GCT. The final line updates the data table that logs vehicle transactions to reflect this most recent transaction of the vehicle across the federate boundaries.

As mentioned, Figure 42 shows the first half of the commands associated with a container and/or vehicle passing between the roadway network model and the port model federates. Figure 43 shows the second half of the associated commands. These commands are relevant only should the arriving vehicle be carrying a container (i.e., if the if the 'If PortContID = 0 Then' statement from Figure 42 is false). In that case, the RTI first queries the federation database to collect the attribute values associated with the incoming container. This occurs using commands similar to those described above for querying incoming vehicle attribute values. The RTI then sets several local RTI variables – 'PortDestID', 'PortDestID2' and 'PortOriginID' – equal to the queried attribute values. Recall that the RTI commands in Figure 42 have already recovered the 'Container ID'

```
        ContainerTempTable = ContainerAdapter.GetDataByContainerID(PortContID)
        ContainerTempRow = ContainerTempTable.FindBycontainer_id(PortContID)
        PortDestID = ContainerTempRow.destination_id
        PortDestID2 = ContainerTempRow.destination_id2
        PortOriginID = ContainerTempRow.origin_id

        VehicleTempTable = VehicleAdapter.GetDataByVehicleID(PortVehID)
        VehicleTempRow = VehicleTempTable.FindByvehicle_id(PortVehID)
        PortVehType = VehicleTempRow.vehicle_type
        PortVehOrigin = VehicleTempRow.origin_id

        ArenaLanguage.VariableArrayValue(1011) = PortContID
        ArenaLanguage.VariableArrayValue(1012) = PortVehID
        ArenaLanguage.VariableArrayValue(1013) = PortVehType
        ArenaLanguage.VariableArrayValue(1014) = PortDestID
        ArenaLanguage.VariableArrayValue(1015) = PortDestID2
        ArenaLanguage.VariableArrayValue(1016) = PortOriginID

        ArenaLanguage.VariableArrayValue(8010) = 1
        ArenaLanguage.VariableArrayValue(8011) = 1

        IndexAdapter.DeleteIndex(array1010)
        ContainerAdapter.UpdateCont(PortContID, PortVehID, PortOriginID, PortDestID,
                                    PortDestID2, 7, fTime, PortContID)
        VehicleAdapter.UpdateVeh(PortVehID, PortVehType, 0, PortVehOrigin, 0, 7, fTime,
                                                            PortContID, PortVehID)
        ContainerAdapter.DeleteCont(PortContID)

        ContainerLogAdapter.Insert(PortContID, PortVehID, PortOriginID, PortDestID,
                                                  PortDestID2, 7, fTime, array1010)
        VehicleLogAdapter.Insert(PortVehID, PortVehType, 0, PortVehOrigin, 0, 7, fTime,
                                                            PortContID, array1010)
    End If
End If
```

**Figure 43. RTI Commands for Vehicles Exiting Roadway to Port Model – Part 2**

attribute value when querying the federation database for the VISSIM©-assigned vehicle 'Index ID' value. The RTI then queries the federation database to recover the vehicle attribute values associated with the incoming vehicle. These commands are the same as shown in Figure 42. The RTI then sets a series of local RTI variables – 'PortVehID,' and 'PortVehType,'– equal to these queried attribute values. Recall that, again, the 'Vehicle ID' attribute value was already recovered when querying the federation database for the VISSIM©-assigned vehicle 'Index ID' value. Next, the RTI utilizes the Arena© COM interface to set a unique series of Arena© global variables equal to the local RTI variables holding the container and vehicle attribute values. These global variables will

be accessed by the **VBA 8** and **VBA 9** blocks shown in Figure 14 to assign the associated attribute values to the vehicle and container entities that will be created in the GCT Gate submodel.

Once all attribute values have been written to the unique global variables in the Arena© model federate, the RTI sets the global "switch" variables '8010' and '8011' equal to one. These variables now trigger both the 'Vehicle Input v2' and 'Container Input v2' blocks to create a vehicle and a container entity in the GCT gate submodel.

The final series of commands in Figure 43 update the federation database tables. The first line deletes the table row from the indexing table that corresponds to the VISSIM©-assigned vehicle 'Index ID' value for the vehicle that has just passed between federations. The data tables containing active container and vehicle data are updated to reflect the entities' new locations and timestamps. Next, the data table row corresponding to the container that has just crossed between federates is deleted as that container has arrived at a terminal location: the GCT. Finally, new rows are inserted in the data tables that log container and vehicle transactions to log the transactions and reflect the new locations of the vehicle and container entities, as well as the simulation time at which the transaction across federate boundaries occurred.

Recall that the process just described is only for port vehicles and containers entering the GCT Gate submodel from the roadway network model federate. A similar process to that which was just described is also undertaken for road vehicles exiting the roadway network model federate. The only significant difference is that a different series of local RTI variable are used. These commands are shown in Appendix A.

3.3.2.2 Management and Recreation of Trucks Diffused by VISSIM©

Recall from earlier discussions that vehicles are periodically deleted, or "diffused," by VISSIM© during runtime to prevent unrealistic deadlock on the roadway network. This section will introduce the RTI commands that detect vehicle diffusion in the federation and recreate diffused vehicles within the Arena© port model federate. Because the VISSIM© COM interface does not provide access to any command that identifies whether or not a vehicle was deleted during the previous time-step, an alternate approach had to be adopted.

The fundamental approach is to search the federation database for any port or road truck that has been traveling in the roadway network model federate for an "abnormally" long time interval. The length of this time interval was determined through iterative model calibration and validation by examining average travel times among the various origin-destination pairs in the federation. For the purpose of this study, an "abnormal travel time interval" of 5000 second was used.

Because of the number and length of commands associated with vehicle diffusion, the code is shown in two parts. Figure 44 shows the first part of RTI code associated with detecting and recreating diffused vehicle.

The vehicle diffusion code is embedded within an 'If-Then' loop that starts only if the federation time ('fTime') is greater than 5000 seconds. The second line then sets a local RTI variable 'DispersionSysTime' equal to the federation time ('fTime') minus 5000 seconds. The time-stamped record of active trucks in the federation database will be compared with this 'DispersionSysTime' time value to determine if a truck has been in the roadway network for an "abnormally" long time interval, suggesting it was diffused.

```
If fTime > 5000 Then

                DispersionSysTime = fTime - 5000
                IndexTempTable = IndexAdapter.GetData
                DispersionAdapter.InsertQuery()
                DispersionTempTable = DispersionAdapter.GetData
                DispersionTempRow = DispersionTempTable.FindBydisp_index(1)
                DispersionVehTime = DispersionTempRow.time_stamp

                If DispersionVehTime < DispersionSysTime Then
                    DispersionIndex = DispersionTempRow.index_id
                    DispersionVehicle = DispersionTempRow.vehicle_id
                    DispersionContainer = DispersionTempRow.container_id

                    VehicleTempTable =
                                    VehicleAdapter.GetDataByVehicleID(DispersionVehicle)
                    VehicleTempRow =
                                    VehicleTempTable.FindByvehicle_id(DispersionVehicle)

                    DispersionType = VehicleTempRow.vehicle_type
                    DispersionDestination = VehicleTempRow.current_destination
                    DispersionOrigin = VehicleTempRow.origin_id

                    ArenaLanguage.VariableArrayValue(8082) = DispersionVehicle
                    ArenaLanguage.VariableArrayValue(8083) = DispersionType
                    ArenaLanguage.VariableArrayValue(8084) = DispersionDestination
                    ArenaLanguage.VariableArrayValue(8080) = 1

                    IndexAdapter.DeleteIndex(DispersionIndex)
```

**Figure 44. RTI Commands for Diffused Vehicle Detection and Recreation – Part 1**

The next line, starting with 'DispersionAdapter.InsertQuery()' executes a query of the federation database that copies the first line of the indexing table (the table where the VISSIM©-assigned vehicle 'Index ID' is recorded along with corresponding 'Vehicle ID' and 'Container ID' valued). The query then inserts that first line into a table in the federation database called DispersionTable. Recall that once a truck is passed from the roadway network model federate back to the port model federate, its record in the indexing table is deleted. Therefore, the indexing table only consists of trucks that are active in the roadway network. Also, as the indexing table is sorted according to time stamp values, the first line from that table which is copied to the DispersionTable will always be the oldest record.

The next three lines of code recover the time-stamp of that single, oldest record that was copied from the indexing table, setting the local RTI variable 'DispersionVehTime' equal to that record's time stamp value.

The next command line starts with an 'If-Then' statement that compares that oldest active record's time stamp (currently held in the local RTI variable 'DispersionVehTime') with the 'DispersionSysTime' value, or the federation time from 5000 seconds prior to the current time. If this 'If-Then' statement is found to be true (meaning that the active truck's recorded time-stamp is over 5000 seconds old) then the RTI assumes that the vehicle has been diffused and then executes a series of commands to recreate the truck in the Vehicle Diffusion Module of the Arena© port model federate.

The next eight command lines in Figure 44, starting with 'DispersionIndex…' recovers all of the attribute values associated with the diffused truck object (and container objects, if applicable). The RTI then writes these attribute values to a set of global Arena© variables ('8082', '8083', and '8084') associated with trucks in the Vehicle Diffusion Module. The RTI then sets the "switch" variable '8080' equal to 1, triggering the Vehicle Diffusion Module to create a truck entity at the 'Vehicle Input v2' block (shown in Figure 38). The final command shown in Figure 44 executes a database query that deletes the diffused vehicle's record from the indexing table.

The second half of the RTI commands concerning vehicle diffusion is shown below in Figure 45. These commands perform two functions. First, they determine if a container object was also diffused, and if so then trigger its recreation in the Vehicle Diffusion Module. Second, these commands update all active container and vehicle tables in the federation database (these tables will be discussed in the next section) and

```
If DispersionDestination = 6 Then

   If DispersionContainer = 0 Then
      VehicleAdapter.DeleteVeh(DispersionVehicle)
      VehicleLogAdapter.Insert(DispersionVehicle, DispersionType, 0, 7777, 0,
                               DispersionDestination, fTime, DispersionContainer,
                               DispersionIndex)
   Else
      ArenaLanguage.VariableArrayValue(8081) = 1
      VehicleAdapter.DeleteVeh(DispersionVehicle)
      VehicleLogAdapter.Insert(DispersionVehicle, DispersionType, 0, 7777, 0,
                               DispersionDestination, fTime, DispersionContainer,
                               DispersionIndex)
      ContainerAdapter.DeleteCont(DispersionContainer)
      ContainerLogAdapter.Insert(DispersionContainer, DispersionVehicle, 7777, 0, 0,
                               DispersionDestination, fTime, DispersionIndex)
   End If

Else

   If DispersionContainer = 0 Then
      VehicleAdapter.UpdateVeh(DispersionVehicle, DispersionType, 0, DispersionOrigin,
                               0, DispersionDestination, fTime, DispersionContainer,
                               DispersionVehicle)
      VehicleLogAdapter.Insert(DispersionVehicle, DispersionType, 0, 7777, 0,
                               DispersionDestination, fTime, DispersionContainer,
                               DispersionIndex)

   Else
      ArenaLanguage.VariableArrayValue(8081) = 1
      VehicleAdapter.UpdateVeh(DispersionVehicle, DispersionType, 0, DispersionOrigin,
                               0, DispersionDestination, fTime, DispersionContainer,
                               DispersionVehicle)
      VehicleLogAdapter.Insert(DispersionVehicle, DispersionType, 0, 7777, 0,
                               DispersionDestination, fTime, DispersionContainer,
                               DispersionIndex)
      ContainerAdapter.DeleteCont(DispersionContainer)
      ContainerLogAdapter.Insert(DispersionContainer, DispersionVehicle, 7777, 0, 0,
                               DispersionDestination, fTime, DispersionIndex)
   End If

End If


End If
DispersionAdapter.DeleteDispersionAll()
Else
DispersionAdapter.DeleteDispersionAll()
End If
```

**Figure 45. RTI Commands for Diffused Vehicle Detection and Recreation – Part 2**

log entries of the diffused container and/or vehicle in separate log tables in the federation database.

The commands in Figure 45 are written as an ‘If-Then-Else’ loop that evaluates the statements ‘If DispersionDestination = 6 Then’. If this statement is true, then it

signifies that the diffused vehicle was en route to destination 6, or the I-16 Junction when it was diffused. This is important for the RTI to know in order to correctly update the active container and vehicle tables in the federation database. For example, when a vehicle arrives at a distribution center or the GCT, the RTI updates the vehicle's active record and logged record in the federation database with the new location, time of arrival, etc. Conversely, when a vehicle arrives at the I-16 Junction, the RTI updates the vehicle's logged record, but deletes the active record in the federation database, because the I-16 Junction is a terminal location for containers and vehicles within the federation. That is, it is not an intermediate destination, but a final destination.

If the 'If-Then' statement is found to be true, the RTI continues to a second 'If-Then' loop that evaluates the statement 'If DispersionContainer = 0 Then', where 'DispersionContainer' is the 'Container ID' attribute value of any container that was being carried by the diffused vehicle. This statement determines whether the diffused vehicle was empty (i.e., 'DispersionContainer' = 0) or whether it was carrying a container (i.e., a non-zero 'DispersionContainer' attribute value). If the diffused vehicle is found to have been empty, then a series of two commands, starting with 'VehicleAdapter…' are executed to record just the vehicle's diffusion in the active and log vehicle tables in the federation database. If the diffused vehicle is found to have been carrying a container when it was diffused, the RTI instead executes five commands starting with 'ArenaLanguage.VariableArrayValue(8081)=1'. This first command sets a "switch" variable equal to 1 to recreate the diffused container entity in the Vehicle Diffusion Module of the Arena© port model federate. The four commands that follow record both

the vehicle and container's diffusion in the relevant active and log tables in the federation database.

If the initial 'If-Then-Else' statement in Figure 45 finds that the 'DispersionDestination' is not equal to 6, indicating that the diffused vehicle was en route to a location other than the I-16 Junction, then the RTI executes an 'If-Then' statement similar to that just described. This 'If-Then' statement first determines whether or not the diffused vehicle was carrying a container, and then executes the appropriate commands to record the container and/or vehicle's diffusion in the corresponding active and log tables in the federation database.

The final command lines in Figure 45, starting with 'DispersionAdapter.DeleteDispersionAll()' execute a database query that clears the DispersionTable populated by the first command lines in Figure 44, effectively resetting the database so that the RTI can check for a diffused vehicle during the next time step.

This method of determining whether vehicles were diffused from the roadway network during simulation runtime creates some limitations for both data collection and model operation. These limitations are discussed at greater length in section 3.4.

### 3.3.3 Federation Database Structure and Query Method

As the role of the RTI is purely one of data, time and object management, it does not store any information during simulation execution. Instead, the RTI stores information in a relational database. This section will describe the structure and querying methods of the federation database, constructed using Microsoft Access 2007©.

The federation database consists of seven data tables. The first six tables are concerned with the runtime operation of the simulation: (1) the Container Table, (2) the Container Log, (3) the Vehicle Table, (4) the Vehicle Log, (5) the Index Table and (6) the Dispersion Table. The function of the Container Table is to hold all of the attribute values associated with each container that is active in the federation. Similarly, the Vehicle Table holds all of the attribute values associated with each port or roadway truck that is active in the federation. The Container Log keeps a record of every transaction where a container entity crosses the federation boundary between federates at one of the interaction points. The Vehicle Log similarly keeps a record of every transaction where a vehicle entity crosses the federation boundary. The Index Table matches paired container and vehicle entities that are currently active only in the roadway network model federate by indexing them according to their corresponding VISSIM©-assigned vehicle 'Index ID.' Lastly, the Dispersion Table is used to examine the oldest record in the Index Table to determine if the truck contained in that record was diffused from the VISSIM© roadway network model federate. This was discussed in the previous section.

The last table, (7) the Queues Table, is concerned with collection of queue length data relating to federation system performance. The RTI periodically records the queue lengths at the various queues in the Arena© port model federate during runtime. These values are recorded in the Queues Table in the federation database for post-processing once the simulation has completed.

The following sections will discuss the construction and querying methods for each of these federation database data tables.

3.3.3.1 Container Table and Container Log Data Table Structures

Both the Container Table and the Container Log have the same basic field structure. Figure 46 gives a side by side comparison of the "Design View" structure of these two tables. Note that there are only two differences between the two tables. The first is that the Container Log table has one additional field for 'index_id.' This field refers to the VISSIM©-generated vehicle 'Index ID' for the vehicle on which the

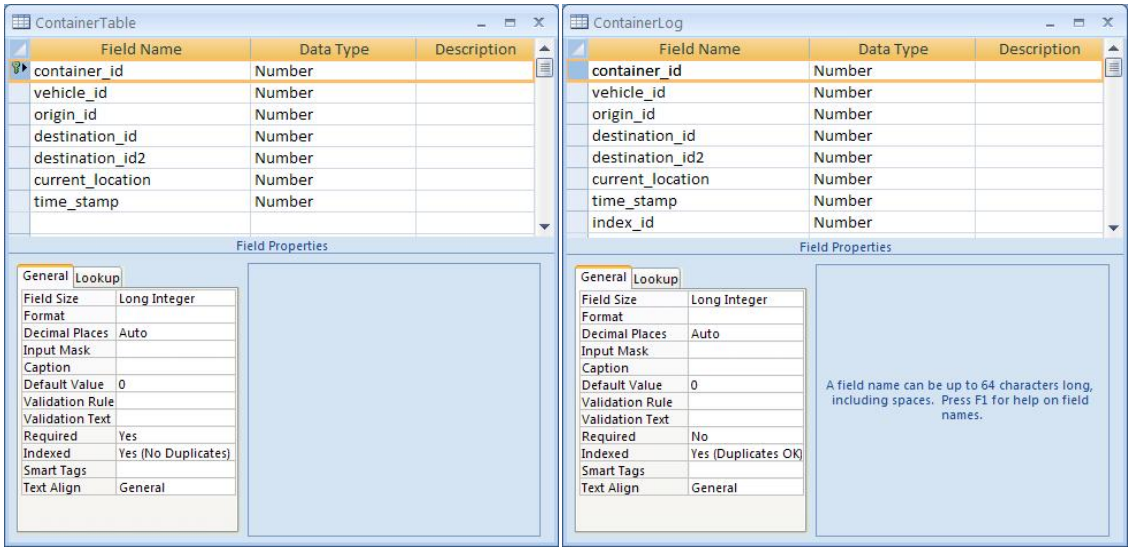
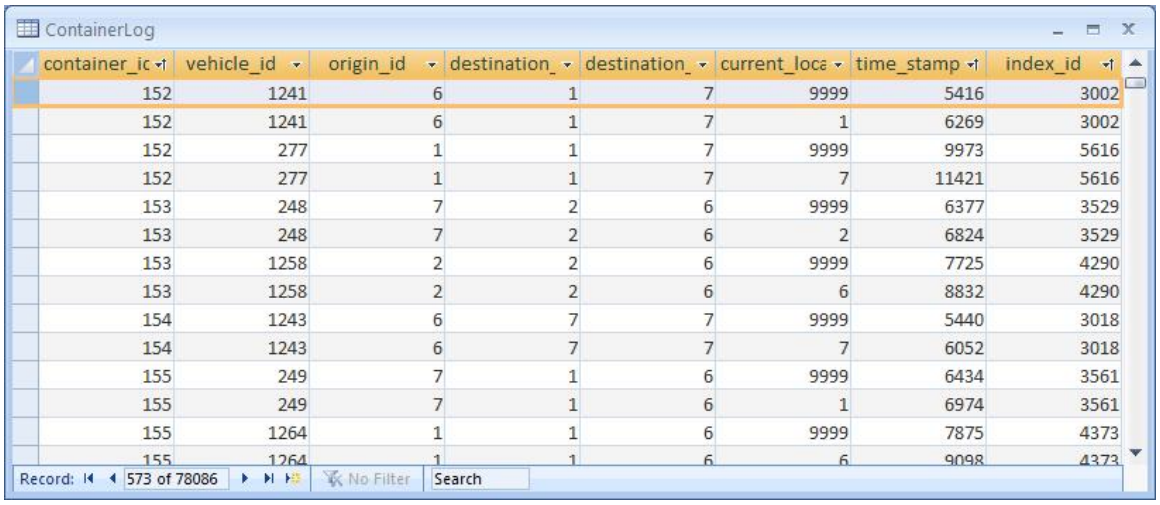



**Figure 46. Federation Database Container Table and Container Log Structure**

container is or was last located. This field was added to the Container Log table primarily for trouble-shooting during development and testing of the federation. The second difference is that the primary key for the Container Table is the 'container_id' field. This ensures that no erroneous duplicate containers are created and recorded during simulation. The Container Log table, however, does not have a primary key. As the purpose of the Container Log table is to provide a record of all of the transactions of

container entities, it is expected that the log table will have multiple entries with the same 'container_id' value.

During simulation execution, these two data tables will be populated with simulation object data and time data. Figure 47 shows an example of the Container Log table populated with data. The first line shows that container 152 was batched with vehicle 1241. The containers origin is the I-16 Junction (origin_id = 6), its first destination is Distribution Center 1 (destination_id = 1) and its second and final



ContainerLog

| container_ic | vehicle_id | origin_id | destination_ | destination_ | current_loca | time_stamp | index_id |
|---|---|---|---|---|---|---|---|
| 152 | 1241 | 6 | 1 | 7 | 9999 | 5416 | 3002 |
| 152 | 1241 | 6 | 1 | 7 | 1 | 6269 | 3002 |
| 152 | 277 | 1 | 1 | 7 | 9999 | 9973 | 5616 |
| 152 | 277 | 1 | 1 | 7 | 7 | 11421 | 5616 |
| 153 | 248 | 7 | 2 | 6 | 9999 | 6377 | 3529 |
| 153 | 248 | 7 | 2 | 6 | 2 | 6824 | 3529 |
| 153 | 1258 | 2 | 2 | 6 | 9999 | 7725 | 4290 |
| 153 | 1258 | 2 | 2 | 6 | 6 | 8832 | 4290 |
| 154 | 1243 | 6 | 7 | 7 | 9999 | 5440 | 3018 |
| 154 | 1243 | 6 | 7 | 7 | 7 | 6052 | 3018 |
| 155 | 249 | 7 | 1 | 6 | 9999 | 6434 | 3561 |
| 155 | 249 | 7 | 1 | 6 | 1 | 6974 | 3561 |
| 155 | 1264 | 1 | 1 | 6 | 9999 | 7875 | 4373 |
| 155 | 1264 | 1 | 1 | 6 | 6 | 9098 | 4373 |

Record: 573 of 78086 No Filter Search

**Figure 47. Federation Database Example Container Log**

destination is the GCT (destination_id2 = 7). It is currently located in the roadway network model (current_location = 9999, where 9999 denotes the roadway network model) and it exited the I-16 Junction for the roadway at simulation time 5416 seconds (time_stamp = 5416). Lastly, the VISSIM© vehicle 'Index ID' (the ID of the actual vehicle object in the roadway network model, or 'index_id') is equal 3002. The next entry in the table shows that container 152 arrived at Distribution Center 1

(current_location = 1) at simulation time 6269 seconds (time_stamp = 6269). All other attribute values for the container remain the same. The third entry in the data table shows that container 152 left Distribution Center 2 on vehicle 277 (vehicle_id = 277) at simulation time 9973 seconds (time_stamp = 9973) and that the VISSIM© vehicle 'Index ID' (index_id) for the vehicle object in the roadway network is 5616. Finally, the fourth line shows that container 152 arrived at the GCT (current_location = 7) at simulation time 11421 (time_stamp = 11421).

Note that because the Container Log table serves as a record of all transactions between federates, row entries in that table will never be deleted during simulation, but will continue to grow in number. The Container Table, on the other hand, is a "working" table, containing only row entries for containers that are active in the system. As the RTI is structured, "active" entities are only those either in-transit in the roadway network model federated, or those located at one of the distribution center submodels awaiting further transit. When a container arrives at the GCT Gate submodel, it is logged in the Container Log table as having arrived, but is deleted from the Container Table as the GCT is considered a final destination for containers in the Arena© port model federate. Similarly, when a container arrives at the I-16 Junction submodel, its arrival is entered as a new row entry in the Container Log table, but it is deleted from the Container Table as the interstate is a final destination and the container has effectively left the federation.

Lastly, when the RTI detects that a container was diffused by the VISSIM© model federate, it records that as an entry in the Container Log table. The record is signified as a diffused container by entering an 'origin_id' value equal to 7777.

### 3.3.3.2 Vehicle Table and Vehicle Log Data Table Structures

The basic structure for both the Vehicle Table and Vehicle Log tables is the same. As above with the container tables, the Vehicle Log table has an additional field for 'index_id' and does not have any specific field identified as the primary key, thereby allowing duplicate entries. Figure 48 shows a side-by-side comparison of the "Design View" structures of the Vehicle Table and Vehicle Log tables.

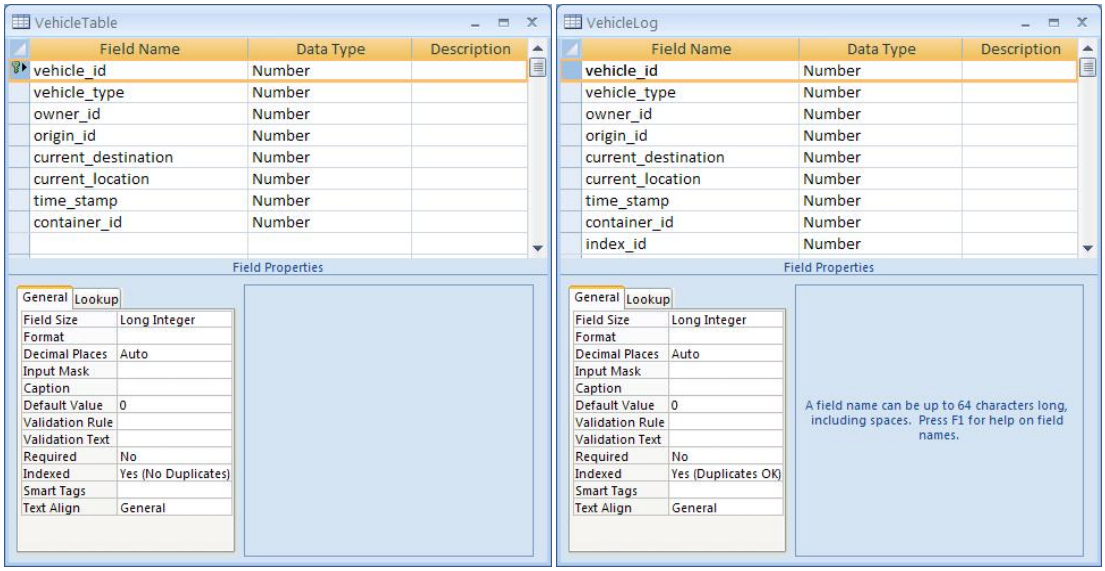
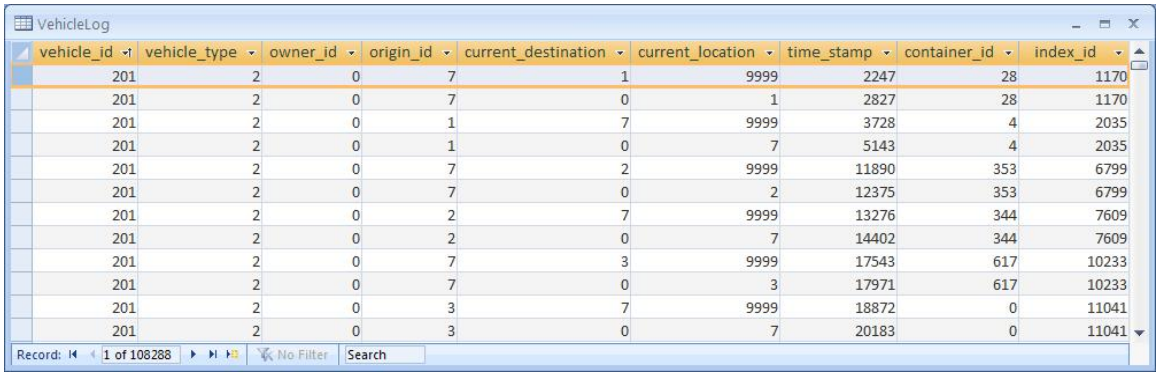



**Figure 48. Federation Database Vehicle Table and Vehicle Log Structure**

During simulation, both the Vehicle Table and the Vehicle Log will be populated with object data and time data for every transaction between model federates. Figure 49 shows an example of the Vehicle Log table populated with data. Note that this table is currently sorted according to 'vehicle_id,' showing each time a vehicle crossed between federates at one of the interaction points. The current table view shows federate transaction records for vehicle 201. The first line shows that vehicle 201 left the GCT



VehicleLog

| vehicle_id | vehicle_type | owner_id | origin_id | current_destination | current_location | time_stamp | container_id | index_id |
|---|---|---|---|---|---|---|---|---|
| 201 | 2 | 0 | 7 | 1 | 9999 | 2247 | 28 | 1170 |
| 201 | 2 | 0 | 7 | 0 | 1 | 2827 | 28 | 1170 |
| 201 | 2 | 0 | 1 | 7 | 9999 | 3728 | 4 | 2035 |
| 201 | 2 | 0 | 1 | 0 | 7 | 5143 | 4 | 2035 |
| 201 | 2 | 0 | 7 | 2 | 9999 | 11890 | 353 | 6799 |
| 201 | 2 | 0 | 7 | 0 | 2 | 12375 | 353 | 6799 |
| 201 | 2 | 0 | 2 | 7 | 9999 | 13276 | 344 | 7609 |
| 201 | 2 | 0 | 2 | 0 | 7 | 14402 | 344 | 7609 |
| 201 | 2 | 0 | 7 | 3 | 9999 | 17543 | 617 | 10233 |
| 201 | 2 | 0 | 7 | 0 | 3 | 17971 | 617 | 10233 |
| 201 | 2 | 0 | 3 | 7 | 9999 | 18872 | 0 | 11041 |
| 201 | 2 | 0 | 3 | 0 | 7 | 20183 | 0 | 11041 |

Record: 1 of 108288 No Filter Search

**Figure 49. Federation Database Example Vehicle Log**

(origin_id = 7) at simulation time 2247 seconds (time_stamp = 2247) carrying container 28 (container_id = 28) and is currently bound for Distribution Center 1 (current_destination = 1). Furthermore, this vehicle is a port vehicle (vehicle_type = 2) and it is currently in-transit in the roadway network model federate (current_location = 9999). Lastly, its VISSIM©-assigned 'Index ID' is equal to 1170 (index_id = 1170).

The second line of Figure 49 shows that vehicle 201 arrived at Distribution Center 1 (currnet_location = 1) at simulation time 2827 seconds (time_stamp = 2827). Because it is not in-transit, the vehicle's current destination is equal to zero.

The third line of Figure 49 shows that vehicle 201 then left Distribution Center 1 (origin_id = 1) at time 3728 (time_stamp = 3728) bound for the GCT (current_destination = 7) carrying container 4 (container_id = 4) and that it's VISSIM©-assigned 'Index ID' is 2035 (index_id = 2035).

The Vehicle Table and Vehicle Log tables contain an additional field titled 'owner_id.' This field is not currently used in simulation but was included to give the future flexibility to track multiple long-distance trucking carriers in the simulation model.

Similar to the container tables, the Vehicle Log table serves as a record of all transactions of vehicles between federates. As such, row entries in that table will never be deleted during simulation, but will continue to grow in number. The Vehicle Table, however, is a working table containing only those vehicles that are currently active in the in federation. Because port vehicles are reused throughout the system and never transport containers to the interstate at the I-16 Junction, they will never be deleted from the working Vehicle Table. Instead, their entries will simply be updated throughout the simulation. Long-distance trucking vehicles, however, will be deleted from the working Vehicle Table upon arrival at the I-16 Junction submodel, were they effectively leave the federation.

Lastly, when the RTI detects that a vehicle was diffused by the VISSIM© model federate, it records that as an entry in the Vehicle Log table. The record is signified as a diffused vehicle by entering an 'origin_id' value equal to 7777.

#### 3.3.3.3 Index Table Data Table Structure

Recall that when a vehicle object is created in VISSIM©, the COM interface does not allow a vehicle ID number to be assigned to the vehicle object by an outside program or RTI. Instead, VISSIM© assigns its own unique vehicle 'Index ID' to the vehicle object that has been created on its network. Therefore, it is necessary to relate the federation 'Vehicle ID' and 'Container ID' entity attributes to the corresponding VISSIM©-generated vehicle 'Index ID.' Accordingly, the Index Table pairs the federation 'Container ID' and 'Vehicle ID' attributes for the entities currently in-transit in the roadway network with their corresponding VISSIM©-assigned vehicle 'Index ID.'

Records in the Index Table also contain time-stamp values as well as a 'dispersion index' or 'disp_index' value (this will be discussed in the next section). Figure 50 shows the "Design View" structure of the Index Table. Note that the 'index_id' field is designated as the primary key for the table to prohibit erroneous duplicate entries for individual vehicle objects created in VISSIM©.

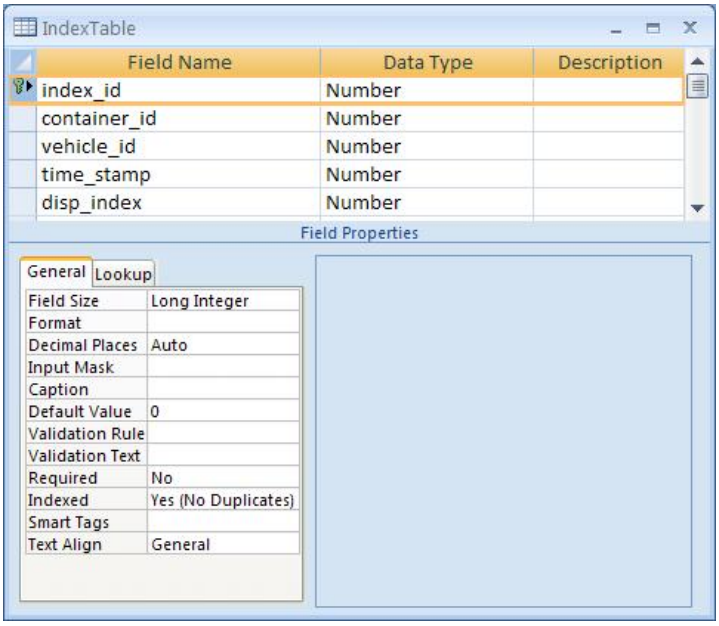



**Figure 50. Federation Database Index Table Structure**

Only vehicle objects that are actively in-transit in the roadway network are represented in the Index Table. When VISSIM© vehicle objects are removed from the roadway network upon arrival at any of the destinations (i.e., it is no longer active in the roadway network model) the corresponding row in the Index Table is deleted. Figure 51 shows an example of the Index Table that is populated with data. The first line shows that container 20475 is currently in transit in the roadway network, carried by vehicle 272. Also, it shows that the 'index_id' is equal to 263478, meaning that within the

VISSIM© model federate, the physical vehicle associated with this batched vehicle/container pair is VISSIM© vehicle number 263478. The record was entered in the Index table at time 430426 seconds (time_stamp = 430426). Also note that for this record, and all records in the Index Table, the Dispersion Index (disp_index) value is equal to 1. The reasoning for this will be explained later.

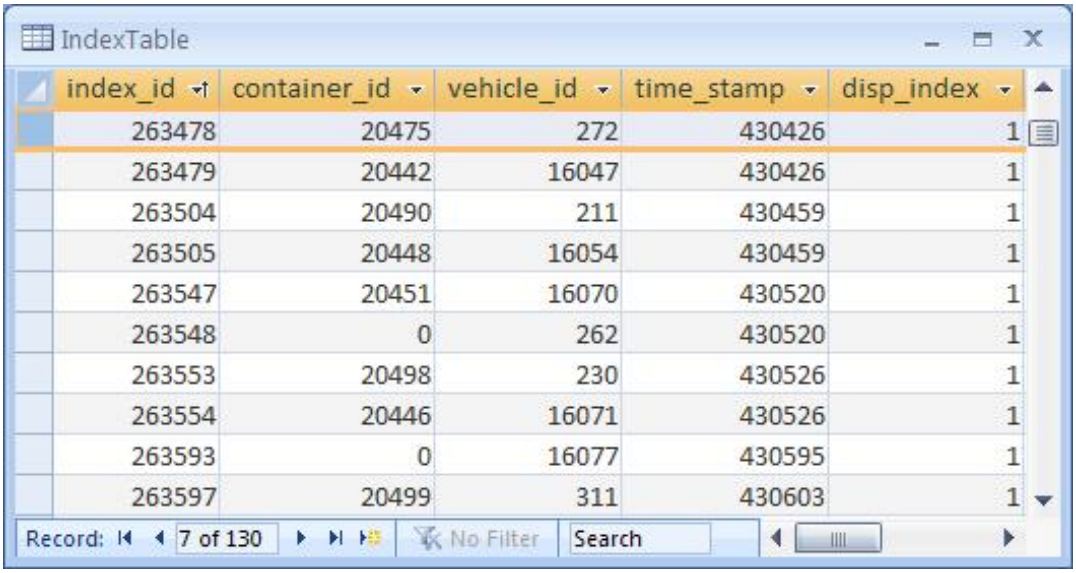



**Figure 51. Federation Database Example Index Table**

3.3.3.4 Queue Data Table Structure

During simulation, the RTI periodically retrieves port facility queue values from the Arena© port model federate. The frequency with which this information is collected is a user-defined value. For the purposes of this study, queue length information is collected every 60 seconds, or 1 minute.

The queue length values that are collected by the RTI are recorded in the Queue Table. Figure 52 shows the "Design View" of each of that table. The 'Type' field

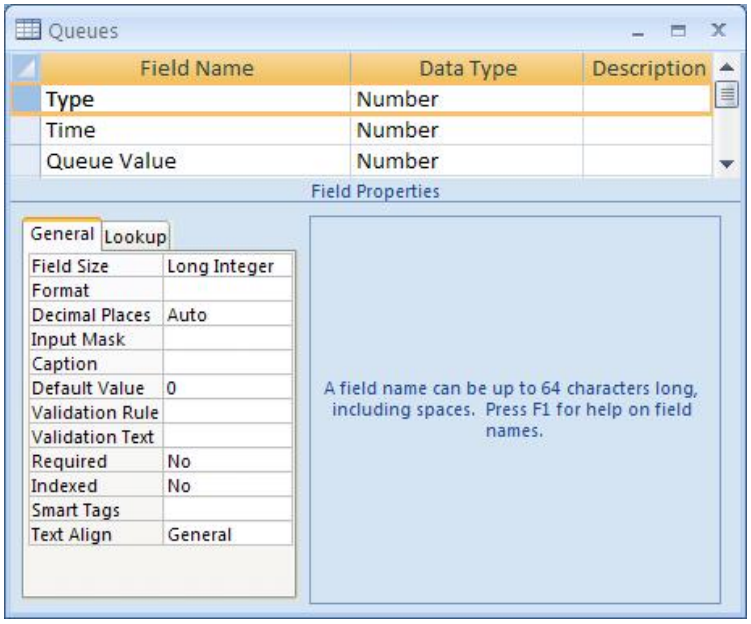



**Figure 52. Federation Database Queue Table Structure**

identifies which queue is recorded in that line. The 'Time' field indicates the simulation time in minutes at which that record was collected. The 'Queue Value' fields indicate the queue length for each recorded entry. Figure 53 shows an example of the Queue Table

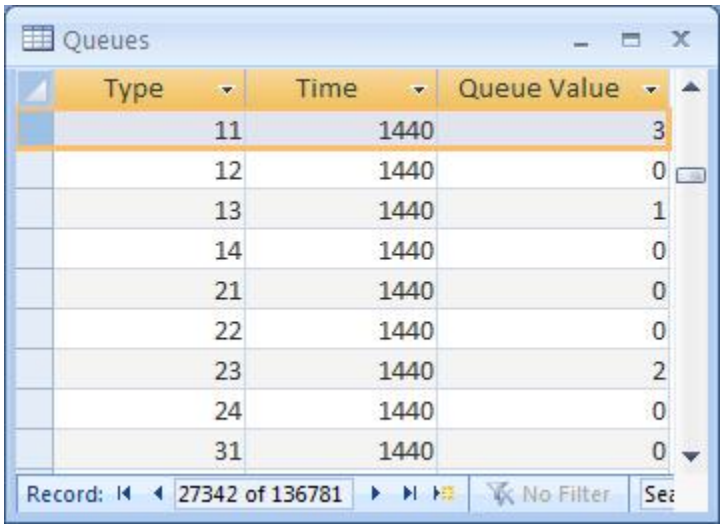



**Figure 53. Federation Database Example Queues Table**

populated with truck queue entries. The first line indicates that queue 'Type' 11, at 'Time' 1440 minutes has a 'Queue Value' equal to 3. As 'Type' 11 corresponds to the 'Dist1VehQp' queue, this entry indicates that at the end of the 1440th minute of simulation, there were at total of 3 port trucks in the port truck queue at Distribution Center 1. The second line indicates that queue 'Type' 12, at 'Time' 1440 minutes has a 'Queue Value' equal to 0. As 'Type' 12 corresponds to the 'Dist1ContQP' queue, this entry indicates that at the end of the 1440th minute of simulation, there were at total of 0 port containers in the port container queue at Distribution Center 1. Table 4 shows each queue 'Type' and its corresponding Arena© queue.

**Table 4. Database Queue Types and Corresponding Arena© Queues**

| Queue Type | Queue | Queue Type | Queue |
|---|---|---|---|
| 11 | Dist1_PortVeh_Queue | 31 | Dist3_PortVeh_Queue |
| 12 | Dist1_ContToOther_Queue | 32 | Dist3_ContToOther_Queue |
| 13 | Dist1_RoadVeh_Queue | 33 | Dist3_RoadVeh_Queue |
| 14 | Dist1_ContToRoad_Queue | 34 | Dist3_ContToRoad_Queue |
| 21 | Dist2_PortVeh_Queue | 71 | Port_PortVeh_Queue |
| 22 | Dist2_ContToOther_Queue | 72 | Port_ContToOther_Queue |
| 23 | Dist2_RoadVeh_Queue | 73 | Port_RoadVeh_Queue |
| 24 | Dist2_ContToRoad_Queue | 74 | Port_ContToRoad_Queue |

3.3.3.5 Dispersion Data Table Structure

The determine whether a vehicle was diffused during the previous time step the oldest record in the Index Table is examined. However, when searching for records in a data table, queries must be performed by querying the "primary key" value associated with the desired record. As primary key values are, by definition, unique to each entry,

this querying method guarantees that only one record will be returned for each query. However, this creates a problem when performing queries on the Index Table because the primary key value ('index_id') of the first record in the Index Table is unknown. To circumvent this problem, the first record in the Index Table is copied to the Dispersion Table, which has a different field set as its primary key. Therefore, the Dispersion Table serves as a temporary table in which records are held while they are examined by the RTI to determine if a vehicle has been diffused by the VISSIM© model federate during runtime. The Dispersion Table has the same exact structure as the Index Table (refer to Figures 50 and 51), with the exception that the Dispersion Table's primary key is set as the 'disp_index' field, not the 'index_id' field.

As detailed in section 3.3.2.2, during runtime the RTI copies the first record in the Index Table to the Dispersion Table. This includes the 'disp_index' value which is always equal to 1 for each record. As the Index Table is organized in ascending order of 'time_stamp' values, the first record in the Index Table is also the oldest. Also, because the final commands shown in Figure 45 clear the Dispersion Table at the end of each time step, this record that the RTI copies from the Index Table to the Dispersion Table at the beginning of each time step will be the only record in the Dispersion Table. Therefore, when the series of commands in Figure 44 are executed that query the Dispersion Table and recover attribute values, the query of the Dispersion Table table (which only contains one record) searches for the in the 'disp_index' primary key value, which is always equal to 1. As mentioned, this query method does not pose a problem as there is only ever one record in the Dispersion Table during any given time step.

3.3.3.6 Federation Database Query Methods – Table Adapters

The RTI queries, writes and retrieves data from the federation database employing a method unique to Visual Studio© called "Table Adapters." Table Adapters utilize a series of Structured Query Language (SQL) statements unique to each data table within the database. Separate Table Adapters were created for each data table for the following four basic data queries: (1) Get Data/Fill Query, (2) Insert Query, (3) Update Query, and (4) Delete Query. However, not all data tables require each of the four Table Adapter query types. For example, row entries in the Index Table are only for vehicles active in the roadway network model; records are inserted when the vehicle enters the roadway network and deleted when the vehicle exits the roadway network. Therefore, an 'Update

**Table 5. Federation Database Table Adapter Queries**

| Data Table Name | Table Adapters | Table Adapter Names |
|---|---|---|
| Index Table | Insert<br>Delete<br>Get Data/Fill | Insert<br>DeleteIndex<br>GetDataByIndexID |
| Dispersion Table | Get Data/Fill<br>Delete<br>Insert | GetData<br>DeleteDispersionAll<br>InsertQuery |
| Container Table | Insert<br>Delete<br>Get Data/Fill<br>Update | Insert<br>DeleteCont<br>GetDataByContainerID<br>UpdateCont |
| Container Log | Insert | Insert |
| Vehicle Table | Insert<br>Delete<br>Get Data/Fill<br>Update | Insert<br>DeleteVeh<br>GetDataByVehicleID<br>UpdateVeh |
| Vehicle Log | Insert | Insert |
| Queues Table | Insert | Insert |

Data' Table Adapter is not needed for management of the Index Table. Table 6 shows the Table Adapters associated with each data table and the unique name associated with each Table Adapter.

While the data tables were created using Microsoft Access©, the Table Adapters were created directly in the RTI in Visual Studio©. This is done by first importing the federation database into the RTI using the 'Add New Data Source' feature of Visual Studio©. Once the federation database has been successfully imported as a data source into the RTI, Table Adapters are created in association with each data table. Figure 54 shows a view of the imported federation database in the Visual Studio© RTI with each set of table adapter queries listed under each respective data table. Note that are no 'Insert' Table Adapter queries listed (with the exception of the Dispersion Table, in which the InsertQuery() Table Adapter executes an SQL command to copy the first line from the Index Table into the Dispersion Table). This is because Visual Studio© creates

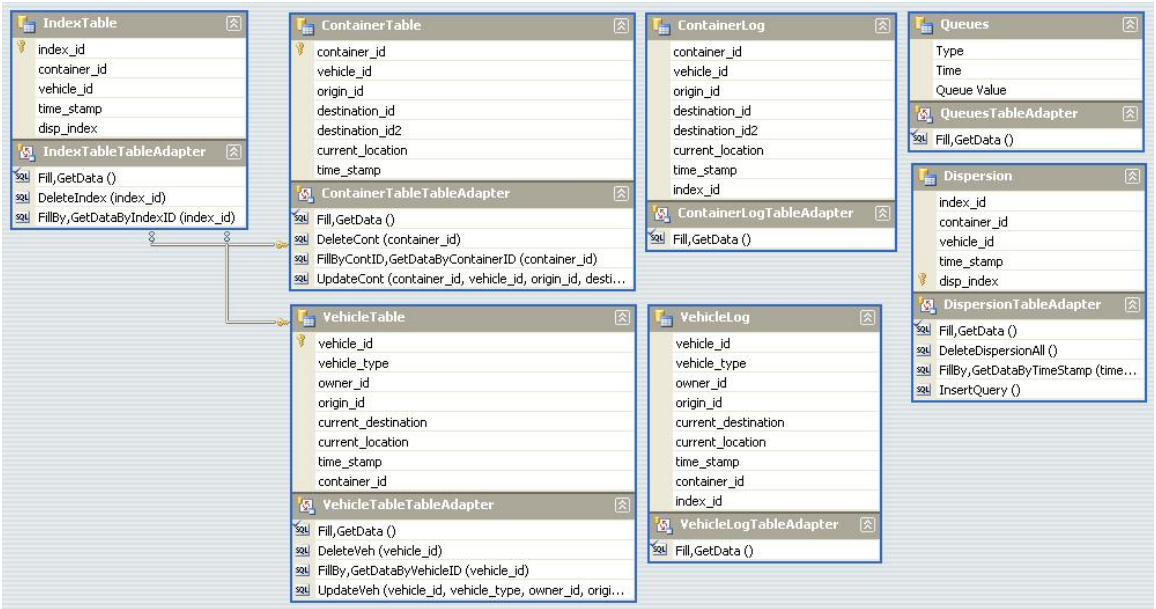



**Figure 54. Federation Database Showing RTI Table Adapter Queries**

a generic 'Insert' Table Adapter query for each unique data table automatically upon import of the database, negating the need for a specialized 'Insert' query.

To provide examples of the SQL statements underlying each Table Adapter, those queries related to the Container Table will be examined. The 'GetDataByContainerID' query allows the RTI to recover all object and time data for a specific container entity. This statement is used at several instances in the RTI command code, however one specific example was shown in the first line of Figure 43 for containers that are exiting the roadway network model federate to enter GCT gate submodel of the port federate. This command line executes the 'GetDataByContainerID' Table Adapter query for the Container Table to recover all data for the container entity whose 'Container ID' attribute value is currently stored in the RTI local variable 'PortContID.' This query's underlying SQL statement for this Table Adapter is:

```
SELECT container_id, vehicle_id, origin_id, destination_id,
destination_id2, current_location, time_stamp FROM
ContainerTable WHERE container_id = ?
```

where for a specified 'container_id' ('WHERE…') the query selects the requested fields ('SELECT…') from the specified data table ('FROM…). The second line in Figure 43 then locates the specific row containing the 'PortContID' number, and the third through fifth lines recover the specific entity attribute values from that row.

An instance of the 'UpdateCont' Table Adapter query is also shown in Figure 43 where a container entity's attribute values, upon arrival at the destination, are updated to reflect the new location (current_location) and the new timestamp (time_stamp). The SQL statement underlying this Table Adapter query is:

```
UPDATE `ContainerTable` SET `container_id` = ?,
`vehicle_id` = ?, `origin_id` = ?, `destination_id` = ?,
`destination_id2` = ?, `current_location` = ?, `time_stamp`
= ? WHERE (`container_id` = ?)
```

where for a specified ‘container_id’ (‘WHERE…’) the query updates the specified fields (‘SET…’) in the specified data table (‘UPDATE…’).

Note that in Figure 43 the ‘UpdateCont’ command includes in parenthesis the local RTI variables/values ‘PortContID’, ‘PortVehID’, ‘PortOriginID’, ‘PortDestID’, ‘PortDestID2’, ‘7’, ‘fTime’ and ‘PortContID’. This is because these local RTI variables contain the entity attribute values that correspond, in the order in which they are listed, to the repsective data table fields identified by the ‘SET’ command in the SQL statement. Note that ‘PortContID’ is listed again at the end of the ‘SET’ command to confirm the original value of the newly updated ‘container_id’ field.

The ‘DeleteCont’ Table Adapter query very simply deletes a specified row from the data table. An instance of this command can also be seen in Figure 43. The ‘DeleteCont’ table adapter executes the simple SQL statement:

```
[DELETE FROM `ContainerTable` WHERE (`container_id` = ?)]
```

where for a specified ‘container_id’ (‘WHERE…’) the query deletes the corresponding row from the specified data table (‘DELETE FROM…’).

The Table Adapter queries for each of the other data tables in the federation database are very similar to those that have just been described for the Container Table. The primary difference is that the data table field names incorporated in each SQL statement are unique to each data table.

### 3.3.4 VISSIM© Roadway Network Model Federate

This section will describe the structure and design considerations of the roadway network model created using VISSIM 5.10© traffic microscopic simulation software. This discussion will cover four modeling elements: (1) general model overview and design considerations, (2) vehicle routing methods and calibration, (3) vehicle detectors and (5) special design considerations for model construction.

#### 3.3.4.1 General Model Overview and Design Considerations

The roadway network model was constructed in VISSIM© using a scaled Google Earth© image of the Port of Savannah as a background and construction base. Building the roadway network over this scaled image allowed for the close approximation and

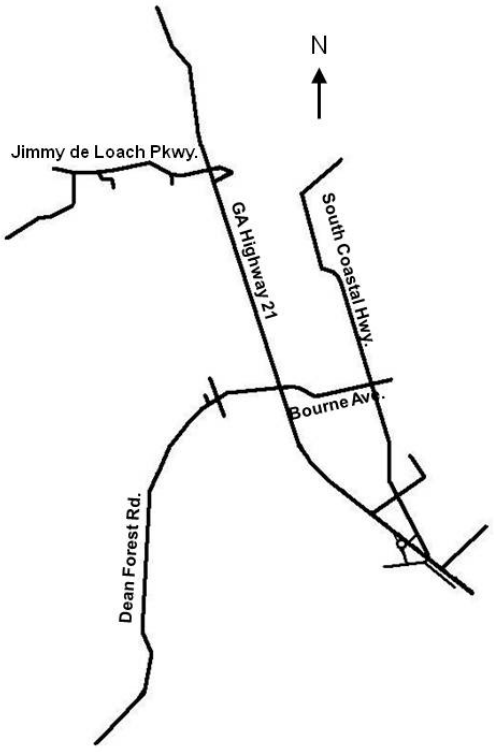



**Figure 55. Port of Savannah Roadway Network Geometry**

replication of the real-world roadway geometry. Figure 55 shows the surrounding roadway network at the Port of Savannah with the major streets labeled. Note, this is similar to that which was shown earlier in Figure 3.

All roadways in the network shown in Figure 55 accurately reflect the intersection layout and geometry (with the exception noted above) of the actual roadway. This includes appropriate exclusive left turn lanes for northbound (NB) and southbound (SB) traffic on GA Highway 21 at the intersection with Bourne Ave/Dean Forest Road. This also includes an exclusive left turn land for SB traffic and an exclusive right turn lane for NB traffic at the intersection of Jimmy de Loach Parkway and GA Highway 21. Also, note that Jimmy de Loach Parkway has an overpass over GA Highway 21 such that eastbound (EB) vehicles approaching GA Highway 21 along Jimmy de Loach Parkway first cross over the highway, curve to the right, and then intersect GA Highway 21 from the east.

All roadways in the network shown in Figure 55 also accurately reflect the number of lanes of the actual roadways. All roadways are two lanes each way with the exception of South Coastal Highway, which is one lane each way, and the southern portion of GA Highway 21 south of Bourne Ave, which transitions into three lanes each way. All driveways connecting the distribution centers with Jimmy de Loach Parkway are also one lane each way.

There are nine signalized intersections in the roadway model, shown in Figure 56. To accommodate the various intersection configurations and lane geometry, nine separate traffic signal control plans were created. These traffic signal control plans are simple pre-timed plans and do not exactly reflect the real-world signal timing plans at these

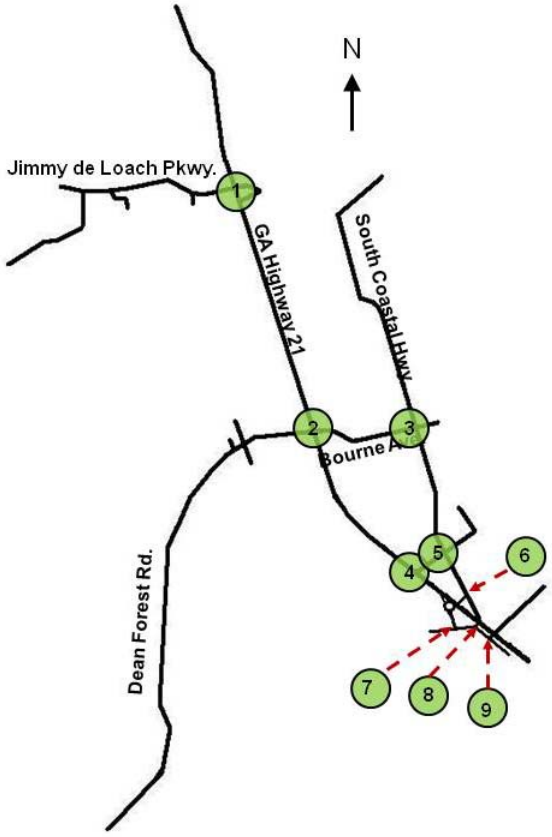



**Figure 56. Roadway Network Model Signalized Intersections**

intersections. Generic traffic signal timing plans were developed that were tailored to the traffic demand modeled. Future roadway network models used for deeper analysis of port system operations should incorporate the actual signal timing plans for these locations. Appendix C shows each traffic signalization plan according to the intersections as numbered in Figure 56.

To simulate non-port roadway traffic, vehicle inputs were located at the beginning links of five major roadways to generate "background" traffic. Note that the background traffic inputs and volumes do not exactly reflect those found in the field. Background

traffic inputs were generated that create a moderate traffic demand on the network. Future models will incorporate field data to more accurately reflect real-world conditions.

The first background vehicle input generates EB traffic at the western-most endpoint of Jimmy de Loach Parkway, the second and third generate NB and SB traffic at the two endpoints of GA Highway 21, fourth generates NB at the southern-most endpoint of Dean Forest Road and the fifth generates SB traffic at the northern-most endpoint of South Coastal Highway.

Throughout the VISSIM© roadway network model federate, vehicle turning movements are determined by user-defined "routing decisions." These decisions can either restrict vehicles from making certain turning movements at intersections, or else determine the percentage of approaching vehicles that execute right turning movements, versus left turning movements, versus through movements.

The routing decisions for the background traffic are set to restrict background traffic to GA Highway 21, Dean Forest Road, South Coastal Highway and Jimmy de Loach Parkway. No background vehicles travel on the distribution center driveways off of Jimmy de Loach Parkway or on the GCT driveway (the extension of Bourne Ave east of South Coastal Highway). At each intersection approach, background traffic routing decisions route 10-20% of approaching background traffic as left-turning movements, 10-20% of approaching background traffic as right-turning movements, and the remainder of approaching background traffic as through movements. Exact proportions depend on the intersection.

3.3.4.2 Port and Roadway Truck Routing Method

A series of truck-specific routing decisions, that route trucks according to their VISSIM© vehicle class, ensure that port and roadway trucks are properly routed between their various origins and destinations. VISSIM© distinguishes vehicles in two different ways, according to 'vehicle type' and 'vehicle class.' VISSIM© defines 'vehicle type' as a "group of vehicles with similar technical characteristics and physical driving behavior." [9] Vehicle types include cars, light trucks, heavy trucks and buses. On the other hand, a "vehicle class represents a logical container for one or more previously defined vehicle types." [9] One way to think about a vehicle class is as a fleet of delivery vehicles, wherein the fleet could consist of cars, light trucks and heavy trucks. The distinction between vehicle types and classes is important because VISSIM© can assign vehicle classes to particular routes, but not vehicle types. However, the COM interface commands only allow the RTI to specify vehicle types when creating vehicles in the network, not vehicle classes.

For the federation, six vehicle classes were created according to destinations: Class 1 – to Distribution Center 1, Class 2 – to Distribution Center 2, Class 3 – to Distribution Center 3, Class 6 – to Long-Distance Trucking/I-16 Junction, Class 7 – to Port/GCT, and Class 10 - Background. Note that classes 4, 6, 8 and 9 are intentionally omitted as they pertain to other port system destinations that are not part of this study, but may be used in future studies.

Within each vehicle class there is only one associated vehicle type. Classes 1, 2, 3, 4, 6, and 7 are assigned the 'HGV' (heavy-goods vehicle) vehicle type. Class 10 – Background is assigned the 'car' vehicle type. However, because of the COM interfaces

inability to allow the RTI to assign vehicle classes, each vehicle type instance in a vehicle class must be unique. Therefore, seven vehicle types were created – six of which are 'HGV' vehicles and one of which is a 'car' vehicle. Each of the seven unique vehicle types is associated with only one of the vehicle classes. For example, Vehicle Type 1 – HGV is assigned only to Vehicle Class 1 – to Distribution Center 2. Similarly, Vehicle Type 7 – HVG is assigned only to Vehicle Class 7 – to GCT/Port, and so on.

Next, a series of routing decisions were then created for each of the major intersections across which port and road truck traverse. Multiple decisions were made per intersection approach to correctly route the port and road trucks through each intersection. For example, Figure 57 shows the routing decision window, with the row highlighted for 'Decision No. 49.' Routing decision 'Decision No. 49' starts on link 51 (as indicated by the green arrow), which is the westbound approach to intersection 2 on Bourne Ave. (see Figure 56), and ends on link 78 (as indicated by the red arrow), which

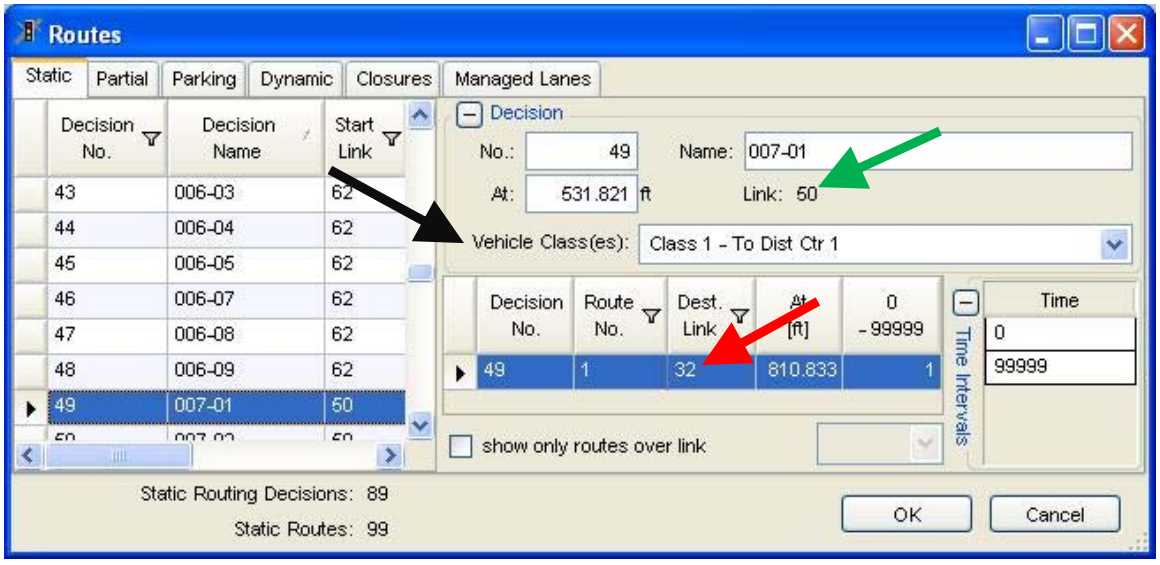



**Figure 57. VISSIM© Roadway Network Sample Routing Decision Window**

is the westbound link leading away from intersection 2 on Dean Forest Rd. Note that routing 'Decision No. 49' only applies to vehicle class 'Class 6 – To I-16 Junction,' as indicated by the black arrow. This means that routing 'Decision No. 49' will only direct 'Class 6 – To I-16 Junction' vehicles to continue straight through intersection 2 towards the I-16 Junction.

Recall from Figure 41 that when the RTI creates a vehicle in VISSIM© (for this example, a vehicle from the GCT) it executes the following command:

```
vehicle = vissim.Net.Vehicles.AddVehicleAtLinkCoordinate
(array5014, 50, 50, 1, 0)
```

The values embedded in the parentheses in this command refer to, in order, the vehicle type (array5014), the desired speed in mph (50), the desired link number on which to create the vehicle (50), the desired lane number on that link on which to create the vehicle (1), and the x-coordinate on that link (0, denoting the beginning of the link). Also, recall from Figure 41 that the local RTI variable 'array5014' refers to the 'Destination ID' attribute value for the current container/vehicle exiting the GCT. For this example, the container is bound for the I-16 Junction (Destination ID = 6), therefore VISSIM© creates a vehicle at the given location that is VISSIM© vehicle type 6. As vehicle type 6 only belongs to vehicle class 6, the newly created vehicle will only follow routing decisions associated with vehicle class 6. In this case, the newly created vehicle will follow only the routing decision straight through intersection 2 westbound, toward the I-16 Junction. The truck encountered a similar routing decision upstream at intersection 3 that routed the vehicle through that intersection, westbound toward intersection 2.

There are multiple routing decisions that begin on the same link (link 51) as 'Decision No. 49,' shown in Figure 57. However, these other routing decisions end at the other various locations and are applicable only to the certain vehicle classes associated with each of those destinations. For example, the routing decision from the GCT to Distribution Center 1 (Destination ID = 1) is only applicable to vehicle class 1, which only contains vehicle type 1. Therefore, if an exiting container/vehicle has a Destination ID equal to 1, the RTI will create in VISSIM© a vehicle of vehicle type 1. As vehicle type 1 is only associated with vehicle class 1, the vehicle will only follow routing decisions that direct it to Distribution Center 1.

3.3.4.3 Roadway Detectors at Destination Links

As noted previously, roadway detectors are incorporated into the VISSIM© roadway model to detect the presence of a vehicle at the end of one of the destination links ready to exit the roadway network model federate. In VISSIM©, detectors must be assigned to a signal controller. Previously, nine signal controllers were created – one for each of the nine signal control plans discussed above and outlined in Appendix C. As assigning the exiting link detectors to one of these three signal controllers would activate erroneous signal control calls, a tenth "dummy" signal controller, with no actual signal phasing, was created expressly for the detectors.

Each detector was placed approximately 100 feet before the end of an exiting destination link and all detectors were set to be 20 feet in length. The rationale for these decisions will be discussed in the next section. Figure 58 shows the detector for the existing link at Distribution Center 1.

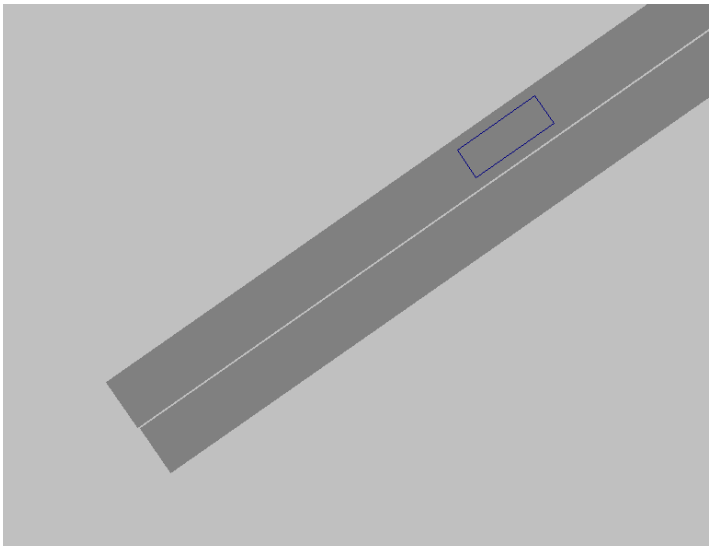


**Figure 58. VISSIM© Detector Location for Distribution Center 1**

3.3.4.4 Roadway Network Model – Special Considerations

During model construction and testing, special considerations had to be made specifically to address two known issues in the VISSIM© roadway network model federate. The first concerns "vehicle diffusion," or the deletion of vehicles from the roadway network during runtime, which has been mentioned previously. The second concerns ensuring that the roadway model did not fail to detect trucks upon their arrival at a destination. Each of these issues will be discussed in the following sections.

*3.3.4.4.1 Diffusion of Vehicles From the Roadway Network*

As has been stated, the primary reason VISSIM© diffuses in-transit vehicles from the roadway model is to avoid deadlock. Deadlock can occur for multiple reasons. However, the primary type of diffusion that was encountered during this study involved lane-change delay. When a vehicle is forced to stop at the "emergency stop position" for a lane-changing maneuver, there is a prescribed amount of time that vehicle will wait for

a sufficient gap to occur before VISSIM© deletes, or "diffuses," the vehicle from the network [9]. The "emergency stop position" is the location on the link at which the connecter for the corresponding movement begins, requiring the lane changing maneuver. In other words, it is the last possible point of opportunity that a vehicle has to execute a lane-changing maneuver.

The initial cause of diffusion of vehicles in this roadway network model was largely a product of insufficient vehicle look-ahead for routing decisions at turning movements. In other words, the starting point of routing decisions were not placed sufficiently far upstream of intersections, causing vehicles to wait until just before a turning movement to change lanes for that movement.

Two special design considerations were incorporated to combat this problem of vehicle deletion. The first special consideration was to move the truck-specific routing decision starting points farther upstream of intersections. In some cases, these starting points were moved up to a mile upstream of the intersection to allow trucks sufficient space in which to execute a lane-changing maneuver, as necessary.

The second special consideration was to increase the maximum allowable time for lane-change delay before a vehicle is deleted, or diffused. The default value in VISSIM© is 60 seconds. For this model, this time was increased to 120 seconds. This effectively doubles the time interval vehicles have in which to find an adequate gap in traffic and execute a lane-changing maneuver.

Even with these two special design considerations it was not possible to completely eliminate deadlocks by making adjustments to the VISSIM© roadway network model. Instead, these design considerations reduced the number of port/road

trucks deadlocks, and therefore diffusion, to a rate of approximately two trucks per hour. However, the compounded effect of this diffusion on a reusable resource, such as port trucks, over several days of simulation can be significant. For example, using this average diffusion rate of two trucks per hour, 144 of the 150 port trucks could potentially be diffused in three days of simulation. Therefore, it was necessary to implement the RTI/Arena© based diffusion module (section 3.3.2.2) as a secondary solution.

*3.3.4.4.2 Vehicles Not Detected by Destination Link Detectors*

Early trials of the federation indicated that numerous vehicles were not being detected by the destination link detectors. It was discovered that there were two reasons for this.

First, on exiting links with multiple lanes, all but the right-most lane was closed to truck traffic. This should have caused all trucks to travel in the right lane. Accordingly, single detectors were only located in the right-most lane of each link. However, consistent with the discussion above concerning inadequate lane-changing behavior, numerous vehicles did not change to the right-most lane prior to exiting the network and were missed by the detectors. The first solution considered was to widen detectors to cover the second lane. However, this could have caused a second vehicle to be missed if two side-by-side vehicles simultaneously cross the widened detector during the same time step.

The second reason that detectors were missing vehicles is related to the detector length. If the detector is too short, a vehicle can pass completely over the detector in a one second time step, missing detection. However, if the detector is too long, two

vehicles in the same lane traveling at close spacing can be simultaneously located over the detector. This would cause the RTI to recover only one 'Vehicle ID' for the detector, thereby missing the second vehicle.

To address these issues, several design elements were considered, all but one of which were incorporated into the roadway model. First, because some trucks occasionally exit the roadway model from the left lane of the destination links, despite lane closures, the first solution was to either merge or divert trucks into single exiting lanes. As the GCT and the I-16 Junction are the only destinations with multilane exiting links, this was only necessary at those locations. Figure 59 shows the condition at the GCT destination link where two lanes of truck traffic are merged into one single entering lane. This enables the use of only one detector in the single exiting lane, as shown.

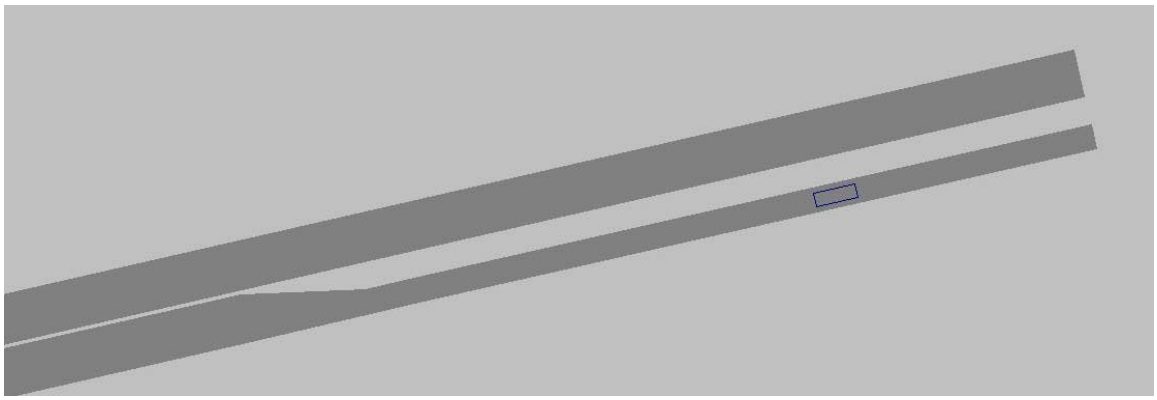


**Figure 59. VISSIM© Roadway Model GCT Exiting Link**

Merging two lanes into one single lane was an option at the GCT because no background traffic is allowed on this link. However, as background traffic is mixed with truck traffic on Dean Forest Road at the I-16 Junction, a different solution was necessary.

Figure 60 shows the condition at the I-16 Junction where truck traffic is diverted to a side road link to exit at the I-16 Junction. Diverting truck traffic to the side road does not interfere or impede the flow of background traffic as would have occurred if the two exiting lanes were merged into one single exiting lane. Also, locating a routing decision several hundred feet upstream, to route trucks onto the side road, forces the occasional left lane truck to merge into the right lane to exit to the side road. These methods of merging and diverting exiting truck traffic lanes solved the problem associated with failing to detect occasional vehicles traveling in the left lanes of exiting links. Also,

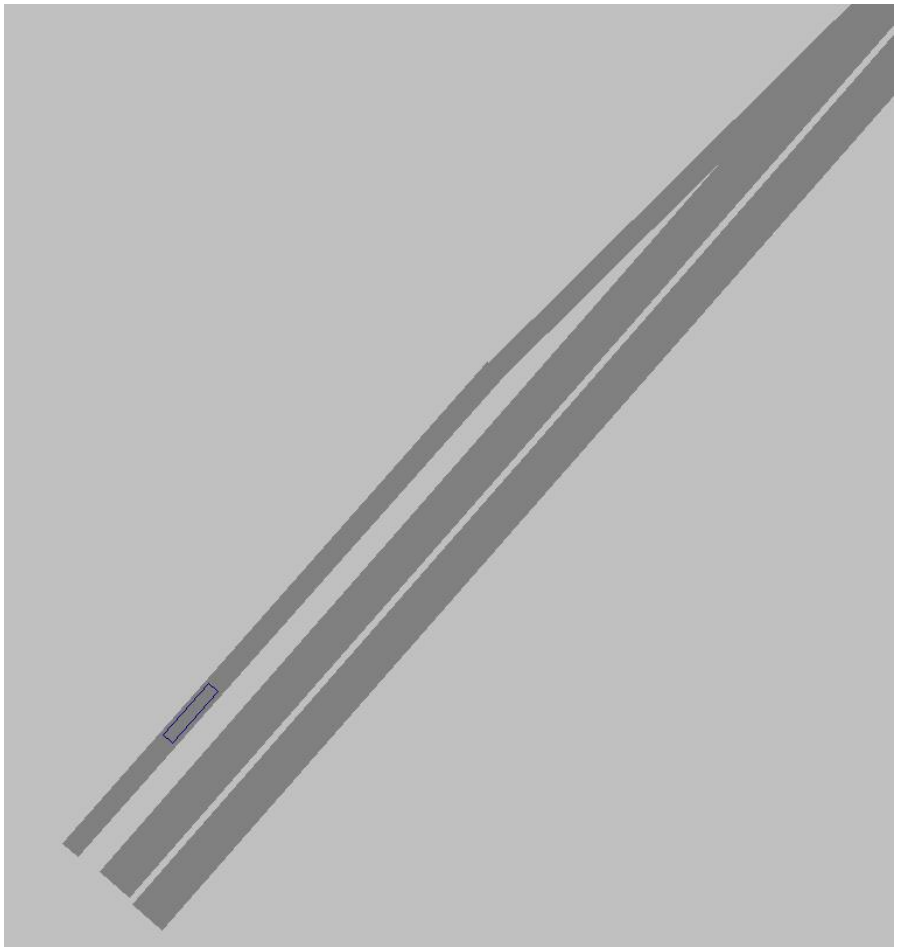


**Figure 60. VISSIM© Roadway Model I-16 Junction Exiting Link**

initial tests indicated that these occasional forced lane-changes associated with the I-16Junction exit do not increase background traffic delay or congestion along that roadway.

Diverting and merging exiting lanes, however, did not solve the problem associated with inadequately short detectors failing to detect crossing vehicles and overly long detectors detecting failing to detect closely spaced vehicles. Generally, this problem is due to having no control over the speed and frequency with which vehicles crossed detectors. Therefore, three solutions were considered to combat this problem. The first was to install a second detector downstream of the first detector. If a truck passed completely over the first detector in a single time step due to inadequate length, the truck would be detected by the second detector.

The second solution would be to utilize a VISSIM© COM command that allows the RTI to extract a list of all vehicles currently located on that exiting link. This list of vehicles from VISSIM© is then be compared with the federation database list of port and roadway trucks currently in the roadway network model federate to determine exiting vehicles.

The third design element considered was to place a stop sign immediately before each detector. This would cause vehicles to cross the detectors at low speeds and reduced frequencies.

The problem identified with adding a second detector in series is two-fold. First, adding a separate second detector would require the duplication of the RTI commands shown in Figures 42 and 43 for the second detector. This would significantly reduce the speed and efficiency of the RTI. Second, if the first detector fails to detect a truck, it is

assumed that the second detector will detect the truck. However, if the first detector fails to detect the vehicle, there is no guarantee that the second detector will not also fail to do so.

The problem identified with using the COM command to populate and compare a list of vehicles on exiting links with the federation database is computing limitations. Given that the lists would have to be populated at the end of every time step, the computing overhead required for this solution would significantly slow down the federated simulation execution. For these reasons, the decision was made to place a stop sign immediately before a single detector at each exit.

There is some concern that installing a stop sign would detract from the reality of the simulation model. This concern is largely unwarranted for vehicles exiting at the distribution centers and the GCT where, in reality, trucks would be preparing to stop to enter the facilities and for unloading. Nonetheless, this concern is reasonable for the I-16 Junction. It is hoped that the incorporation of a separate exiting link exclusively for trucks mitigates some of concern that stopping trucks would disrupt background traffic flow, especially if the exiting link has sufficient queuing space to prevent spillback from impeding background traffic flow on the mainline. Note that all travel time data associated with port and road way trucks is collected along segments that end upstream of these stop controlled exits. Therefore, any delays associated with these stop controlled exits are not reflected in travel time data. Future versions of the model will seek a better method to detect trucks exiting the system.

## 3.4 Known Model Limitations

Several known limitations exist in various federation and federate model components that should be addressed in future port system simulation and modeling efforts. This section will outline these limitations by federate component.

### 3.4.1 Arena© Port Model Federate Limitations

Throughout federation development, the assumption was made that the federation time-step interval would uniformly be 1 second in length. As a result, this 1 second time-step interval has been "hard-coded" into several elements of the federation and model federates. One example of this hard-coding is found in the 'Vehicle Input v2' and 'Container Input v2' **Input** blocks discussed in Section 3.3.1.1.4. Recall that these **Input** blocks were developed specifically for this model using the template development capability of Arena©. In these blocks, entities are created at 1 second intervals in the "switch variable series" of logic (Figure 24) which controls the blockage, release, and therefore creation, of vehicle/container entities from the **Input** block.

As was stated earlier, if the 1 second interval in that template block were set to a smaller time value, it could allow erroneous duplicate entities to be created by the **Input** block. Similarly, if the interval were set to a larger time value, entering vehicles could be missed or not created by the **Input** block. This problem is a direct byproduct of the fundamental assumption that all time-stepping in the federation would occur in 1 second intervals.

The implication of this hard-coded 1 second time-step interval is that there is no flexibility to adjust the resolution of the federation. That is, there is no flexibility to set

the model to run at lower time resolutions (e.g., longer time-steps: 2 seconds, 5 seconds, etc.) or to run at higher time resolutions (e.g., shorter time-steps: 0.5 seconds, 0.1 seconds, etc.). Depending on the analysis and evaluation needs of future simulation and modeling efforts, this inflexibility in time resolution could prove to be a limitation.

There is a second limitation in the Arena© port model federate, which has to do with truck object rerouting. Currently, the rerouting of empty vehicles follows a "GCT-centric" logic. That is, if empty port truck objects are not needed (i.e., are in excess) in the current distribution center submodel, they are automatically rerouted to the GCT Gate submodel. This somewhat limits the ability of the port model federate to balance the distribution and utilization of port truck objects throughout the system as there is no capability for excess port truck objects at one distribution center to be rerouted to another distribution center where they may be needed.

Similarly, if road truck objects are in excess at one of the distribution center submodels, the road truck objects are either rerouted to the GCT Gate submodel, or if unneeded at the GCT Gate submodel, they are rerouted to the I-16 Junction submodel. There is currently no option to reroute excess road truck objects from the GCT Gate submodel to one of the distribution center submodels. Again, this creates a limitation in the port model federate's ability to adaptively balance the distribution and utilization of road truck objects throughout the system by considering all submodel needs, and not primarily those of the GCT Gate submodel.

### 3.4.2 VISSIM© Roadway Network Model Federate Limitations

As the purpose of this study is the development and validation a federation methodology, several assumptions and special design considerations were incorporated to simplify the construction and operation of the roadway network model federate. These assumptions may prove to be a limitation for future simulation modeling efforts intended for a deeper analysis of the port system operations.

As stated in Section 3.3.4.1, both the background traffic input volumes and signal timing plans are assumed. Future analysis of port systems operations should incorporate field-based values for both traffic input volumes and signal timing plans, which likely utilize actuated or adaptive intersection signalization strategies.

The second roadway network model limitation concerns the destination links at federation interaction points. First, the use of stop controlled single-lane exits to reduce the frequency and increase the uniformity of truck objects passing over the detectors. Because of the practical challenges associated with using multiple or duplicate detectors – both in the VISSIM© model and the RTI – this technique was implemented. However, the obvious inconsistency with field-conditions of using stop controls at these exiting links, especially the I-16 Junction, will limit the model's reflection of the real system's operations. Further calibration of detectors or innovative RTI command coding in future modeling efforts may prove capable of negating the need for single-lane, stop controlled exiting links.

The third limitation is that the current VISSIM© COM interface library does not allow outside programs, such as the RTI, to access vehicle diffusion information during runtime. Therefore, it is not possible to determine, during runtime, if a vehicle was

diffused during the previous time-step. For this reason, the 5000 second "look back" method was created because the federation database *could* be accessed during runtime, and provided time-stamped records capable of identifying vehicle diffusion occurrences.

One issue with this is that the hard-coded 5000 second look-back delay may have some minor impact in the clarity of the truck utilization and location values that are collected during runtime. That is, it may cause these values to indicate that a greater number of trucks are present in the roadway network than is accurate. It is not until 5000 seconds after a truck is diffused from the roadway network model that it is detected by the RTI, the truck is recreated in the Arena© model, and these truck utilization and location values are corrected. Future versions of the model will seek better solutions to account for vehicle diffusion without such significant delays in detection.

Another limitation of this method for determining vehicle diffusion is that if the roadway network is highly congested, the RTI may determine that a vehicle was diffused when it in fact that vehicle was just significantly delayed by the congestion. the RTI determines that a vehicle was diffused when in fact that vehicle was merely just significantly delayed in the network, the RTI generates an error message and the federated simulation stops.

### 3.4.3 Runtime Infrastructure (RTI) Limitations

The primary limitation of the RTI is the command code's current inability to record when two or more containers are loaded onto a truck. As has been mentioned above, the Arena© port model federate submodels could easily be altered to enable multiple containers entities to be batched with individual truck entities. However, the

current RTI code is not able to record the association of multiple containers with a single truck. To enable this capability would require significant changes in the RTI structure and minor changes to the federation database structure.

## 3.5 Chapter Summary

This chapter presented the methodology of the federated simulation effort undertaken in this study. Section 3.1 provided with an operational overview of the port and roadway network systems being modeled. Section 3.2 provided an overview of the federation components was then presented discussing the general operation of the Arena© port model federate, the VISSIM© roadway network model federate, the runtime infrastructure (RTI), and the federation database. Section 3.3 described each federation component in detail. Section 3.4 provided a discussion of the known limitations in each federation component.

The next chapter introduces the experiment that is conducted on the federated simulation model. This includes a background discussion of the type of experiment conducted, the types of data that are collected, and the statistically based methods by which the output data is analyzed

# CHAPTER 4

# DESIGN OF EXPERIMENT

The purpose of this experiment is to test for the presence of feedback loops between the two model federates and understand the characteristics of how the model readjusts to steady-state operation following adjustments to federation input parameters during run-time. To show this, a simple time-lag experiment will be conducted in which two federate parameters are changed in sequence, or out of phase, from one another during run-time. The effects of these parameter changes, as well as the propagation time of these changes, will be analyzed in the federation output data.

## 4.1 Time-Lag Experiment Background

Much of the theory explaining traffic flow characteristics has a basis in fluid dynamics. Indeed, the foundational research of Lighthall and Whitman (1955) used some concepts from fluid dynamics (most notably the concept of mass conservation, wherein a traffic input volume should be equal to the output volume plus some roadway storage along a section of roadway) and wave theory to develop the kinematic wave model of traffic flow [29]. Essentially, this theory suggests that as events occur in a traffic stream, the effects of those events propagate upstream through the traffic as a time dependent "shock wave." For example, consider a vehicle on a crowded arterial that brakes suddenly. Not all drivers upstream will brake at the same time as that first driver, but will instead brake sequentially in response to the vehicle in front of them. This sudden reduction in speed of the traffic stream propagates as a kinematic "shock wave" upstream

from the initial vehicle. Therefore, there will be some time-delay between when that initial driver brakes and the time at which another driver several hundred feet upstream in traffic is forced to brake.

This treatment of traffic as a fluid also occurs in the realm of computer simulation. Ni (2006) has discussed how macroscopic computer models of traffic networks generally treat traffic flow as a compressible fluid [1]. This concept is further reflected in studies conducted by Daganzo and Geroliminis (2007, 2007 & 2008) that relate traffic flow and average density on urban roadway networks through a macroscopic fundamental diagram (MFD) by monitoring the input and outflow of traffic around the perimeter of a network [30-33]. This is analogous to measuring the flow rates into and out of a water tank or fluid pipe network [33]. These studies present the idea that given this general knowledge, conclusions can be made about the density of traffic (and therefore congestion level) within the urban roadway network itself [30-32].

The concept of a time-lag experiment in this study builds on this historical foundation of using fluid dynamics to better understand both kinematic wave propagation through traffic streams and macroscopic traffic networks.

The basic concept of the time-lag experiment has been borrowed from the field of hydrology. As has been explained, there is some precedent for using hydraulics and hydrology to explain transportation engineering and traffic flow phenomena. Therefore, the underlying concept is analogous to the measurement of changes in streamflow out of a drainage basin during and after a rainfall event [34]. As rain falls on a basin, the rainwater flows downhill to some stream that drains the basin. However, because of the time associated with the rainfall collecting in the basin and flowing downhill toward the

drainage stream, the increase in flow of the drainage stream is not immediate. Instead, there is a time-lag; that is, the flow-volume of the basin drainage stream increases over some time interval during and after the rainfall event. The flow rate of the basin drainage stream eventually peaks and then subsides as no additional water is input once the rainfall event has stopped. The difference in time between the beginning of the rainfall event and the peak of the drainage streamflow is called the *time of rise,* and the time from the midpoint, or center of mass, of the rainfall event to the peak of the drainage streamflow is called the *lag time* [34]. Figure 61 shows a generalized hydrograph illustrating the relationship of these times between the rainfall event and the peak streamflow.

In this example of time-lag (lag time) from hydrology, the change in flow, or output, of the system lags behind the increase in the input of that system. This time

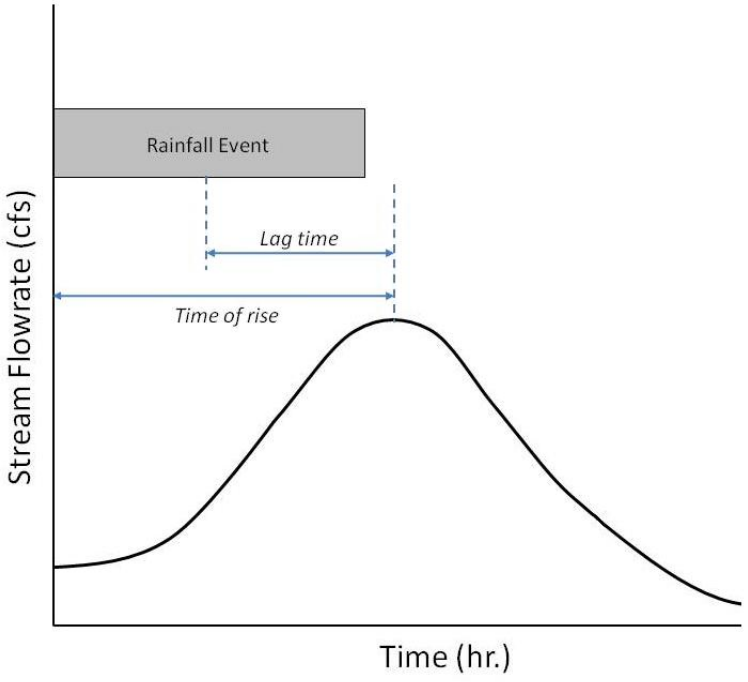



**Figure 61. Hydrograph Characteristics and Time Relationships (Figure Credit: Hydrology and Floodplain Analysis [34])**

difference between the change in an input parameter in a system and the observable effects of that change as it propagates throughout the system forms the basis of the experiment conducted in this study. Specifically, two input parameters of the Port of Savannah federated simulation will be sequentially changed and the amount of time required for the effects of those changes to propagate throughout the system will be determined.

## 4.2 Experimental Design

The previous section provides the background and foundation for conducting a time-lag experiment on the Port of Savannah federated simulation. This section provides a discussion of the specific design elements that this type of experiment entails. This begins with a discussion of the overarching structure of the experiment. This is followed by a discussion of the input parameters (independent variables) that will be changed during the experiment. Next is a discussion of the types of system performance and output data (dependent variables) that are collected throughout the simulation. Lastly, a discussion of the statistical methods used to analyze this output data is presented.

### 4.2.1 Experimental Overview

One of the fundamental objectives of this study is to investigate the effects of port operations on the surrounding roadway network, and conversely to investigate the effects of roadway network performance on port operations. Therefore, input parameters of both systems must be adjusted during simulation runtime. Within the port model federate, the input parameter that is adjusted is the volume of containers arriving to the port from

container ships. This is the first independent variable – port container volume. Within the roadway network model federate, the input parameter that is adjusted is the volume of background traffic circulating in the network. This is the second independent variable – network background traffic volume.

To avoid the risk of confounding data associated with changing two independent variables during one replication of the federated simulation, the two independent variables are adjusted separately, out of phase from one another. Figure 62 shows how

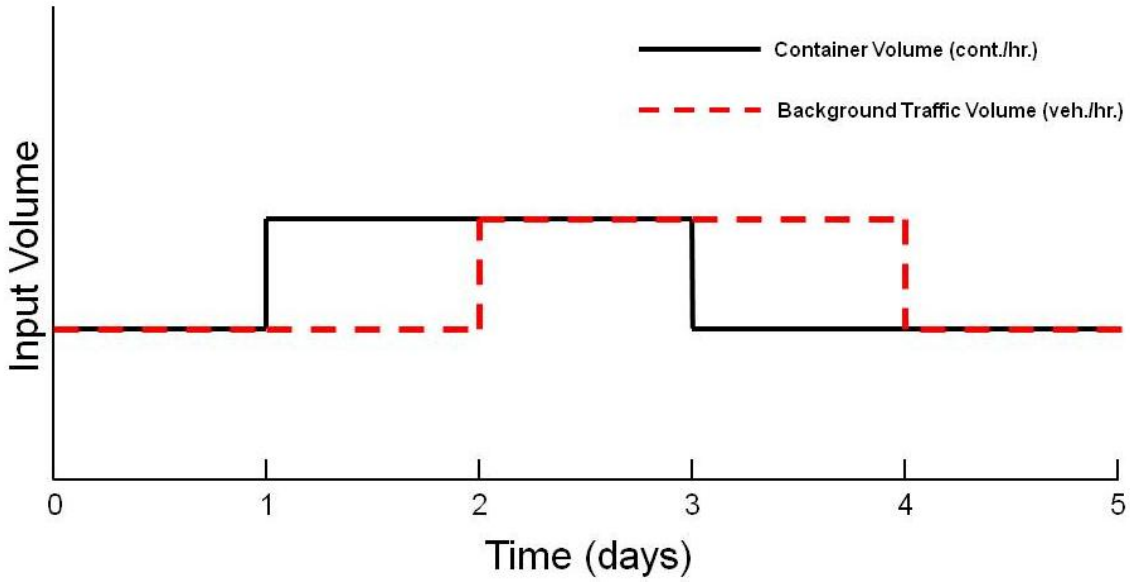



**Figure 62. Sequencing of Independent Variable Changes**

this sequencing occurs. For the first day of simulation time, both the container volume and the background traffic volume are held constant, providing time for the federated simulation to reach steady state. At the end of day one, the container volume is increased, while the background traffic volume is held constant, and the simulation continues in this state for a second day of simulation time. At the end of day two, the

background traffic volume is increased. At this point, both port container volume and background traffic volume variables are in their increased states, and the simulation continues for a third day of simulation time. At the end of day three, the container volume is reduced to its original starting volume, while the background traffic volume remains in its increased state. The simulation continues for a fourth day of simulation. At the end of day four, the background traffic volume is reduced to its original starting volume. At this point, both port container and background traffic input volumes are back at their original levels. The simulation then continues for a fifth and final day of simulation time.

Stepping up both independent variables one day out of phase from one another, and then stepping them back down to their original states, allows several phenomena to be observed and assumptions to be tested. First, the assumption is made that the federated simulation achieves steady state operation at some point during each successive day of operation. Using various output data, steady state operation can be calculated (the statistical methods for this calculation are discussed in section 4.2.4). The second assumption, building on the first, is that there is some time-constant associated with the time-lag between when an input parameter is changed and when the simulation readjusts to a new steady state condition. In other words, while an input parameter can be adjusted nearly instantaneously (as shown by the sharp stepping of the lines in Figure 62), various dependent variables (e.g., travel times, number of trucks in the network, port queue lengths) will adjust to these new conditions over some time interval, or time-lag. Furthermore, by both increasing and decreasing port container volume and background

traffic volume, any differences in time-lag intervals associated with the direction of change can be observed.

### 4.2.2 Independent Input Variables

As has been described throughout Chapter 3 – Methodology, there are numerous independent variables that can be adjusted throughout simulation to affect the performance of both the port model federate and the roadway network model federate. The purpose of this section is to discuss the two input parameters (independent variables) that will be adjusted and how they will be adjusted. Also, this section discusses which other independent variables are held constant throughout the federated simulation.

#### 4.2.2.1 Transient Independent Variables

As mentioned in section 4.2.1, two input parameters are adjusted during simulation runtime: port container input volume, and roadway network background traffic input volume.

Container input volume is adjusted only in the GCT aggregated submodel. Recall from section 3.3.1.1.1 and Figure 7 that containers arrive to the GCT Gate submodel at the **Station** block 'Receiving from Truck Transfer.' Upstream from this **Station** block, containers are generated within the aggregated GCT submodels by a separate submodel (developed by Peesapati and Gbologah [8]) that represents container ships arriving to the port. In the base condition, day one, ships arrive at a constant rate, one ship every two hours. The number of containers on each ship is determined using a random uniform distribution with minimum and maximum values of 225 and 275 containers per ship,

respectively. Therefore, the average container volume in the base condition is 250 containers per ship arrival, or 125 containers per hour. Container volume is then increased at the end of day one by shortening the interarrival time of ships to the port. Instead of ships arriving every two hours, the higher container volume scenario has ships arriving every hour and a half. The number of containers on each ship is determined using the same random uniform distribution with minimum and maximum values of 225 and 275, respectively, as before. Therefore, the average container volume in the high container volume scenario is 166.67 containers per hour. At the end of day three, the ship interarrival time is reset to two hours.

The base background traffic input volume is 250 vehicles per hour on all links entering the roadway network (excluding the distribution center and GCT driveways) for days one and two. At the end of day two, the background traffic input volume is increased to 500 vehicles per hour on all links entering the roadway network with the exception of three: the entering links of northbound GA Highway 21, southbound GA Highway 21 and northbound Dean Forest Rd. For these three inputs, the background traffic input volume is increased to 850 vehicles per hour. This increase occurs at the end of day two. These increased volumes (500 and 850 veh/hr, depending on the input link) constitute the high volume background traffic scenario. At the end of day four, the background traffic input volume is reset to 250 vehicles per hour for all entering links.

#### 4.2.2.2 Constant Independent Variables

During earlier discussions in Chapter 3 – Methodology, numerous other "user-defined" independent variables were identified. All of these other independent variables

are held constant for the time-lag experiment conducted in this study. This section provides a brief description of these variables and their constant values.

Recall from section 3.3.1.3 that the I-16 Junction/Highway submodel generates both empty road trucks and road trucks carrying containers during runtime to simulate the arrival of trucks and containers from the interstate highway system. It is important that the average rate at which trucks are generated in the I-16 Junction submodel (i.e., the combined rate of both empty and container-laden trucks) be at least equivalent to the average rate at which containers are generated at the GCT. This is due to the fact that all containers generated at the GCT are assigned the I-16 Junction as their final destination, independent of whether or not they are assigned an intermediate destination (i.e., one of the distribution centers). Therefore, the number of road trucks entering the federation must balance (i.e., be equal to or greater than) the number of containers entering the federation at the GCT. It follows that the base port container volume described in the previous section would require a lesser number of road trucks to be generated at the I-16 Junction submodel than would be required for the higher port container volume scenario. Therefore, a road truck generation rate of 200 vehicles per hour was chosen as it provides a sufficient number of road trucks for both high and low container volume scenarios, with some excess capacity.

Each container arriving at the GCT Gate model is previously assigned a pair of intermediate and final destinations by the aggregated GCT submodels. Ten percent of the containers that arrive at the GCT Gate submodel are not assigned an intermediate destination, but are assigned only a final destination at the I-16 Junction. The remaining 90 percent of arriving containers are assigned an intermediate destination at one of the

distribution centers (divided equally, 30 percent to each distribution center) and a final destination at the I-16 Junction. Containers and empty road trucks that are generated at the I-16 Junction are assigned destinations according to these same distributions to balance the number and distribution of containers entering at the GCT.

Recall that all three distribution center submodels and the GCT Gate submodel contain **Delay** blocks to simulate the unloading time associated with arriving trucks and containers. For both arriving container and truck logic processes in all submodels, the unloading delay time is set to a constant 15 minute interval. These **Delay** blocks also delay empty trucks upon arrival, not just those carrying containers.

Recall also that each distribution center reroutes both empty port and road trucks according to various maximum queue lengths within the port model federate. All distribution centers are set to reroute empty port trucks to the GCT if the queue of available port trucks at that distribution center is greater than or equal to three. Similarly, empty road trucks are rerouted if the queue of available road trucks at that distribution center is greater than or equal to three. These empty road trucks are then rerouted to the I-16 Junction if the queue of road trucks at the GCT is greater than or equal to five. Otherwise, they were rerouted to the GCT.

The GCT Gate submodel also reroutes empty road trucks according to queues of available road trucks at the GCT. In this case, if the queue of available road trucks in the GCT Gate submodel is greater than or equal to five, excess empty road trucks are rerouted to the I-16 Junction.

Lastly, the look-back time associated with evaluating vehicle diffusion in the RTI is set to a constant 5000 seconds. Therefore, the RTI does not begin evaluating vehicle

diffusion until the simulation time has advanced to a value greater than 5000 seconds. This time was determined by conducting several investigative replications of the federation and was found to provide sufficient look-back time for both the low volume and high volume traffic scenarios without generating a system error.

### 4.2.3 Dependent Variables and Data Output

Throughout simulation, data is collected relating to four performance measures (dependent variables) within the federation: facility queue lengths, roadway travel times truck utilization and location, and facility processing rates. The purpose of this section is to briefly introduce these four types of data and how they are collected.

#### 4.2.3.1 Queue Length Data

The queue length data collected during simulation runtime represents the number of container and truck objects waiting in each of the port submodels for batching and release to the roadway network model federate. Four types of queue length data are collected separately for the GCT Gate submodel, and each of the three distribution centers. No queue length data are collected for the I-16 Junction submodel as no queues occur in that submodel.

Each of the four submodels mentioned above contain four queues: (1) port trucks waiting to be batched with outgoing containers, (2) containers waiting to be batched with outgoing port trucks, (3) road trucks waiting to be batched with outgoing containers, and (4) containers waiting to be batched with outgoing road trucks. Figure 7 shows these

queues for the GCT Gate submodel, and Figures 28 and 29 show these queues for the distribution center submodels.

Queue length data is collected by the RTI directly from the port model federate **Queue** blocks (via the built-in COM interface) and written to the Queues table in the federation database. Queue length data is collected every 60 seconds and provides a snapshot of the various queue lengths at 60 second intervals.

#### 4.2.3.2 Roadway Travel Time Data

Travel time data is collected for both background traffic and port and road truck traffic over numerous road segments within the network. This data is collected by VISSIM© during simulation runtime, which records this information directly to text files. Only compiled travel time data is collected, meaning that travel times are averaged for the number of vehicles that completely traverse a travel time segment during a set time interval. For this study, travel time data is compiled for 30 minute time intervals.

As mentioned, travel time data is collected for both background traffic and truck traffic. Port and road truck travel times are collected for "door-to-door" origin-destination travel time segments. For example, travel times are recorded for trucks traveling from the GCT to Distribution Center 1, separately from trucks traveling from the GCT to Distribution Center 2.

As this study's focus is on the interaction and impacts of port operations on roadway performance, and vice-versa, travel time segments for background traffic are only placed on roads on which Port of Savannah truck traffic and background traffic interact. Therefore, six travel time segments were selected for background traffic. The first two

segments consist of the entire length of GA Highway 21 in both the northbound and southbound directions. The second two segments consist of GA Highway 21 (both northbound and southbound) between the intersection of Bourne Ave/Dean Forest Rd. and Jimmy de Loach Pkwy. These segments are shown in red in Figure 63. The final two segments cover of Dean Forest Rd., northbound and southbound, between the I-16 Junction and the intersection with GA Highway 21. These segments are shown in green in Figure 63. Note that the partial travel time segments shown in Figure 63 do not include the intersections themselves, but instead start on the far side of each intersection and end just prior to each intersection.

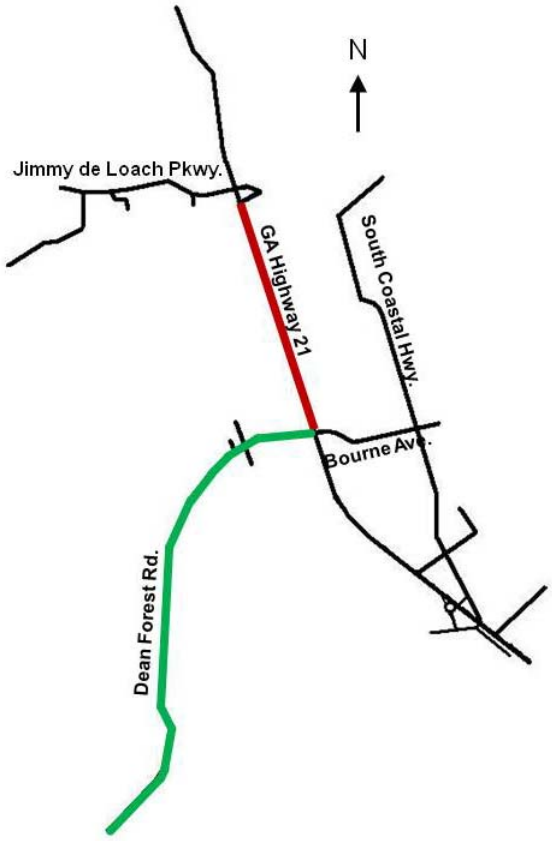



**Figure 63. Partial Travel Time Segments for Background Traffic**

#### 4.2.3.3 Truck Utilization and Location Data

Three types of truck utilization and location data are determined separately for both port and roadway trucks. These are (1) the total number of trucks in the federation system (i.e., trucks in both roadway network and port model federates), (2) the total number of trucks active in the roadway network model federate (both empty and container-laden trucks), and (3) the total number of trucks in the roadway network model federate carrying a container.

Recall that each truck entry recorded in the Vehicle Log table of the federation database specifics the origin, current destination, current location, container ID, and time stamp for every truck transaction. This enables the truck utilization and location data to be determined by post-processing the Vehicle Log data tables.

#### 4.2.3.4 Submodel Facility Processing Data

Processing data counts the number of containers and trucks entering or leaving from each of the five port federate submodels to the roadway network model federate during a given time interval. This data collection type was originally implemented as a troubleshooting device during the development of the port model federate. It is unlikely that the submodel facility processing data will provide significant insight into the operation and performance of the federated system. This is in part due to the fact that it aggregates many port truck and road truck transactions into a single metric. Nonetheless, processing data is collected as a best-practice effort during this study. For this reason, the details of this data collection are outlined below.

Facility processing data is collected by the various **Count** blocks located throughout the port submodels as time-stamped records. Arena© writes this count data directly to a text file. Post-processing then determines the number of trucks and containers entering or exiting a port submodel in every successive 900 second, or 15 minute, time interval.

At each of the three distribution center submodels, five types of processing data are collected: (1) the number of trucks entering the distribution center, (2) the number of containers entering the distribution center, (3) the number of trucks leaving the distribution center with a container, (4) the number of empty port trucks rerouted to the roadway network, and (5) the number of empty road trucks rerouted to the roadway network.

At the GCT Gate submodel, five types of processing data are collected: (1) the number of containers arriving at the GCT Gate submodel from incoming freight ships (2) the number of trucks entering the GCT Gate submodel, (3) the number of containers entering the GCT Gate submodel, (4) the number of trucks leaving the GCT Gate submodel with a container, and (5) the number of roadway trucks rerouted to the roadway network model bound for the I-16 Junction.

At the I-16 Junction submodel, four types of processing data are collected: (1) the number of truck/container combos created (i.e., those entering the roadway network model), (2) the number of empty road trucks created (i.e., those entering the roadway network model), (3) the number of containers leaving the roadway network model to enter the I-16 Junction submodel, and (4) the number of trucks leaving the roadway network model to enter the I-16 Junction submodel.

### 4.2.4 Data Analysis Methods

Analysis of the federated simulation output data occurs in two phases. First, a broad overview of the entire five-day simulation period is conducted to identify major trends in output data. This provides a general understanding of the federated system's operation and identifies trends in output that may influence statistical analysis. Second, the simulation output is separated into single-day intervals to capture both the transient and steady state phases following each end-of-day input parameter adjustment (as described previously). Statistical data analysis methods are then applied to each single-day output interval.

Statistical data analysis methods in this study are used to determine two characteristics of the output data, each related to one of the two assumptions discussed in section 4.2.1. One task is to determine from the output data if the simulation reaches steady state in any or all of the five one-day periods. The second task is then to determine the time-lag (often called the "initial transient phase") between when a federation model parameter is changed and when the model readjusts from its initial steady state to a new steady state condition. The first section below will introduce the general procedure that is required. The second two sections below then introduces the specific methods used to determine these two characteristics from the output data.

#### 4.2.4.1 General Approach to Statistical Output Analysis

The federated simulation in this study will be conducted as a non-terminating simulation. This means that an arbitrary terminating condition will be used to define the length of the federated simulation execution (in this case, when the simulation has run for

five days of simulation time) because "there is no natural event E to specify the length of a run." [35] Law (2007) notes that this is common "when we are designing a new system or changing an existing system, and we are interested in the behavior of the system in the long run when it is operating 'normally.'" [35] However, it is well documented that within non-terminating time sequence simulations, steady state observations (i.e., sequential data points) are correlated and therefore not independent [35-38]. Thus, it is necessary to conduct the federated simulation using the method of independent replications wherein the output data from numerous replications of a simulation are combined. As each of these replications are conducted independently, the new combined set of observations are therefore independent. Law (2007) and Welch (1983) note that $M$ = 5 to 10 replications provides a good starting point, however the exact number is dependent on data variability and execution time [35, 37]. For this study, $M = 10$ replications are conducted to produce 10 independent data sets which are then combined.

As we are interested in the time-lag associated with the readjustment to steady state, each individual day of simulation time will be examined separately to determine the extent of the time-lag (initial transient phase) and steady state phases within that individual day following some change of a federation input parameter. Moreover, each individual day of simulation time will have to be examined in chronological order. This is due to the fact that if the initial transient phase of one day extends into the subsequent day, this will affect the determination of time-lag and steady state simulation in subsequent days.

Data sets from the 10 independent replications will be combined using a method first outlined by Welch (1983) wherein the sequence of sample means is determined for

*M* replications of some output sequence of observations $\{V_n: n = 1,2,\ldots\}$ [37]. This is illustrated by the equation:

$$\hat{\mu}_n = \bar{V}_n = \frac{1}{M}\sum_{m=1}^{M} V_{mn}, \quad n = 1, \ldots, N,$$

Where: $\hat{\mu}_n$ = the sequence of sample means from *M* replications

Therefore, values in the sequence $\hat{\mu}_n$ are independent as they are derived from 10 independent replications. Welch (1983) notes that if M is sufficiently large, $E[\hat{\mu}_n] = \mu_n$ . [37]

Once this independent sequence of sample means is determined, the analysis of output data must be broken into two phases. The first phase is a simple visual examination of the output data in a graphical plot. This examination will identify the suspected extent of the initial transient phase and the presence of apparent steady state operation. The outcome of this visual examination will either determine that the initial transient phase is suspected to terminate at some point during that day of simulation (followed by a steady state phase), or otherwise that the initial transient phase does not apparently terminate during that day of simulation.

If the visual examination determines that the initial transient phase appears to terminate, then more rigorous methods will be employed to confirm the extent of the initial transient phase of the output data. Conversely, if the visual examination cannot identify that the initial transient phase terminates, then subsequent days of simulation and other data outputs will have to be examined to explain this phenomenon.

#### 4.2.4.2 Determination of Steady State Simulation

Section 4.2.4.1 described a simple process by which simulation output is visually examined to determine whether steady state operation occurs in a given day of simulation. Welch (1983) proposes a more statistically rigorous method of determining steady state operation [37]. The central premise is that a histogram showing the distribution or frequency of occurrence of an observation from a series of replicated simulations should approach some common distribution as one plots histograms for observations that occur successively later in the dataset. In effect, the "histogram is an estimate of the probability distribution" [37] for a given output value. For example, consider a sequence of observations $\{V_n: n = 1, 2,\ldots,200\}$ for which we plot the histograms of $V_{40}$, $V_{80}$, $V_{120}$, $V_{160}$, and $V_{200}$. If the data set is at steady state, these histograms, which examine successively later-occurring observations, should converge to some common probability distribution. See Welch (1983) for several examples of this method [37].

There are some significant challenges associated with applying Welch's method for determining steady state simulation to this study. First, the method requires more simulation replications than is practical in this study. For the example provided in the text, Welch (1983) conducts 1000 replications to obtain a dataset sufficient to plot histograms that approximate probability distribution functions [37]. Given the amount of time required to execute one five day federated simulation in this study (approximately 2.5 days of computing time using a single-CPU workstation computer is required per execution), it is not practical to obtain 1000 replications of data to confirm steady state operation.

The second concern with Welch's method is as it applies to transportation-related simulation. Much of the data collected from transportation simulations exhibit cyclic trends. For example, vehicle arrival rates at two closely spaced intersections often reflect the cyclic traffic signal phasing of the upstream intersection. Therefore, while successive histograms may converge to a common distribution in a truly random simulation, as Welch (1983) suggests, it is possible that successive histograms would exhibit cyclic behavior when applied to transportation-related simulation data. It is not clear that a recognized method exists to determine steady state operation that can account for this cyclic tendency often seen in transportation-related simulation output. Therefore, steady state behavior in the federated simulation output is confirmed only as "apparent steady state" behavior using the visual method outlined in section 4.2.4.1. Future efforts will delve more deeply into the development of more rigorous approaches to the identification of the presence of steady state operations.

#### 4.2.4.3 Determination of Time-Lag or the Initial Transient Phase

If the initial visual examination of sequenced sample means indicates that a bounded initial transient phase exists within a given day of simulation, then a more precise extent of these bound must be determined through further analysis. Numerous methods exist to determine the extent of the initial transient (see [38] for a complete discussion of various methods). For this study, two practical methods are considered which fall under the category of truncation heuristics: the Marginal Standard Error Rule (MSER) [39] and a moving averages graphical technique [37]. However, Sandikci and Sabuncuoglu (2006) discuss that the MSER method "is very sensitive to outliners

(extreme values)," noting that eight outliers in a dataset of 6000 observations caused the method to indicated that 4876 observations should be truncated (i.e., included in the initial transient phase) as compared to 339 observations, once the eight outliers were removed. Initial test runs of the federated Port of Savannah simulation indicated that moderate to significant variability could exist in the output data. Therefore, the moving averages graphical technique is used in this study.

The moving averages graphical technique is also presented by Welch (1983) and further explained by Law (2007) [35, 37]. This method builds upon the sequence of sample means approach that was discussed in the previous section, by taking a moving average to "smooth out the high-frequency oscillations in [a dataset] but leave the low frequency oscillations or long-run trend of interest." [35] By defining the number of values included in each calculation of the moving average as $2K + 1$, where $K$ is an arbitrarily chosen constant, and using the sample mean sequence data, the moving average is defined as:

$$\bar{\mu}(n:K) = \begin{cases} (2K+1)^{-1} \sum_{k=-K}^{K} \hat{\mu}_{n+k} & \text{if n} \geq K+1 \\ (2n-1)^{-1} \sum_{k=-(n-1)}^{n-1} \hat{\mu}_{n+k} & \text{if n} < K+1 \end{cases}$$

Given this new sequence of moving averages we can visually approximate a value of $n$ beyond which $\bar{\mu}_n$ "appears to have converged" toward some constant value associated with the steady state phase [35].

# CHAPTER 5

# RESULTS, ANALYSIS AND DISCUSSION

This section presents the results of the federated simulation experiment, analyzes specific days of data within the datasets, and discusses the significance of the large trends exhibited by the data. This begins with a broad overview of the five-day output to provide a better understanding of the federated system's general operation and identify general trends in the data. This data is then analyzed in individual, successive single-day intervals. General trends and specific analysis results are then discussed.

## 5.1 General Results and Data Trends

Upon completion of the federated simulation replications, the queue length, travel time, truck utilization and location, and facility processing data was output and then aggregated into sequences of sample means using the method described in section 4.2.4.1. This section provides an overview of the sample mean output data across the full five days of simulation time. Note that this overview of results focuses heavily on performance of the GCT Gate submodel, and the performance of the port trucks in the roadway network model federate. The results at the distribution centers are not covered in any depth as all three distribution centers exhibit similar trends. Therefore, only results from Distribution Center 1 are examined in depth.

### 5.1.1 Queue Length Data

Queue length data was collected at the GCT Gate submodel and each of the three distribution center submodels. Queue length data was collected at the end of every minute of simulation time, therefore each data point in the figures that follow represent the previous minute of simulation time. Figure 64 shows the queue lengths for idle port trucks waiting to be batched with outgoing containers and the queue lengths for

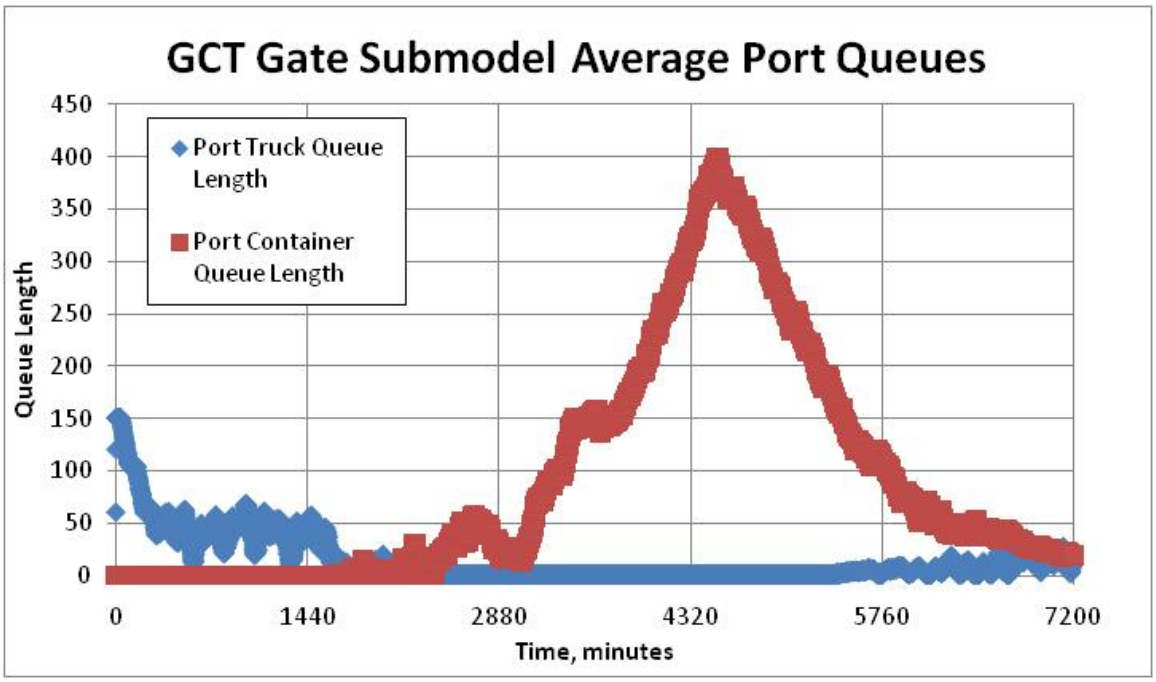



**Figure 64. GCT Gate Submodel Average Port Queue Lengths**

containers that have been received from the aggregated GCT submodels (i.e., cargo ships) and are waiting to be batched for transport to one of the three distribution centers. One day is equivalent to 1440 minutes, therefore vertical lines have been placed at 1440 minute intervals for reference.

Several interesting phenomena are shown in Figure 64. Note that the supply of port trucks starts at 150 trucks (recall that this was a user-defined input), but quickly converges to what appears to be a steady state queue length of approximately 50 port trucks, and maintains that state for the first day of simulation. Consequently, as there is an excess supply of port trucks available for the first day, a queue of port containers does not accumulate. At the beginning of the second day (simulation time 1440 minutes), the volume of containers arriving to the port via cargo ships is increased. This increase in demand of port trucks for container transport reduces the port truck queue length over the second day of simulation. The port truck queue first reaches a zero average queue length at simulation time 1779 minutes. A second consequence of the increase in container demand is to cause a queue of port containers to accumulate. The beginning of this port container accumulation coincides roughly with the dissipation of the port truck queue, with the first non-zero average queue length of 0.2 containers at time = 1639 minutes.

At the beginning of the third day of simulation, at 2880 minutes, the roadway network background traffic volume is increased. Note from Figure 64 that this roughly coincides with the sharp increase in the port container queue at the GCT Gate submodel. One explanation for this is that the supply of available port trucks becomes significantly delayed by the increased roadway congestion. This sharp increase in port container queue length equates to a significant surplus demand for port trucks to transport these containers. Therefore, the port truck average queue length is zero for all of day three.

At the beginning of the fourth day of simulation (time = 4320 minutes) the volume of containers arriving to the GCT Gate submodel from the aggregated GCT submodels (i.e., cargo ships) is reduced to its original volume. At simulation time 4525

minutes, the port container queue reaches its maximum average queue length of 399.4 containers, after which the queue begins to dissipate.

At the beginning of the fifth day of simulation (simulation time 5760 minutes) the roadway network traffic volume is reduced to its original volume. This roughly coincides with a queue of port trucks starting to accumulate at the GCT Gate submodel.

Recall from section 4.2.2.2 that road trucks are generated at the I-16 Junction submodel at a rate of 200 vehicles per hour. This is done to provide a sufficient number of road trucks for both high and low container volume scenarios, with some excess capacity. Also recall from section 4.2.2.2 that the condition for rerouting empty road trucks is set to a road truck queue length of 5 vehicles. Given these conditions, Figure 65 shows the queue lengths for idle road trucks waiting to be batched with outgoing containers and the queue lengths for containers that have been received from the

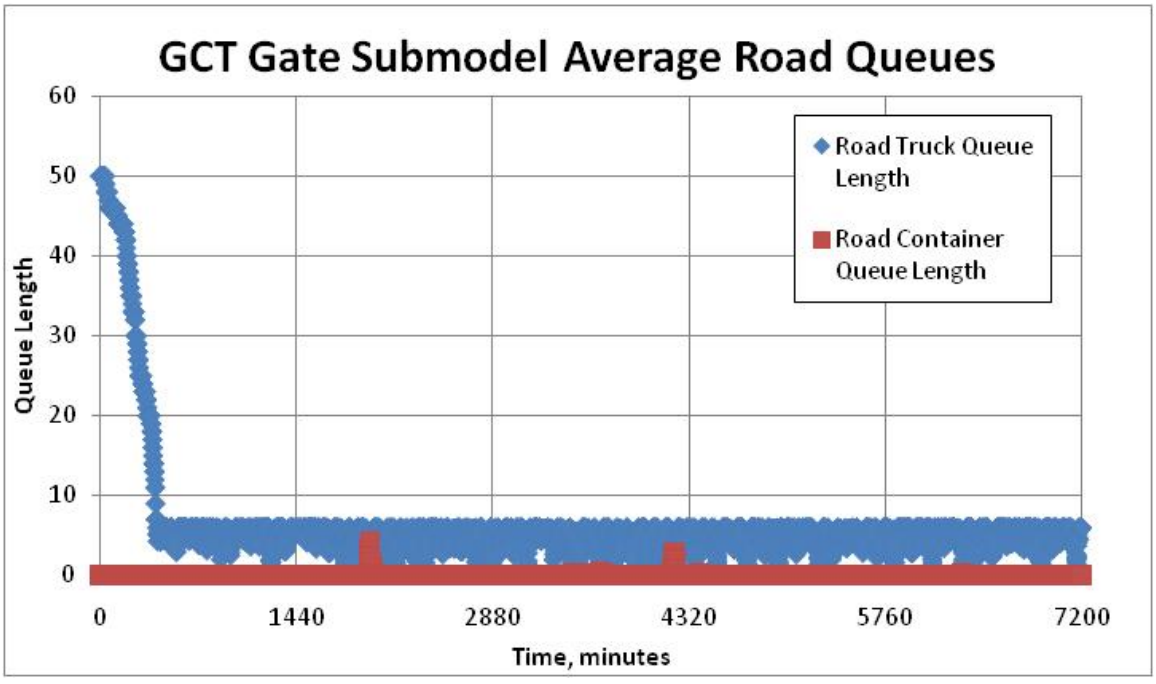



**Figure 65. GCT Gate Submodel Average Road Queue Lengths**

aggregated GCT submodels (i.e., cargo ships) and are waiting to be batched for transport to the I-16 Junction. At simulation time 0, an initial quantity of 50 road trucks is created at the GCT Gate submodel, which dissipates after some initial transient period. The road truck queue length then maintains what appears to be a steady state queue length near 5 road trucks. This state is maintained for the entire five days of simulation. Because of this excess supply of road trucks, a road container queue (that is, containers bound directly for the I-16 Junction) seldom accumulates.

Average queue lengths at Distribution Centers 1 – 3 all exhibit similar behavior. Therefore, only plots of average queue length for Distribution Center 1 are presented. Figure 66 shows the average port truck and port container queue lengths at Distribution Center 1. Similarly, Figure 67 shows the average road truck and road container queue lengths at Distribution Center 1.

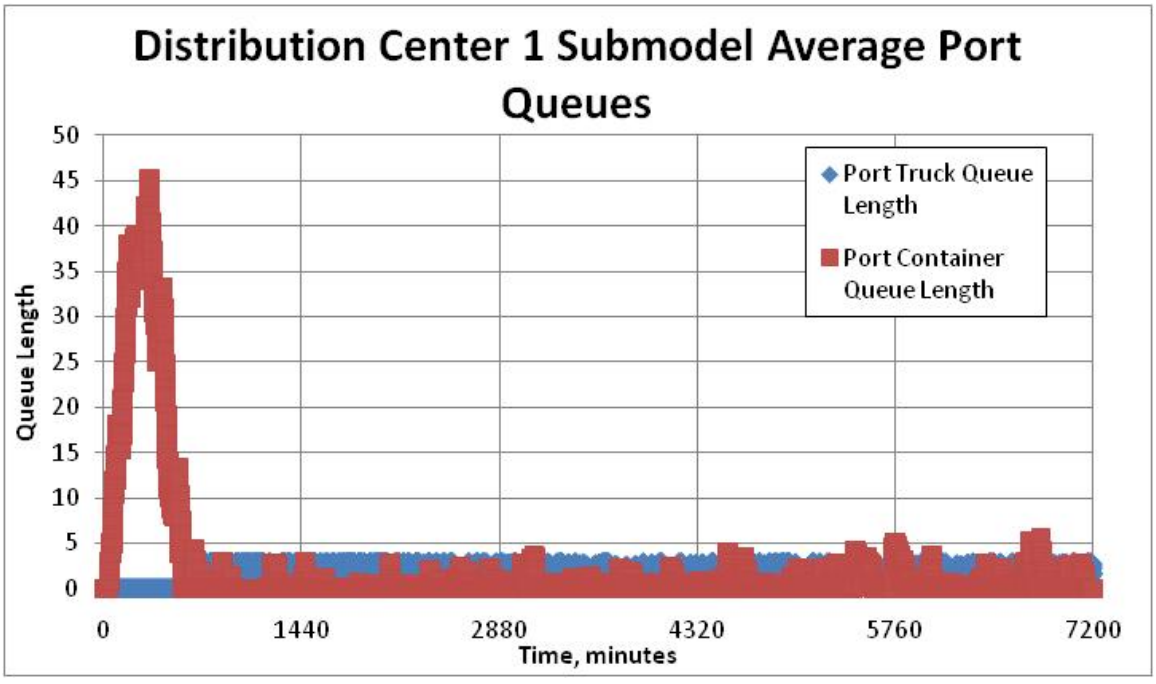



**Figure 66. Distribution Center 1 Submodel Average Port Queue Lengths**

In the case of Figure 66, note that after some initial transient period, no significant queue of port containers accumulates, but instead it appears to reach some steady state queue length. Similarly, a queue of port trucks does not accumulate above the specified rerouting queue length value of 3 trucks.

Figure 67, pertaining to road trucks, exhibits some greater variation in average queue lengths for both road trucks and road containers. Note, however, that the average road container length queue seldom increases to a value greater than eight containers. Also, note that the average road truck queue length does not increase to a value greater than three, which is the rerouting condition for road trucks at distribution centers.

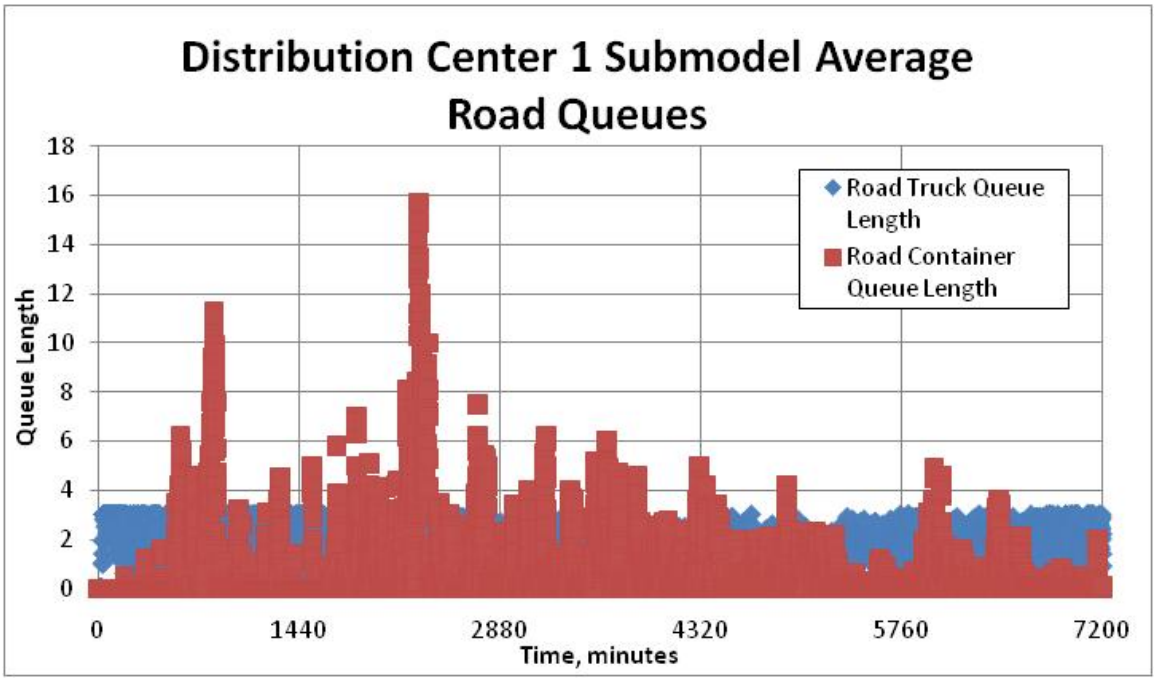



**Figure 67. Distribution Center 1 Submodel Average Road Queue Lengths**

### 5.1.2 Travel Time Data

Travel time data was collected for both port and road trucks, as well as background traffic in the roadway network model federate. Travel time data was compiled in 30 minute intervals; therefore, each data point in the figures that follow represents the aggregation of 30 minutes of data collection. The primary focus of the travel time data is on those travel times associated with background traffic; therefore only travel times associated with background traffic are presented.

Figure 68 shows the plot of travel times for background traffic, both northbound and southbound, along the segment of Highway 21 between Jimmy de Loach Pkwy and Bourne Ave/Dean Forest Rd. Several key phenomena should be noted from this plot.

Travel times along the segment in both directions begin at roughly the same value. This makes some sense as the lane configuration and base traffic volumes are the

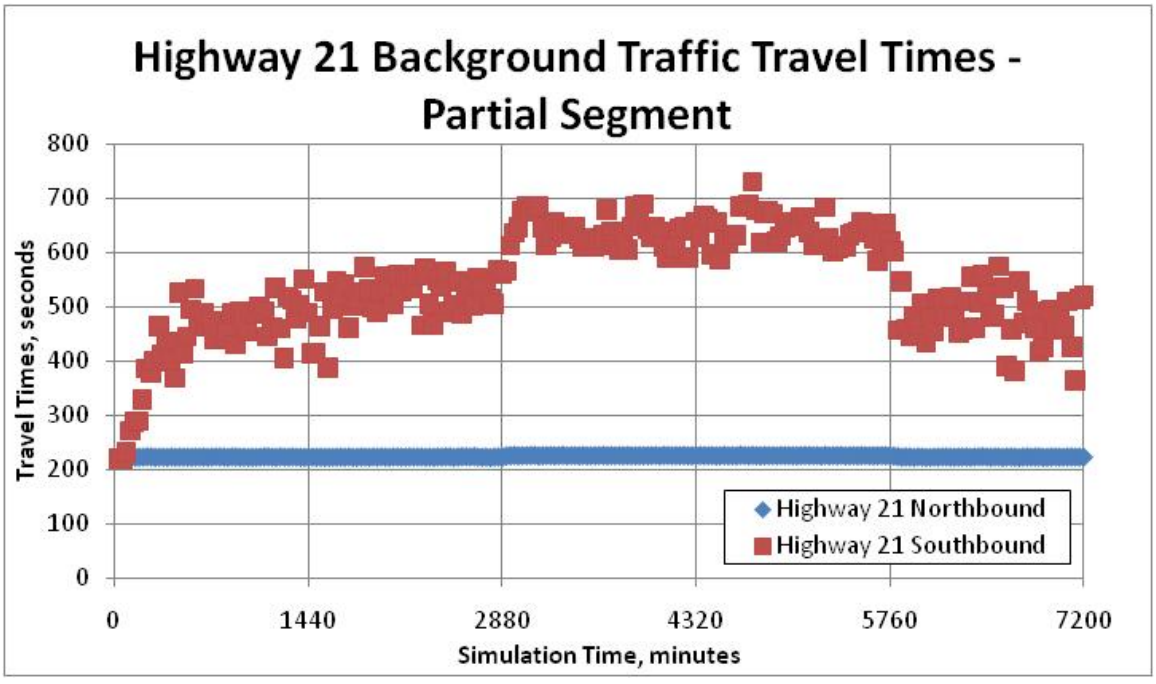



**Figure 68. Highway 21 Background Traffic Travel Times – Partial Segment**

same for both directions. However, after some initial transient period, southbound travel times increase until reaching what appears to be a steady state value near 475 seconds.

At 1440 minutes, the port container volume is increased. This appears to roughly coincide with a slight increase in southbound travel times which reach a new steady state value near 525 seconds. At 2880 minutes, the background traffic volume is increased. This coincides with an increase in southbound travel times, which after a short initial transient period, appear to reach a steady state value near 625 seconds. This steady state southbound travel time of approximately 625 seconds is maintained throughout the third and fourth days of simulation. However, recall that the port container volume is decreased at 4320 minutes, yet there is no apparent effect on the travel times shown in Figure 68. At time 5760, background traffic volume is reduced to its original level. This coincides with a decrease in southbound travel times, which after an initial transient appears to reach a steady state value near 500 seconds.

Note, however, that there is significantly greater variability in southbound travel time than in northbound travel times, which do not change significantly over the five days of simulation.

The travel times for port traffic between the GCT, distribution centers and the I-16 Junction exhibit trends identical to those shown for Highway 21 background traffic in Figure 68, differing only in magnitude of travel times. Because of these similarities, further discussion of these other travel time plots is not necessary.

The one travel time plot that does vary somewhat significantly from the Highway 21 travel times shown in Figure 68 relates to background traffic along Dean Forest Rd. between Highway 21 and the I-16 Junction. This travel time data is shown in Figure 69.

Note first that travel times for both directions along Dean Forest Rd. are much closer in value than those shown for Highway 21 in Figure 68. Also note that northbound travel times appear to be affected by the increase in background traffic volume between 2880 minutes and 5760 minutes, whereas southbound travel times do not appear to be affected by the increase in background traffic. Also, there is some localized variation in northbound travel times that coincides with decreased port container traffic at time 4320 minutes, but the apparent steady state travel time of northbound traffic does not change.

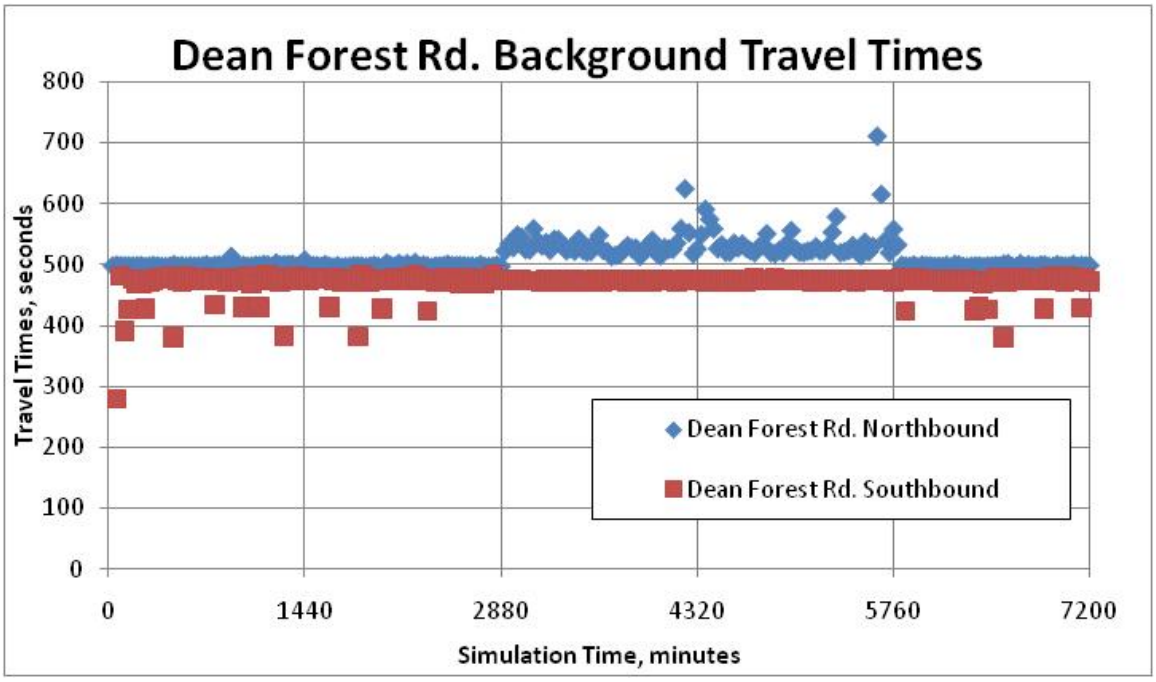



**Figure 69. Dean Forest Rd. Background Traffic Travel Times**

### 5.1.3 Truck Location and Utilization Data

Truck location and utilization data was collected at the end of every minute of simulation time, therefore each data point in the figures that follow represent the previous

minute of simulation time. Road truck location data (i.e., how many trucks are located in the port model federate and how many trucks are located in the roadway model federate) is presented first, and is examined separately from port truck location information.

Figure 70 shows the total number of road trucks that are in the federated system (i.e., traveling on the roadway network, or queued at one of the port facilities) during the simulation. At the beginning of simulation, there is a short initial transient before the number of trucks appears to reach a steady state value near 130 trucks. At simulation time 1440 minutes, the port container volume is increased, yet the steady state value of 130 trucks appears to persist. At 2880 minutes the background traffic volume is increased. From this figure it is unclear if the number of trucks value achieves steady state in the third day of simulation. At time 4320 minutes, the port container volume is reduced. After some initial transient, which may be a continuation of the transient phase

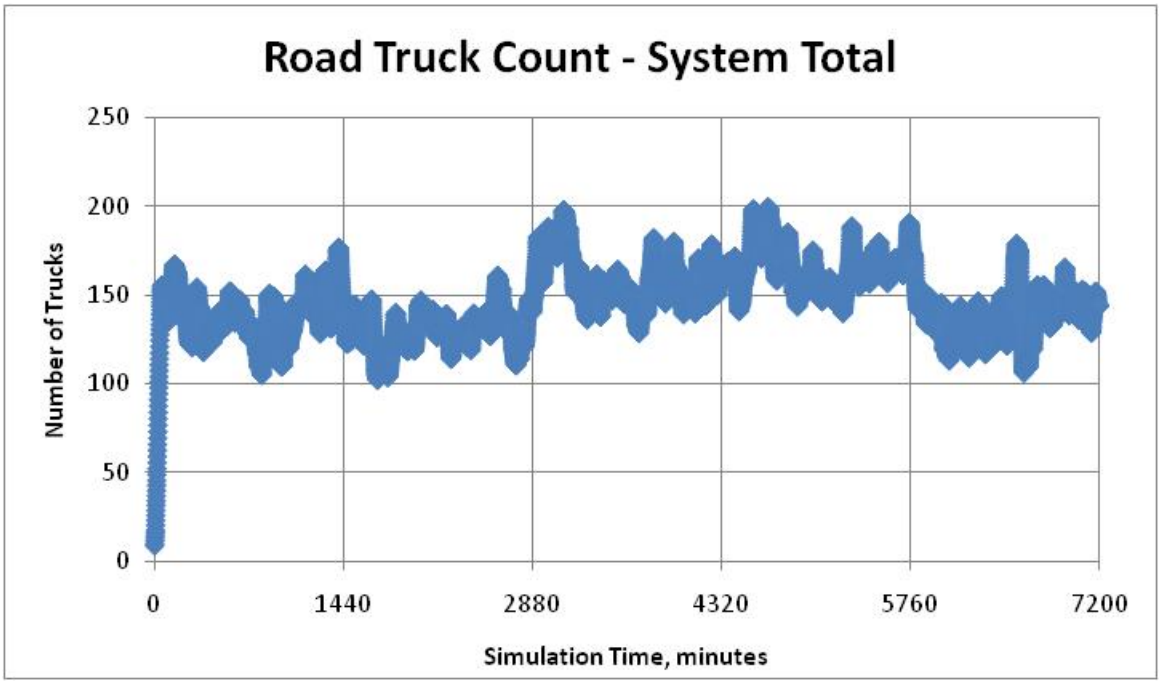



**Figure 70. Road Truck Location Count – System Total**

from the previous day, the truck count value appears to reach a steady state value near 155 trucks. At time 5760 minutes, the background traffic is reduced. Again, from this figure it is unclear if the number of trucks achieves steady state in the fifth day of simulation, however the trend does appear to converge towards the initial day one value of 130 trucks.

Figure 71 shows the total number of road trucks that are in the roadway network during simulation. Note that the "On Roadway Total" road truck count values shown in Figure 71 exhibit similar trends to "System Total" road truck count values shown in Figure 70, differing only in magnitude.

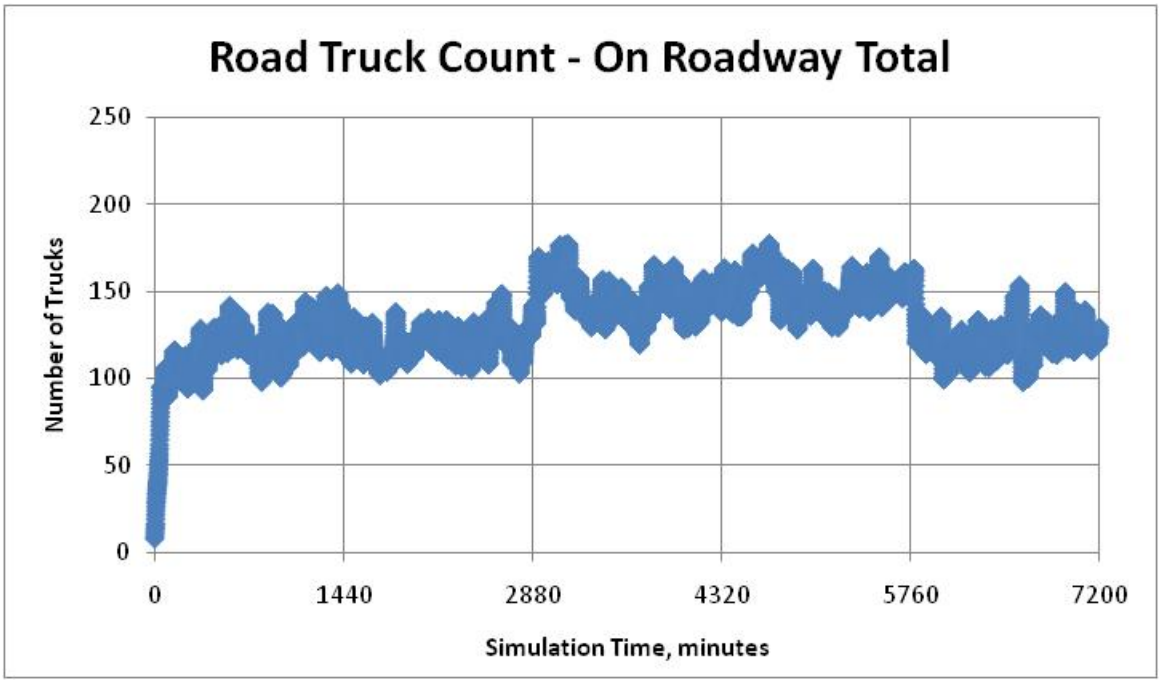



**Figure 71. Road Truck Location Count – On Roadway Total**

Figure 72 is the final road truck location/utilization figure and shows the number of trucks in the roadway network during simulation that are carrying a container (i.e., is

utilized to carry a container). Note that the vertical scale in Figure 72 is smaller than in previous figures to accentuate the variability in the data. Although the general trend of "With Container Total" road truck utilization shown in Figure 72 exhibits many of the same trends as the "System Total" and "On Roadway Total" count values, there is one key difference. In Figures 70 and 71, the number of trucks remained at roughly the same steady state value both before and after the increase in port container volume at time 1440 minutes. In Figure 72, conversely, there is an apparent increase in the steady state utilization value sometime after 1440 minutes.

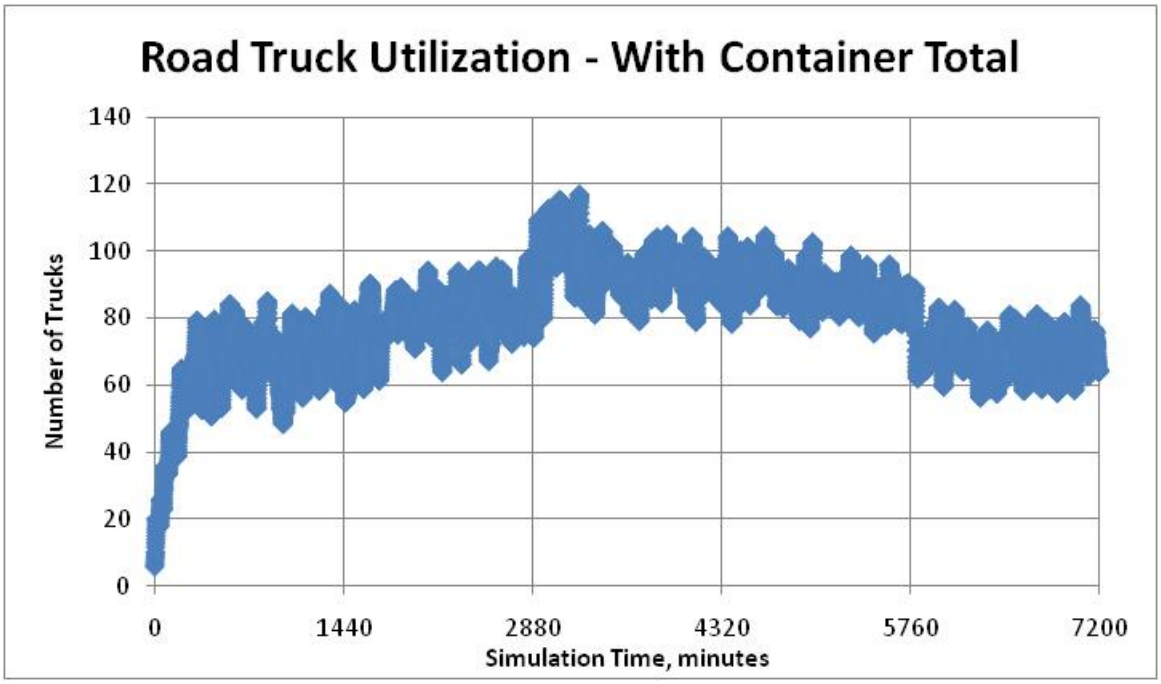



**Figure 72. Road Truck Utilization – With Container Total**

The second type of truck location and utilization data pertains to the number and utilization of available port trucks during simulation. The port truck count "System Total" (that is, the total number of port trucks that are in the federated system) remains

constant during simulation, with some very minor variation due to vehicle diffusion. However, there is significant variation in both the "On Roadway Total" count values and the "With Container Total" port truck utilization data.

Figure 73 shows the port truck "On Roadway Total" count values for the entire simulation. Note first that after some initial transient, the number of trucks appear to reach a steady state value near 55 trucks. After the increase in port container volume at time 1440 minutes, there is an initial transient before the number of trucks appear to reach a new steady state value near 75 trucks. At time 2880 minutes the background traffic volume is increased. This coincides with a slight increase in the number of port trucks in the roadway, however it is unclear from this figure if the data achieves steady state operation in the third day. At 4320 minutes the port container volume is decreased. This appears to coincide with a gradual decrease in the number of port trucks in the

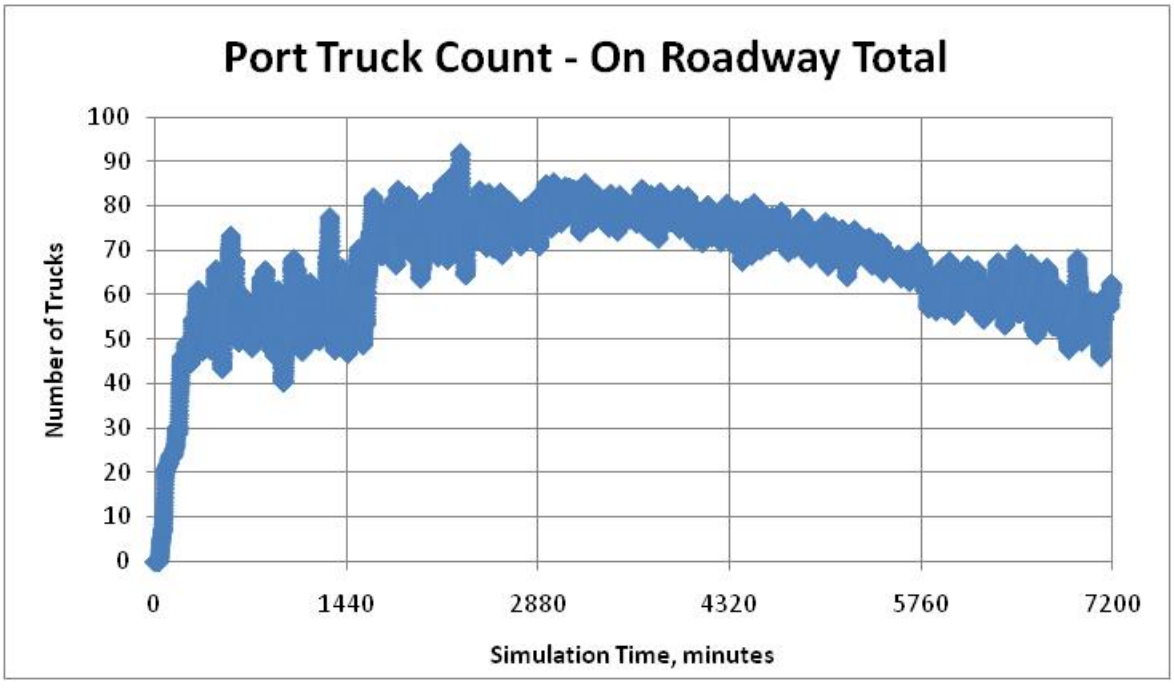



**Figure 73. Port Truck Location Count – On Roadway Total**

roadway. Recall from Figure 64 that the queue of port containers at the GCT Gate steadily increased after time 2880 minutes and then peaked at some point just after time 4320 minutes. It is likely that there is a connection between the peaking trends in Figures 64 and Figure 73. As the queue of port containers needing transport diminishes (and therefore the demand) one could expect to see some decrease in the number of port trucks in the roadway.

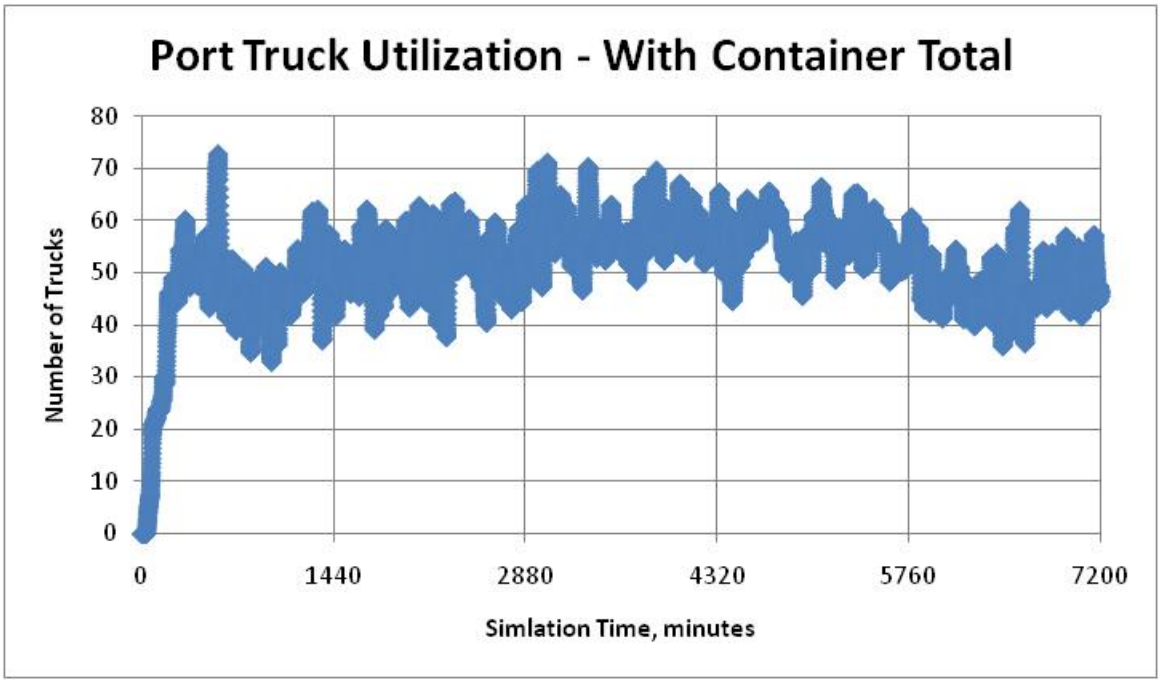



**Figure 74. Port Truck Utilization – With Container Total**

Figure 74 shows the port truck count associated with the number of trucks in the roadway network carrying a container throughout the simulation. During the first day it is unclear if the simulation has had sufficient time to achieve steady state operation. However, after the increase in port container volume at time 1440 minutes there is an apparently slight increase in port truck utilization to a steady state value near 55 trucks.

To reconcile this with Figure 73, one must consider that the increase in port container volume equates to an increase in containers that must move *away* from the GCT, but not back *towards* the GCT (i.e., containers from the I-16 Junction). One would not expect to see the same magnitude change between days one and two in Figure 74 as in Figure 73. This is due to the fact that the "On Roadway Total" truck count reflects both trucks with containers moving away from the GCT to distribution centers, as well as empty port trucks that are rerouted *back* to the GCT to service the excess demand there.

At time 2880 minutes the background traffic volume is increased. This appears to coincide with a slight increase in port truck utilization which converges to an apparent steady state near 60 trucks. At time 4320 minutes the port container volume is decreased. At this time, there appears to be a slight reduction in the steady state value of port truck utilization, to a new value near 55 trucks. At time 5760 minutes, the background traffic volume is reduced. After an initial transient phase, it appears that the port truck utilization data converges towards a steady state value near 55 trucks.

### 5.1.4 Facility Processing Data

As discussed earlier, the facility processing data was initially collected to troubleshoot during the federation development. Nonetheless, it was collected during this study's federation execution as a best practice effort. However, because of its general nature and non-specificity with respect to port truck versus road truck, it provides very little additional insight into the operation of the federated system in this modeling effort. The facility processing data is therefore intentionally omitted from this discussion of results.

## 5.2 Analysis of Results

This section analyzes the data presented in the previous section by applying the statistical analysis methods outlined in Chapter 4 – Design of Experiment. This statistical analysis is conducted by examining individual days of data from the simulation output to determine the extent and presence of the steady state phase, and the extent of the initial transient phase, for that day of simulation. However, having examined the data output and results from the previous section, it is clear that some data sets do not achieve steady state in either individual or multiple days from the five day simulation. Therefore, the analysis in this chapter is limited to only those data sets where steady state operation is likely achieved in one or more days of simulation time.

Furthermore, recall that an excess supply of road trucks was created at the I-16 Junction throughout the simulation. These excess, empty road trucks were continually rerouted back to the I-16 Junction from the GCT and the distribution centers during simulation runtime; therefore, the queues associated with road vehicles maintained an artificial steady state consistent with the maximum queue value of the rerouting decision. Because of this artificial steady state in road truck queues, as well as the excess supply of road trucks to meet the demand of road containers, neither road truck nor road container queues will be analyzed as they do not exhibit true steady state operation.

### 5.2.1 Queue Data Analysis

Analysis of queue data to determine the extent of the initial transient phase and steady state phase examines two data sets: (1) the GCT Gate submodel average port truck queue, and (2) the distribution center submodel average port container queue.

The first task is to identify the presence and steady state operation. Figure 75 shows the GCT Gate submodel port truck queue data for the first day of simulation (time 0 to 1440 minutes). This data is composed of the moving average of the sequence of sample means from the ten federated simulation replications. The K value used to calculate the moving averages is K = 15.

Recall from Figure 64 that only the first day of simulation exhibited apparent steady state behavior the initial transient phase. Looking more closely at the data presented in Figure 75, it appears that steady state behavior does occur within the first day of simulation. However, the minimum queue length values of approximately 20 vehicles that occur at approximately time 600 minutes, 840 minutes, 1080 minutes, and 1320 minutes suggest that there may be a four hour cyclic trend present. While further

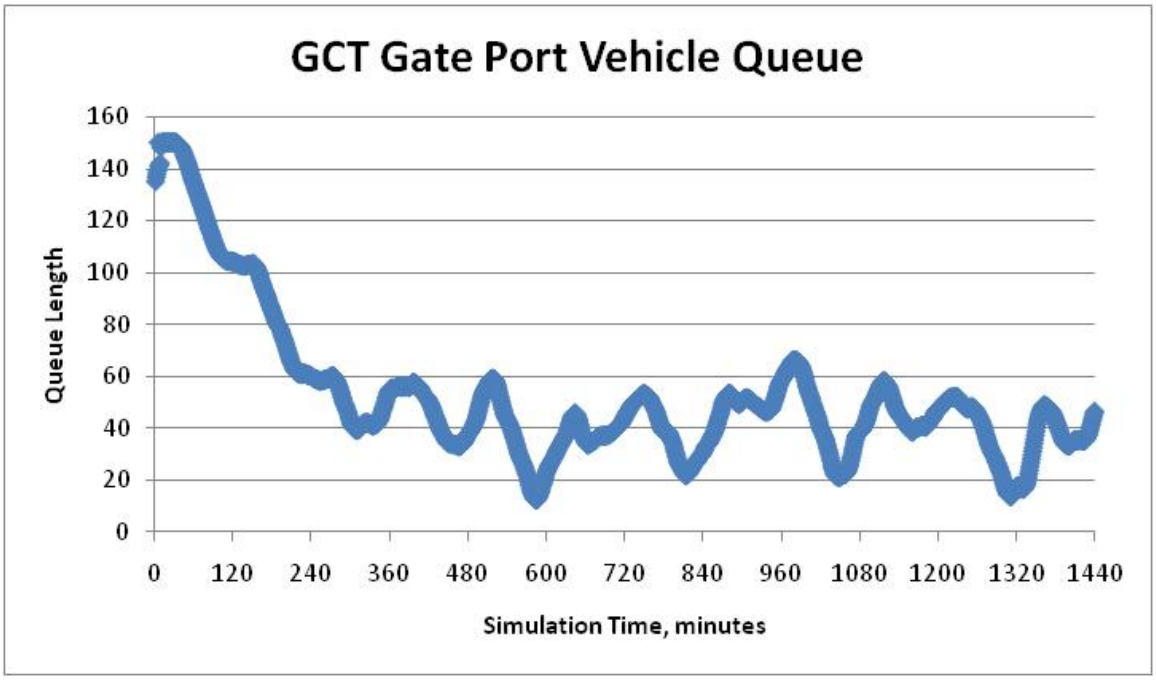



**Figure 75. GCT Gate Port Vehicle Average Queue Length (K=15), Day 1**

analysis would be necessary to confirm the presence of such cyclic trends, this observation does suggest that the method of histograms proposed by Welch (1983) to determine steady state operation would not be effective in this application.

The second task is to determine the extent of the initial transient phase. Applying Welch's visual method [37], it appears that the initial transient phase terminates at approximately 240 minutes. This also coincides with the arrival of the second container ship to the GCT, therefore it is logical that the surge in demand for port trucks associated with these extra containers could help dissipate the starting queue of port trucks created during federated simulation initialization.

Figure 76 shows the queue of port containers (containers bound for the GCT, to be carried by port trucks) at Distribution Center 1 for the first day of simulation. Recall

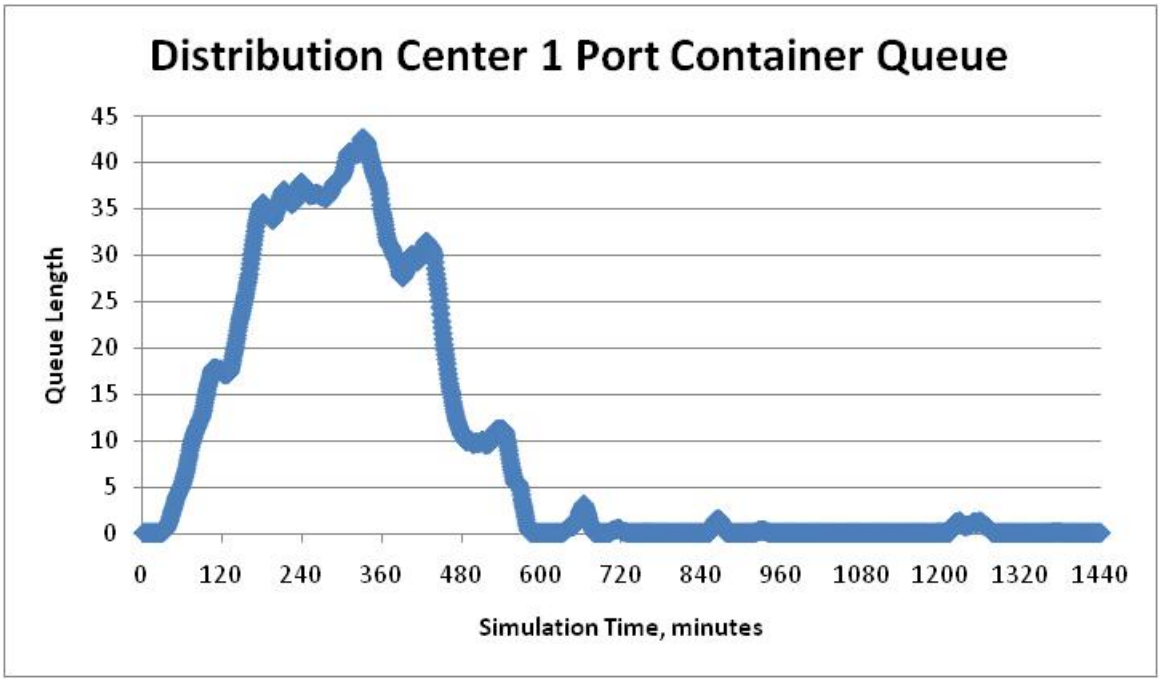



**Figure 76. Distribution Center 1 Port Vehicle and Container Average Queue Length (K=15), Day 1**

from Figure 66 that at some point during the first day of simulation, the data begins to exhibit apparent steady state operation. Figure 76 confirms that apparent steady state operation occurs at some point after 600 minutes. Furthermore, this plot indicates that the initial transient phase terminates at approximately 580 minutes of simulation time. This initial transient is likely associated with the time required for a sufficient quantity of port trucks to circulate and distribute throughout the federated system, and therefore provide the transport capacity necessary to dissipate the queue of containers that has accumulated at Distribution Center 1. As noted earlier, the trends and behavior exhibited in the queue length output data of Distribution Center 1 reflects trends nearly identical to those exhibited in the output data of Distribution Centers 2 and 3. Therefore, the data for these other two distribution centers are not discussed or analyzed further.

### 5.2.2 Travel Time Data Analysis

Analysis of the travel time data to determine the extent of the initial transient and steady state phases examines four travel time segments: (1 & 2) Highway 21 northbound and southbound between Jimmy de Loach Parkway and Bourne Ave/Dean Forest Rd., and (3 & 4) Dean Forest Rd northbound and southbound between Highway 21 and the I-16 Junction. As noted previously, background traffic and truck travel time data exhibit similar trends. Therefore, for both of these segments, only travel times for background traffic are examined. The K value used to calculate the moving averages for travel time data is K = 7. A smaller value was used for travel time than was used for queue length as travel time data consists of fewer data points.

Figure77 shows Highway 21 travel times in both directions for the first day of simulation. Note first that there is no significant variation in travel times for northbound traffic, which maintain a steady state value of approximately 220 seconds immediately upon execution. Southbound travel times, however, exhibit much more significant variation. As indicated earlier in Figure 68, southbound travel times reach an apparent

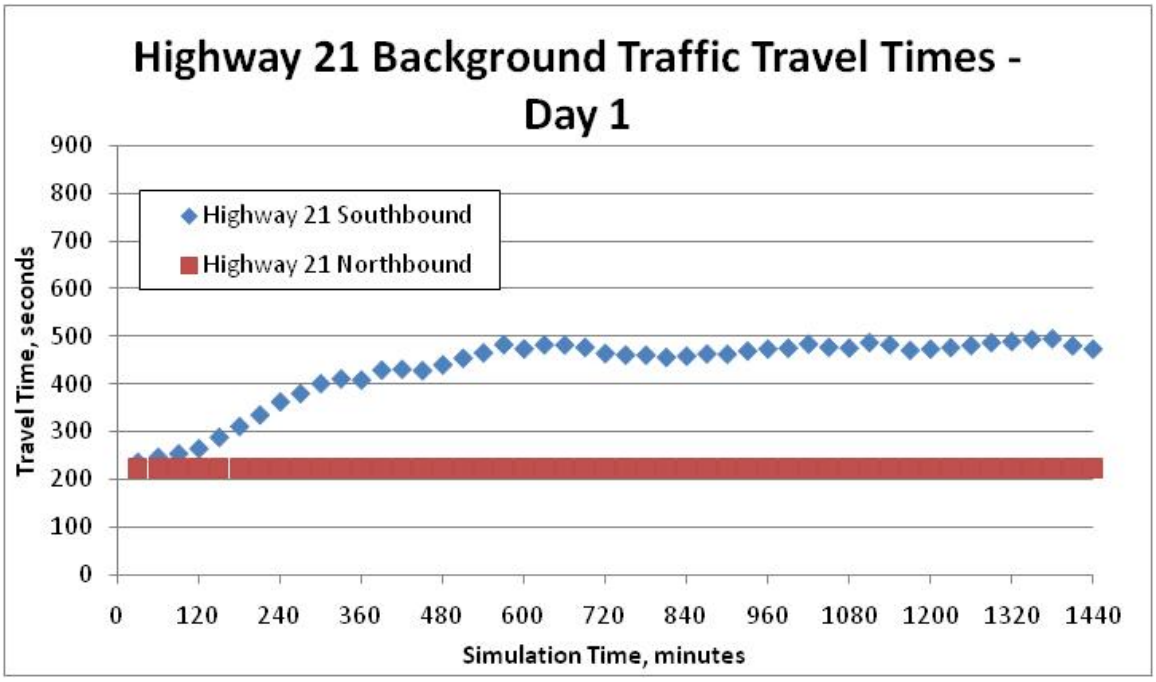



**Figure 77. Highway 21 Background Traffic Average Travel Times (K=7), Day 1**

steady state value during the first day of simulation. Figure 77 confirms this, and that the steady state travel time value is approximately 480 seconds. Figure 77 also indicates that the initial transient phase terminates at approximately 500 minutes, indicating a 500 minute time-lag in travel times.

At simulation time 1440 minutes, the port container volume is increased for the second day of simulation. Figure 78 shows the background traffic travel times for

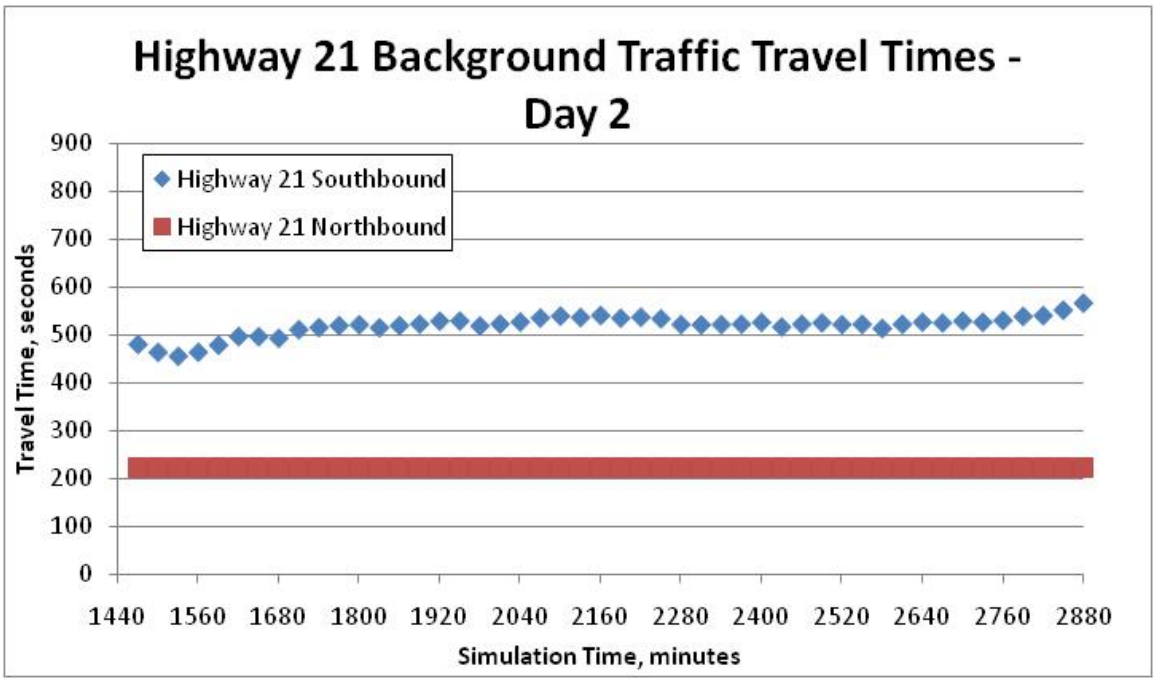



**Figure 78. Highway 21 Background Traffic Average Travel Times (K=7), Day 2**

Highway 21 in the second day of simulation. Note that the northbound travel times appear to remain unchanged. Southbound travel times, however, exhibit a slight increase to a new steady state value of approximately 520 seconds. The initial transient phase for this readjustment to the new steady state travel time value appears to terminate at simulation time 1700 minutes, indicating a 260 minute time-lag for travel time.

At simulation time 2880 the background traffic volume is increased. Figure 79 shows the background traffic travel times for Highway 21 in the third day of simulation. Note that there is no change in the northbound travel times, but that the steady state value of southbound travel times increases to an approximate value of 800 seconds. The initial transient phase appears to terminate at approximately 3000 minutes, indicating a time-lag of approximately 120 minutes for travel time.

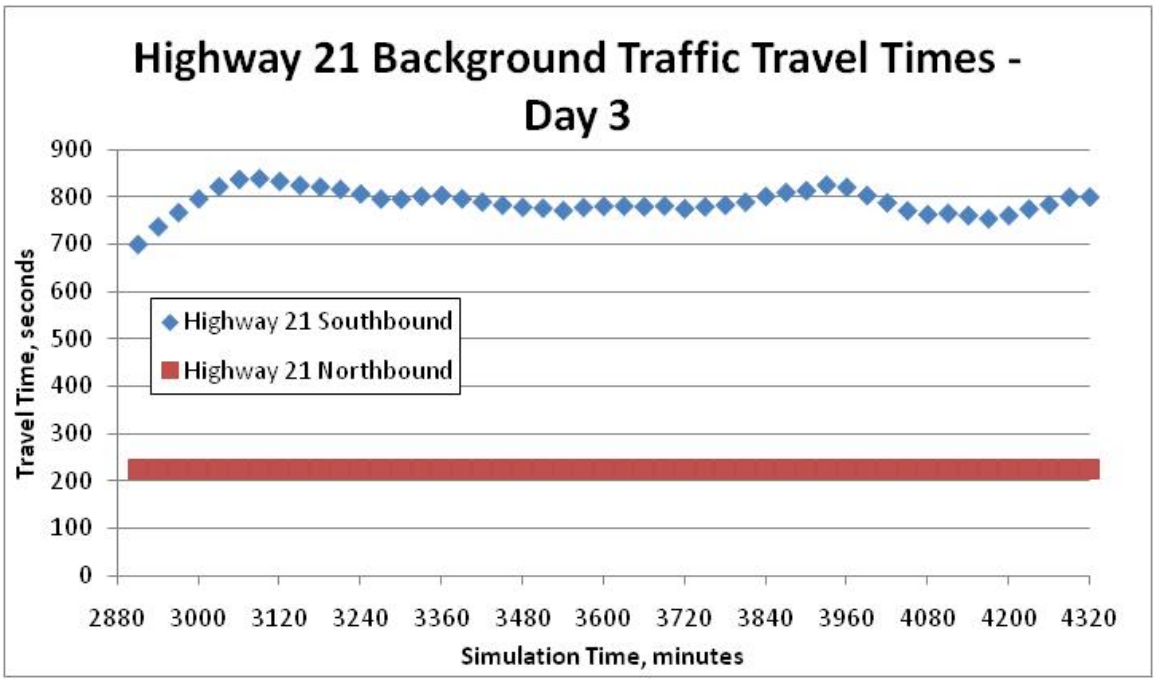



**Figure 79. Highway 21 Background Traffic Average Travel Times (K=7), Day 3**

Travel times for Highway 21 on the fourth day of simulation exhibit steady state travel time values for both northbound and southbound traffic that are similar to those exhibited on the third day of simulation. That is, the southbound travel time steady state value remains approximately 800 seconds, and the northbound value remains approximately 220 seconds. This is regardless of the reduction in port container volume that occurs at simulation time 4320 minutes. Because these trends remain largely unchanged from the previous day, the plotted data for day four are not shown.

At simulation time 5760 seconds, the background traffic volume is reduced to its original level. Figure 80 shows the background traffic travel times for Highway 21 in the fifth day of simulation. Note that northbound travel times again remain unchanged. However, southbound travel times decrease to a new steady state value of approximately 560 seconds. Note that the original southbound travel time value in the low traffic

volume scenario was approximately 480 seconds, and that the travel times shown in Figure 80 have not returned to these original levels. This is likely a result of the higher port trucks activity during day five (as the container queue build up for days three continues to be cleared through day 5) resulting in increased congestion and travel times similar to the higher container volume scenario in day two. The initial transient phase shown in Figure 80 appears to terminate at simulation time 5880 minutes, indicating a time-lag of approximately 120 minutes. It is expected when the container queue at the GCT finally dissipates the travel times will again return to the condition of day one, however a longer simulation run would be required to confirm this result.

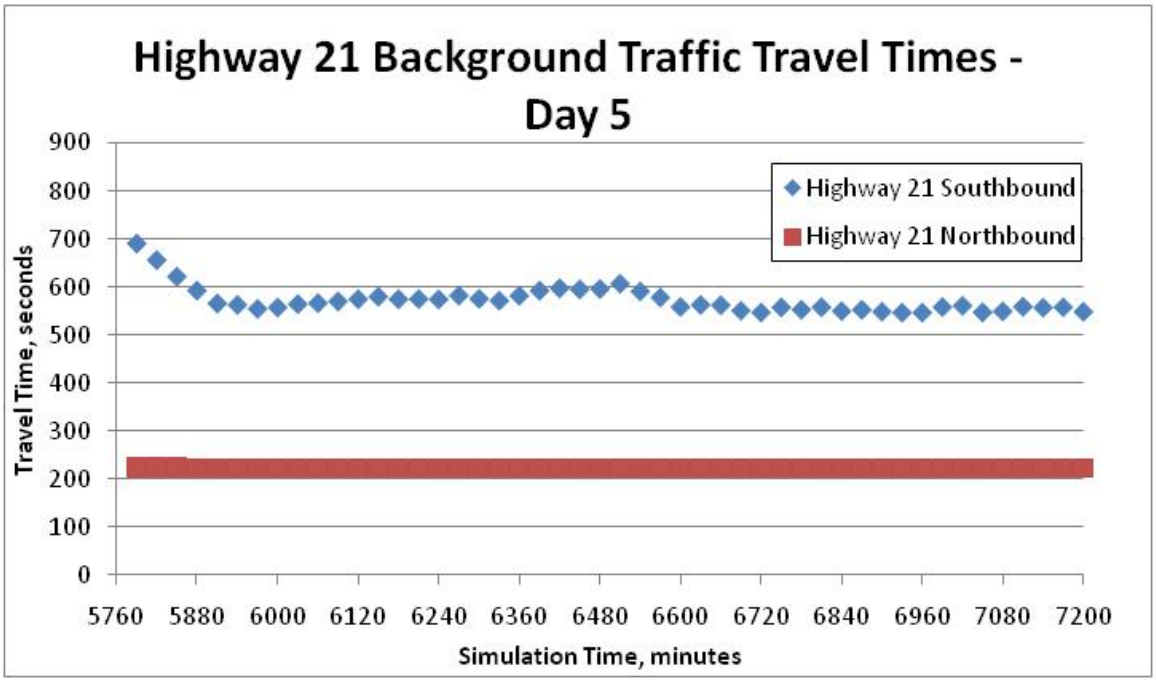



**Figure 80. Highway 21 Background Traffic Average Travel Times (K=7), Day 5**

The second travel time segment analyzed is Dean Forest Rd. between the I-16 Junction and Highway 21. Figure 81 shows the background traffic travel times for Dean

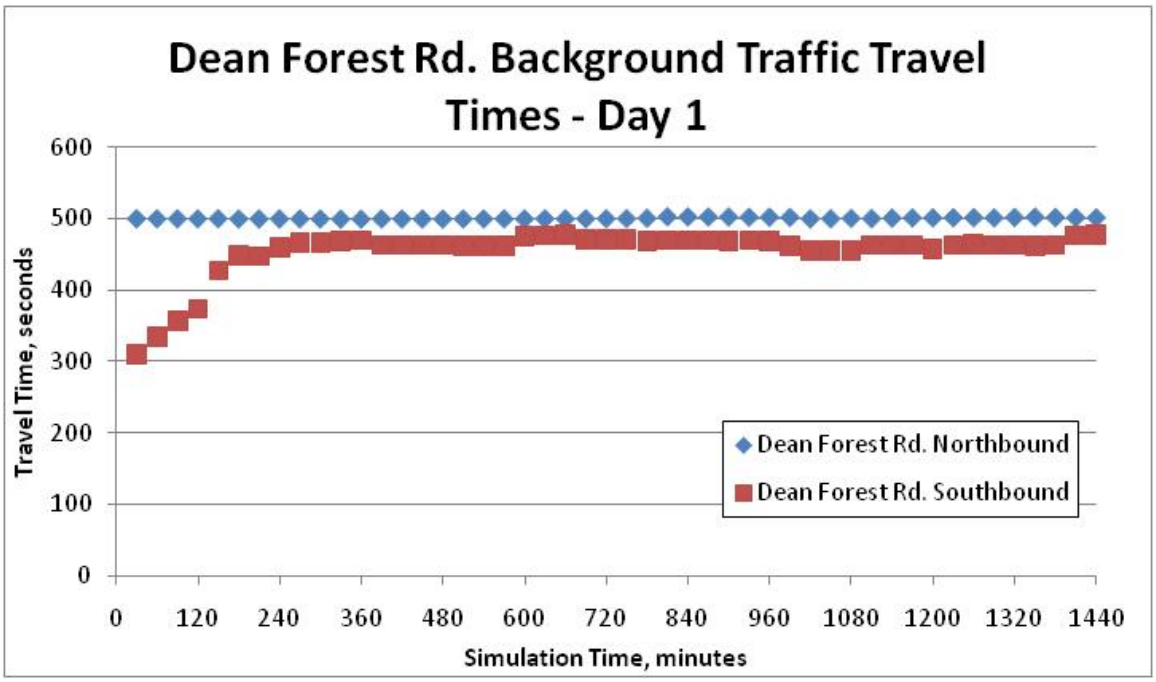



**Figure 81. Dean Forest Rd. Background Traffic Average Travel Times (K=7), Day 1**

Forest Rd. in the first day of simulation. Note first that the northbound travel times maintain a steady state value of approximately 500 seconds, and that there is no apparent initial transient phase, or time-lag associated with that direction of travel. The southbound travel times appear to converge towards a steady state travel time value of approximately 470 seconds. The initial transient phase for southbound travel times terminates at approximately simulation time 240 minutes, indicating a time-lag of approximately 240 minutes.

At simulation time 1440 minutes the port container volume is increased, however travel times in both directions along Dean Forest Rd. appear unaffected. The northbound travel times appear to remain at a steady state value of approximately 500 seconds while the southbound travel times appear to remain at a steady state value near 465 seconds.

As the second day travel time trends along Dean Forest Rd. exhibit the same steady state trends (without any initial transient phase) as those shown for day 1 in Figure 81, the plotted data for second day travel times are not shown.

At simulation time 2880 the background traffic volume is increased for the third day of simulation. Figure 82 shows the travel times along Dean Forest Rd. for the third

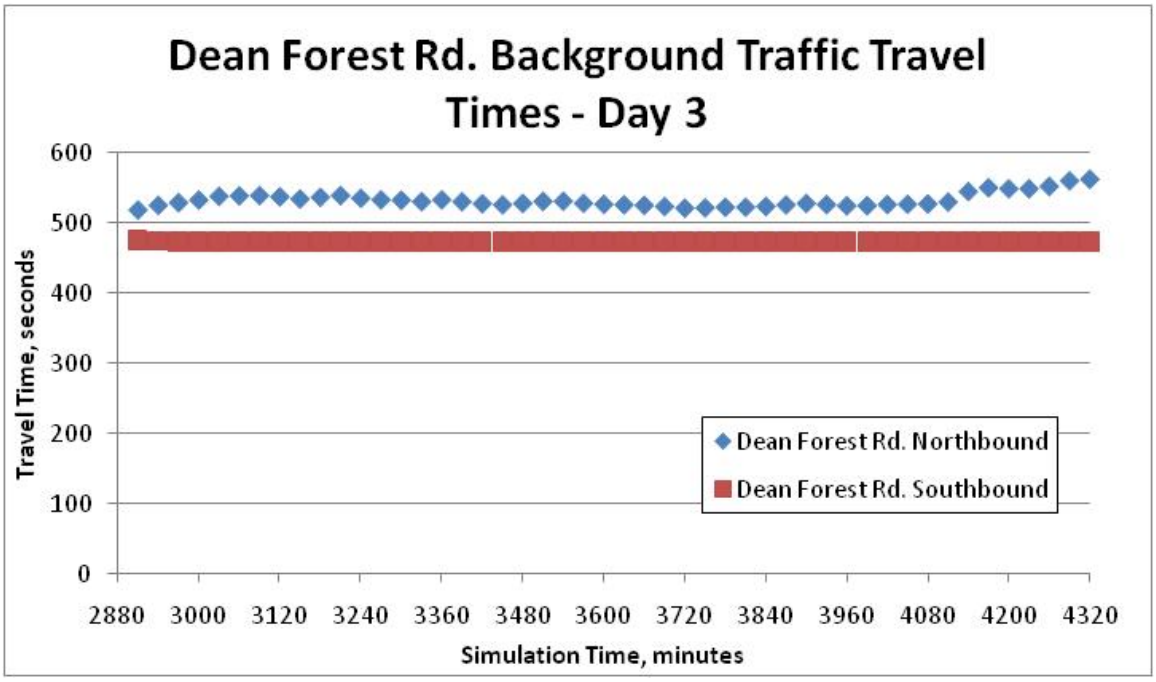



**Figure 82. Dean Forest Rd. Background Traffic Average Travel Times (K=7), Day 3**

day of simulation. Note that the southbound travel times appear to stabilize somewhat as compared with the day one data shown in Figure 81. Northbound travel times exhibit what might be a slight increase in travel times, from 500 seconds to a new steady state value near 530 seconds. The initial transient phase for northbound travel times appears to terminate at simulation time 3000 minutes, indicating a lag-time of approximately 100

minutes. It should also be noted that at approximately 4100 minutes, there is a slight increase in the northbound travel times shown. Given this analysis method, it cannot be conclusively determined whether or not this anomaly is a function of random variability in the simulation or some other factor. However, this perceived increase must be taken into account when considering the next day's travel time information along this roadway segment.

At simulation time 4320 minutes the port container volume is reduced to its original level. Figure 83 shows the travel times along Dean Forest Rd. for this fourth day of simulation. Note that southbound travel times remain largely unchanged with a steady state value of approximately 470 seconds. Northbound travel times appear to remain at their earlier steady state value near 530 seconds. The anomalous variation at the end of

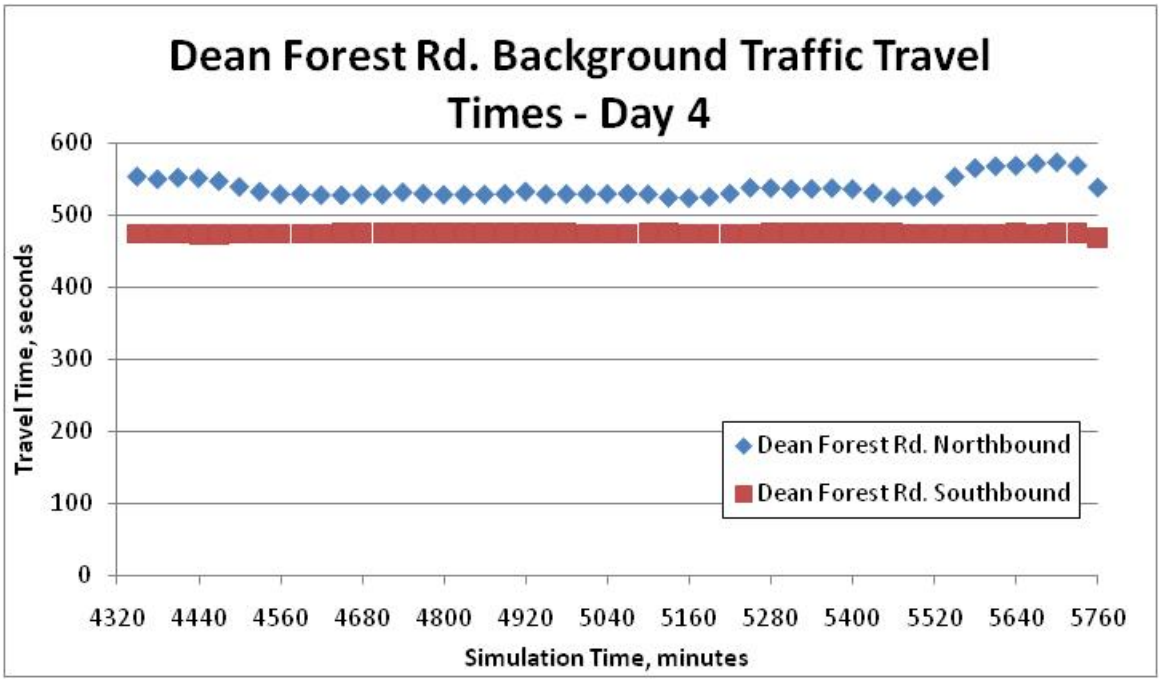



**Figure 83. Dean Forest Rd. Background Traffic Average Travel Times (K=7), Day 4**

the previous day (see Figure 82) appears to recover within first 500 seconds of simulation time in Figure 83. However, there is another anomalous rise in northbound travel times starting at approximately simulation time 5520 minutes. This must also be taken into account when considering travel times for the next day of simulation. As no notable changes have been made to the steady state operation of Dean Forest Rd. travel times for the fourth day of simulation, no initial transient phases need to be determined.

At the end of day four, simulation time 5760 minutes, the background traffic volume is reduced to its original level. Figure 84 shows the Dean Forest Rd. travel times associated with the fifth day of simulation. Note first that there is no significant change in southbound travel times, and that the southbound steady state travel time value remains approximately 470 seconds. There is, however, a reduction in the northbound travel time

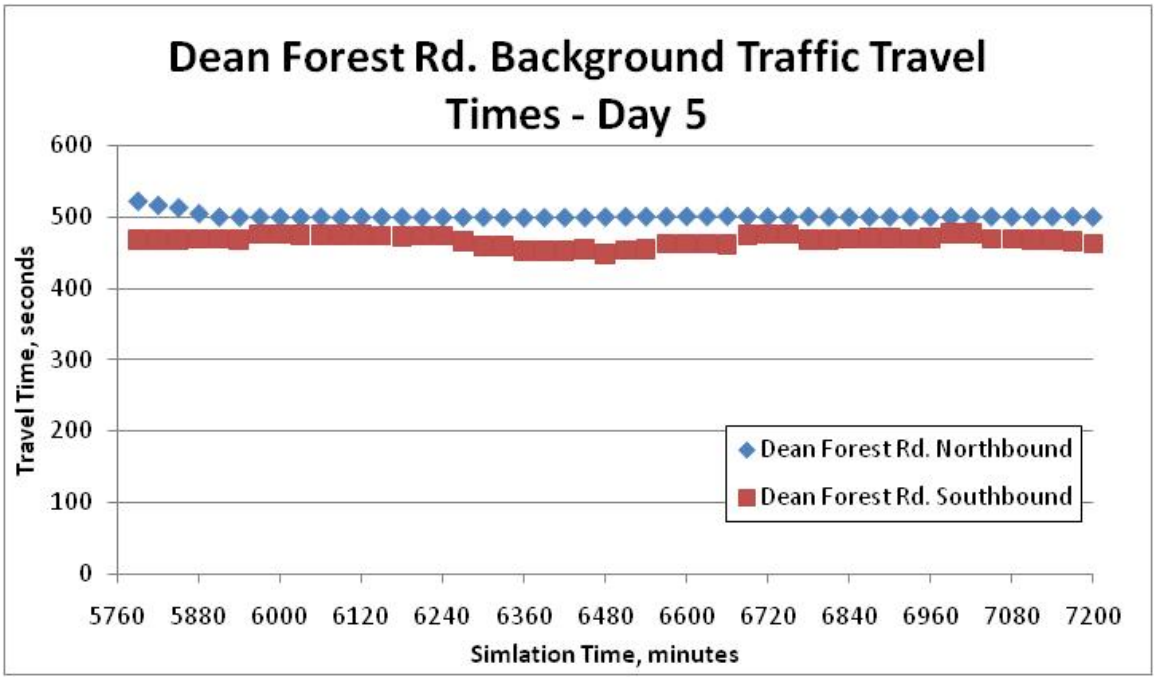



**Figure 84. Dean Forest Rd. Background Traffic Average Travel Times (K=7), Day 5**

values to a new steady state value of approximately 500 seconds. Note that 500 seconds is the original northbound travel time value along Dean Forest Rd. shown earlier in Figure 81. The initial transient phase for northbound travel times appears to terminate approximately at simulation time 5880 minutes, indicating a time-lag 120 minutes.

### 5.2.3 Truck Location and Utilization Data Analysis

The analysis of truck utilization and location data focuses primarily on port trucks. Because of the planned excess supply of road trucks created at the I-16 Junction during simulation, it is difficult to analyze and draw meaningful conclusions about road truck utilization and counts. The larger trends of this somewhat artificial road truck location and utilization data are helpful in understanding the overall operation of the model, but are a poor candidate for steady state analysis. Therefore, only the first day of road truck utilization data will be examined to determine the extent of the initial transient associated with federated simulation start-up.

Given the truck location and utilization data presented in section 5.1.3, analysis of the port truck location and utilization data will only examine the first two full days of simulation. After day two, general trends in port truck location and utilization suggest that steady state operation was not achieved, but rather that some cyclic or transient behavior occurred. Also, as the port truck "System Total" is a fixed quantity, varying only due to the periodic diffusion of trucks during runtime, port truck system total information is not examined. The K value used to calculate the moving averages for truck location and utilization data is K = 15.

5.2.3.1 Road Truck Location and Utilization Data Analysis

As mentioned, the analysis of road truck location and utilization data will only examine data for the first full day of simulation to determine the extent of the initial transient phase associated with federated simulation start-up.

Figure 85 shows the road truck location and utilization data for the first day of simulation. This figure shows the "System Total" values (total number of road trucks within the federated system), the "On Roadway Total" values (total number of road trucks in the

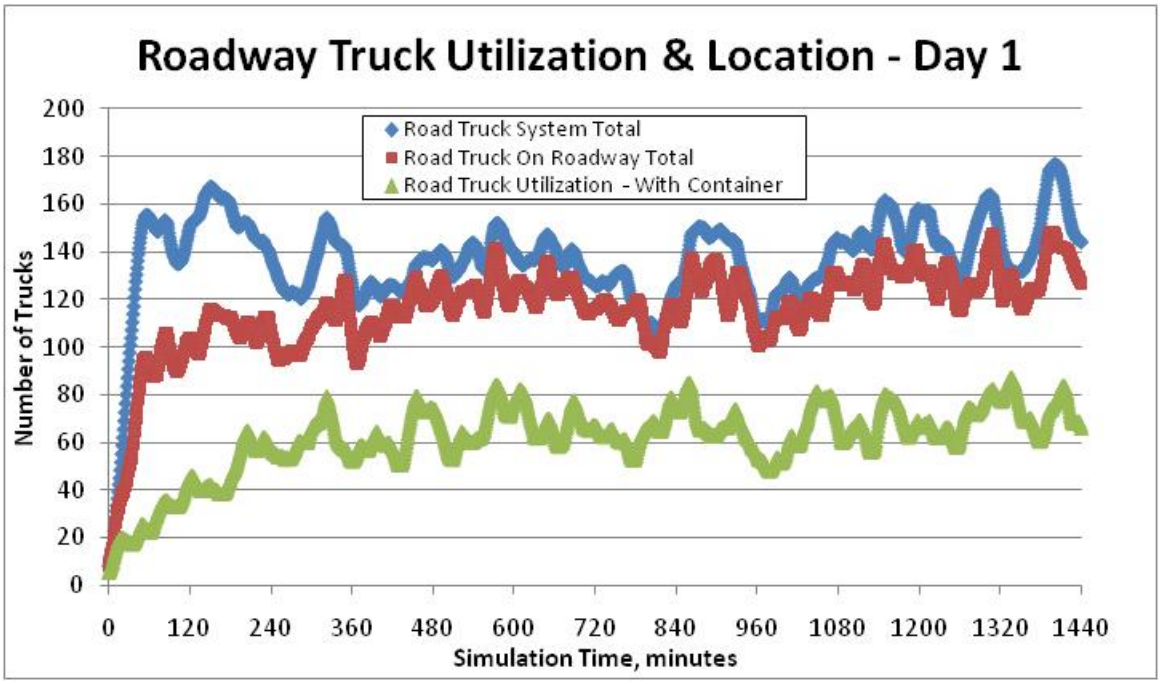



**Figure 85. Average Road Truck Location and Utilization (K=15), Day 1**

roadway network), and the "With Container Total" values (total number of road trucks in the roadway network that are carrying a container). Note that all three data sets appear to converge towards steady state at some point during this first day of simulation. "System

Total" road truck location count appears to converge towards a steady state value of approximately 130 trucks, "On Roadway Total" road truck location count appears to converge towards a steady state value of approximately 120 trucks, and "With Container Total" road truck utilization appears to converge towards a steady state value of approximately 60 trucks.

From Figure 85, we can also determine the approximate extents of the initial transient phases for the three road truck utilization and location data sets. For the "System Total" road truck location count, the initial transient phase appears to terminate at approximately 50 minutes, indicating a start-up time-lag of 50 minutes. For the "On Roadway Total" road truck location count, the initial transient phase also appears to terminate at approximately 50 minutes, indicating a start-up time-lag of 50 minutes. For the "With Container Total" road truck utilization, the initial transient phase appears to terminate at 190 minutes, indicating a start-up time-lag of 190 minutes.

5.2.3.2 Port Truck Location and Utilization Data Analysis

As discussed, the port truck utilization and location data analysis is limited to the first two days of federated simulation. Day one is examined to understand the start-up time-lag characteristics of the federated system for port trucks. Day two is examined to understand the extent of the initial transient phase following an increase in port container volume.

Figure 86 shows the port truck data for "On Roadway Total" truck location values and "With Container Total" truck utilization counts for the first day of simulation. Note that the "With Container Total" number is never higher than the "With Container Total,"

but that at times (e.g., simulation time 0 to 420 minutes and 500 to 560 minutes) the two numbers coincide exactly, indicating 100% port truck utilization. Recall that the "System Total" truck values remain roughly constant throughout the federated simulation (the only variation is due to periodic truck diffusion), and have therefore been omitted from this figure.

Although there is significant variation in the "On Roadway Total" port truck location count dataset, it appears that the data converges towards a steady state value near 55 trucks. This would suggest that the initial transient phase for "On Roadway Total" port truck location data terminates at approximately 300 minutes, indicating a start-up time-lag of 300 minutes.

Given the data for the "With Container Total" port truck utilization, it is somewhat inconclusive as to whether true steady state is achieved. Note the trend of

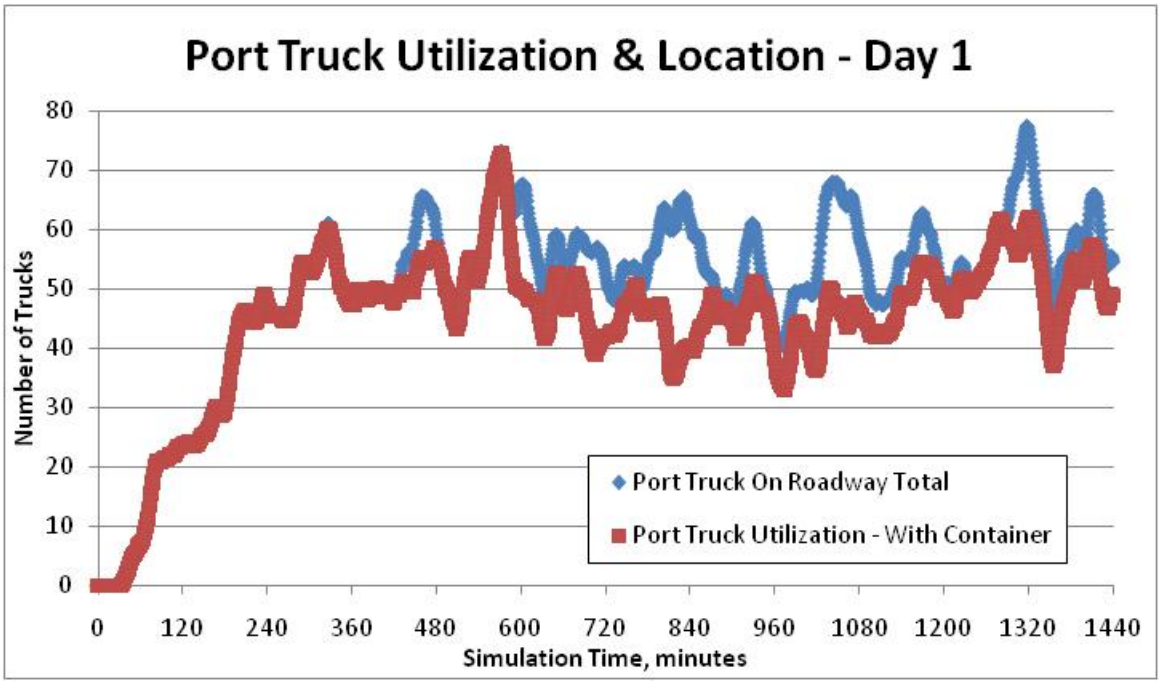



**Figure 86. Average Port Truck Location and Utilization (K=15), Day 1**

slightly reduced truck utilization values between approximately 600 minutes and 1200minutes. From this data and analysis method, it cannot be conclusive determined whether or not this anomaly is a function of random variability in the simulation or some other factor. If we treat this as random variability in the model and consider the entire "With Container Total" dataset for the first day, then the data appears to approach a steady state value of approximately 50 trucks. Regardless of this trend, we can approximate the termination point of the initial transient phase as approximately simulation time 300 minutes, indicating a start-up time-lag of 300 minutes.

Figure 87 shows the port truck n data for "On Roadway Total" port truck location values and "With Container Total" port truck utilization values for the second day of simulation. Recall that at the end of day one (simulation time 1440 minutes) the port container volume was increased. Note first that the "On Roadway Total" steady state

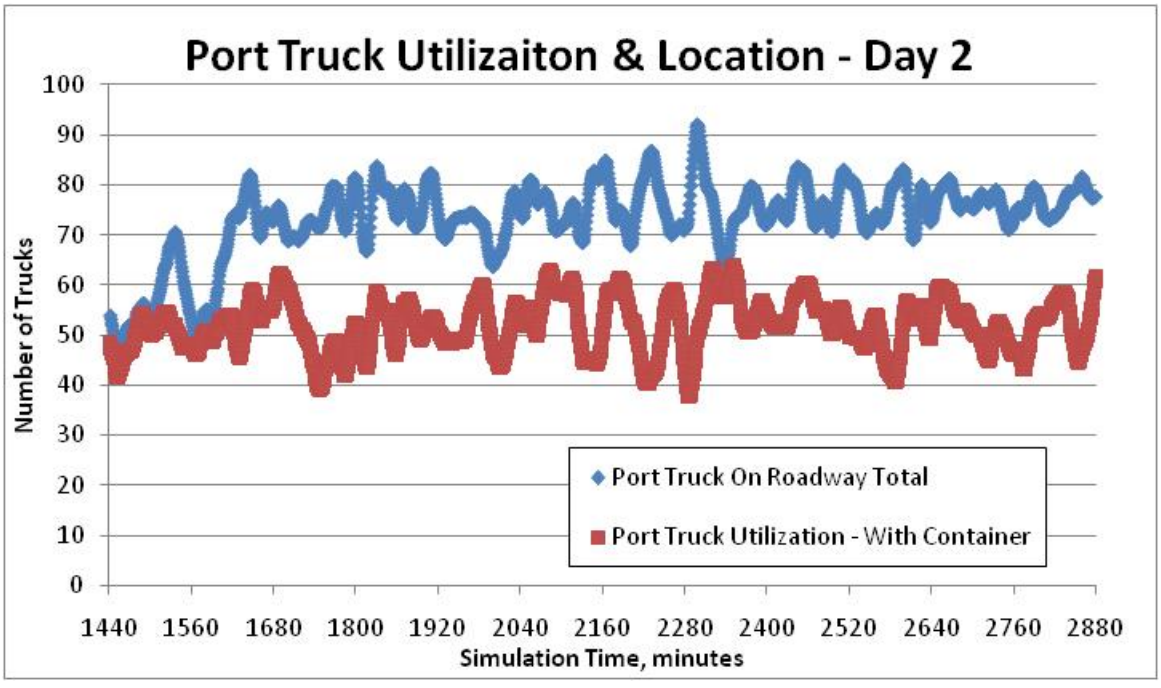



**Figure 87. Average Port Truck Location and Utilization (K=15), Day 2**

value increases to a value of approximately 75 trucks. The initial transient appears to terminate at 1660 minutes, indicating a time-lag of 220 minutes.

As the steady state analysis of "With Container Total" port truck utilization data was inconclusive for the previous day of simulation (see Figure 86), an initial transient phase for the second day of data cannot be determined. However, the second day of data does exhibit more conclusive steady state data, wherein the data appears to maintain a steady state value near 50 trucks. Therefore, this could suggest that an initial transient for this data does not in fact exist.

## 5.3 Discussion of Results

The results and analysis presented in the previous two sections identify several operational characteristics of port system as it has been modeled in this study. This section will briefly discuss several of these characteristics: (1) the effect of background traffic volume variations, (2) the effect of container volume variations, (3) the effect of limited port truck resources, and (4) general observations about initial transient phase and steady state behavior.

### 5.3.1 Effect of Background Traffic Volume Variations

The results suggest that background traffic and congestion levels can have a noticeable impact on the federated system, specifically the simulated port operations. Most notably, Figures 68 and 69 show that increasing background traffic volume can significantly increase travel times along southbound Highway 21 and northbound Dean Forest Rd. Recall from Figure 64 that this increase in travel times coincides with a sharp

increase in the port container queue that forms at the GCT. This suggests that port trucks are increasingly delayed by the higher volume of background traffic, and therefore cannot complete as many trips as are necessary to dissipate the GCT port container queue.

Additionally, recall from section 5.1.3 that both port and roadway truck utilization and location data exhibited an increase in the number of trucks on the roadway during days of higher traffic volume. This reinforces the suggestion that port and roadway trucks become increasingly delayed by roadway network congestion.

### 5.3.2 Effect of Container Volume Variations

Although increases in roadway network traffic volume appear to have greater impacts on the federated system's performance, variable container volumes do impact the simulation results to some extent. It has been noted earlier that the increased volume of containers arriving to the GCT equates to an increased demand for transport *away* from the GCT. However, as the inbound container volume generated at the I-16 Junction towards the GCT remains constant, there is no change in demand for container transport *to* the GCT. This is reflected in Figure 73 where an increase in the port container volume coincides with an increase in the number of port trucks on the roadway network, likely as a result of increased empty port truck rerouting back to the GCT. However, Figure 74 shows that only a minor increase in the number of port trucks on the roadway network carrying a container coincides with increased port container volume. This suggests that the port trucks were either at or near capacity in their ability to move containers away from the port, or that increases in container volume only have subtle impacts on port truck utilization.

The model results do not indicate that background traffic travel times are noticeably impacted by increased port container volume at the levels used in this study. Recall from Figure 68 that the increase in container volume coincides with a notable increase in southbound Highway 21 travel times. However, Figure 68 does not show a decrease in southbound Highway 21 travel times at time 4320 minutes when the container volume was decreased. This is likely due to significant port container queue accumulation at the GCT which delayed the decrease in container volume *to the roadway* until that queue either dissipated or reached some steady state value in day five. Conversely, background traffic travel times northbound on Highway 21 and both directions on Dean Forest Rd. appear unaffected by both the increase and decrease in port container volume over the course of the simulation.

### 5.3.3 Effect of Limited Port Truck Resources

During the first day of simulation, port truck utilization and location counts, port truck queues, and port container queues indicate that the initial supply of 150 port trucks is sufficient to meet the transport demand under the combined low traffic and low container volumes. Figure 64, showing port container and truck queues, indicates that the supply of 150 trucks may also be sufficient for the higher port container volume scenario when background traffic volumes are low. However, when background traffic volumes are increased, the accumulation of port containers at the GCT suggests that supply of 150 port trucks is insufficient to service the transport needs of the port, as modeled. There is an indication that this is caused by excessive truck delays in the roadway network, where higher background traffic volumes lead to increased network congestion. This higher

background traffic volume prevents the port trucks from completing the number of roundtrips necessary to service the demand of the port containers queued at the GCT, resulting in an accumulation in that queue.

Figure 64 also suggests that it is the compounded effect of higher background traffic volumes and increased container volume that causes the port container queue to accumulate at the GCT. Shortly after time 4320 minutes, when the container volume is reduced but the background traffic is still at the higher volume scenario, the port container queue at the GCT begins to dissipate. Therefore, there must be a relationship between the number of port trucks necessary and the combined effects of roadway network congestion and port container volume.

### 5.3.4 Steady State Behavior and Initial Transient Phases

The experiment presented in Chapter 4 and the results presented earlier in this chapter suggest that the federated simulation, as modeled, is capable of achieving steady state operation and importantly, capturing the impact of each model on the steady state of the other. All truck utilization and location, container and truck queue lengths, and travel time data exhibit steady state behavior at some point during the five days of simulation. Much of the truck utilization and location data after day two is inconclusive as to whether steady state operation was achieved. This suggests that in many cases, one day is not a sufficiently long time interval to allow the model to readjust to steady state operation after the input parameters changes used in this study. Similarly, queue length data indicates that the model is capable of steady state operation, but again that one day is not a sufficiently long time interval to readjust to steady state. This is evident in Figure 64

where it is inconclusive whether the port container queue achieves steady state in day two before its sharp increase in day three. Conversely, all travel time data exhibits steady state behavior across all five days of simulation. This suggests that one day is a sufficient time interval for travel times to readjust to steady state output.

The initial transient phases determined earlier in this chapter confirm that a time-lag can be defined between when input parameters are adjusted and when the effects of those adjustments are visible in the federated system. The queue length data suggests that the startup time-lag is greater for distribution centers submodels than for the GCT Gate submodel. This is likely explained by the additional time needed for the initial supply of trucks and containers to circulate and distribute throughout the federated system.

The travel time data and initial transient phase analysis suggest that there is a longer time-lag associated with federation start-up than following adjustments to input parameters during runtime. The data also suggest that once the federated simulation has initialized, time-lag for travel times associated with variations in background traffic volumes on the network are generally between 100 minutes and 300 minutes.

Truck utilization and location data indicates that a longer start-up time-lag is associated with port trucks and containers than with road trucks and containers. This can be explained by the added time necessary for arriving containers to process through the aggregated GCT submodels and arrive at the GCT Gate submodel. Road trucks and containers are simply created at the I-16 Junction submodel and immediately passed to the roadway network model federate with much less processing delay. Therefore, the start-up time-lag associated with road trucks and containers is comparably shorter. The

data is inconclusive as to the truck utilization and location time-lag associated with adjustments to model input parameters during runtime. Indeed, the only time-lag that could be determined is for changes to the port truck "On Roadway Total" location count in the second day following the increase in port container volume. Nonetheless, the time-lag associated with this change is comparably shorter than the start-up time-lag, as one would expect.

# CHAPTER 6

# CONCLUSION

This study has developed a federated simulation method to model the combined operations of a port system and roadway network using disparate modeling software packages. This federation method has been inspired by the High Level Architecture (HLA), acting as a guide to better inform the development process. The federated simulation method consists of four components: (1) a port model developed in Arena©, (2) a roadway network model developed in VISSIM©, (3) a runtime infrastructure (RTI) developed in Visual Studio 2005©, and (4) a federation database developed in Access 2003©. The federated simulation was then tested using a time-lag experiment to understand characteristics of the federated system's operations. This experiment also tests for the presence of feedback loops between federate model components wherein variations in input parameters of one federate model can be seen to affect system performance of the other.

This chapter discusses conclusions about the development and implementation of the federated simulation method. It then discusses the method's ability to capture the interaction between the modeled port components as seen in the experimental results. This section then discusses the challenges associated with federating two disparate transportation simulation platforms and proposes future research needs in this area.

## 6.1 Federated Transportation Simulation Development and Implementation

As has been discussed, the HLA rules have been used in this study to inform the development of the proposed federated simulation method. These include the use of an RTI for data, object, and time management, and to facilitate the interaction of federate model components. The HLA rules have provided a good platform for the federation of two disparate transportation simulators, and future efforts to develop and refine such federation methods should incorporate greater HLA compliance.

One of the primary challenges to federating two disparate transportation simulators is the issue of time management. As time is addressed differently in continuous time-stepped models (e.g., VISSIM©) and event-based models (e.g., Arena©), one key function of the federation is to reconcile these differences. This study has proposed one method to coordinate the runtime operation of two disparate transportation simulators by first advancing the continuous time-stepped model, pausing that model, and then allowing the event-based model to advance through as many steps as are necessary to "catch up." One challenge is that this method can create up to a one second time disparity in model federate simulation clocks. Depending on the required resolution of simulation results, this may or may not be problematic.

The second challenge has been to effectively exchange objects and data between model federates. This study has developed an RTI to facilitate this exchange and a federation database to maintain a time-stamped record of all transactions between model federates during simulation runtime.

## 6.2 Dynamic Interaction of Model Federates

One of the primary objectives of this study was to determine if the proposed federation method could establish feedback loops between two modeled systems and therefore capture the dynamic interaction between the two models during runtime. To accomplish this, an experiment was conducted wherein input parameters for both models were sequentially adjusted during runtime, and various output data were collected and analyzed.

The experiment has shown that adjustments to input parameters in one model federate can be seen in the performance output of the other model, and therefore that feedback loops exist between model federates. One example of this is the sharp increase in the port container queue length (a performance metric for the port model) as a result of an increase in the background traffic volume on the roadway network (an input parameter of the roadway network model). As second example is the slight increase in several background traffic travel times (a performance metric for the roadway network) as a result of an increase in the port container volume (an input parameter of the port model).

A secondary objective of this study was to identify characteristics of the operation of the federated system, as it has been modeled. To accomplish this, the experimental output data was analyzed to determine the presence and extent of steady state operation and initial transient phases, or time-lags, in the output data. This study has shown that the federated simulation is capable of steady state operation, but that variable time intervals are required to allow the federated system to adjust to steady state operation. Similarly, this study has shown that initial transient phases can be determined and that

there are time-lag constants associated with the changes in the output data following several changes to model input parameters.

### 6.3 Challenges and Future Research Needs

By developing a method for federating port and transportation simulators, this study has provided a foundation for future work in this area. This section outlines several of the challenges and related future research needs for the federation of transportation-related simulation.

First, future efforts should endeavor to make federations fully HLA compliant. As the HLA provides a recognized standard for federated simulation, incorporating greater compliance to the rules outlined therein will increase the flexibility and reusability of future federated simulations.

Throughout earlier discussions, several elements of the federated simulation have been identified as "hard coded" into the RTI and model components. Examples of this include the one second time-resolution of the federation's execution, and the 5000 second "look-back" time associated with the vehicle diffusion module. These hard coded values effectively limit the flexibility of the federated simulation. In the example of the one second time-resolution, this prevents modeling efforts that require either greater or lesser data resolution. Future work should avoid hard coded federation values, instead incorporating user-defined values, and thereby increasing the flexibility and reusability of model federates and the federation method.

Several model limitations were identified throughout this study, most notably the detection of exiting vehicles and diffused vehicles in the roadway network model

federate. Future work should investigate alternate methods for detecting port vehicles exiting the roadway network that avoid stop-controlled and single-lane exit links. Furthermore, the 5000 second delay associated with detecting a diffused vehicle in the roadway network model federate creates some data clarity issues. Future editions of VISSIM© may incorporate COM interface objects that allow the RTI to access vehicle diffusion information during runtime, however more immediate efforts should investigate diffused vehicle detection methods that minimize the delay between diffusion and detection.

One current practical limitation of the federation method proposed in this study is the amount of time required to execute one replication of federated simulation. Given the discussion of parallel and distributed computing in Chapter 2, this federation method is a good candidate for such computing methods. Distributing simulation model federates and federation components across either several computing platforms or several processers in a single computing platform could likely increase the execution time, and therefore practicality, of this federated simulation method.

Finally, many assumptions about the roadway network and port facility operations have been made during this study. For example, the traffic signal timing plans were developed that were tailored to the traffic demand modeled, but are not accurate reflections of conditions found in the field. Similarly, assumed truck quantities and container generation rates were chosen for this study that do not necessarily reflect the operational characteristics of the Port of Savannah. Future modeling efforts should incorporate design and operational parameters that more accurately reflect actual port and roadway operations.

# APPENDIX A

# RUNTIME INFRASTRUCTURE COMMAND CODE

```
Imports Microsoft.Office.Interop.Excel
Imports Microsoft.Office.Interop
Imports Microsoft.Vbe.Interop.Forms
Imports Microsoft.Office.Interop.Access
Imports VISSIM_COMSERVERLib
Imports Arena
Imports System.Text
Imports System.Net.Sockets
Imports System.Convert
Imports ADODB
Imports System.Data.Odbc

Public Class Form1
    '*************************************************************************************
    '************************   Define Programs   ***********************************

    'Define Program Directories
     Dim DirectoryArena As String = "C:\Documents and Settings\twall3\Desktop\Port
        Federation\Arena Working Models\Savannah Port\"
     Dim DirectoryVISSIM As String = "C:\Documents and Settings\twall3\Desktop\Port
        Federation\VISSIM Working Models\Port Network\Util Version 3 HV ContainerFirst\"
     Dim DirectoryAccess As String = "C:\Documents and Settings\twall3\Desktop\Port
        Federation\Access Working Database\"

    'Define Arena Program
    Dim ArenaSim As Arena.Application
    Dim ArenaModel As Arena.Model
    Dim ArenaLanguage As SIMAN

    'Define VISSIM Program
    Dim vissim As Vissim    'VISSIM model components
    Dim simulation As Simulation    'VISSIM simulation
    Dim vehicle As Vehicle  'VISSIM vehicle
    Dim links As Links  'Collection of links in VISSIM
    Dim detectors As Detectors 'Collection of detectors in VISSIM
    Dim detector As Detector 'Individual Detector

    'Define Access Program
    Dim StarfleetDataSet As StarfleetDataSet

    Dim ContainerTable As StarfleetDataSet.ContainerTableDataTable
    Dim ContainerTempTable As StarfleetDataSet.ContainerTableDataTable
     Dim ContainerAdapter As StarfleetDataSetTableAdapters.ContainerTableTableAdapter
    Dim ContainerTempRow As StarfleetDataSet.ContainerTableRow

    Dim IndexTable As StarfleetDataSet.IndexTableDataTable
    Dim IndexTempTable As StarfleetDataSet.IndexTableDataTable
    Dim IndexAdapter As StarfleetDataSetTableAdapters.IndexTableTableAdapter
    Dim IndexTempRow As StarfleetDataSet.IndexTableRow

    Dim VehicleTable As StarfleetDataSet.VehicleTableDataTable
    Dim VehicleTempTable As StarfleetDataSet.VehicleTableDataTable
     Dim VehicleAdapter As StarfleetDataSetTableAdapters.VehicleTableTableAdapter
    Dim VehicleTempRow As StarfleetDataSet.VehicleTableRow

    Dim DispersionTable As StarfleetDataSet.DispersionDataTable
    Dim DispersionRow As StarfleetDataSet.DispersionRow
    Dim DispersionTempRow As StarfleetDataSet.DispersionRow
    Dim DispersionTempTable As StarfleetDataSet.DispersionDataTable
      Dim DispersionAdapter As StarfleetDataSetTableAdapters.DispersionTableAdapter
```

```
    Dim ContainerLogTable As StarfleetDataSet.ContainerLogDataTable
      Dim ContainerLogAdapter As StarfleetDataSetTableAdapters.ContainerLogTableAdapter

    Dim VehicleLogTable As StarfleetDataSet.VehicleLogDataTable
    Dim VehicleLogTempTable As StarfleetDataSet.VehicleLogDataTable
    Dim VehicleLogTempRow As StarfleetDataSet.VehicleLogRow
     Dim VehicleLogAdapter As StarfleetDataSetTableAdapters.VehicleLogTableAdapter

    Dim UtilizationTable As StarfleetDataSet.UtilizationDataTable
      Dim UtilizationAdapter As StarfleetDataSetTableAdapters.UtilizationTableAdapter

    Dim QueuesTable As StarfleetDataSet.QueuesDataTable
    Dim QueuesAdapter As StarfleetDataSetTableAdapters.QueuesTableAdapter

'******************Assign Variables   ******************

    'Define variable values to be used in the program
    Dim Time As Long    'Time used in the Simulation Loop
    Dim DurationTime As Long
    Dim aTime As Long   'Arena Time
    Dim fTime As Long   'VISSIM Time / Federation Time
    Dim systemtime As Date

    'Define VISSIM objects
    Dim PortDetector As Detector
    Dim Dist1Detector As Detector
    Dim Dist2Detector As Detector
    Dim Dist3Detector As Detector
    Dim RoadDetector As Detector
    Dim portcount As Long
    Dim distcount As Long
    Dim roadcount As Long

    'Dim Port Veh/Cont Information Arrays
    Dim array5011 As Long
    Dim array5012 As Long
    Dim array5013 As Long
    Dim array5014 As Long
    Dim array5015 As Long
    Dim array5016 As Long
    Dim array5017 As Long
    Dim array5018 As Long
    Dim array5019 As Long
    Dim array5110 As Long
    Dim array5111 As Long

    'Dim Port Release Information Arrays
    Dim array9210 As Long
    Dim array9211 As Long
    Dim array9212 As Long

    'Dim Dist1 Veh/Cont Information Arrays
    Dim array5021 As Long
    Dim array5022 As Long
    Dim array5023 As Long
    Dim array5024 As Long
    Dim array5025 As Long
    Dim array5026 As Long
    Dim array5027 As Long
    Dim array5028 As Long
    Dim array5029 As Long
    Dim array5120 As Long
    Dim array5121 As Long
    Dim array5122 As Long

    'Dim Dist2 Veh/Cont Information Arrays
    Dim array5031 As Long
    Dim array5032 As Long
    Dim array5033 As Long
    Dim array5034 As Long
    Dim array5035 As Long
```

```
Dim array5036 As Long
Dim array5037 As Long
Dim array5038 As Long
Dim array5039 As Long
Dim array5130 As Long
Dim array5131 As Long
Dim array5132 As Long

'Dim Dist3 Veh/Cont Information Arrays
Dim array5041 As Long
Dim array5042 As Long
Dim array5043 As Long
Dim array5044 As Long
Dim array5045 As Long
Dim array5046 As Long
Dim array5047 As Long
Dim array5048 As Long
Dim array5049 As Long
Dim array5140 As Long
Dim array5141 As Long
Dim array5142 As Long

'Dim Road Veh/Cont Information Arrays
Dim array5065 As Long
Dim array5066 As Long
Dim array5067 As Long
Dim array5068 As Long
Dim array5069 As Long
Dim array5160 As Long

'Dim Dist 1 to Port Rerouting Arrays
Dim array9022 As Long
Dim array9023 As Long
Dim array9024 As Long

'Dim Dist 2 to Port Rerouting Arrays
Dim array9032 As Long
Dim array9033 As Long
Dim array9034 As Long

'Dim Dist 3 to Port Rerouting Arrays
Dim array9042 As Long
Dim array9043 As Long
Dim array9044 As Long

'Dim Road Empty Vehicle Creation Arrays
Dim array9110 As Long
Dim array9111 As Long
Dim array9112 As Long

'Dim Detector VEHICLEID Arrays
Dim array1010 As Long
Dim array2020 As Long
Dim array2030 As Long
Dim array2040 As Long
Dim array3060 As Long

Dim VehIDPort1 As Long
Dim VehIDPort2 As Long
Dim VehIDDist11 As Long
Dim VehIDDist12 As Long
Dim VehIDDist13 As Long
Dim VehIDDist21 As Long
Dim VehIDDist22 As Long
Dim VehIDDist23 As Long
Dim VehIDDist31 As Long
Dim VehIDDist32 As Long
Dim VehIDDist33 As Long
Dim VehIDRoad1 As Long
Dim VehIDRoadEmpty As Long
Dim VehIDPortRelease As Long
```

```
Dim PortContID As Long
Dim PortVehID As Long
Dim PortDestID As Long
Dim PortDestID2 As Long
Dim PortVehType As Long
Dim PortOriginID As Long

Dim Dist1ContID As Long
Dim Dist1VehID As Long
Dim Dist1DestID As Long
Dim Dist1DestID2 As Long
Dim Dist1VehType As Long
Dim Dist1OriginID As Long

Dim Dist2ContID As Long
Dim Dist2VehID As Long
Dim Dist2DestID As Long
Dim Dist2DestID2 As Long
Dim Dist2VehType As Long
Dim Dist2OriginID As Long

Dim Dist3ContID As Long
Dim Dist3VehID As Long
Dim Dist3DestID As Long
Dim Dist3DestID2 As Long
Dim Dist3VehType As Long
Dim Dist3OriginID As Long

Dim RoadContID As Long
Dim RoadVehID As Long
Dim RoadDestID As Long
Dim RoadDestID2 As Long
Dim RoadVehType As Long
Dim RoadOriginID As Long

Dim PortVehOrigin As Long
Dim Dist1VehOrigin As Long
Dim Dist2VehOrigin As Long
Dim Dist3VehOrigin As Long
Dim RoadVehOrigin As Long

Dim RouteIndex As Long
Dim RerouteIndex As Long

Dim dummy As Long
Dim countport As Long
Dim countd1 As Long
Dim countd2 As Long
Dim countd3 As Long

Dim DispersionSysTime As Long
Dim DispersionVehTime As Long
Dim DispersionContainer As Long
Dim DispersionVehicle As Long
Dim DispersionIndex As Long
Dim DispersionType As Long
Dim DispersionDestination As Long
Dim DispersionOrigin As Long
Dim DispersionArray() As Long
Dim i As Long

Dim PortUtil1 As Long
Dim PortUtil2 As Long
Dim PortUtil3 As Long
Dim RoadUtil1 As Long
Dim RoadUtil2 As Long
Dim RoadUtil3 As Long

Dim Dist1VehQp As Long
Dim Dist1ContQp As Long
```

```
    Dim Dist1VehQh As Long
    Dim Dist1ContQh As Long

    Dim Dist2VehQp As Long
    Dim Dist2ContQp As Long
    Dim Dist2VehQh As Long
    Dim Dist2ContQh As Long

    Dim Dist3VehQp As Long
    Dim Dist3ContQp As Long
    Dim Dist3VehQh As Long
    Dim Dist3ContQh As Long

    Dim gctVehQp As Long
    Dim gctContQp As Long
    Dim gctVehQh As Long
    Dim gctContQh As Long

    Dim RoadDispCount As Long
    Dim PortDispCount As Long

    Dim Dist1Delay1 As Long
    Dim Dist1Delay2 As Long
    Dim Dist1Delay3 As Long
    Dim Dist1Delay4 As Long

    Dim Dist2Delay1 As Long
    Dim Dist2Delay2 As Long
    Dim Dist2Delay3 As Long
    Dim Dist2Delay4 As Long

    Dim Dist3Delay1 As Long
    Dim Dist3Delay2 As Long
    Dim Dist3Delay3 As Long
    Dim Dist3Delay4 As Long

    Dim GCTDelay1 As Long
    Dim GCTDelay2 As Long
    Dim GCTDelay3 As Long


    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles Button1.Click
        '*********************************************************************
'***********************   OPEN PROGRAMS   *******************************

'Generate Access DB Components
StarfleetDataSet = New StarfleetDataSet
ContainerTable = New StarfleetDataSet.ContainerTableDataTable()
ContainerAdapter = New
        StarfleetDataSetTableAdapters.ContainerTableTableAdapter
IndexTable = New StarfleetDataSet.IndexTableDataTable()
IndexAdapter = New StarfleetDataSetTableAdapters.IndexTableTableAdapter
VehicleTable = New StarfleetDataSet.VehicleTableDataTable
VehicleAdapter = New StarfleetDataSetTableAdapters.VehicleTableTableAdapter
ContainerLogTable = New StarfleetDataSet.ContainerLogDataTable
ContainerLogAdapter = New
        StarfleetDataSetTableAdapters.ContainerLogTableAdapter
VehicleLogTable = New StarfleetDataSet.VehicleLogDataTable
VehicleLogAdapter = New StarfleetDataSetTableAdapters.VehicleLogTableAdapter
DispersionTable = New StarfleetDataSet.DispersionDataTable
DispersionAdapter = New StarfleetDataSetTableAdapters.DispersionTableAdapter

UtilizationTable = New StarfleetDataSet.UtilizationDataTable
UtilizationAdapter = New
        StarfleetDataSetTableAdapters.UtilizationTableAdapter

QueuesTable = New StarfleetDataSet.QueuesDataTable
QueuesAdapter = New StarfleetDataSetTableAdapters.QueuesTableAdapter
```

```
'Open Vissim File
vissim = CreateObject("vissim.vissim")
vissim.LoadNet(DirectoryVISSIM & "savannahRun1r2HV.inp")
vissim.LoadLayout(DirectoryVISSIM & "vissim.ini")

'Open Arena and Assign SIMAN operators
ArenaSim = CreateObject("Arena.Application")
ArenaModel = ArenaSim.Models.Open(DirectoryArena & "Three_Port_Model_v14r21
       HV Port First.doe")
ArenaLanguage = ArenaModel.SIMAN
DurationTime = 432000
ArenaModel.ReplicationLength = DurationTime


       '*************************************************************************
'***********************   Prepare Simulation  ***************************

'Assign simulation characteristics into VISSIM
simulation = vissim.Simulation
simulation.Period = DurationTime
simulation.Resolution = 1
simulation.Speed = 1000

'Reset simulation time values
aTime = 0   'Simulation Time in Arena
fTime = 0   'Simulation Time in VISSIM
portcount = 0
distcount = 0
roadcount = 0

'Assign VISSIM detectors - Detectors for traffic ENTERING each locatoin
detectors =
       vissim.Net.SignalControllers.GetSignalControllerByNumber(1).Detectors
Dist1Detector = detectors.GetDetectorByNumber(1)
Dist2Detector = detectors.GetDetectorByNumber(2)
Dist3Detector = detectors.GetDetectorByNumber(3)
RoadDetector = detectors.GetDetectorByNumber(6)
PortDetector = detectors.GetDetectorByNumber(7)

'*************************************************************************
'***********************   Simulation Runs *****************************

       '*************************************************************************
'***********************   Entities Exiting VISSIM  ************************

For Time = 1 To simulation.Period

       'Advancing VISSIM Model/Time
      simulation.RunSingleStep()    'Run a single step of VISSIM
      fTime = vissim.Simulation.AttValue("ELAPSEDTIME")   'Obtain the current
               simulation time in VISSIM

       'Check Vehicles Entering ARENA at PORT
      If PortDetector.AttValue("IMPULSE") = 1 Then   'causes following
               sequence if vehicle is present at detector
            array1010 = PortDetector.AttValue("VEHICLEID")  'identifies which
                     Vehicle from Index table is entering
            vissim.Net.Vehicles.RemoveVehicle(array1010)
            ArenaLanguage.VariableArrayValue(1010) = array1010  'writes
                     vehicle Index ID to Arena variable

            IndexTempTable = IndexAdapter.GetDataByIndexID(array1010)
            IndexTempRow = IndexTempTable.FindByindex_id(array1010)
            PortContID = IndexTempRow.container_id
            PortVehID = IndexTempRow.vehicle_id

            If PortContID = 0 Then
               VehicleTempTable =
                           VehicleAdapter.GetDataByVehicleID(PortVehID)
                  VehicleTempRow =
                           VehicleTempTable.FindByvehicle_id(PortVehID)
```

```
                PortVehType = VehicleTempRow.vehicle_type
                PortVehOrigin = VehicleTempRow.origin_id

                ArenaLanguage.VariableArrayValue(1012) = PortVehID
                ArenaLanguage.VariableArrayValue(1013) = PortVehType
                ArenaLanguage.VariableArrayValue(8010) = 1

                IndexAdapter.DeleteIndex(array1010)
                VehicleAdapter.UpdateVeh(PortVehID, PortVehType, 0,
                            PortVehOrigin, 0, 7, fTime, PortContID, PortVehID)
                VehicleLogAdapter.Insert(PortVehID, PortVehType, 0,
                            PortVehOrigin, 0, 7, fTime, PortContID, array1010)
          Else
                ContainerTempTable =
                            ContainerAdapter.GetDataByContainerID(PortContID)
                ContainerTempRow =
                            ContainerTempTable.FindBycontainer_id(PortContID)
                PortDestID = ContainerTempRow.destination_id
                PortDestID2 = ContainerTempRow.destination_id2
                PortOriginID = ContainerTempRow.origin_id

                VehicleTempTable =
                            VehicleAdapter.GetDataByVehicleID(PortVehID)
                VehicleTempRow =
                            VehicleTempTable.FindByvehicle_id(PortVehID)
                     PortVehType = VehicleTempRow.vehicle_type
                     PortVehOrigin = VehicleTempRow.origin_id

                     ArenaLanguage.VariableArrayValue(1011) = PortContID
                     ArenaLanguage.VariableArrayValue(1012) = PortVehID
                     ArenaLanguage.VariableArrayValue(1013) = PortVehType
                     ArenaLanguage.VariableArrayValue(1014) = PortDestID
                     ArenaLanguage.VariableArrayValue(1015) = PortDestID2
                     ArenaLanguage.VariableArrayValue(1016) = PortOriginID

                     ArenaLanguage.VariableArrayValue(8010) = 1       'triggers
                            vehicle creation in Arena
                ArenaLanguage.VariableArrayValue(8011) = 1

                IndexAdapter.DeleteIndex(array1010)
                ContainerAdapter.UpdateCont(PortContID, PortVehID,
                            PortOriginID, PortDestID, PortDestID2, 7, fTime,
                            PortContID)
                VehicleAdapter.UpdateVeh(PortVehID, PortVehType, 0,
                            PortVehOrigin, 0, 7, fTime, PortContID, PortVehID)
                ContainerAdapter.DeleteCont(PortContID)

                ContainerLogAdapter.Insert(PortContID, PortVehID,
                            PortOriginID, PortDestID, PortDestID2, 7, fTime,
                            array1010)
                VehicleLogAdapter.Insert(PortVehID, PortVehType, 0,
                            PortVehOrigin, 0, 7, fTime, PortContID, array1010)
           End If
    End If
  'Check Vehicles Entering ARENA at DIST CTR 1.
  If Dist1Detector.AttValue("IMPULSE") = 1 Then
    array2020 = Dist1Detector.AttValue("VEHICLEID")
         vissim.Net.Vehicles.RemoveVehicle(array2020)
         ArenaLanguage.VariableArrayValue(2020) = array2020

         IndexTempTable = IndexAdapter.GetDataByIndexID(array2020)
         IndexTempRow = IndexTempTable.FindByindex_id(array2020)
         Dist1ContID = IndexTempRow.container_id
         Dist1VehID = IndexTempRow.vehicle_id

         If Dist1ContID = 0 Then
            VehicleTempTable =
                            VehicleAdapter.GetDataByVehicleID(Dist1VehID)
               VehicleTempRow =
                            VehicleTempTable.FindByvehicle_id(Dist1VehID)
               Dist1VehType = VehicleTempRow.vehicle_type
```

```
            Dist1VehOrigin = VehicleTempRow.origin_id

            ArenaLanguage.VariableArrayValue(2022) = Dist1VehID
            ArenaLanguage.VariableArrayValue(2023) = Dist1VehType
            ArenaLanguage.VariableArrayValue(8020) = 1

            IndexAdapter.DeleteIndex(array2020)
            VehicleAdapter.UpdateVeh(Dist1VehID, Dist1VehType, 0,
                        Dist1VehOrigin, 0, 1, fTime, Dist1ContID, Dist1VehID)
            VehicleLogAdapter.Insert(Dist1VehID, Dist1VehType, 0,
                        Dist1VehOrigin, 0, 1, fTime, Dist1ContID, array2020)
        Else
                ContainerTempTable =
                        ContainerAdapter.GetDataByContainerID(Dist1ContID)
            ContainerTempRow =
                        ContainerTempTable.FindBycontainer_id(Dist1ContID)
            Dist1DestID = ContainerTempRow.destination_id
            Dist1DestID2 = ContainerTempRow.destination_id2
            Dist1OriginID = ContainerTempRow.origin_id

            VehicleTempTable =
                        VehicleAdapter.GetDataByVehicleID(Dist1VehID)
            VehicleTempRow =
                        VehicleTempTable.FindByvehicle_id(Dist1VehID)
            Dist1VehType = VehicleTempRow.vehicle_type
            Dist1VehOrigin = VehicleTempRow.origin_id

            ArenaLanguage.VariableArrayValue(2021) = Dist1ContID
            ArenaLanguage.VariableArrayValue(2022) = Dist1VehID
            ArenaLanguage.VariableArrayValue(2023) = Dist1VehType
            ArenaLanguage.VariableArrayValue(2024) = Dist1DestID
            ArenaLanguage.VariableArrayValue(2025) = Dist1DestID2
            ArenaLanguage.VariableArrayValue(2026) = Dist1OriginID

            ArenaLanguage.VariableArrayValue(8020) = 1
            ArenaLanguage.VariableArrayValue(8021) = 1

            IndexAdapter.DeleteIndex(array2020)
                If Dist1DestID2 = 1 Then
                        ContainerAdapter.DeleteCont(Dist1ContID)
                Else
                        ContainerAdapter.UpdateCont(Dist1ContID,
                                Dist1VehID, Dist1OriginID, Dist1DestID,
                                Dist1DestID2, 1, fTime, Dist1ContID)
                End If
                VehicleAdapter.UpdateVeh(Dist1VehID, Dist1VehType, 0,
                        Dist1VehOrigin, 0, 1, fTime, Dist1ContID, Dist1VehID)

            ContainerLogAdapter.Insert(Dist1ContID, Dist1VehID,
                        Dist1OriginID, Dist1DestID, Dist1DestID2, 1, fTime,
                        array2020)
            VehicleLogAdapter.Insert(Dist1VehID, Dist1VehType, 0,
                        Dist1VehOrigin, 0, 1, fTime, Dist1ContID, array2020)
        End If
  End If
'Check Vehicles Entering ARENA at DIST CTR 2.
If Dist2Detector.AttValue("IMPULSE") = 1 Then
  array2030 = Dist2Detector.AttValue("VEHICLEID")
      vissim.Net.Vehicles.RemoveVehicle(array2030)
      ArenaLanguage.VariableArrayValue(2030) = array2030

      IndexTempTable = IndexAdapter.GetDataByIndexID(array2030)
      IndexTempRow = IndexTempTable.FindByindex_id(array2030)
      Dist2ContID = IndexTempRow.container_id
      Dist2VehID = IndexTempRow.vehicle_id

      If Dist2ContID = 0 Then

                VehicleTempTable =
                        VehicleAdapter.GetDataByVehicleID(Dist2VehID)
            VehicleTempRow =
```

```
                    VehicleTempTable.FindByvehicle_id(Dist2VehID)
               Dist2VehType = VehicleTempRow.vehicle_type
               Dist2VehOrigin = VehicleTempRow.origin_id

               ArenaLanguage.VariableArrayValue(2032) = Dist2VehID
               ArenaLanguage.VariableArrayValue(2033) = Dist2VehType
               ArenaLanguage.VariableArrayValue(8030) = 1

               IndexAdapter.DeleteIndex(array2030)
               VehicleAdapter.UpdateVeh(Dist2VehID, Dist2VehType, 0,
                    Dist2VehOrigin, 0, 2, fTime, Dist2ContID, Dist2VehID)
               VehicleLogAdapter.Insert(Dist2VehID, Dist2VehType, 0,
                    Dist2VehOrigin, 0, 2, fTime, Dist2ContID, array2030)
          Else
          ContainerTempTable =
                    ContainerAdapter.GetDataByContainerID(Dist2ContID)
               ContainerTempRow =
                    ContainerTempTable.FindBycontainer_id(Dist2ContID)
               Dist2DestID = ContainerTempRow.destination_id
               Dist2DestID2 = ContainerTempRow.destination_id2
               Dist2OriginID = ContainerTempRow.origin_id

                    VehicleTempTable =
                    VehicleAdapter.GetDataByVehicleID(Dist2VehID)
               VehicleTempRow =
                    VehicleTempTable.FindByvehicle_id(Dist2VehID)
               Dist2VehType = VehicleTempRow.vehicle_type
               Dist2VehOrigin = VehicleTempRow.origin_id

               ArenaLanguage.VariableArrayValue(2031) = Dist2ContID
               ArenaLanguage.VariableArrayValue(2032) = Dist2VehID
               ArenaLanguage.VariableArrayValue(2033) = Dist2VehType
               ArenaLanguage.VariableArrayValue(2034) = Dist2DestID
               ArenaLanguage.VariableArrayValue(2035) = Dist2DestID2
               ArenaLanguage.VariableArrayValue(2036) = Dist2OriginID

               ArenaLanguage.VariableArrayValue(8030) = 1
               ArenaLanguage.VariableArrayValue(8031) = 1

               IndexAdapter.DeleteIndex(array2030)
               If Dist2DestID2 = 2 Then
                    ContainerAdapter.DeleteCont(Dist2ContID)
               Else
                    ContainerAdapter.UpdateCont(Dist2ContID, Dist2VehID,
                              Dist2OriginID, Dist2DestID, Dist2DestID2, 2,
                              fTime, Dist2ContID)
               End If
               VehicleAdapter.UpdateVeh(Dist2VehID, Dist2VehType, 0,
                    Dist2VehOrigin, 0, 2, fTime, Dist2ContID, Dist2VehID)
               ContainerLogAdapter.Insert(Dist2ContID, Dist2VehID,
                    Dist2OriginID, Dist2DestID, Dist2DestID2, 2, fTime,
                    array2030)
               VehicleLogAdapter.Insert(Dist2VehID, Dist2VehType, 0,
                    Dist2VehOrigin, 0, 2, fTime, Dist2ContID, array2030)
          End If
     End If
'Check Vehicles Entering ARENA at DIST CTR 3.
     If Dist3Detector.AttValue("IMPULSE") = 1 Then
     array2040 = Dist3Detector.AttValue("VEHICLEID")
          vissim.Net.Vehicles.RemoveVehicle(array2040)
          ArenaLanguage.VariableArrayValue(2040) = array2040

          IndexTempTable = IndexAdapter.GetDataByIndexID(array2040)
          IndexTempRow = IndexTempTable.FindByindex_id(array2040)
          Dist3ContID = IndexTempRow.container_id
          Dist3VehID = IndexTempRow.vehicle_id

          If Dist3ContID = 0 Then
               VehicleTempTable =
                    VehicleAdapter.GetDataByVehicleID(Dist3VehID)
               VehicleTempRow =
```

```
                    VehicleTempTable.FindByvehicle_id(Dist3VehID)
            Dist3VehType = VehicleTempRow.vehicle_type
            Dist3VehOrigin = VehicleTempRow.origin_id

                   ArenaLanguage.VariableArrayValue(2042) = Dist3VehID
            ArenaLanguage.VariableArrayValue(2043) = Dist3VehType
                   ArenaLanguage.VariableArrayValue(8040) = 1

            IndexAdapter.DeleteIndex(array2040)
            VehicleAdapter.UpdateVeh(Dist3VehID, Dist3VehType, 0,
                    Dist3VehOrigin, 0, 3, fTime, Dist3ContID, Dist3VehID)
            VehicleLogAdapter.Insert(Dist3VehID, Dist3VehType, 0,
                    Dist3VehOrigin, 0, 3, fTime, Dist3ContID, array2040)
            Else
            ContainerTempTable =
                    ContainerAdapter.GetDataByContainerID(Dist3ContID)
               ContainerTempRow =
                    ContainerTempTable.FindBycontainer_id(Dist3ContID)
               Dist3DestID = ContainerTempRow.destination_id
               Dist3DestID2 = ContainerTempRow.destination_id2
               Dist3OriginID = ContainerTempRow.origin_id

               VehicleTempTable =
                    VehicleAdapter.GetDataByVehicleID(Dist3VehID)
               VehicleTempRow =
                    VehicleTempTable.FindByvehicle_id(Dist3VehID)
               Dist3VehType = VehicleTempRow.vehicle_type
               Dist3VehOrigin = VehicleTempRow.origin_id

               ArenaLanguage.VariableArrayValue(2041) = Dist3ContID
               ArenaLanguage.VariableArrayValue(2042) = Dist3VehID
               ArenaLanguage.VariableArrayValue(2043) = Dist3VehType
               ArenaLanguage.VariableArrayValue(2044) = Dist3DestID
               ArenaLanguage.VariableArrayValue(2045) = Dist3DestID2
               ArenaLanguage.VariableArrayValue(2046) = Dist3OriginID

               ArenaLanguage.VariableArrayValue(8040) = 1
               ArenaLanguage.VariableArrayValue(8041) = 1

               IndexAdapter.DeleteIndex(array2040)
               If Dist3DestID2 = 3 Then
                    ContainerAdapter.DeleteCont(Dist3ContID)
               Else
                    ContainerAdapter.UpdateCont(Dist3ContID, Dist3VehID,
                                Dist3OriginID, Dist3DestID, Dist3DestID2, 3,
                                fTime, Dist3ContID)
                    End If
               VehicleAdapter.UpdateVeh(Dist3VehID, Dist3VehType, 0,
                    Dist3VehOrigin, 0, 3, fTime, Dist3ContID, Dist3VehID)
               ContainerLogAdapter.Insert(Dist3ContID, Dist3VehID,
                    Dist3OriginID, Dist3DestID, Dist3DestID2, 3, fTime,
                    array2040)
               VehicleLogAdapter.Insert(Dist3VehID, Dist3VehType, 0,
                    Dist3VehOrigin, 0, 3, fTime, Dist3ContID, array2040)
            End If
  End If
'Check Vehicles Entering Arena at ROADWAY.
If RoadDetector.AttValue("IMPULSE") = 1 Then
  array3060 = RoadDetector.AttValue("VEHICLEID")
       vissim.Net.Vehicles.RemoveVehicle(array3060)
       ArenaLanguage.VariableArrayValue(3060) = array3060

       IndexTempTable = IndexAdapter.GetDataByIndexID(array3060)
       IndexTempRow = IndexTempTable.FindByindex_id(array3060)
       RoadContID = IndexTempRow.container_id
       RoadVehID = IndexTempRow.vehicle_id

       If RoadContID = 0 Then
          VehicleTempTable =
                    VehicleAdapter.GetDataByVehicleID(RoadVehID)
          VehicleTempRow =
```

```
                    VehicleTempTable.FindByvehicle_id(RoadVehID)
               RoadVehType = VehicleTempRow.vehicle_type
               RoadVehOrigin = VehicleTempRow.origin_id

               ArenaLanguage.VariableArrayValue(3062) = RoadVehID
               ArenaLanguage.VariableArrayValue(3063) = RoadVehType
               ArenaLanguage.VariableArrayValue(8060) = 1

               IndexAdapter.DeleteIndex(array3060)
               VehicleAdapter.UpdateVeh(RoadVehID, RoadVehType, 0,
                    RoadVehOrigin, 0, 6, fTime, RoadContID, RoadVehID)
               VehicleAdapter.DeleteVeh(RoadVehID)
               VehicleLogAdapter.Insert(RoadVehID, RoadVehType, 0,
                    RoadVehOrigin, 0, 6, fTime, RoadContID, array3060)
          Else
          ContainerTempTable =
                    ContainerAdapter.GetDataByContainerID(RoadContID)
               ContainerTempRow =
                    ContainerTempTable.FindBycontainer_id(RoadContID)
               RoadDestID = ContainerTempRow.destination_id
               RoadDestID2 = ContainerTempRow.destination_id2
               RoadOriginID = ContainerTempRow.origin_id

               VehicleTempTable =
                    VehicleAdapter.GetDataByVehicleID(RoadVehID)

                   VehicleTempRow =
                    VehicleTempTable.FindByvehicle_id(RoadVehID)
               RoadVehType = VehicleTempRow.vehicle_type
               RoadVehOrigin = VehicleTempRow.origin_id

               ArenaLanguage.VariableArrayValue(3061) = RoadContID
               ArenaLanguage.VariableArrayValue(3062) = RoadVehID
                   ArenaLanguage.VariableArrayValue(3063) = RoadVehType
               ArenaLanguage.VariableArrayValue(3064) = RoadDestID
               ArenaLanguage.VariableArrayValue(3065) = RoadDestID2
               ArenaLanguage.VariableArrayValue(3066) = RoadOriginID

               ArenaLanguage.VariableArrayValue(8060) = 1
               ArenaLanguage.VariableArrayValue(8061) = 1

               IndexAdapter.DeleteIndex(array3060)
               ContainerAdapter.UpdateCont(RoadContID, RoadVehID,
                    RoadOriginID, RoadDestID, RoadDestID2, 6, fTime,
                    RoadContID)
               VehicleAdapter.UpdateVeh(RoadVehID, RoadVehType, 0,
                    RoadVehOrigin, 0, 6, fTime, RoadContID, RoadVehID)

               ContainerAdapter.DeleteCont(RoadContID)
               VehicleAdapter.DeleteVeh(RoadVehID)

               ContainerLogAdapter.Insert(RoadContID, RoadVehID,
                    RoadOriginID, RoadDestID, RoadDestID2, 6, fTime,
                    array3060)
               VehicleLogAdapter.Insert(RoadVehID, RoadVehType, 0,
                    RoadVehOrigin, 0, 6, fTime, RoadContID, array3060)
          End If
      End If

'Advancing Arena Time
While Int(aTime) <= fTime   'While Arena time is less than or equal to
          VISSIM Time
      ArenaModel.Step()                  'Step Arena
      aTime = ArenaLanguage.RunCurrentTime    'Obtain the current
                simulation time in Arena
End While

If fTime = i * 60 Then

      Dist1VehQp = ArenaLanguage.QueueNumberOfEntities(100)
          Dist1ContQp = ArenaLanguage.QueueNumberOfEntities(101)
```

```
        Dist1VehQh = ArenaLanguage.QueueNumberOfEntities(102)
        Dist1ContQh = ArenaLanguage.QueueNumberOfEntities(103)
        Dist2VehQp = ArenaLanguage.QueueNumberOfEntities(200)
        Dist2ContQp = ArenaLanguage.QueueNumberOfEntities(201)
        Dist2VehQh = ArenaLanguage.QueueNumberOfEntities(202)
        Dist2ContQh = ArenaLanguage.QueueNumberOfEntities(203)
        Dist3VehQp = ArenaLanguage.QueueNumberOfEntities(300)
        Dist3ContQp = ArenaLanguage.QueueNumberOfEntities(301)
        Dist3VehQh = ArenaLanguage.QueueNumberOfEntities(302)
        Dist3ContQh = ArenaLanguage.QueueNumberOfEntities(303)
        gctVehQp = ArenaLanguage.QueueNumberOfEntities(700)
        gctContQp = ArenaLanguage.QueueNumberOfEntities(701)
        gctVehQh = ArenaLanguage.QueueNumberOfEntities(702)
        gctContQh = ArenaLanguage.QueueNumberOfEntities(703)

        QueuesAdapter.Insert(11, i, Dist1VehQp)
        QueuesAdapter.Insert(12, i, Dist1ContQp)
        QueuesAdapter.Insert(13, i, Dist1VehQh)
        QueuesAdapter.Insert(14, i, Dist1ContQh)
        QueuesAdapter.Insert(21, i, Dist2VehQp)
        QueuesAdapter.Insert(22, i, Dist2ContQp)
        QueuesAdapter.Insert(23, i, Dist2VehQh)
        QueuesAdapter.Insert(24, i, Dist2ContQh)
        QueuesAdapter.Insert(31, i, Dist3VehQp)
        QueuesAdapter.Insert(32, i, Dist3ContQp)
        QueuesAdapter.Insert(33, i, Dist3VehQh)
        QueuesAdapter.Insert(34, i, Dist3ContQh)
        QueuesAdapter.Insert(71, i, gctVehQp)
        QueuesAdapter.Insert(72, i, gctContQp)
        QueuesAdapter.Insert(73, i, gctVehQh)
        QueuesAdapter.Insert(74, i, gctContQh)

        i = i + 1
    End If

'*********************************************
'Check for Diffused (Deleted) Vehicles

If fTime > 5000 Then

    DispersionSysTime = fTime - 5000
        IndexTempTable = IndexAdapter.GetData
        DispersionAdapter.InsertQuery()
        DispersionTempTable = DispersionAdapter.GetData
        DispersionTempRow = DispersionTempTable.FindBydisp_index(1)
        DispersionVehTime = DispersionTempRow.time_stamp

        If DispersionVehTime < DispersionSysTime Then
           DispersionIndex = DispersionTempRow.index_id
              DispersionVehicle = DispersionTempRow.vehicle_id
              DispersionContainer = DispersionTempRow.container_id

              VehicleTempTable =
                          VehicleAdapter.GetDataByVehicleID(DispersionVehicle)
              VehicleTempRow =
                          VehicleTempTable.FindByvehicle_id(DispersionVehicle)

              DispersionType = VehicleTempRow.vehicle_type
              DispersionDestination = VehicleTempRow.current_destination
              DispersionOrigin = VehicleTempRow.origin_id

              ArenaLanguage.VariableArrayValue(8082) = DispersionVehicle
              ArenaLanguage.VariableArrayValue(8083) = DispersionType
              ArenaLanguage.VariableArrayValue(8084) =
                          DispersionDestination
              ArenaLanguage.VariableArrayValue(8080) = 1

              IndexAdapter.DeleteIndex(DispersionIndex)

              If DispersionDestination = 6 Then
                   If DispersionContainer = 0 Then
```

```
                              VehicleAdapter.DeleteVeh(DispersionVehicle)
                              VehicleLogAdapter.Insert(DispersionVehicle,
                                           DispersionType, 0, 7777, 0,
                                           DispersionDestination, fTime,
                                           DispersionContainer, DispersionIndex)
                        Else
                              ArenaLanguage.VariableArrayValue(8081) = 1
                              VehicleAdapter.DeleteVeh(DispersionVehicle)
                              VehicleLogAdapter.Insert(DispersionVehicle,
                                           DispersionType, 0, 7777, 0,
                                           DispersionDestination, fTime,
                                           DispersionContainer, DispersionIndex)

                                    ContainerAdapter.DeleteCont(DispersionContainer
                                           )

                              ContainerLogAdapter.Insert(DispersionContainer,
                                           DispersionVehicle, 7777, 0, 0,
                                           DispersionDestination, fTime,
                                           DispersionIndex)
                     End If
                     Else
                     If DispersionContainer = 0 Then
                              VehicleAdapter.UpdateVeh(DispersionVehicle,
                                           DispersionType, 0, DispersionOrigin, 0,
                                           DispersionDestination, fTime,
                                           DispersionContainer, DispersionVehicle)
                              VehicleLogAdapter.Insert(DispersionVehicle,
                                           DispersionType, 0, 7777, 0,
                                           DispersionDestination, fTime,
                                           DispersionContainer, DispersionIndex)
                        Else
                              ArenaLanguage.VariableArrayValue(8081) = 1
                              VehicleAdapter.UpdateVeh(DispersionVehicle,
                                           DispersionType, 0, DispersionOrigin, 0,
                                           DispersionDestination, fTime,
                                           DispersionContainer, DispersionVehicle)
                                    VehicleLogAdapter.Insert(DispersionVehicle,
                                           DispersionType, 0, 7777, 0,
                                           DispersionDestination, fTime,
                                           DispersionContainer, DispersionIndex)

                                    ContainerAdapter.DeleteCont(DispersionContainer
                                           )
                              ContainerLogAdapter.Insert(DispersionContainer,
                                           DispersionVehicle, 7777, 0, 0,
                                           DispersionDestination, fTime,
                                           DispersionIndex)
                     End If
                End If
                End If
                 DispersionAdapter.DeleteDispersionAll()
            Else
                 DispersionAdapter.DeleteDispersionAll()
       End If
    '*************************************************************************
'************************   Entities Exiting Arena to VISSIM ****************

'************************   Exiting Port to VISSIM **************************
'********  Other  ********
       If ArenaLanguage.VariableArrayValue(5012) > 0 Then
       array5011 = ArenaLanguage.VariableArrayValue(5011) 'other
                     container id

            array5012 = ArenaLanguage.VariableArrayValue(5012) 'other vehicle
                     id
            array5013 = ArenaLanguage.VariableArrayValue(5013) 'port vehicle
                     type
            array5014 = ArenaLanguage.VariableArrayValue(5014) 'other
                     destination id 1
            array5019 = ArenaLanguage.VariableArrayValue(5019) 'other
```

```
                destination id 2
        array5111 = ArenaLanguage.VariableArrayValue(5111) 'other origin
                id

        ArenaLanguage.VariableArrayValue(5012) = 0
        ContainerTempTable = ContainerAdapter.GetData
        If ContainerTempTable.Rows.Contains(array5011) = True Then
           ContainerAdapter.UpdateCont(array5011, array5012,
                        array5111, array5014, array5019, 9999, aTime,
                        array5011)
        Else
           ContainerAdapter.Insert(array5011, array5012, array5111,
                        array5014, array5019, 9999, aTime)
        End If
        VehicleTempTable = VehicleAdapter.GetData
        If VehicleTempTable.Rows.Contains(array5012) = True Then
           VehicleAdapter.UpdateVeh(array5012, array5013, 0, 7,
                        array5014, 9999, aTime, array5011, array5012)
           Else
           VehicleAdapter.Insert(array5012, array5013, 0, 7,
                        array5014, 9999, aTime, array5011)
        End If
        vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5014,
                50, 50, 1, 0)
        VehIDPort1 = vehicle.ID
        IndexAdapter.Insert(VehIDPort1, array5011, array5012, aTime, 1)


        ContainerLogAdapter.Insert(array5011, array5012, array5111,
                array5014, array5019, 9999, aTime, VehIDPort1)
        VehicleLogAdapter.Insert(array5012, array5013, 0, 7, array5014,
                9999, aTime, array5011, VehIDPort1)
    End If

'********  Road  ********
If ArenaLanguage.VariableArrayValue(5016) > 0 Then
    array5015 = ArenaLanguage.VariableArrayValue(5015) 'road
                container id
           array5016 = ArenaLanguage.VariableArrayValue(5016) 'road vehicle
                id
           array5017 = ArenaLanguage.VariableArrayValue(5017) 'road vehicle
                type
        array5018 = ArenaLanguage.VariableArrayValue(5018) 'road
                destination id
        array5110 = ArenaLanguage.VariableArrayValue(5110) 'road origin
                id

        ArenaLanguage.VariableArrayValue(5016) = 0
        ContainerTempTable = ContainerAdapter.GetData
        If ContainerTempTable.Rows.Contains(array5015) = True Then
           ContainerAdapter.UpdateCont(array5015, array5016,
                        array5110, array5018, array5018, 9999, aTime,
                        array5015)
        Else
           ContainerAdapter.Insert(array5015, array5016, array5110,
                        array5018, array5018, 9999, aTime)
        End If
        VehicleTempTable = VehicleAdapter.GetData
        If VehicleTempTable.Rows.Contains(array5016) = True Then
           VehicleAdapter.UpdateVeh(array5016, array5017, 0, 7,
                        array5018, 9999, aTime, array5015, array5016)
        Else
           VehicleAdapter.Insert(array5016, array5017, 0, 7,
                        array5018, 9999, aTime, array5015)
        End If
        vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5018,
                50, 50, 1, 0)
        VehIDPort2 = vehicle.ID
        IndexAdapter.Insert(VehIDPort2, array5015, array5016, aTime, 1)
```

```
      ContainerLogAdapter.Insert(array5015, array5016, array5110,
                array5018, array5018, 9999, aTime, VehIDPort2)
      VehicleLogAdapter.Insert(array5016, array5017, 0, 7, array5018,
                9999, aTime, array5015, VehIDPort2)
 End If

'********  Road Release ********
If ArenaLanguage.VariableArrayValue(9210) > 0 Then
 array9210 = ArenaLanguage.VariableArrayValue(9210) 'reroute
                vehicle id
      array9211 = ArenaLanguage.VariableArrayValue(9211) 'reroute
                vehicle type
      array9212 = ArenaLanguage.VariableArrayValue(9212) 'reroute
                destination
      ArenaLanguage.VariableArrayValue(9210) = 0
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array9210) = True Then
         VehicleAdapter.UpdateVeh(array9210, array9211, 0, 7,
                        array9212, 9999, aTime, 0, array9210)
      Else
         VehicleAdapter.Insert(array9210, array9211, 0, 7,
                        array9212, 9999, aTime, 0)
      End If

      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array9212, 50,
                        50, 1, 0)
 VehIDPortRelease = vehicle.ID
      IndexAdapter.Insert(VehIDPortRelease, 0, array9210, aTime, 1)

      VehicleLogAdapter.Insert(array9210, array9211, 0, 7, array9212,
                9999, aTime, 0, VehIDPortRelease)
 End If
'************************   Exiting Dist Ctr 1 to VISSIM  *************
'********  Dist 1 Other  ********
If ArenaLanguage.VariableArrayValue(5022) > 0 Then
 array5021 = ArenaLanguage.VariableArrayValue(5021) 'other
                container id
      array5022 = ArenaLanguage.VariableArrayValue(5022) 'other vehicle
                id
      array5023 = ArenaLanguage.VariableArrayValue(5023) 'port vehicle
                type
      array5024 = ArenaLanguage.VariableArrayValue(5024) 'other
                destination id 1
      array5029 = ArenaLanguage.VariableArrayValue(5029) 'other
                destination id 2
      array5121 = ArenaLanguage.VariableArrayValue(5121) 'other origin
                id

      ArenaLanguage.VariableArrayValue(5022) = 0
      ContainerTempTable = ContainerAdapter.GetData
      If ContainerTempTable.Rows.Contains(array5021) = True Then
         ContainerAdapter.UpdateCont(array5021, array5022,
                        array5121, array5024, array5029, 9999, aTime,
                        array5021)
      Else
         ContainerAdapter.Insert(array5021, array5022, array5121,
                        array5024, array5029, 9999, aTime)
      End If
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array5022) = True Then
         VehicleAdapter.UpdateVeh(array5022, array5023, 0, 1,
                        array5029, 9999, aTime, array5021, array5022)
      Else
         VehicleAdapter.Insert(array5022, array5023, 0, 1,
                array5029, 9999, aTime, array5021)
      End If
      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5029,
                50, 39, 1, 0)
```

```
      VehIDDist11 = vehicle.ID
      IndexAdapter.Insert(VehIDDist11, array5021, array5022, aTime, 1)

      ContainerLogAdapter.Insert(array5021, array5022, array5121,
                array5024, array5029, 9999, aTime, VehIDDist11)
      VehicleLogAdapter.Insert(array5022, array5023, 0, 1, array5029,
                9999, aTime, array5021, VehIDDist11)
 End If

'********  Dist 1 Road  ********
If ArenaLanguage.VariableArrayValue(5026) > 0 Then
 array5025 = ArenaLanguage.VariableArrayValue(5025) 'road
                container id
      array5026 = ArenaLanguage.VariableArrayValue(5026) 'road vehicle
                id
      array5027 = ArenaLanguage.VariableArrayValue(5027) 'road vehicle
                type
      array5028 = ArenaLanguage.VariableArrayValue(5028) 'road
                destination id 1
      array5122 = ArenaLanguage.VariableArrayValue(5122) 'road
                destination id 2
      array5120 = ArenaLanguage.VariableArrayValue(5120) 'road origin
                id

      ArenaLanguage.VariableArrayValue(5026) = 0
      ContainerTempTable = ContainerAdapter.GetData
      If ContainerTempTable.Rows.Contains(array5025) = True Then
         ContainerAdapter.UpdateCont(array5025, array5026,
                        array5120, array5028, array5122, 9999, aTime,
                        array5025)
      Else
         ContainerAdapter.Insert(array5025, array5026, array5120,
                        array5028, array5122, 9999, aTime)
 End If
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array5026) = True Then
         VehicleAdapter.UpdateVeh(array5026, array5027, 0, 1,
                        array5122, 9999, aTime, array5025, array5026)
      Else
         VehicleAdapter.Insert(array5026, array5027, 0, 1,
                        array5122, 9999, aTime, array5025)
      End If
      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5122,
                50, 39, 1, 0)
         VehIDDist12 = vehicle.ID
      IndexAdapter.Insert(VehIDDist12, array5025, array5026, aTime, 1)

      ContainerLogAdapter.Insert(array5025, array5026, array5120,
                array5028, array5122, 9999, aTime, VehIDDist12)
         VehicleLogAdapter.Insert(array5026, array5027, 0, 1, array5122,
                9999, aTime, array5025, VehIDDist12)
 End If

'********  To Dist 1 ReRoute ********
If ArenaLanguage.VariableArrayValue(9022) > 0 Then
 array9022 = ArenaLanguage.VariableArrayValue(9022) 'reroute
                vehicle id
      array9023 = ArenaLanguage.VariableArrayValue(9023) 'reroute
                vehicle type
         array9024 = ArenaLanguage.VariableArrayValue(9024) 'reroute
                destination
      ArenaLanguage.VariableArrayValue(9022) = 0
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array9022) = True Then
         VehicleAdapter.UpdateVeh(array9022, array9023, 0, 1,
                        array9024, 9999, aTime, 0, array9022)
      Else
         VehicleAdapter.Insert(array9022, array9023, 0, 1,
                        array9024, 9999, aTime, 0)
         End If
```

```
      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array9024,
                50, 39, 1, 0)
      VehIDDist13 = vehicle.ID
      IndexAdapter.Insert(VehIDDist13, 0, array9022, aTime, 1)
      RerouteIndex = 0

      VehicleLogAdapter.Insert(array9022, array9023, 0, 1, array9024,
                9999, aTime, 0, VehIDDist13)
 End If

'************************   Exiting Dist Ctr 2 to VISSIM  ************
'********  Dist 2 Other  ********
If ArenaLanguage.VariableArrayValue(5032) > 0 Then
 array5031 = ArenaLanguage.VariableArrayValue(5031) 'other
                container id
      array5032 = ArenaLanguage.VariableArrayValue(5032) 'other vehicle
                id
      array5033 = ArenaLanguage.VariableArrayValue(5033) 'port vehicle
                type
      array5034 = ArenaLanguage.VariableArrayValue(5034) 'other
                destination id 1
      array5039 = ArenaLanguage.VariableArrayValue(5039) 'other
                destination id 2
      array5131 = ArenaLanguage.VariableArrayValue(5131) 'other origin
                id

      ArenaLanguage.VariableArrayValue(5032) = 0
      ContainerTempTable = ContainerAdapter.GetData
      If ContainerTempTable.Rows.Contains(array5031) = True Then
         ContainerAdapter.UpdateCont(array5031, array5032,
                   array5131, array5034, array5039, 9999, aTime,
                   array5031)
      Else
         ContainerAdapter.Insert(array5031, array5032, array5131,
                   array5034, array5039, 9999, aTime)
      End If
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array5032) = True Then
         VehicleAdapter.UpdateVeh(array5032, array5033, 0, 2,
                   array5039, 9999, aTime, array5031, array5032)
      Else
      VehicleAdapter.Insert(array5032, array5033, 0, 2, array5039,
                9999, aTime, array5031)
      End If
      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5039,
                50, 29, 1, 0)
      VehIDDist21 = vehicle.ID
      IndexAdapter.Insert(VehIDDist21, array5031, array5032, aTime, 1)

      ContainerLogAdapter.Insert(array5031, array5032, array5131,
                array5034, array5039, 9999, aTime, VehIDDist21)
      VehicleLogAdapter.Insert(array5032, array5033, 0, 2, array5039,
                9999, aTime, array5031, VehIDDist21)
 End If

'********  Dist 2 Road  ********
If ArenaLanguage.VariableArrayValue(5036) > 0 Then
 array5035 = ArenaLanguage.VariableArrayValue(5035) 'road
                container id
      array5036 = ArenaLanguage.VariableArrayValue(5036) 'road vehicle
                id
      array5037 = ArenaLanguage.VariableArrayValue(5037) 'road vehicle
                type
      array5038 = ArenaLanguage.VariableArrayValue(5038) 'road
                destination id 1
      array5132 = ArenaLanguage.VariableArrayValue(5132) 'road
                destination id 2
      array5130 = ArenaLanguage.VariableArrayValue(5130) 'road origin
                id
```

```
      ArenaLanguage.VariableArrayValue(5036) = 0
      ContainerTempTable = ContainerAdapter.GetData
      If ContainerTempTable.Rows.Contains(array5035) = True Then
                ContainerAdapter.UpdateCont(array5035, array5036,
                        array5130, array5038, array5132, 9999, aTime,
                        array5035)
      Else
         ContainerAdapter.Insert(array5035, array5036, array5130,
                        array5038, array5132, 9999, aTime)
      End If
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array5036) = True Then
         VehicleAdapter.UpdateVeh(array5036, array5037, 0, 2,
                        array5132, 9999, aTime, array5035, array5036)
      Else
         VehicleAdapter.Insert(array5036, array5037, 0, 2,
                        array5132, 9999, aTime, array5035)
      End If
      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5132,
                50, 29, 1, 0)
      VehIDDist22 = vehicle.ID
      IndexAdapter.Insert(VehIDDist22, array5035, array5036, aTime, 1)

      ContainerLogAdapter.Insert(array5035, array5036, array5130,
                array5038, array5132, 9999, aTime, VehIDDist22)
      VehicleLogAdapter.Insert(array5036, array5037, 0, 2, array5132,
                9999, aTime, array5035, VehIDDist22)
 End If

'********  To Dist 2 ReRoute ********
If ArenaLanguage.VariableArrayValue(9032) > 0 Then
 array9032 = ArenaLanguage.VariableArrayValue(9032) 'reroute
                vehicle id
      array9033 = ArenaLanguage.VariableArrayValue(9033) 'reroute
                vehicle type
      array9034 = ArenaLanguage.VariableArrayValue(9034) 'reroute
                destination
      ArenaLanguage.VariableArrayValue(9032) = 0
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array9032) = True Then
         VehicleAdapter.UpdateVeh(array9032, array9033, 0, 2,
                        array9034, 9999, aTime, 0, array9032)
      Else
                VehicleAdapter.Insert(array9032, array9033, 0, 2,
                        array9034, 9999, aTime, 0)
      End If

      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array9034,
                50, 29, 1, 0)
      VehIDDist23 = vehicle.ID
      IndexAdapter.Insert(VehIDDist23, 0, array9032, aTime, 1)
      RerouteIndex = 0

      VehicleLogAdapter.Insert(array9032, array9033, 0, 2, array9034,
                9999, aTime, 0, VehIDDist23)
 End If
'************************   Exiting Dist Ctr 3 to VISSIM  ***********
'********  Dist 3 Other  ********
If ArenaLanguage.VariableArrayValue(5042) > 0 Then
 array5041 = ArenaLanguage.VariableArrayValue(5041) 'other
                container id
      array5042 = ArenaLanguage.VariableArrayValue(5042) 'other vehicle
                id
      array5043 = ArenaLanguage.VariableArrayValue(5043) 'port vehicle
                type
      array5044 = ArenaLanguage.VariableArrayValue(5044) 'other
                destination id 1
      array5049 = ArenaLanguage.VariableArrayValue(5049) 'other
```

```
                destination id 2
        array5141 = ArenaLanguage.VariableArrayValue(5141) 'other origin
                id

        ArenaLanguage.VariableArrayValue(5042) = 0
        ContainerTempTable = ContainerAdapter.GetData
        If ContainerTempTable.Rows.Contains(array5041) = True Then
           ContainerAdapter.UpdateCont(array5041, array5042,
                        array5141, array5044, array5049, 9999, aTime,
                        array5041)
        Else
              ContainerAdapter.Insert(array5041, array5042, array5141,
                        array5044, array5049, 9999, aTime)
        End If
        VehicleTempTable = VehicleAdapter.GetData
        If VehicleTempTable.Rows.Contains(array5042) = True Then
           VehicleAdapter.UpdateVeh(array5042, array5043, 0, 3,
                        array5049, 9999, aTime, array5041, array5042)
        Else
           VehicleAdapter.Insert(array5042, array5043, 0, 3,
                        array5049, 9999, aTime, array5041)
        End If
        vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5049,
                50, 77, 1, 0)
        VehIDDist31 = vehicle.ID
        IndexAdapter.Insert(VehIDDist31, array5041, array5042, aTime, 1)
        ContainerLogAdapter.Insert(array5041, array5042, array5141,
                array5044, array5049, 9999, aTime, VehIDDist31)
        VehicleLogAdapter.Insert(array5042, array5043, 0, 3, array5049,
                9999, aTime, array5041, VehIDDist31)
    End If

    '********  Dist 3 Road  ********
    If ArenaLanguage.VariableArrayValue(5046) > 0 Then
      array5045 = ArenaLanguage.VariableArrayValue(5045) 'road
                container id
        array5046 = ArenaLanguage.VariableArrayValue(5046) 'road vehicle
                id
        array5047 = ArenaLanguage.VariableArrayValue(5047) 'road vehicle
                type
        array5048 = ArenaLanguage.VariableArrayValue(5048) 'road
                destination id 1
        array5142 = ArenaLanguage.VariableArrayValue(5142) 'road
                destination id 2
        array5140 = ArenaLanguage.VariableArrayValue(5140) 'road origin
                id

           ArenaLanguage.VariableArrayValue(5046) = 0
        ContainerTempTable = ContainerAdapter.GetData
        If ContainerTempTable.Rows.Contains(array5045) = True Then
           ContainerAdapter.UpdateCont(array5045, array5046,
                        array5140, array5048, array5142, 9999, aTime,
                        array5045)
        Else
           ContainerAdapter.Insert(array5045, array5046, array5140,
                        array5048, array5142, 9999, aTime)
        End If
        VehicleTempTable = VehicleAdapter.GetData
        If VehicleTempTable.Rows.Contains(array5046) = True Then
           VehicleAdapter.UpdateVeh(array5046, array5047, 0, 3,
                        array5142, 9999, aTime, array5045, array5046)
        Else
           VehicleAdapter.Insert(array5046, array5047, 0, 3,
                        array5142, 9999, aTime, array5045)
        End If
        vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5142,
                50, 77, 1, 0)
        VehIDDist32 = vehicle.ID
        IndexAdapter.Insert(VehIDDist32, array5045, array5046, aTime, 1)
```

```
      ContainerLogAdapter.Insert(array5045, array5046, array5140,
                array5048, array5142, 9999, aTime, VehIDDist32)
      VehicleLogAdapter.Insert(array5046, array5047, 0, 3, array5142,
                9999, aTime, array5045, VehIDDist32)
  End If

'********  To Dist 3 ReRoute ********
If ArenaLanguage.VariableArrayValue(9042) > 0 Then
  array9042 = ArenaLanguage.VariableArrayValue(9042) 'reroute
                vehicle id
      array9043 = ArenaLanguage.VariableArrayValue(9043) 'reroute
                vehicle type
      array9044 = ArenaLanguage.VariableArrayValue(9044) 'reroute
                destination
      ArenaLanguage.VariableArrayValue(9042) = 0
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array9042) = True Then
         VehicleAdapter.UpdateVeh(array9042, array9043, 0, 3,
                        array9044, 9999, aTime, 0, array9042)
      Else
         VehicleAdapter.Insert(array9042, array9043, 0, 3,
                        array9044, 9999, aTime, 0)
      End If

      vehicle =
                vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array9044,
                50, 77, 1, 0)
         VehIDDist33 = vehicle.ID
      IndexAdapter.Insert(VehIDDist33, 0, array9042, aTime, 1)
      RerouteIndex = 0

      VehicleLogAdapter.Insert(array9042, array9043, 0, 3, array9044,
                9999, aTime, 0, VehIDDist33)
  End If
'************************   Exiting Road to VISSIM *************
'********  Road  ********
If ArenaLanguage.VariableArrayValue(5066) > 0 Then
  array5065 = ArenaLanguage.VariableArrayValue(5065) 'road
                container id
         array5066 = ArenaLanguage.VariableArrayValue(5066) 'road vehicle
                id
      array5067 = ArenaLanguage.VariableArrayValue(5067) 'road vehicle
                type
      array5068 = ArenaLanguage.VariableArrayValue(5068) 'road
                destination id 1
      array5069 = ArenaLanguage.VariableArrayValue(5069) 'road
                destination id 2
      array5160 = ArenaLanguage.VariableArrayValue(5160) 'road origin
                id

         ArenaLanguage.VariableArrayValue(5066) = 0

      ContainerTempTable = ContainerAdapter.GetData
      If ContainerTempTable.Rows.Contains(array5065) = True Then
         ContainerAdapter.UpdateCont(array5065, array5066,
                        array5160, array5068, array5069, 9999, aTime,
                        array5065)
         Else
         ContainerAdapter.Insert(array5065, array5066, array5160,
                        array5068, array5069, 9999, aTime)
      End If
      VehicleTempTable = VehicleAdapter.GetData
      If VehicleTempTable.Rows.Contains(array5066) = True Then
         VehicleAdapter.UpdateVeh(array5066, array5067, 0, 6,
                        array5068, 9999, aTime, array5065, array5066)
      Else
         VehicleAdapter.Insert(array5066, array5067, 0, 6,
                        array5068, 9999, aTime, array5065)
      End If
```

```
            vehicle =
                      vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array5068,
                      50, 76, 1, 0)
               VehIDRoad1 = vehicle.ID
            IndexAdapter.Insert(VehIDRoad1, array5065, array5066, aTime, 1)
            RouteIndex = 0

            ContainerLogAdapter.Insert(array5065, array5066, array5160,
                      array5068, array5069, 9999, aTime, VehIDRoad1)
            VehicleLogAdapter.Insert(array5066, array5067, 0, 6, array5068,
                      9999, aTime, array5065, VehIDRoad1)
         End If
        '********  Road Empty Vehicle ********
        If ArenaLanguage.VariableArrayValue(9110) > 0 Then
         array9110 = ArenaLanguage.VariableArrayValue(9110) 'reroute
                      vehicle id
            array9111 = ArenaLanguage.VariableArrayValue(9111) 'reroute
                      vehicle type
            array9112 = ArenaLanguage.VariableArrayValue(9112) 'reroute
                      destination
               ArenaLanguage.VariableArrayValue(9110) = 0
            VehicleTempTable = VehicleAdapter.GetData

               VehicleAdapter.Insert(array9110, array9111, 0, 6, array9112,
                      9999, aTime, 0)

            vehicle =
                      vissim.Net.Vehicles.AddVehicleAtLinkCoordinate(array9112,
                             50, 76, 1, 0)
            VehIDRoadEmpty = vehicle.ID
            IndexAdapter.Insert(VehIDRoadEmpty, 0, array9110, aTime, 1)

            VehicleLogAdapter.Insert(array9110, array9111, 0, 6, array9112,
                      9999, aTime, 0, VehIDRoadEmpty)
         End If
Next
End Sub
End Class
```

# APPENDIX B

# ARENA© GLOBAL VARIABLES

| Submodel | Global Variable Name | Global Variable Number |
|---|---|---|
| Distribution Center 1 | dist1_port_origin_id | 5121 |
| Distribution Center 1 | dist1_road_origin_id | 5120 |
| Distribution Center 1 | Dist1_RoadVehType_Var | 5027 |
| Distribution Center 1 | Dist1_OtherVehType_Var | 5023 |
| Distribution Center 1 | Dist1_RoadDestID2_Var | 5122 |
| Distribution Center 1 | Dist1_OtherDestID2_Var | 5029 |
| Distribution Center 1 | Dist1_RoadDestID_Var | 5028 |
| Distribution Center 1 | Dist1_OtherDestID_Var | 5024 |
| Distribution Center 1 | Dist1_RoadContID_Var | 5025 |
| Distribution Center 1 | Dist1_OtherContID_Var | 5021 |
| Distribution Center 1 | Dist1_RoadVehID_Var | 5026 |
| Distribution Center 1 | Dist1_OtherVehID_Var | 5022 |
| Distribution Center 1 | Dist1SwitchVar_Veh | 8020 |
| Distribution Center 1 | Dist1SwitchVar_Cont | 8021 |
| Distribution Center 1 | dist1_enter_originID | 2026 |
| Distribution Center 1 | Dist1_Enter_DestinationID2 | 2025 |
| Distribution Center 1 | Dist1_Enter_DestinationID1 | 2024 |
| Distribution Center 1 | Dist1_Enter_Vehicle_Type | 2023 |
| Distribution Center 1 | Dist1_Enter_VehicleID | 2022 |
| Distribution Center 1 | Dist1_Enter_ContainerID | 2021 |
| Distribution Center 1 | Dist1_Enter_IndexID | 2020 |
| Distribution Center 1 | Dist1_Reroute_Destination | 9024 |
| Distribution Center 1 | Dist1_PortVehReroute_VehID | 9022 |
| Distribution Center 1 | Dist1_PortVehReroute_VehType | 9023 |
| Distribution Center 2 | dist2_port_origin_id | 5131 |
| Distribution Center 2 | dist2_road_origin_id | 5130 |
| Distribution Center 2 | Dist2_RoadVehType_Var | 5037 |
| Distribution Center 2 | Dist2_OtherVehType_Var | 5033 |
| Distribution Center 2 | Dist2_RoadDestID2_Var | 5132 |
| Distribution Center 2 | Dist2_OtherDestID2_Var | 5039 |
| Distribution Center 2 | Dist2_RoadDestID_Var | 5038 |
| Distribution Center 2 | Dist2_OtherDestID_Var | 5034 |
| Distribution Center 2 | Dist2_RoadContID_Var | 5035 |
| Distribution Center 2 | Dist2_OtherContID_Var | 5031 |

| Submodel | Global Variable Name | Global Variable Number |
|---|---|---|
| Distribution Center 2 | Dist2_RoadVehID_Var | 5036 |
| Distribution Center 2 | Dist2_OtherVehID_Var | 5032 |
| Distribution Center 2 | Dist2SwitchVar_Veh | 8030 |
| Distribution Center 2 | Dist2SwitchVar_Cont | 8031 |
| Distribution Center 2 | dist2_enter_originID | 2036 |
| Distribution Center 2 | Dist2_Enter_DestinationID2 | 2035 |
| Distribution Center 2 | Dist2_Enter_DestinationID1 | 2034 |
| Distribution Center 2 | Dist2_Enter_Vehicle_Type | 2033 |
| Distribution Center 2 | Dist2_Enter_VehicleID | 2032 |
| Distribution Center 2 | Dist2_Enter_ContainerID | 2031 |
| Distribution Center 2 | Dist2_Enter_IndexID | 2030 |
| Distribution Center 2 | Dist2_Reroute_Destination | 9034 |
| Distribution Center 2 | Dist2_PortVehReroute_VehID | 9032 |
| Distribution Center 2 | Dist2_PortVehReroute_VehType | 9033 |
| Distribution Center 3 | dist3_port_origin_id | 5141 |
| Distribution Center 3 | dist3_road_origin_id | 5140 |
| Distribution Center 3 | Dist3_RoadVehType_Var | 5047 |
| Distribution Center 3 | Dist3_OtherVehType_Var | 5043 |
| Distribution Center 3 | Dist3_RoadDestID2_Var | 5142 |
| Distribution Center 3 | Dist3_OtherDestID2_Var | 5049 |
| Distribution Center 3 | Dist23_RoadDestID_Var | 5048 |
| Distribution Center 3 | Dist3_OtherDestID_Var | 5044 |
| Distribution Center 3 | Dist3_RoadContID_Var | 5045 |
| Distribution Center 3 | Dist3_OtherContID_Var | 5041 |
| Distribution Center 3 | Dist3_RoadVehID_Var | 5046 |
| Distribution Center 3 | Dist3_OtherVehID_Var | 5042 |
| Distribution Center 3 | Dist3SwitchVar_Veh | 8040 |
| Distribution Center 3 | Dist3SwitchVar_Cont | 8041 |
| Distribution Center 3 | dist3_enter_originID | 2046 |
| Distribution Center 3 | Dist3_Enter_DestinationID2 | 2045 |
| Distribution Center 3 | Dist3_Enter_DestinationID1 | 2044 |
| Distribution Center 3 | Dist3_Enter_Vehicle_Type | 2043 |
| Distribution Center 3 | Dist3_Enter_VehicleID | 2042 |
| Distribution Center 3 | Dist3_Enter_ContainerID | 2041 |
| Distribution Center 3 | Dist3_Enter_IndexID | 2040 |
| Distribution Center 3 | Dist3_Reroute_Destination | 9044 |
| Distribution Center 3 | Dist3_PortVehReroute_VehID | 9042 |
| Distribution Center 3 | Dist3_PortVehReroute_VehType | 9043 |

| Submodel | Global Variable Name | Global Variable Number |
|---|---|---|
| I-16 Junction | road_origin_id | 5016 |
| I-16 Junction | Road_RoadVehType_Var | 5067 |
| I-16 Junction | Road_RoadDestID2_Var | 5069 |
| I-16 Junction | Road_RoadDestID_Var | 5068 |
| I-16 Junction | Road_RoadContID_Var | 5065 |
| I-16 Junction | Road_RoadVehID_Var | 5066 |
| I-16 Junction | Road_Emprty_VehicleID | 9110 |
| I-16 Junction | Road_Empty_VehicleType | 9111 |
| I-16 Junction | Road_Emprty_VehicleDestination | 9112 |
| GCT Gate | port_port_origin_id | 5111 |
| GCT Gate | port_road_origin_id | 5110 |
| GCT Gate | Port_RoadVehType_Var | 5017 |
| GCT Gate | Port_PortVehType_Var | 5013 |
| GCT Gate | Port_PortDestID2_Var | 5019 |
| GCT Gate | Port_RoadDestID_Var | 5018 |
| GCT Gate | Port_PortDestID_Var | 5014 |
| GCT Gate | Port_RoadContID_Var | 5015 |
| GCT Gate | Port_OtherContID_Var | 5011 |
| GCT Gate | Port_RoadVehID_Var | 5016 |
| GCT Gate | Port_OtherVehID_Var | 5012 |
| GCT Gate | PortSwitchVar_Veh | 8010 |
| GCT Gate | PortSwitchVar_Cont | 8011 |
| GCT Gate | Port_Enter_OriginID | 1016 |
| GCT Gate | Port_Enter_DestinationID2 | 1015 |
| GCT Gate | Port_Enter_DestinationID | 1014 |
| GCT Gate | Port_Enter_Vehicle_Type | 1013 |
| GCT Gate | Port_Enter_VehicleID | 1012 |
| GCT Gate | Port_Enter_ContainerID | 1011 |
| GCT Gate | Port_Enter_IndexID | 1010 |
| GCT Gate | Port_Road_Release_VehID | 9210 |
| GCT Gate | Port_Road_Release_VehType | 9211 |
| GCT Gate | Port_Road_Release_Destiation | 9212 |

# APPENDIX C

# ROADWAY NETWORK INTERSECTION SIGNALIZATION PLANS

| Intersection | Phase | All Red | Amber | Green |
|---|---|---|---|---|
| 4 | 1 | 2 | 3 | 30 |
| | 2 | 2 | 3 | 10 |
| | 3 | 2 | 3 | 35 |
| 5 | 1 | 2 | 3 | 25 |
| | 2 | 2 | 3 | 35 |
| | 3 | 2 | 3 | 15 |
| 6 | 1 | 2 | 3 | 25 |
| | 2 | 2 | 3 | 25 |
| 7 | 1 | 2 | 3 | 30 |
| | 2 | 2 | 3 | 10 |
| | 3 | 2 | 3 | 35 |
| 8 | 1 | 2 | 3 | 20 |
| | 2 | 2 | 3 | 20 |
| | 3 | 2 | 3 | 35 |
| 9 | 1 | 2 | 3 | 25 |
| | 2 | 2 | 3 | 35 |
| 10 | 1 | 2 | 3 | 30 |
| | 2 | 2 | 3 | 10 |
| | 3 | 2 | 3 | 35 |
| 11 | 1 | 2 | 3 | 30 |
| | 2 | 2 | 3 | 15 |
| | 3 | 2 | 3 | 55 |
| 12 | 1 | 2 | 3 | 30 |
| | 2 | 2 | 3 | 10 |
| | 3 | 2 | 3 | 35 |

# REFERENCES


1. Ni, D. *A Framework for New Generation Transportation Simulation*. in *Proceedings of the 2006 Winter Simulation Conference*. 2006. Monterey, CA: IEEE.

2. Dahmann, J.S. *High Level Architecture for Simulation*. in *First International Workshop on Distributed Interactive Simulation and Real Time Applications, 1997*. 1997. Eilat: IEEE.

3. Authority, G.P. *Future Expansion: The Future of Trade*. 2009 [cited 2009 4 Oct. 2009]; Available from: http://www.gaports.com/Facilities/GardenCityTerminal/FutureExpansion/tabid/269/Default.aspx.

4. Puglisi, C.M., *Issues and Challenges of Federating Between Different Transportation Simulators*, in *School of Civil & Environmental Engineering*. 2008, Georgia Institute of Technology: Atlanta. p. 75.

5. Kelton, W.D., R.P. Sadowski, and D.T. Sturrock, *Simulation With Arena*. 4th ed. 2007, Boston: McGraw Hill Higher Education. 630.

6. Hunter, M. and R. Machemehl. *Development and Validation of a Flexible, Open Architecture, Transportation Simulation*. in *Proceedings of the 2003 Winter Simulation Conference*. 2003: IEEE.

7. Law, A.M., *Simulation Modeling and Analysis*. 4th ed. 2007, Boston: McGraw Hill Higher Education.

8. Gbologah, F., *Development of a Multimodal Port Freight Model for Evaluation of Container Shipments*, in *School of Civil and Environmental Engineering*. Forthcoming (2010), Georgia Institute of Technology: Atlanta.

9. PTV, *VISSIM 5.10 User Manual*. 2008, Karlsruhe, Germany: Planung Transport Verkehr AG.

10. Chung, C.A., *Simulation Modeling Handbook: A Practical Approach*. Industrial and Manufacturing Engineering Series, ed. H.R. Parsaei. 2004, New York: CRC Press.

11. Fujimoto, R.M., *Parallel and Distributed Simulation Systems*. Wiley Series on Parallel and Distributed Computing, ed. A.Y. Zomaya. 2000, New York: John Wiley & Sons, Inc. 300.

12. Pegden, C.D. *Future Directions in Simulation Modeling*. in *Proceedings of the 2005 Winter Simulation Conference*. 2005: IEEE.

13. Chronopoulos, A.T. and C.M. Johnston, *A Real-Time Traffic Simulation System.* IEEE Transactions on Software Engineering, 1998. **47**(1): p. 321-331.

14. Potuzak, T. and P. Herout. *Use of Distributed Traffic Simulation in the JUTS Project*. in *Proceedings of the International Conference on Computer as a Tool. EUROCON, 2007.* 2007. Warsaw: IEEE.

15. Chandy, K.M. and J. Misra, *Distributed Simulation: A Case Study in Design and Verification of Distributed Programs.* IEEE Transactions on Software Engineering, 1979. **SE-5**(5): p. 440-452.

16. Reynolds, P.F. *A Spectrum of Options for Parallel Simulation*. in *Proceedings of the 1988 Winter Simulation Conference*. 1988: IEEE.

17. Kewley, R., et al. *Federated Simulations for Systems of Systems Integration*. in *Proceedings of the 2008 Winter Simulation Conference*. 2008. Austin, TX: IEEE.

18. Weatherly, R.M., et al. *Advanced Distributed Simulation through the Aggregate Level Simulation Protocol*. in *Proceedings of the 29th Hawaii International Conference on System Sciences*. 1996. Wailea, HI: IEEE.

19. Ferenci, S.L., K.S. Perumalla, and R.M. Fujimoto. *An Approach for Federating Parallel Simulators*. in *Proceedings of the Fourteenth Workshop on Parallel and Distributed Simulation, 2000*. 2000. Bologne: IEEE.

20. DoD. *The Department of Defense (DoD) Modeling and Simulation Coordination Office (M&S CO).* 2009 [cited 2009 5 Oct. 2009]; Available from: http://www.msco.mil/.

21. Dahmann, J.S. and R. Lutz. *Persistent Federations*. in *Proceedings of the 1998 Spring Simulation Interoperability Workshop*. 1998: SISO.

22. Tan, F., A. Persson, and R. Ayani. *HLA Federate Migration*. in *Proceedings of the 38th Annual Simulation Symposium*. 2005: IEEE.

23. Nance, R.E. *Distributed Simulation with Federated Models: Expectations, Realizations and Limitations*. in *Proceedings of the 1999 Winter Simulation Conference*. 1999. Phoenix, AZ: IEEE.

24. Potuzak, T. *Distributed Traffic Simulation and the Reduction of Inter-Process Communication Using Traffic Flow Characteristics Transfer*. in *Proceedings of the Tenth International conference on Computer Modeling and Simulation*. 2008. Cambridge, UK: IEEE.

25. Yoginath, S.B. and K.S. Perumalla. *Parallel Vehicular Traffic Simulation using Reverse Computation-Based Optimistic Execution*. in *22nd Workshop on Principles of Advanced and Distributed Simulation*. 2008. Roma: IEEE.

26. Klein, U., T. Schulze, and S. Straburger. *Traffic Simulation Based on the High Level Architecture*. in *Proceedings of the 1998 Winter Simulation Conference*. 1998. Washington, DC: IEEE.

27. Google, *Google Earth*. 2009, Google, Inc.: Mountain View, CA.

28. PTV, *VISSIM 5.10-06 COM Interface Manual*. 2008, Karlsruhe, Germany: Planung Transport Verkehr AG.

29. Lighthill, M.J. and G.B. Whitman, *On kinematic waves II. A theory of traffic flow on long crowded streets.* Proc. Royal Society of London, Part A, 1955.

30. Geroliminis, N. and C. Daganzo, *Existence of urban-scale macrosocopic fundamental diagrams: Some experimental findings*. 2007, UC Berkeley: UC Berkeley Center for Future Urban Transport: A Volvo Center fof Excellence: Berkeley.

31. Geroliminis, N. and C.F. Daganzo, *Macroscopic modeling of traffic in cities*, in *TRB 86th Annual Meeting*. 2007: Washington DC.

32. Daganzo, C. and N. Geroliminis, *An analytical approximation for the macroscopic fundamental diagram of urban traffic.* 2008, UC Berkeley: UC Berkeley Center for Future Urban Transport: A Volvo Center of Excellence: Berkeley.

33. Daganzo, C., *Presentation to the Georgia Transportation Institute. Some Laws of Urban Traffic Dynamics: Analysis, Physical Evidence and Control Applications*, GTI, Editor. October 2008: Atlanta.

34. Bedient, P.B. and W.C. Huber, *Hydrology and Floodplain Analysis*. 3rd ed. 2002, Upper Saddle River, NJ: Prentice Hall.

35. Law, A.M., *Simulation Modeling and Analysis*. 4th ed. McGraw-Hill Series in Industrial Engineering and Management Science, ed. K.E. Case and P.M. Wolfe. 2007, New York: McGraw Hill.

36. Robinson, S., *A statistical process control approach to selecting a warm-up period for a discrete-event simulator.* European Journal of Operational Research, 2007. **176**: p. 332-346.

37. Welch, P.D., *The Statistical Analysis of Simulation Results*, in *Computer Performance Modeling Handbook*, S.S. Lavenberg, Editor. 1983, Academic Press: New York.

38. Sandikci, B. and I. Sabuncuoglu, *Analysis of the behavior of the transient period in non-terminting simulations.* European Journal of Operational Research, 2006. **173**: p. 252-267.

39. White, K.P., M.J. Cobb, and S.C. Spratt. *A comparison of five steady-state truncation heuristics for Simulation*. in *2000 Winter Simulation Conference*. 2000. Orlando.