NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

QUANTIFYING POLITICAL DISRUPTIONS ON FUEL SUPPLY CHAINS AND MILITARY MICROGRID OPERATIONS IN WEST AFRICA

by

Abigail E. Staffnik

December 2023

Thesis Advisor: Douglas L. Van Bossuyt
Co-Advisors: Ronald E. Giachetti
Emily L. Meierding

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE QUANTIFYING POLITICAL DISRUPTIONS ON FUEL SUPPLY CHAINS AND MILITARY MICROGRID OPERATIONS IN WEST AFRICA	**5. FUNDING NUMBERS**
6. AUTHOR(S) Abigail E. Staffnik	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis addresses the challenge of ensuring resilient energy for Department of Defense (DOD) microgrids in politically volatile regions. It proposes an integrated model that assesses political risks, simulates fuel supply chain logistics, and evaluates microgrid systems using AnyLogic software. Utilizing AnyLogic and Global Terrorism Database insights, the study simulates disruptions over ten years, measuring power outages and availability in DOD microgrids. The results show that a diesel generator-powered military installation in Africa can benefit from over 10% increase in power availability and a decrease in the number of power outages and their duration when a military installation is supplemented with photovoltaic panels and battery storage systems, even under politically unstable conditions, as compared to a military installation solely reliant on diesel generators. Although installing photovoltaic panels and battery storage systems is subject to many conditions, such as space, maintenance availability, and funding, these findings highlight the need for robust energy solutions in volatile regions and provide a quantitative basis for strategic energy planning.

14. SUBJECT TERMS microgrid, resilience, political, non-violent protest, cost modeling	**15. NUMBER OF PAGES** 211
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

QUANTIFYING POLITICAL DISRUPTIONS ON FUEL SUPPLY CHAINS AND MILITARY MICROGRID OPERATIONS IN WEST AFRICA

Abigail E. Staffnik
Captain, United States Army
BSE, Arizona State University, Tempe, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2023

Approved by: Douglas L. Van Bossuyt
Advisor

Ronald E. Giachetti
Co-Advisor

Emily L. Meierding
Co-Advisor

Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis addresses the challenge of ensuring resilient energy for Department of Defense (DOD) microgrids in politically volatile regions. It proposes an integrated model that assesses political risks, simulates fuel supply chain logistics, and evaluates microgrid systems using AnyLogic software. Utilizing AnyLogic and Global Terrorism Database insights, the study simulates disruptions over ten years, measuring power outages and availability in DOD microgrids. The results show that a diesel generator-powered military installation in Africa can benefit from over 10% increase in power availability and a decrease in the number of power outages and their duration when a military installation is supplemented with photovoltaic panels and battery storage systems, even under politically unstable conditions, as compared to a military installation solely reliant on diesel generators. Although installing photovoltaic panels and battery storage systems is subject to many conditions, such as space, maintenance availability, and funding, these findings highlight the need for robust energy solutions in volatile regions and provide a quantitative basis for strategic energy planning.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The Department of Defense (DOD) is increasingly relying on microgrid technology to enhance energy security and resilience at overseas bases, mitigating reliance on centralized grids and fuel supplies vulnerable to disruption by political events (Ton and Smith 2012; 10 C.F.R. § 451). This study investigates how political factors, specifically terrorism and protests, influence the energy resilience of military installations in Africa, utilizing the AnyLogic simulation to integrate political data, including the Global Terrorism Database, into microgrid resilience models. The research spans a political risk assessment, a fuel supply chain simulation, and a microgrid power balance model within a West African context, aiming to provide a foundational framework for energy resilience against political risks (Anuat, Van Bossuyt, and Pollman 2021; Genova and Falola 2003; Giroux 2009; Lambrechts and Blomquist 2017).

This study uses an integrative approach to understand how political risks impact fuel supply chains and microgrid operations in West Africa. The study is scoped to the African continent for its political complexity, limited supply chain infrastructure, and dependence on diesel generators as the primary power source for many military locations. Using AnyLogic's advanced simulation capabilities, the research constructs a case study that combines qualitative political risk assessment, supply chain modeling, and microgrid evaluation, as pictured in Figure 1.

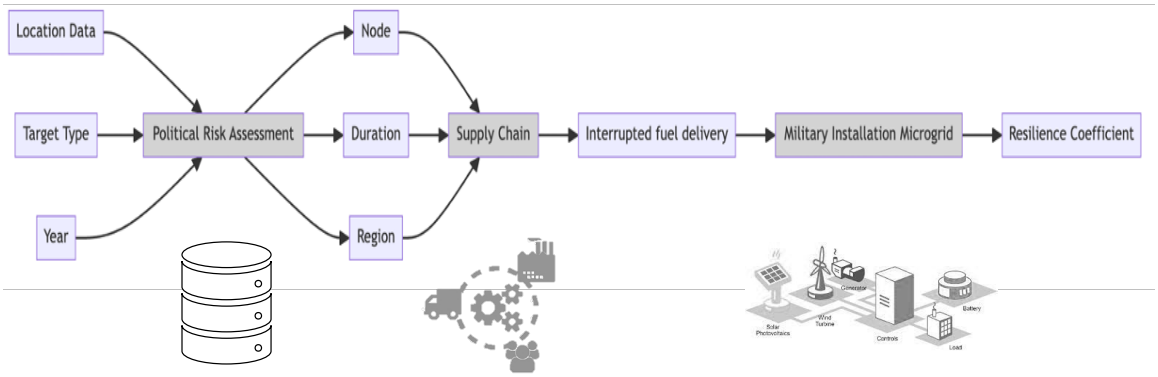


Figure 1. Methodology

The case study illustrated in Figure 2 and demonstrated through the simulation in Figure 3 examines the oil supply chain from the Netherlands to military bases in Nigeria, Morocco, and Western Sahara. It is based on insights from existing literature as well as expert consultations. By simulating disruptions caused by political events, the study measures the impact, through availability measures, on microgrid operations and assesses the supply chain's resilience. The methodology uses AnyLogic, considering a range of parameters to simulate realistic operational conditions and disruptions. The outcome is an assessment of power availability within the simulated microgrids. Appendix A contains the complete code and model parameters. The Java code for the model was generated using OpenAI's GPT-4 language model.

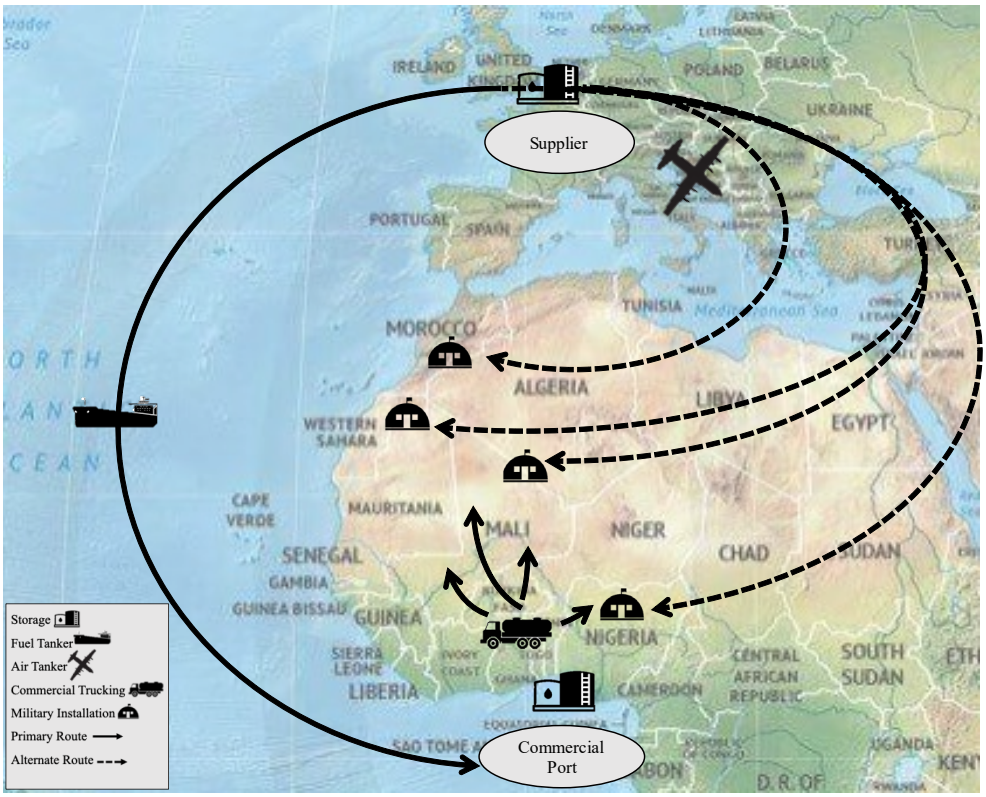


Figure 2. West Africa Case Study

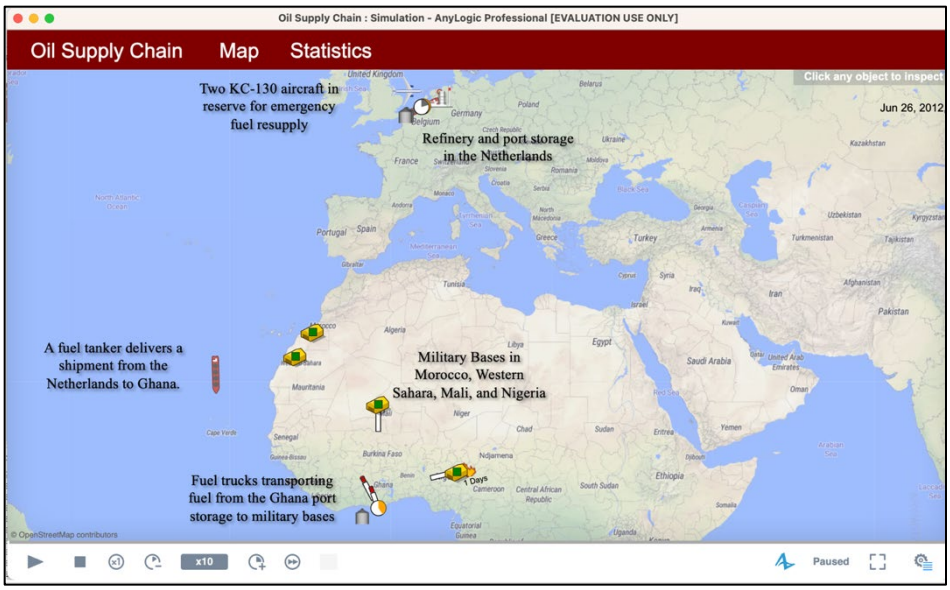


Figure 3. Case Study in the Simulation

This thesis employs AnyLogic software to conduct a series of experiments that examine the power availability of microgrids in military installations under conditions influenced by political disruptions, as summarized in Table 1. The initial experiment indicates that diesel generators maintain 100% availability without political interference. However, when factoring in supply chain attacks, the optimization process falls to 82% availability, well below the 99% standard set by the Department of Defense (DOD), until fuel storage increases to 32 days, which is unfeasible for many installations. When integrating a microgrid with solar and battery storage, however, the improved setup achieves 100% availability, over a 10% improvement compared to the diesel-only installation.

Table 1: Experiment Results Summary

	Main Experiment (Baseline)	Optimization Experiment 1 (Status-Quo)	Optimization Experiment 1 (Status-Quo) Expanded Parameters	Optimization Experiment 2 (Microgrid)	Main Experiment (Status-Quo)	Main Experiment (Microgrid)
	Diesel generators only, no political disruptions	*Finds the optimum fuel storage and diesel generation parameters for maximum availability with political disruptions.*	*Finds the optimum fuel storage and diesel generation parameters for maximum availability with political disruptions. Expanded parameters, as the first experiment was well below 99% availability.*	*Finds the optimum solar power, battery storage, fuel storage, and diesel generation parameters for maximum availability with political disruptions.*	*Availability with diesel only, post-optimization parameters*	*Availability with diesel, solar, battery, post-optimization parameters*
Parameters	*Initial parameters from Table 2*	*Optimization Range*	*Optimization Range*	*Optimization Range*	*Best parameters from Optimization Experiment 1*	*Best parameters from Optimization Experiment 2*
	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65
	Fuel Storage (days): 7	Fuel Storage (days): 7-10	Fuel Storage (days): 10-32	Fuel Storage (days): 7	Fuel Storage (days): 7	Fuel Storage (days): 7
	Battery (kWh): 0	Battery (kWh): 0	Battery (kWh): 0	Battery (kWh): 660-2840	Battery (kWh): 0	Battery (kWh): 660
	PV (kW): 0	PV (kW): 0	PV (kW): 0	PV (kW): 55-220	PV (kW): 0	PV (kW): 165
Results	*Met availability threshold with no supply chain interruptions.*	*Unable to meet 99% between 7-10 days fuel*	*Unable to meet 99% with up to 32 days of fuel*	*Met 99% availability threshold with optimized parameters.*	*82% availability, below DoD standard even with optimized parameters.*	*Over 10% increase in availability with a microgrid when compared to a diesel-generator only installation.*
	Applied SCN Disruptions: 0	Applied SCN Disruptions: 70/yr	Applied SCN Disruptions: 70/yr.	Applied SCN Disruptions: 70/yr.	Applied SCN Disruptions: 70/yr.	Applied SCN Disruptions: 70/yr.
	Availability: 100%	Availability: 89%	Availability: 100%	Availability: 100%	Availability: 82%	Availability: 100%
	#Outages / yr.: 0	Generator Rating (kW): 65	Generator Rating (kW): 65	Battery (kWh): 660	#Outages / yr.: 7	#Outages / yr.: 1.36
	Avg outage duration (day) : 0	Fuel Storage (days): 10	Fuel Storage (days): 32	PV (kW): 165	Avg outage duration (days): 8	Avg outage duration (days): 0.01

The simulation experiments demonstrate that incorporating photovoltaic (PV) and battery systems into military installations in politically unstable regions enhances power availability by an average of 10%. However, installing a 660-kWh battery system and a 165-kW PV system poses significant spatial challenges, requiring approximately a quarter acre or 1/4 of a football field. The diverse terrains of Africa, with its political complexity, limited supply chain infrastructure, and dependence on diesel generators as the primary power source, make it difficult to meet the space requirements and necessary maintenance and security measures. An alternative approach to increase the base fuel contingency from a seven-day to a 32-day supply incurs substantial real-estate costs.

This thesis recognizes the limitations inherent in the modeling approach, especially in the simplified political risk assessment, supply chain modeling, and microgrid simulation. The political risk assessment, which relies on a Poisson distribution, fails to accurately capture the complexities of real-world scenarios. Moreover, despite providing useful insights, the fuel supply chain simulation oversimplifies the intricate logistics dynamics in politically sensitive areas. For example, a military installation in Western Sahara is more likely to receive fuel from a port in Morocco rather than Ghana, as suggested by the case study. The simulated microgrids lack the power-balancing mechanisms that a typical microgrid would have. These limitations highlight the need for future models to refine and become more complex.

In conclusion, this thesis offers a methodology for integrating political risk into the energy security and resilience framework for DOD microgrids, particularly in politically volatile regions. It underscores the importance of considering political factors in energy resilience strategies and provides a quantitative basis for strategic energy planning. As the global political landscape continues to evolve, so too must our approaches to modeling and strategy development, ensuring that energy security remains a pivotal aspect of military planning, especially in regions where political instability poses significant challenges.

References

Anuat, Edward, Douglas L. Van Bossuyt, and Anthony Pollman. 2021. "Energy Resilience Impact of Supply Chain Network Disruption to Military Microgrids." *Infrastructures* 7(1): 4. https://doi.org/10.3390/infrastructures7010004.

Genova, Ann, and Toyin Falola. 2003. "Oil in Nigeria: A Bibliographical Reconnaissance." *History in Africa* 30: 133–56. https://doi.org/10.1017/S0361541300003181.

Giroux, Jennifer. 2009. "Targeting Energy Infrastructure: Examining the Terrorist Threat in North Africa and Its Broader Implications." *Elcano Newsletter*, no. 53 (February): 10 p.

Lambrechts, Derica, and Lars B. Blomquist. 2017. "Political–Security Risk in the Oil and Gas Industry: The Impact of Terrorism on Risk Management and Mitigation." *Journal of Risk Research* 20 (10): 1320–37. https://doi.org/10.1080/13669877.2016.1153502.

Ton, Dan T., and Merrill A. Smith. 2012. "The U.S. Department of Energy's Microgrid Initiative." *The Electricity Journal* 25 (8): 84–94. https://doi.org/10.1016/j.tej.2012.09.013.

ACKNOWLEDGMENTS

My deepest gratitude goes to my advisors and chair within the Systems Engineering Department and the Department of National Security. I swore off statistics in the eighth grade, but here I am! This is no light testament to the exceptional faculty here. I owe a profound debt of thanks to Dr. Giachetti, Dr. Van Bossuyt, and Dr. Meierding for their invaluable insights and support. The experience and time you dedicated to my academic journey is not lost on me and is a gift I will carry forward in my professional career.

On a personal note, I would like to thank my brothers, Anders and Alek, for being a constant inspiration to me. You both make the world look easy, and anything I accomplish is a credit to your inspiration.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Ensuring a reliable and resilient energy supply for its bases, especially those located overseas, is a critical challenge the DoD faces. Microgrids offer a solution to improve energy security by enabling localized generation and distribution, reducing dependence on vulnerable centralized power grids and fuel supply chains. However, the current models used to evaluate and justify investments in energy security do not have a data-based system to incorporate the complex role political events, such as terrorism and protest disruptions, play in affecting critical energy sources like refined oil.

The U.S. Department of Energy characterizes a microgrid as "a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that act as a single controllable entity with respect to the [energy] grid. A microgrid can connect and disconnect from the [energy] grid to enable it to operate in both grid-connected or island mode" (Ton and Smith 2012). Therefore, a microgrid's resilience refers to its capability to maintain operational functionality and swiftly recover from interruptions, irrespective of the grid's status (10 C.F.R. § 451). The DoD prioritizes microgrid technology for its dual benefits of energy security and resilience, particularly its ability to operate independently in the face of diverse threats, ranging from environmental challenges to intentional sabotage (Anuat, Van Bossuyt, and Pollman 2021).

According to Toft et al. (2010), sudden, short-term disruptions can pose severe threats to energy security and can be triggered by many events, including terrorism and political tensions. This makes it imperative for the DoD to consider these factors when evaluating energy security, particularly in volatile regions like Africa.

Several case studies examining the Nigerian oil industry strikes (Kennedy-Darling et al. 2008; Genova and Falola 2003) and Tunisian fuel protests (McCarthy 2022) demonstrate how non-violent political events can significantly impact the refined oil supply chain, which can leave generator-reliant military installations vulnerable. Additionally, studies on the vulnerability of energy infrastructure to terrorist threats (Giroux 2009;

Lambrechts and Blomquist 2017) add another layer of complexity in assessing the resilience of microgrids.

B. OBJECTIVES

The primary objective of this research is to investigate the extent to which political and related factors, such as terrorism and other political events, impact the energy resilience of military installations in Africa. Specifically, this study assesses whether microgrid systems can mitigate the vulnerabilities in fuel supply chains that are often affected by these political factors. To achieve this, the research will utilize AnyLogic as the primary simulation tool, adapting existing microgrid resilience models to incorporate political data sourced from databases like the Global Terrorism Database (GTD) (START (National Consortium for the Study of Terrorism and Responses to Terrorism) 2022).

C. SCOPE

This research integrates a political risk assessment of fuel supply chains, a fuel supply chain simulation model, and a microgrid power balance model to understand the implications of any fuel delivery disruptions. The study is scoped to the African continent. It was selected for its political complexity, limited supply chain infrastructure, and dependence on diesel generators as the primary power source for many military locations. The political risk assessment, fuel supply chain, and microgrid model are founded in a notional case study and, therefore, do not fully capture the complexities of each. This thesis's scope is intended to build a baseline framework upon which further case studies can modify.

D. RESEARCH METHOD DESCRIPTION

The research method employs AnyLogic simulation software to determine the effects of regional political disruptions on a fuel supply chain and a military installation microgrid.

(1) Case Study

This thesis is based on a case study that draws on existing literature and insights from DLA Europe and Africa, as well as the U.S. Africom J4 Logistics Branch. The thesis explores the simulation of microgrids in regions susceptible to political disruptions. The case study on which the simulation is built focuses on a hypothetical fuel supply chain in West Africa, specifically in Nigeria, Morocco, and Western Sahara.

(2) Simulation Model

Based on the case study, the simulation model incorporates a political risk assessment, a supply chain, and a military installation microgrid. First, the model uses the GTD to determine the probability of regional terrorist attacks that could impact the fuel supply chain, like attacks on utilities or roads. Then, the probability of an attack is applied to the fuel supply chain, from its initial production in an oil field to its delivery to a military installation in Africa. The military installation is the final consumer in the supply chain, where its generators consume the fuel from the supply chain. The installation microgrid has a dynamic power generation system, including solar photovoltaic panels, battery storage, and generators. The impact of the simulated supply chain disruptions caused by the attacks is then measured for an installation running only on diesel generators and one with a complete system of solar panels, battery storage, and generators.

(3) Verification and Validation

The model is first verified to ensure that the political risk assessment, supply chain, and microgrid perform as expected as independent systems. The model is then validated using multiple simulation experiments over ten years to produce consistent output metrics.

(4) Data Analysis

The model is run over ten simulation years to determine the power availability of an installation running on generators compared to an installation supplemented with a battery storage system and solar panels.

E. ORGANIZATION

This thesis is structured into five chapters. Chapter II explores literature on DoD energy resilience and microgrid models and how political factors like terrorism affect supply chains. Chapter III details the AnyLogic simulation methodology, integrating a political risk assessment with supply chain and microgrid modeling. Chapter IV presents the analysis of simulation data, revealing the effects of political risks on microgrid operations. Finally, Chapter V synthesizes conclusions and outlines recommendations for integrating political risk into resilience planning.

II. LITERATURE REVIEW

This literature review examines the literature on energy security and energy resilience of the DoD, specifically regarding microgrid resilience and political disruptions. The review is divided into five main sections: models for microgrid resilience, the complexities of the refined oil supply chain in the DoD, approaches for modeling these supply chains, real-world examples illustrating the effects of political unrest on refined oil delivery, and valuable data sources for measuring this impact.

A. ENERGY RESILIENCE IN THE DOD

Resilience in the context of microgrids refers to the system's ability to maintain its operational functionality in the face of disruptions (10 C.F.R. § 451). It measures how well the microgrid can adapt to and recover from adverse conditions, ensuring a continuous supply of energy (Giachetti et al. 2022).

Resilience in microgrids is assessed through a set of metrics and formulas that quantify the system's robustness and adaptability. A metric defined by the DOD is the Resilience Coefficient (R_C):

$$R_C = \frac{T_U}{T_U + T_D}$$

T denotes the total assessment period, T_U, denotes the length of time the system receives sufficient energy ("uptime"), and T_D denotes the duration of insufficient energy ("downtime") (DoD 2020). While this coefficient offers an insight into the availability of power, it is not entirely reflective of resilience, which is a broader concept encompassing a system's robustness and adaptability to adverse conditions (Giachetti et al. 2022). A more appropriate term for R_C might be "energy availability coefficient," as it primarily measures the sufficiency of power rather than the holistic resilience of the system. This distinction will be considered in the analysis of this thesis.

Giachetti et al. (2022) illustrate the behavior of power performance in a microgrid before, during, and after a disruption in Figure 1. The curve outlines three key phases: the

microgrid's resistance to initial performance degradation when a disruption occurs, followed by its stabilization, and finally, its ability to recover to the required performance levels. This curve serves as a comprehensive tool for understanding and assessing a microgrid's resilience across different stages of disruption. This thesis uses the Resilience Coefficient, R_C, as a metric to evaluate the resilience of simulated installation microgrids.

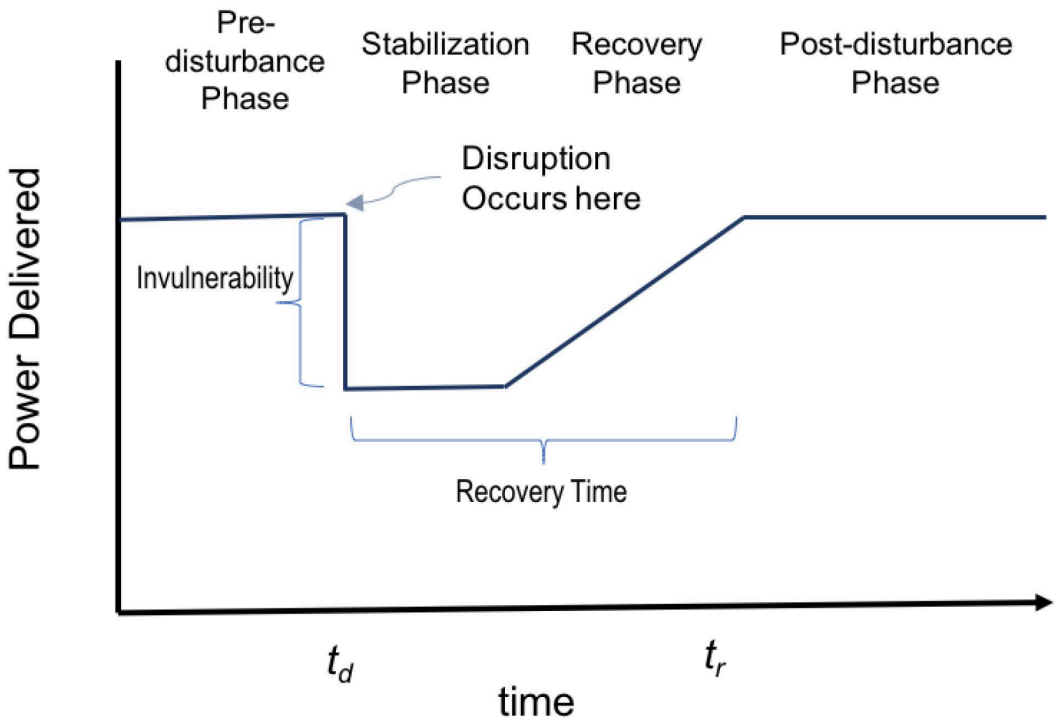


Figure 1. Resilience Curve Depicting Variation in Power Performance Behavior before, during, and after a Disruption. Source: Giachetti et al. (2022).

B. EXISTING MICROGRID RESILIENCE MODELS

Several studies have examined the resilience of microgrids. This section examines the existing models that have been developed to quantify microgrid resilience, specifically focusing on the extent to which these models account for political disruptions.

Several studies have been conducted on quantitative resilience metrics, such as those by Giachetti et al. (2022), Peterson et al. (2021a), and Hildebrand (2020). However, these studies focus on microgrid behavior and metrics rather than incorporating external

systems, such as political data analysis, although they acknowledge the risks of political disruptions.

Giachetti et al. (2022) suggest that an optimized resilience model should prioritize stored and renewable energy by exploring the balance between operational parameters and maintenance. Peterson et al. (2021a) build on this idea by aligning resilience with mission-specific objectives instead of purely economic criteria. Hildebrand (2020) further advances the studies by introducing life-cycle cost modeling as an effective method for evaluating microgrid resilience.

Recent adaptations of these foundational models, such as Bell et al. (2022), have broadened the scope to include extreme weather conditions, while Anglani et al. (2022) focus explicitly on arctic conditions. These models incorporate historical data sets for weather but are not scoped to capture political volatility.

Anuat et al.'s (2021) contribution to the study of microgrid resilience is unique from preceding studies because it presents a method for examining the impact of supply chain disruptions on energy resilience. To explore this, they employ a mathematical framework that combines discrete-time Markov chains (DTMCs) with Dynamic Bayesian Networks (DBNs). This fusion allows them to study how disturbances in one part of the Supply Chain Network (SCN) can trigger a cascading "ripple effect" through the entire network.

As described by the authors, Markovian analysis forecasts the future states of a system based solely on its current status, essentially treating the system as "memoryless." For example, in a board game, a player's next move is determined only by the token's current placement, regardless of the path it took to get there.

DBNs offer a more nuanced view. They capture the conditional probabilities between interconnected nodes over time, providing a framework to consider how one event can influence future events across the network. DBNs employ Conditional Probability Tables (CPTs) and interconnected time slices to model these relationships effectively (Anuat, Van Bossuyt, and Pollman 2021).

Anuat et al. (2021) use a combination of DTMCs and DBNs to model supply chain disruptions. The resulting model captures both the "memoryless" attributes and complex

interconnections within a supply chain, allowing for the simulation of disruption ripple effects.

These "ripple effect" disruptions are applied to the SCN as risks. The authors identify two types of risks within an SCN: operational risks and disruption risks. Operational risks are low-impact but high-probability events like safety recalls, whereas disruption risks are high-impact but low-probability events usually induced by external forces. Table 1 provides examples of SCN risks compiled by Anuat et al. (2021). A disruption or operational risk at one node can cause a cascade effect through the network.

Table 1. Example Supply Chain Node Disruptions. Source: Anuat, Van Bossuyt, and Pollman (2021).

Initial Impact	Disruption Risk	Example
Single node	Deliberate attack	Insider threat [111] Cyberattack [112] Terrorist attack [113]
	Logistics delay	Inclement weather [114] Transportation accident [115] Port congestion [116]
Multi-nodal	Natural disaster	Hurricane [117] Earthquake [118] Wildfire [119]
	Material shortage	Trade tariffs [120] Shipping route blockage [121] Civil unrest [122]
	Financial crisis	Market volatility [123] Economic recession [124] Global pandemic [125]

Anuat et al. (2021) consider single and multi-nodal risks in their model case study. They demonstrate the impact of these risks by modeling a specific disruption between a manufacturer (M2) and a distributor (D2), which negatively affects three customers downstream (C2, C3, and C4), as shown in Figure 2.

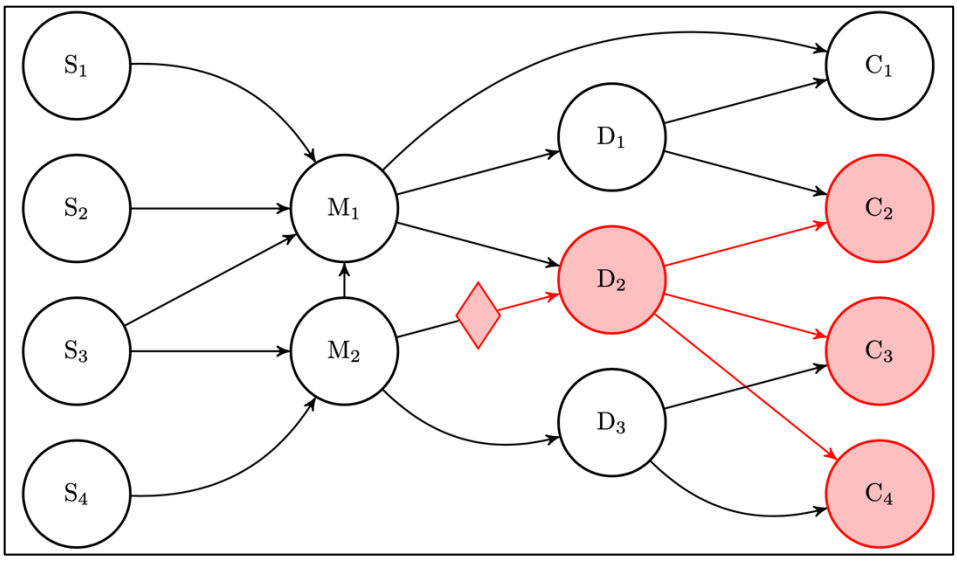


Figure 2. SCN Disruption (Red Diamond) between the Manufacturer (M2) and Distributor (D2), Affecting the Distributor and Three Downstream Customers (C2, C3, and C4). Source: Anuat, Van Bossuyt, and Pollman (2021).

The study results produced a supply chain impact on an installation microgrid and its resilience.

This research adopts Anuat et al.'s methodology for modeling SCNs and their downstream impact on installation microgrids, with key modifications. The research maintains the use of DBNs to capture the conditional probabilities between interconnected nodes over time. However, this research explicitly defines a supply chain and will not use DTMCs to capture regionally specific political volatility according to historical data sets. While DTMCs apply to macro-level analysis, they may not fully capture regional SCN disruptions. The research also incorporates historical data sets to model this regional political volatility.

C. REFINED OIL SUPPLY CHAIN WITHIN THE DOD

Defense Logistics Agency Energy (DLA Energy) plays a primary role in managing the refined oil supply chain essential for powering U.S. military bases in Africa. It holds the exclusive authority to establish international fuel agreements and is responsible for global fuel-related service acquisitions (Defense Logistics Agency 2023). DLA Energy has

faced challenges unique to the DoD in recent news and has unique capabilities to address them.

Specifically, DLA Energy has demonstrated a capability for rapid response and resupply internationally and domestically. Internationally, the agency has worked closely with NATO to manage the Central Europe Pipeline System, highlighting its agility in responding to geopolitical shifts, such as the ongoing Russian activities in Ukraine (Smith 2023). Domestically, DLA Energy's Customer Support Representatives (CSRs) have effectively resolved supply chain issues such as sourcing back-ordered fuel bladders for the Marines and resolving reliance on civilian infrastructure (Reece 2019). This rapid response capability is further exemplified in DLA Energy's role in the U.S. strategy to reposition military assets in the Pacific. The agency spearheaded significant infrastructure improvements at Naval Base Point Loma and Marine Corps Air Station Miramar to facilitate this strategic shift (Shawn 2015).

However, these capabilities are continually tested by challenges such as political unrest. Global incidents have underscored military installations' vulnerability to fuel shortages. Terrorist violence and strikes in West and East Africa have necessitated emergency DoD fuel resupply via aircraft or armed convoys (Hunkuyi and Emmanuel 2021; Department of Defense 2019; Benverren 2022). Events like the unrest in South Africa in July 2021 and persistent fuel shortages in Nigeria further emphasize the external factors that can disrupt the fuel supply chain (Njanji 2021; Asadu 2022)

Works such as "Move That Gas" (Kreisher 2010) and "The Journey of Jet Fuel from Refinery to Flight" (Inman 2017) further demonstrate the complex dynamics of fuel distribution to military units. Kreisher underscores the collaborative effort among U.S. agencies, military services, and civilian contractors, with DLA Energy as the central entity. Notably, the Air Force, the largest consumer of DoD's fuel, encounters challenges in global fuel transportation, especially in regions under the U.S. Central Command (CENTCOM), where complex and risky supply routes are used. These works stress the importance of coordinated supply chains and reliable delivery for military operations and highlight that disruptions, whether due to deficient infrastructure or precarious terrain, significantly impact end-users due to delivery delays.

Adding another layer of complexity, the DoD's refined oil supply chain is politically vulnerable, as revealed by a recent study, "Mapping U.S. Military Dependence on Russian Fossil Fuels" by Gard-Murray and Shanks in 2022. The study indicates that prior to Russia's actions in Ukraine, U.S. military bases in Europe sourced approximately 30% of their annual energy needs from Russian fossil fuels. This presents a political risk that could disrupt the DoD's refined oil supply chain and underscores the importance of incorporating such political risks into microgrid resilience models. Although it does not fully complement DLA Energy's capabilities, this thesis's simulation model will capture the ability to conduct emergency resupply.

D. MODELING THE DOD SUPPLY CHAIN

Modeling the DoD's fuel supply chain differs significantly from the civilian industry, primarily because of differing views on Supply Chain Risk Management (SCRM) (Anuat, Van Bossuyt, and Pollman 2021). The DoD's policy on SCRM is founded on protecting mission-critical infrastructure, while many industries are instead driven by inventory and cash-flow risk ("DoD Instruction 5200.44" 2018; Fan and Stevenson 2018; Faisal, Banwet, and Shankar 2007). Ultimately, both industry and the DoD largely accept models using a combination of DBNs and DTMCs as adopted by Anuat et al. (2021) to capture the downtrace "ripple effects" of disruptions throughout the supply chain. This thesis adopts a similar model approach to the SCN's "ripple effects" founded both on Anuat et al.'s. work and a 2018 study by Rossetti and Bright, "Bulk Petroleum Supply Chain Simulation Modeling," done for DLA Energy.

Anuat et al. (2021) used MATLAB to model the impact of Supply Chain Network (SCN) disruptions on military microgrids. Their model employed DTMCs and DBNs to simulate supply chain risks and capture a resultant R_C on the impacted microgrid. This thesis explicitly defines a supply chain network, negating the need for a DTMC, but the model will be inherently Bayesian. This thesis also measures the effects on the model microgrid using power availability.

Rosetti and Bright's methodology is adopted by this thesis to model the explicitly defined supply chain. Rossetti and Bright (2018) developed a simulation framework using

discrete-event methodology to model the refined oil supply chain of the Defense Logistics Agency Energy. The framework includes classes and interfaces that represent terminals, resources, carriers, and transportation modes. The simulation allows for a detailed evaluation of supply chain disruptions, which helps in planning for resilience in complex decision-making situations. The following is a detailed review of Rosetti and Bright's (2018) methodology, as it will be used to model the supply chain used for this thesis's case study.

According to Rossetti and Bright (2018), the bulk fuel supply chain system's main objective is to cater to end users' fuel requirements over a designated time frame. They present a conceptualization of the bulk fuel supply chain in Figure 3, outlining the system's components, which involve suppliers and customers situated at the terminals of the supply chain. These parties engage with products, primarily encompassing bulk petroleum fuels and additives. The infrastructure comprises various facilities such as defense fuel supply points (DFSPs), intermediate terminals, and bases. The means of transportation connecting these facilities include tanker ships, river barges, and tanker trucks (Rossetti and Bright 2018).

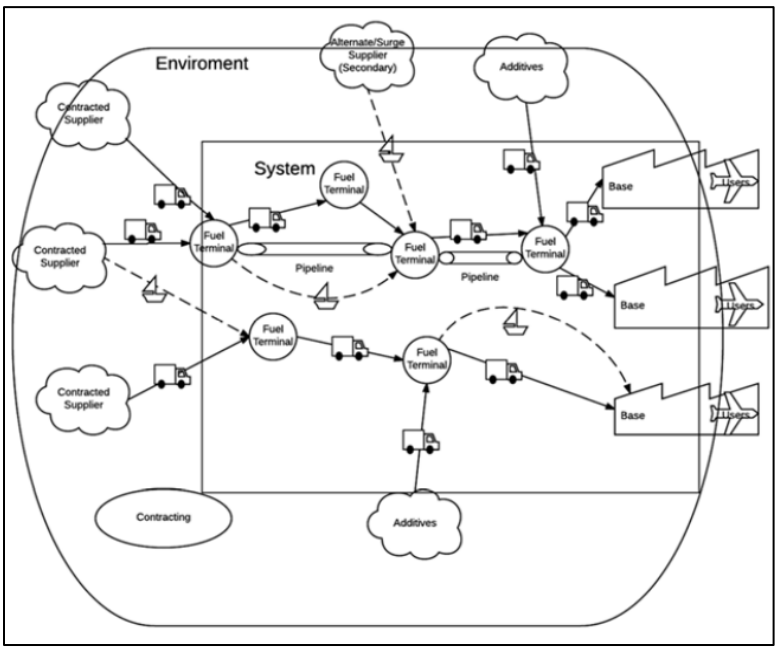


Figure 3. Elements of a Fuel Supply Chain. Source: Rossetti and Bright (2018).

Figure 4 illustrates the operational dynamics of a terminal, a node connecting a mode of transport, such as a truck or ship. This figure illustrates how terminals facilitate the interaction between suppliers, customers, and product inventory.

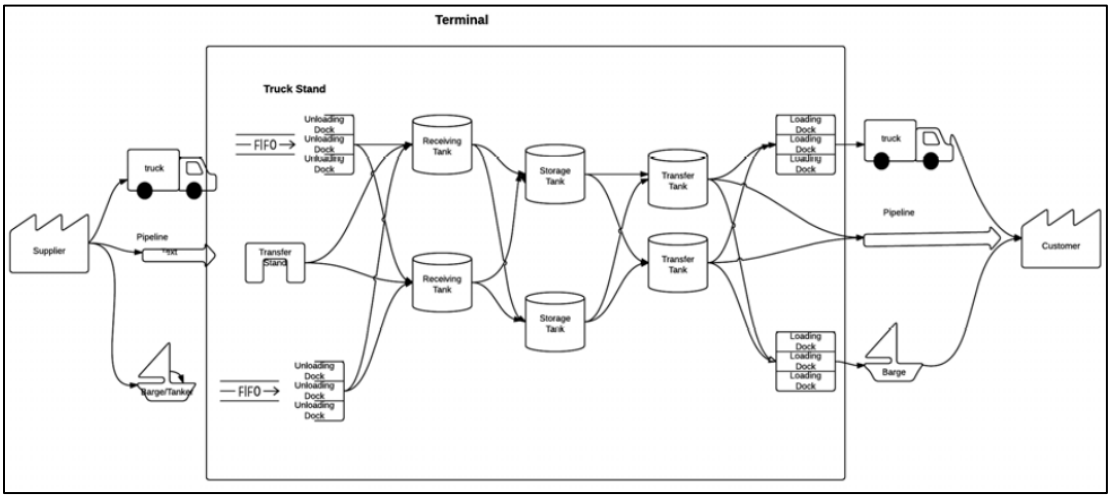


Figure 4. Terminal Illustration. Source: Rossetti and Bright (2018).

According to Rossetti and Bright (2018), terminals have the capability to receive bulk fuel through various transportation modes, including pipelines, tanker ships, barges, tanker rail cars, and tanker trucks. For a terminal to accept a specific mode of fuel transport, it must possess the necessary equipment or facilities to accommodate the given mode. Resources dictate fuel crossing the terminal boundary and can only operate unidirectionally within the resource. If the terminal has the requisite receiving equipment or facilities, bulk fuel is transferred into an available receiving tank, subject to availability.

A conceptual representation of a fuel tank and its measures of performance are shown in Figure 5. Each fuel tank is dedicated to a single fuel type, with a safe fill level excluding non-usable sections.

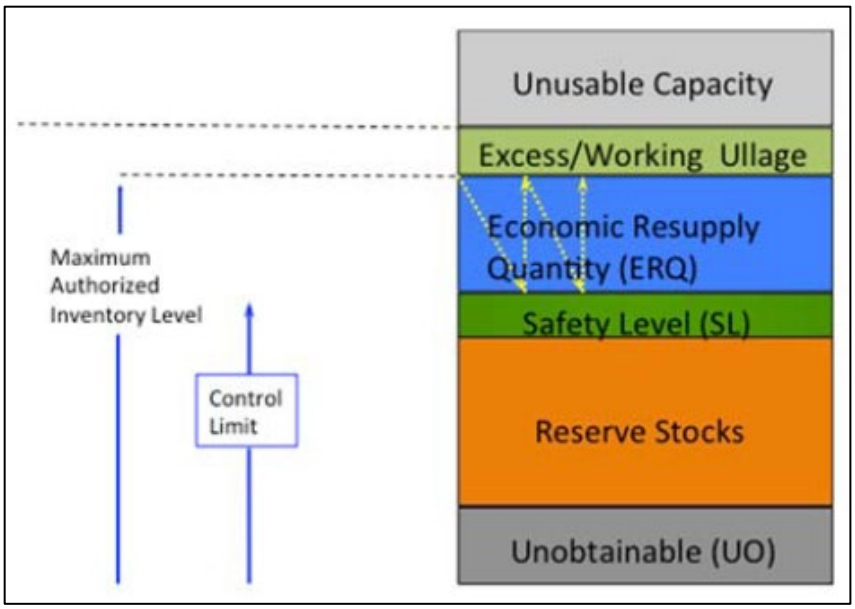


Figure 5. Fuel Tank Illustration. Source: Rossetti and Bright (2018).

To successfully operate, a terminal must process fuel from one mode of transport to another. Truck racks, docks, rail yards, and pipelines function similarly in processing fuel, with specific locations for fuel transfer. Truck racks have designated truck stands for loading/unloading, with some accommodating both, displayed in Figure 6. Docks comprise berths for ship/barge fuel transfer, while rail yards handle rail cars, each accommodating multiple cars. Pipelines connect terminals, transferring fuel in batches sequentially, as shown in Figure 7.

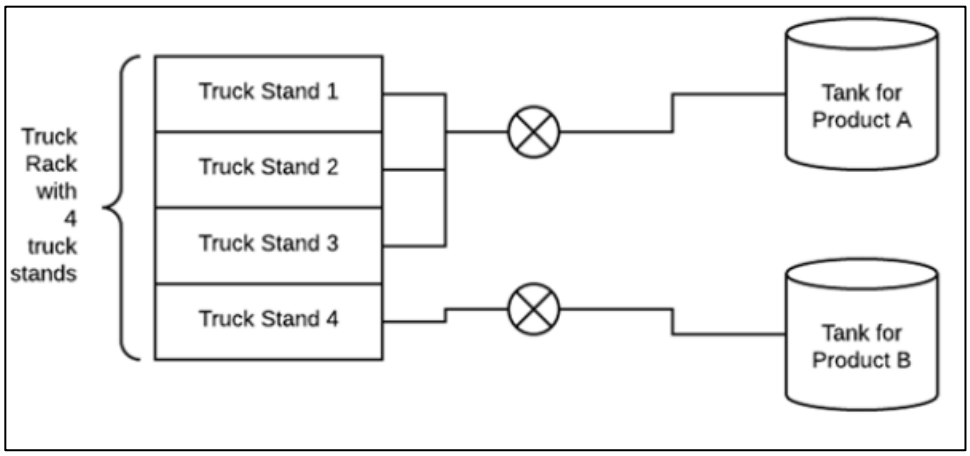


Figure 6. Truck Rack Illustration. Source: Rossetti and Bright (2018).

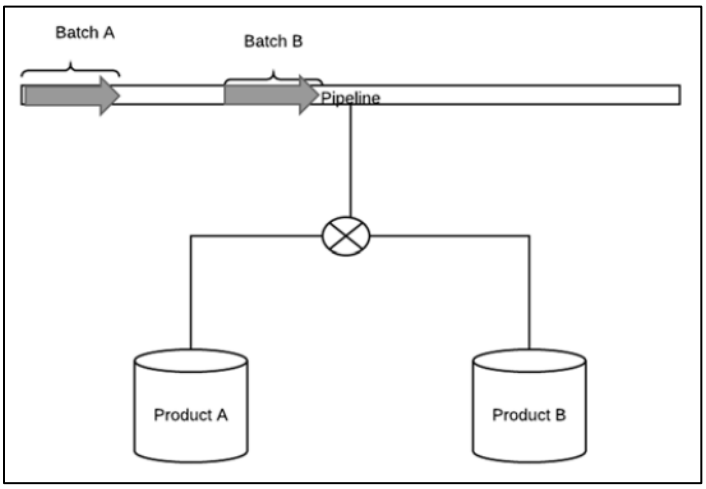


Figure 7. Pipeline Illustration. Source: Rossetti and Bright (2018).

This successful transfer of fuel via the terminal facilitates a supply chain. Rossetti and Bright (2018) used this supply chain to present a notional case study of the DLA Energy East supply chain. Figure 8 depicts this notional example, illustrating terminals, external suppliers, and transportation options. The model is designed to have input disruptions, such as weather or political disruptions, to simulate consequences on supply chain operations.

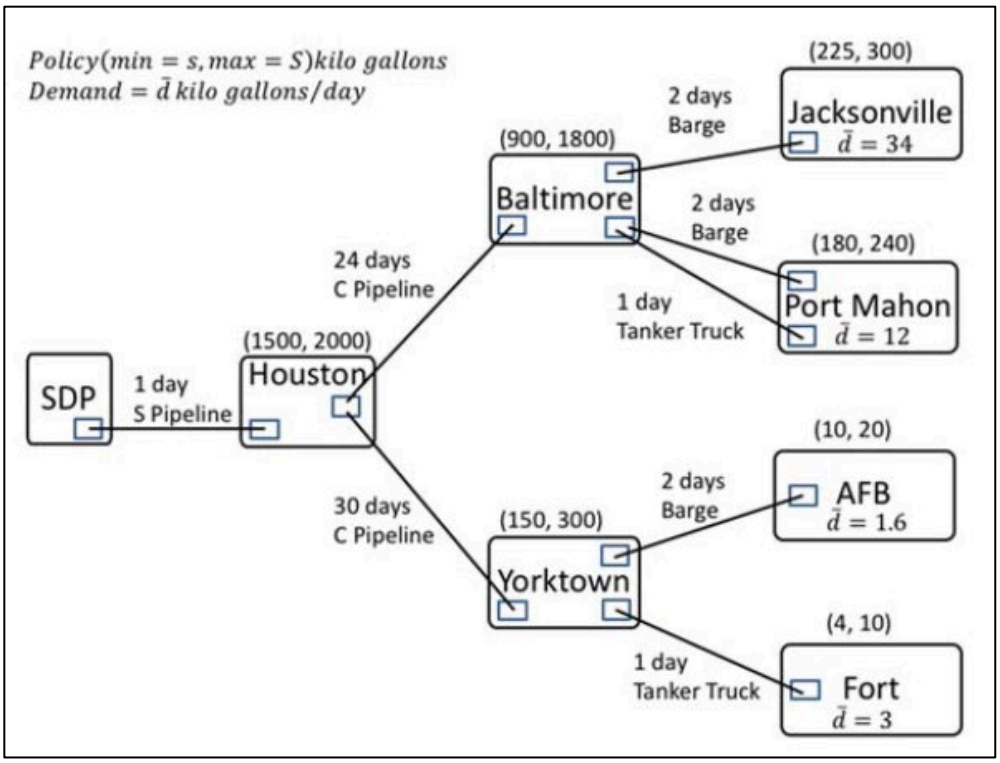


Figure 8. Notional Case Study: DLA Energy East. Source: Rossetti and Bright (2018).

A notional case study such as that presented by Rosetti and Bright can be built using their methodology to model political disturbances to the supply chain.

E. CASE STUDIES: REGIONAL POLITICAL UNREST AND DISRUPTIONS IN THE REFINED OIL SUPPLY CHAIN

Microgrids are engineered with resilience in mind, yet they remain vulnerable to various factors, including political events that can significantly impact supply chains and the availability of energy sources. This literature review examines case studies that highlight the significance of these political factors in microgrid resilience modeling, focusing specifically on non-violent protests and terrorist attacks.

Several case studies highlight the impacts of non-violent protests and political events on the delivery of oil, including Kennedy-Darling et al.'s (2008) and Genova et al.'s (2003) articles on Nigerian oil disruption. Both studies examined how Nigeria, as a major oil-producing country, is susceptible to refined oil supply chain disruptions due to workers

in the Nigerian oil industry, including refinery workers and tanker drivers, going on strike to demand better working conditions, higher wages, and protest government policies. According to both Kennedy-Darling and Genova et al., these strikes have led to disruptions in oil production, refining, and distribution, impacting the country's delivery of refined oil products.

The insights gained from this case study highlight how non-violent protests can impact a microgrid fuel source, namely diesel. This disruption in oil delivery, caused by the non-violent protests, could impact microgrid resiliency in a way that we cannot yet model.

In addition to Kennedy-Darling et al.'s and Genova et al.'s studies, there are several case studies on Tunisian fuel protests. These studies also indicate a potential correlation between non-violent protests and the delivery of refined oil, further emphasizing the potential impact on the resilience of microgrids. McCarthy's (2022) article on the Kamour Campaign, a protest movement in Tunisia after the democratic transition, was particularly insightful in this regard. The campaign mobilized unemployed individuals and targeted oil transport routes (McCarthy 2022). The author explains how the protests escalated with roadblocks, a protest camp, and, most importantly, fuel pipeline closures. The repertoire of non-violent protests included sit-ins, roadblocks, and strikes. All these non-violent protests could negatively impact the resiliency of a DoD expeditionary microgrid, should it be deployed in an Area of Operation (AOR) experiencing similar political unrest, as seen during Tunisia's Kamour Campaign.

Moreover, the threat of terrorism to energy infrastructure is also significant. For instance, Giroux (2009) analyzed the terrorist threat to energy infrastructure in North Africa and its implications for regional security and global energy markets. The study examined the motives and methods of terrorist groups targeting energy infrastructure and assessed the impact on local economies and political systems. This case study could be used to construct a scenario on which to model and evaluate how local fuel disruption could impact the resiliency of an expeditionary DoD microgrid.

Similarly to Giroux's (2009) article, Lambrechts and Blomquist (2017) examine the effects of terrorism on risk management and mitigation in the oil and gas industry. The article highlights the industry's vulnerability to terrorist attacks and the potential disruptions to operations, infrastructure, and personnel. The authors discuss the importance of effective risk management practices, including risk assessments, security protocols, intelligence sharing, and collaborations with security forces. Case studies from different regions demonstrate the diverse challenges faced by the industry. Overall, the article emphasizes the need for robust risk management strategies to address terrorism-related risks and ensure the safety and continuity of oil and gas operations. This study could be used to construct a case study that models how susceptible a region is to supply chain disruption of oil delivery to microgrids due to political events.

By synthesizing the information from these case studies with the supply chain modeling methodologies proposed earlier, this research aims to develop a more comprehensive approach to microgrid resilience. This approach incorporates a range of political factors, including non-violent protests and terrorist attacks, into the broader context of supply chain impacts on microgrid resilience.

F. DATA SOURCES CAPTURING REGIONAL POLITICAL DISRUPTION

To integrate political disruptions into existing cost models, it is important to explore quantitative measures that can capture the influence of these disruptions. While a direct metric to measure such influence could not be found in the literature, several sources were investigated for potentially relevant data.

The Uppsala Conflict Data Program (UCDP) is one such source that collects and analyzes information on armed conflicts within and between states worldwide (Department of Peace and Conflict Research 2021). The UCDP maintains a database of armed conflicts, peace agreements, and other related information that researchers and policymakers can use to better understand and address conflicts. In addition to the database, the UCDP also produces reports, publications, and other resources that aim to provide insights into conflict dynamics and potential avenues for conflict resolution. The data and reports generated by

the UCDP may provide insight into regional instability that may impact microgrid fuel sources.

Another relevant data source is the Non-violent and Violent Campaigns and Outcomes (NAVCO) Data Project, a research project of Erica Chenoweth, a professor of public policy at Harvard Kennedy School, who specializes in the study of non-violent resistance and civil resistance movements (Chenoweth, Pinckney, and Lewis 2019). The NAVCO project collects and analyzes data on non-violent and violent campaigns worldwide. This data may prove valuable in quantifying how movements such as transportation strikes can influence microgrid fuel sources.

The GTD provides a comprehensive data source on terrorist incidents worldwide since 1970, including attacks on energy infrastructure such as oil and gas pipelines, power plants, and other facilities (START 2022). According to the GTD website, data includes information on the location and date of the attack, the type of energy infrastructure targeted, and the number of casualties resulting from the attack. When translated into a resilience model, this data can potentially inform decisions on microgrid design.

Finally, the Energy Infrastructure Attack Database (EIAD) is a database that, although no longer publicly available, demonstrates the type of valuable data that could be collected for future microgrid resilience modeling. The database contains data on attacks and threats to energy infrastructure worldwide, including sectors such as oil, gas, coal, nuclear, and renewable energy. It covers the period from 1980 to 2011 and provides information on the type of attacks, instruments used, and the geographic distribution of incidents (Giroux 2009).

When considering all the available sources for quantifying the impact of political disruptions on microgrid fuel sources and integrating them into existing resilience models, the GTD appears to be the most valuable in the context of this research. This is because the DOD is highly vulnerable to terrorist attacks due to its reliance on infrastructure, as evidenced by the case studies of political unrest. The GTD is an open-source database that covers terrorist incidents worldwide from 1970 to 2020, with plans for yearly updates in the future. Unlike many other event databases, the GTD includes structured data on both

domestic and international terrorist events that have occurred during this period, with over 200,000 instances recorded to date. Therefore, the GTD has been chosen as the primary source for the initial model.

G. ASSESSING THE PROBABILITY OF A POLITICAL EVENT

Civil disruptions, terrorist attacks, and protests are multifaceted events with multiple dependencies, and estimating their frequency of occurrence requires complex modeling techniques. Such techniques can include event trees, which map out a chain of events and potential outcomes following an initiating event; decision trees, which represent choices and their possible consequences in a tree-like graph; and fault trees, which are used to deduce failure probabilities within a system. More sophisticated approaches like Bayesian belief networks model probabilistic relationships among variables; game theory analyzes strategic interactions among rational players; and agent-based models simulate autonomous agents to assess their effects on the system (Huamaní, Mantari, and Roman-Gonzalez 2020; Clauset and Woodard 2012). However, the purpose of this research project is to integrate three systems—political risk assessment, supply chain, and microgrid resilience assessment—rather than evaluate political risk assessment methods. Consequently, the research adopts a straightforward approach of using a Poisson distribution to calculate a rate for the regional risk probability. This method can be improved in future research by applying more complex methods, including those mentioned above, which may be necessary to address the complex issue of terrorism (Ezell et al. 2010).

H. LITERATURE REVIEW SUMMARY

In summary, the existing literature offers insights into microgrid resilience, the complexities of the DoD refined oil supply chain, and the impact of political disruptions. However, a gap remains in research on how to cohesively model the probability of political disruption on a supply chain and, subsequently, an installation's resilience. While foundational studies on microgrid resilience focus predominantly on operational and economic factors, they generally omit how to specifically integrate and model the influence of political disruptions. Moreover, although the literature addresses the susceptibility of the

DoD's fuel supply chain to various disruptions, including political ones, it does not extend this understanding to microgrid resilience. Case studies from regions like Nigeria and Tunisia underline the need to integrate political risk into resilience modeling. Data sources such as the UCDP, NAVCO, and GTD offer potential metrics for quantifying political disruptions, but a unified model for their effective integration into microgrid resilience cost models remains absent. This review underscores the need for a comprehensive approach that holistically considers political disruptions in the DoD's microgrid resilience cost models, aiming to enhance their fidelity and utility for more informed decision-making in politically volatile environments.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

This study synthesizes three different analysis frameworks—a political risk assessment utilizing qualitative data, a simulated oil supply chain, and a microgrid. The methodology is divided into three main sections based on the system: Political Risk Assessment, Supply Chain Modeling, and Microgrid Evaluation. Each segment has its own set of inputs and outputs, which are explained below and illustrated in Figure 9.

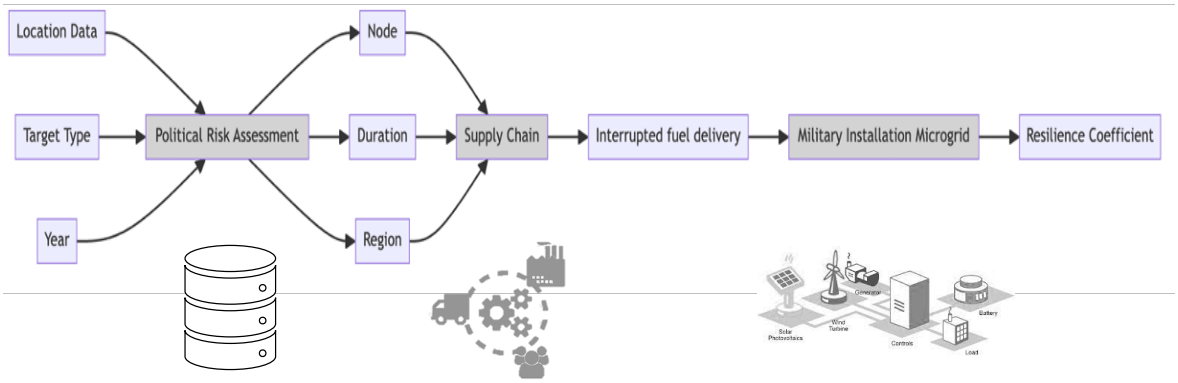


Figure 9. Methodology

A. CASE STUDY CONSTRUCTION

A notional case study is constructed based on insights gathered from the literature review, supplemented with insights from DLA Europe and Africa and the U.S. Africom J4 Logistics Branch. The case study provides a real-world context for the simulation and analysis of microgrid designs under conditions of political instability and targeted infrastructure attacks.

The case study focuses on the oil supply chain to military installations in West Africa, specifically in Nigeria, Morocco, and southern Moroccan regions. Refined oil is transported via commercial vessels from the Netherlands to a Tema port in Ghana. From there, the oil is stored and then transferred to commercial trucking companies for delivery to the military bases, as depicted in Figure 10. While it may seem excessive to transport fuel from Ghana to Morocco by road, given that there are oil ports in Morocco, this case

study is intended to highlight the challenges of road transport of fuel across international borders. In situations where commercial options are disrupted due to political reasons, the military has had to provide fuel via guarded convoys. In the event of a disruption in the primary fuel supply chain, air tankers are used to transport fuel directly to military installations.

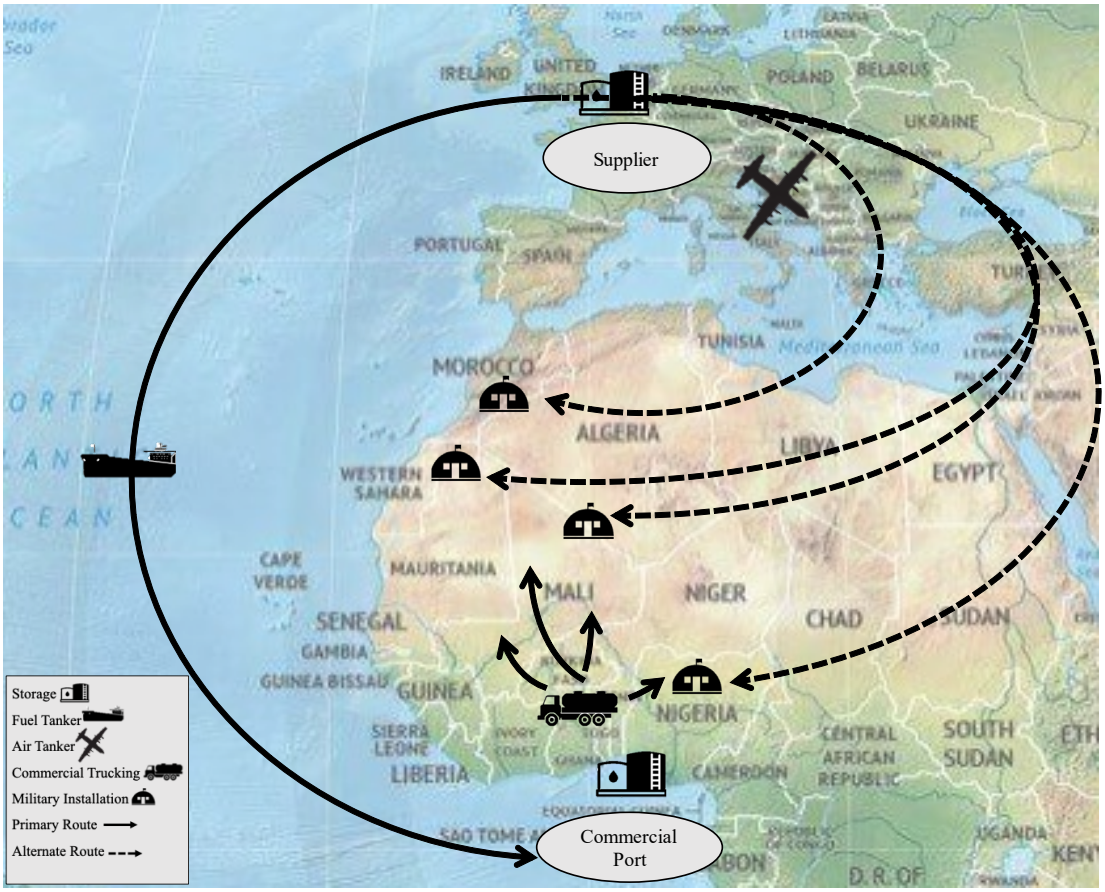


Figure 10. West Africa Case Study

The supply chain is vulnerable to various instances of political disturbances, such as terrorist attacks and strikes. These disruptions pose a significant threat to the steady supply of fuel, as they can disrupt DLA's fuel delivery contracts as well as the fuel transportation mechanisms. For instance, in some countries DLA Energy can only contract fuel delivery with a countries' specified vendors. This severely limits the availability of fuel in certain instances, like commercial shortages and strikes, prompting emergency fuel

resupply via truck or air. This can affect the resilience and operational capability of the military installations' microgrids.

The primary objective is to assess the total projected fuel delivery disruption over ten years. Given the uncertainty in the fuel supply chain, this assessment will provide insights into the resilience of the simulated installation microgrids.

B. ANYLOGIC MODELING

AnyLogic is used in this research for its ability to integrate a political risk assessment with its downstream effects on a fuel supply chain network and an associated installation microgrid. The software enables the explicit definition of SCNs, such as oil refineries or storage facilities, through its Agent-Based Modeling (ABM) and Geographic Information Systems (GIS) capabilities. These nodes, or agents, are placed on a map, and their interactions are governed by programmed behavior and GIS data. Additionally, AnyLogic facilitates the direct import and probabilistic and statistical analysis of large databases employed to model supply chain disruptions based on the GTD. In addition, AnyLogic can model microgrids through process modeling. The platform's Java-based customization options further extend its versatility ("AnyLogic: Simulation Modeling Software Tools & Solutions for Business" 2023). The Java code for the model was generated using OpenAI's GPT-4 language model. All parameters are purely notional and are designed to illustrate a method rather than a real-world simulation. Appendix A contains the full code and model parameters.

1. Supply Chain (Agent-Based GIS Modeling)

The model supply chain simulates the delivery of oil to military installations in West Africa. Figure 11 depicts the agents from the case study as they look in the simulation, and Table 2 shows a summary of the primary parameters that govern the agents. The initial parameters shown in Table 2 are used as a baseline to perform optimization experiments during analysis. The agents include military installations, called "Base Microgrids" in the simulation, fuel tankers, KC-130 aircraft (Air Tankers), fuel trucks, fuel storages, refineries, and pipelines.

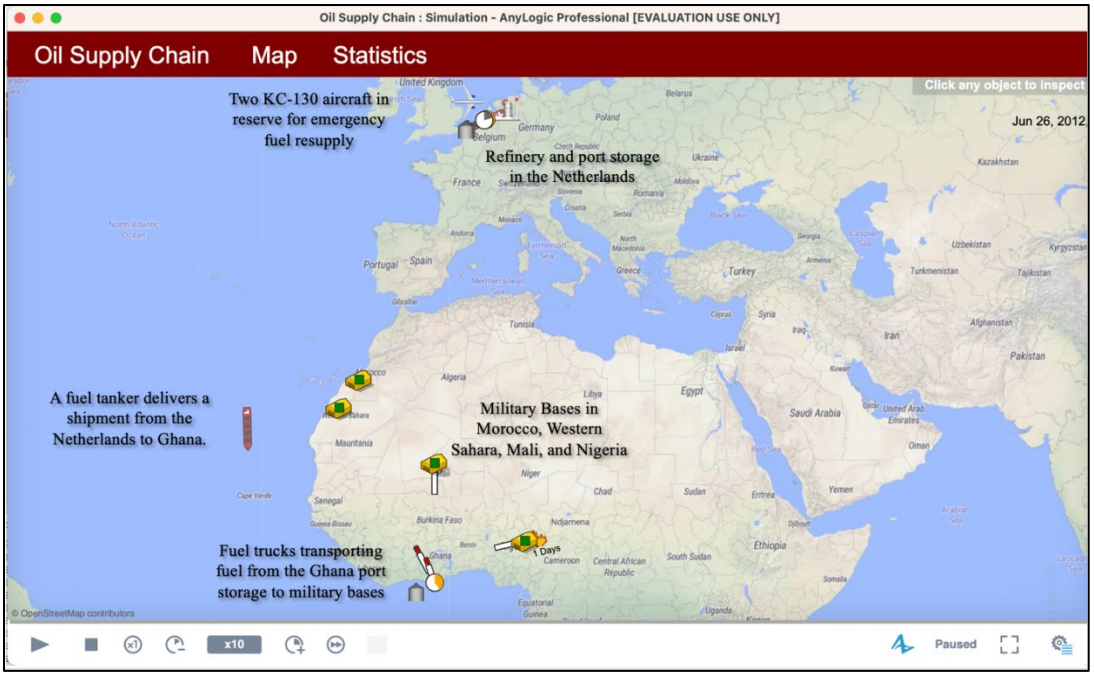


Figure 11. Case Study in the Simulation

Table 2. Primary Model Initial Parameters, Pre-optimization, Summarized

Parameter		**Value**	**Units**
Microgrid (Main)	**Mean Hourly Power Load	55	kWh / hr
	Fuel Storage Capacity	3.18 (7 Days of Supply)	m^3
Microgrid (Diesel Generator)	Power	65	kW
	Load	variable	%
	Min Load	0.6	%
	Peak Consumption Rate	5	gal/hr
	Startup Delay	0	hours
Microgrid (PV)	Photovoltaic Power	50	kW
Microgrid (BESS)	Energy	100	kWh
	Discharge Power	100	kWh
	Discharge Efficiency	0.95	%
	Charge Power	100	kW
	Charge Efficiency	0.95	%
	Min SOC	0.2	%
	Max SOC	0.99	%

Parameter		Value	Units
KC-130 Aircraft	Speed	650	km/hr
	Capacity	32.2	m^3
	Unloading Rate	0.038	m^3/s
Fuel Tanker Ship	Speed	100	knots
	Capacity	3000	m^3
	Unloading Rate	0.1	m^3/s
Fuel Truck	Speed	55	mph
	Capacity	18	
	Unloading Rate	0.0126	m^3/s
Refineries	Refining Rate	2.17	m^3/s
	Capacity	164237710.2	m^3
Storages	Capacity	50000	m^3

** Mean Hourly Power Load varies by 90%-110% of 55-kWh/hr. in daytime hours and 45%-55% at night.

Initially, crude oil is shipped via tankers from crude oil fields to a refinery in Rotterdam. Once the refinement process is complete, the fuel is then piped into storage tanks and transported to port storage in Ghana. The refined fuel is then transported to the military installations via fuel trucks. In case of emergencies, air tankers are deployed to resupply the installations directly. The delivery and reception of fuel are modeled by AnyLogic fluid library elements, as shown in Figure 12.

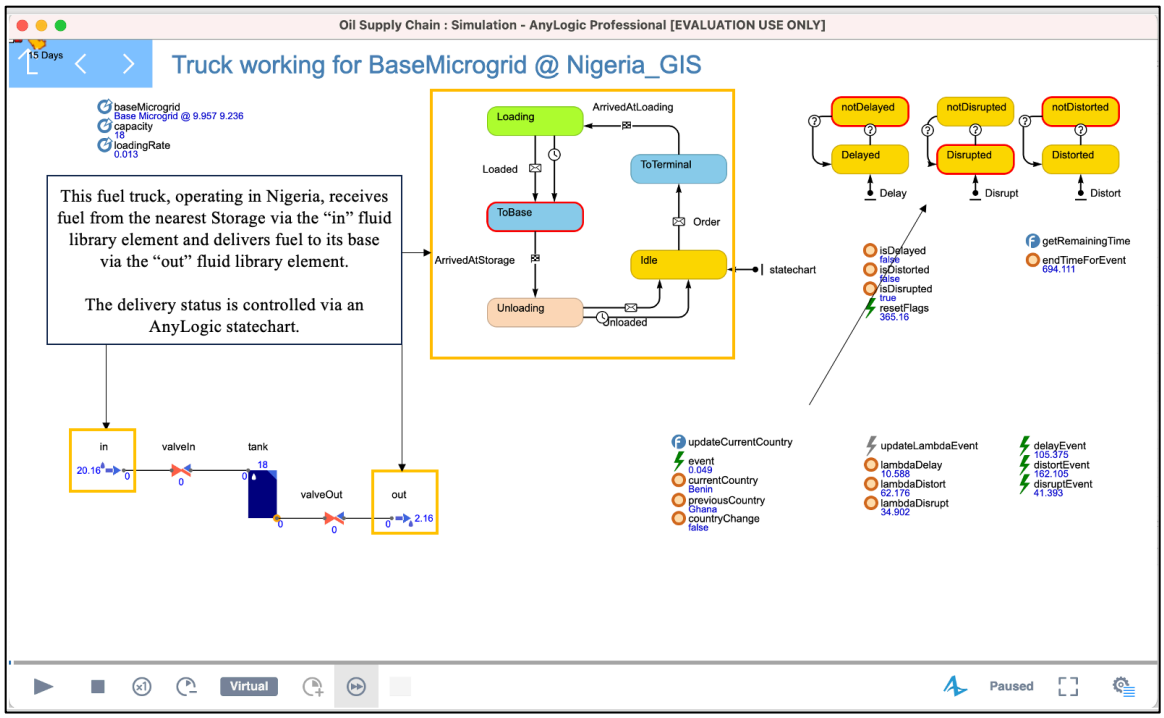


Figure 12. Receipt and Delivery of Fuel via Fluid Library Elements

Military installations' demand for this fuel is modeled stochastically, varying around a specified Mean Hourly Power Load, and discussed in more detail in a subsequent subsection. To meet this demand, the installations follow a simple inventory policy and reorder fuel from the nearest terminal once levels dip below a set threshold.

The model responds to this varying energy and fuel demand with its agents (storages, refineries, military installations, pipelines, trucks, and tankers). The agents' configurations are GIS-driven to enable the simulation of regional attacks to SCNs. The model evaluates the projected 10-year fuel disruption to inform microgrid planning for the specified military bases. Although the model's geographic and numeric values are notional, it explicitly defines the supply route from the Netherlands to Ghana by truck and identifies potential risks, such as targeted terrorist attacks on roads and vendors.

a. Tankers

Tankers transport crude oil from crude oil fields to the port in Rotterdam, Netherlands. The tankers receive fuel according to an AnyLogic fluid process, as pictured in Figure 13. Tankers operate within generic published capacities and speeds per the set parameters (Staff 2020).

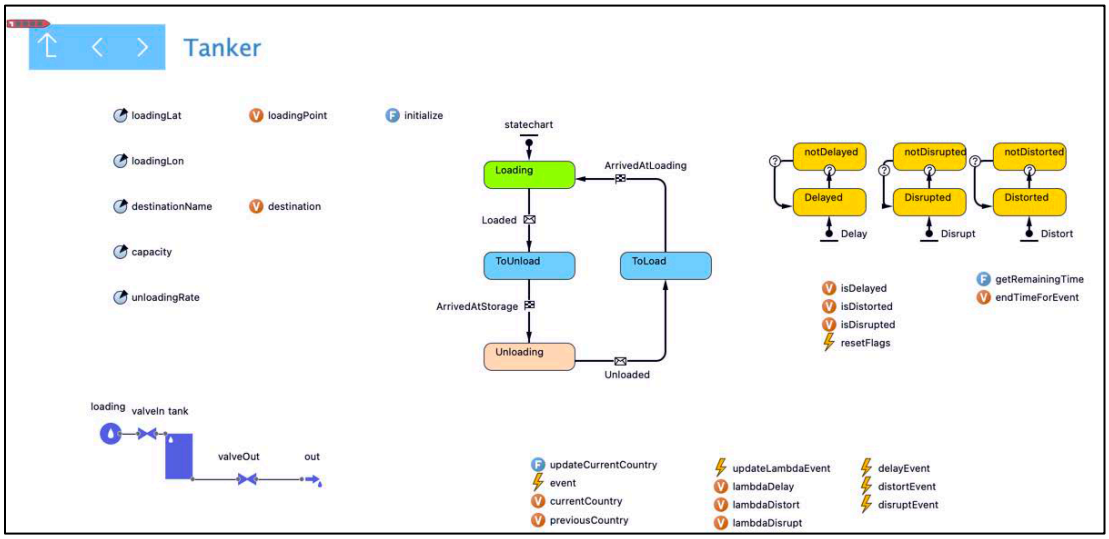


Figure 13. Tanker Agent Sample Configurations

b. Refineries

The Rotterdam refinery receives crude oil from the tankers via its “in” fluid library element, pictured in Figure 14. Refinery parameters are set according to the Rotterdam Refineries published specifications (Port of Rotterdam 2023).

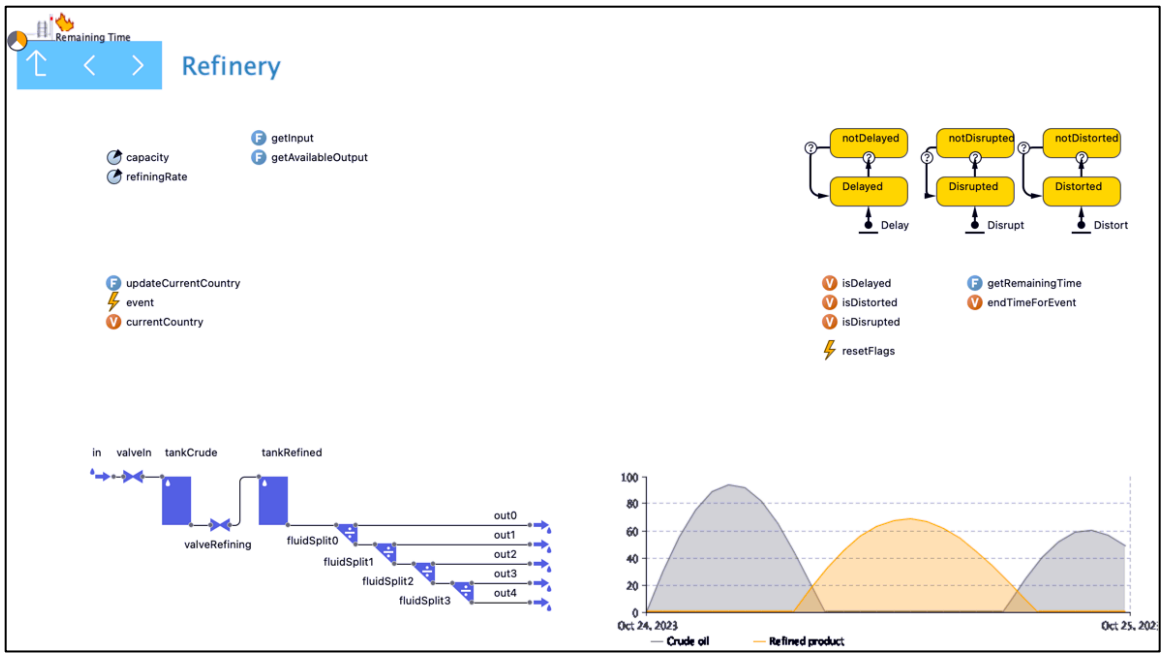


Figure 14. Refinery Agent Sample Configurations

c. Pipelines

A pipeline transports refined petroleum products from the Rotterdam refinery to storage facilities via fluid library elements, pictured in Figure 15. The simulation only specifies one fuel storage facility, although there are many in Rotterdam.

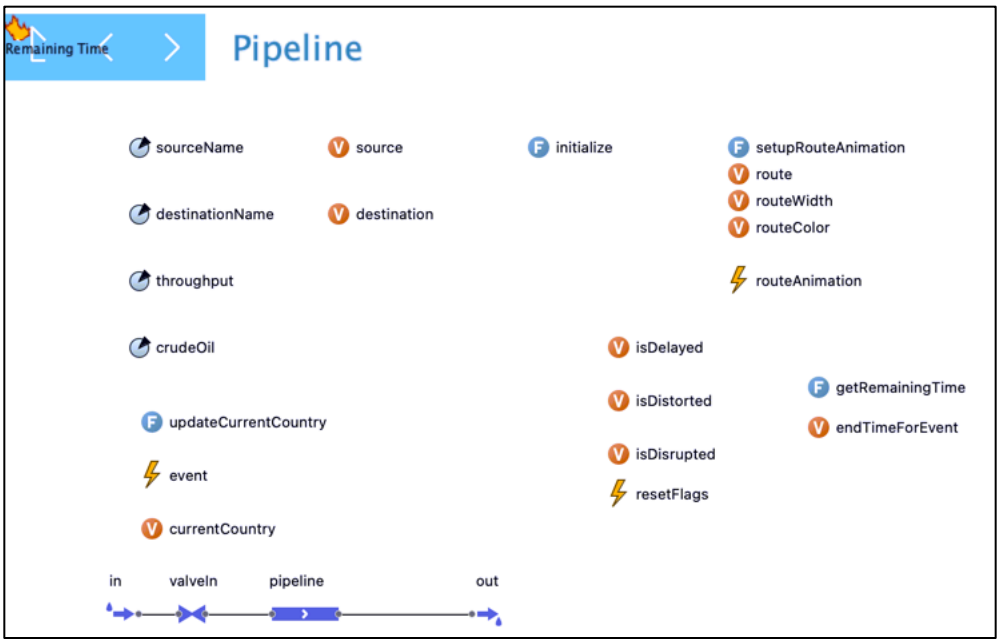


Figure 15. Pipeline Agent Sample Configurations

d. Fuel Tankers

Fuel tankers collect fuel from the storage facility at Rotterdam Port and transport it through a shipping trade route in the Atlantic Ocean to Tema Oil Port in West Africa's Ghana. Fuel Tankers operate within generic published capacities and speeds as per the set parameters (Staff 2020).

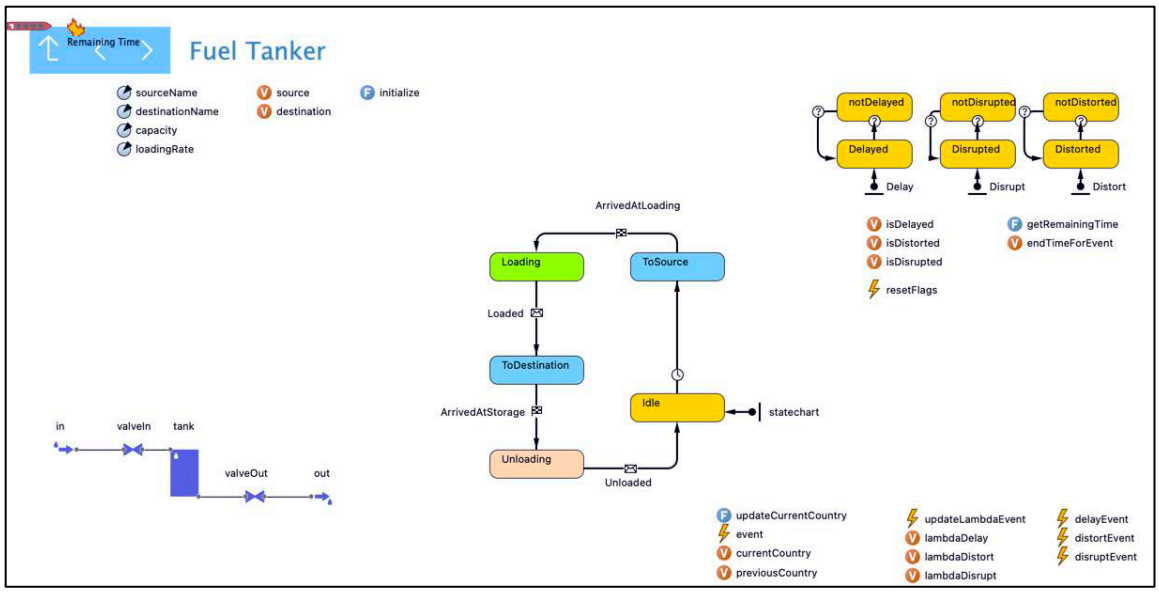


Figure 16. Fuel Tanker Agent Sample Configurations

e. Storages

Storages receive fuel from Fuel Tanker agents through their fluid library elements, as depicted in Figure 17. Storage parameters are based on the parameters published by Ghana's Tema Oil Terminal (Tema Oil Terminal 2023).

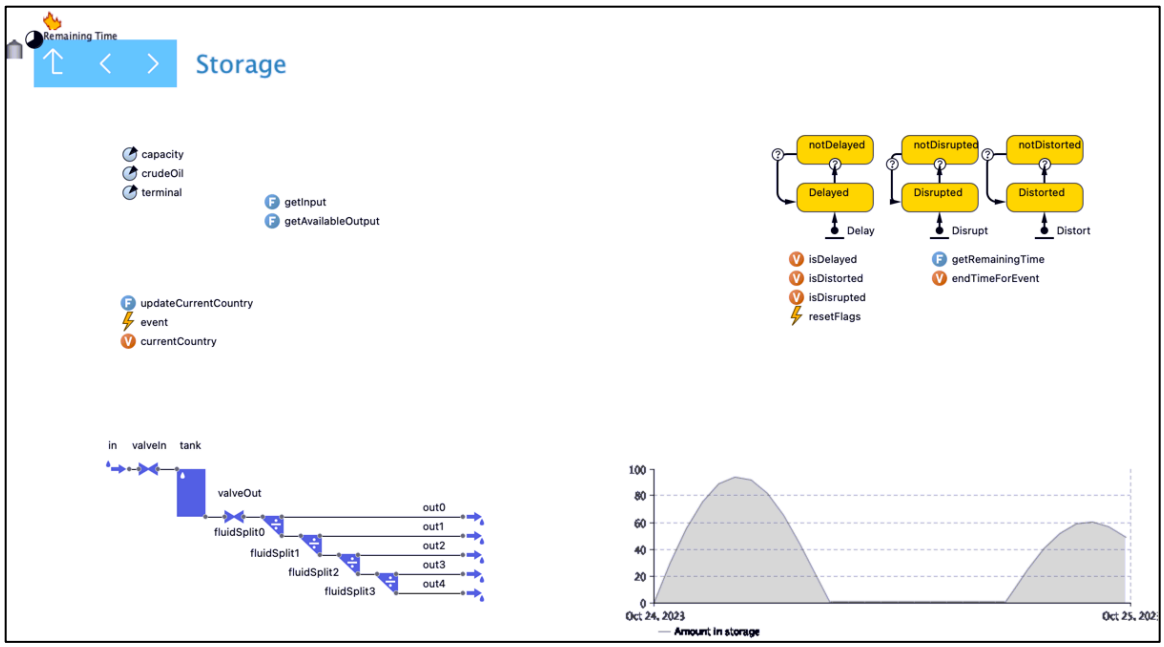


Figure 17. Storage Agent Sample Configurations

f. Military Installations (Base Microgrid Agents)

Military installation, or Base Microgrid agents in the simulation, manage the receipt and consumption of fuel for their respective microgrids. While each installation has a microgrid process flow within it, the purpose of the military installation agent is to manage fuel. The installation first signals varying energy demand and subsequent fuel consumption through AnyLogic "events" and fluid library elements. The 'VaryDemand' event, which modulates the energy load to mimic daily fluctuations, is pictured in Figure 18. The energy demand is varied and is adjusted using multipliers. The multipliers are based on whether it is day or night, with the day demand set between 90% and 110% and the night demand set between 45% and 55% of the Mean Hourly Power Load. The time of the day is determined by a flag called 'isSunlightAvailable,' which influences the multiplier used. The appropriate multiplier is applied to the Mean Hourly Power Load to calculate the new energy load. This mechanism simulates the variable power requirements within military installations.

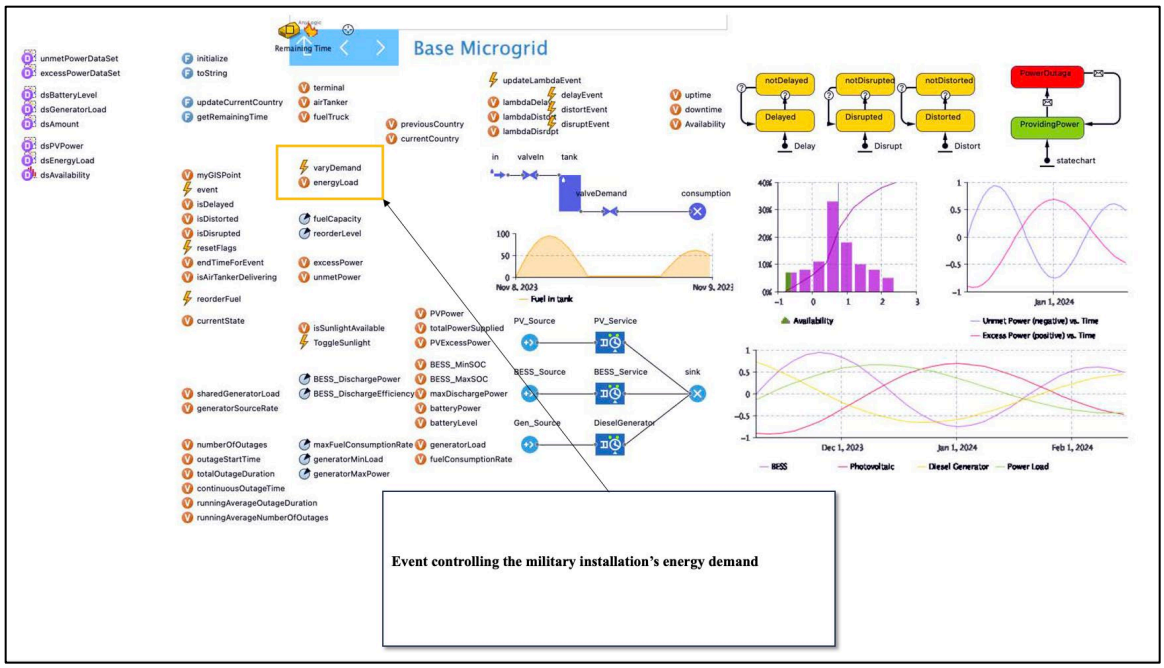


Figure 18. Simulating a Varying Energy Demand

The variable energy demand then signals the military installation's fuel demand through the amount of fuel consumed by the generators in their microgrids to meet this demand. When the fuel level in the installation tank falls below 42%, it sends a signal to the Fuel Truck agents to deliver fuel from the nearest storage facility. The model assumes that unlimited Fuel Truck agents are available if the installation is not under attack. For simulation purposes, a fuel reorder threshold of 42% ensures that each installation maintains a seven-day fuel supply on hand with no disruptions to the supply chain. The calculations are as follows.

1. Determine the total fuel consumption per day:

1. If the consumption rate is 5 gallons per hour, then the daily consumption is:
2. 5 gallons / hour × 24 hours / day = 120 gallons / day

2. Calculate the seven-day consumption to maintain a 100% fuel level:

3. 120 gallons / day x 7 days = 840 gallons

3. Considering the lead time:

4. Since the lead time is a 73-hour drive, which is approximately three days, the installation should reorder fuel when it has three days of fuel left.

5. 120 gallons / day x 3 days = 360 gallons

4. Calculate the reorder point:

6. The reorder point as a percentage of the full seven-day supply is:

7. (360 gallons) / (840 gallons) × 100 % ≈ 42.86 %

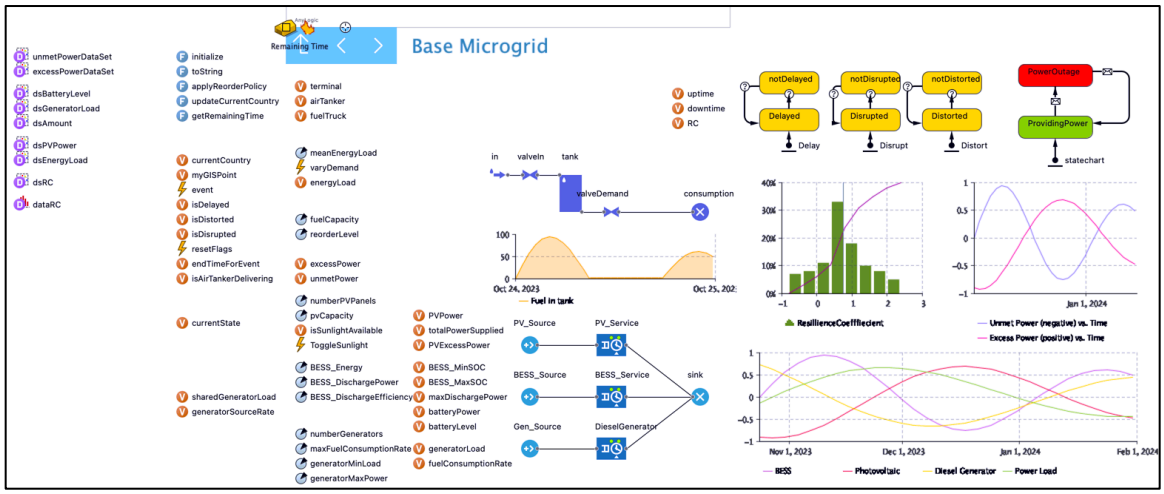


Figure 19. Base Microgrid Agent Sample Configurations

g. Fuel Trucks

Fuel Truck agents are generated based on the demand from the Base Microgrids. They deliver and receive fuel via fluid library elements, as pictured in Figure 20. Fuel trucks travel along AnyLogic's organic GIS road maps, choosing the shortest route and traveling at 40 mph to Morocco, southern Morocco, Mali, and Nigeria. Fuel Truck parameters are based on the parameters published by Ghana's Tema Oil Terminal (Tema Oil Terminal 2023).

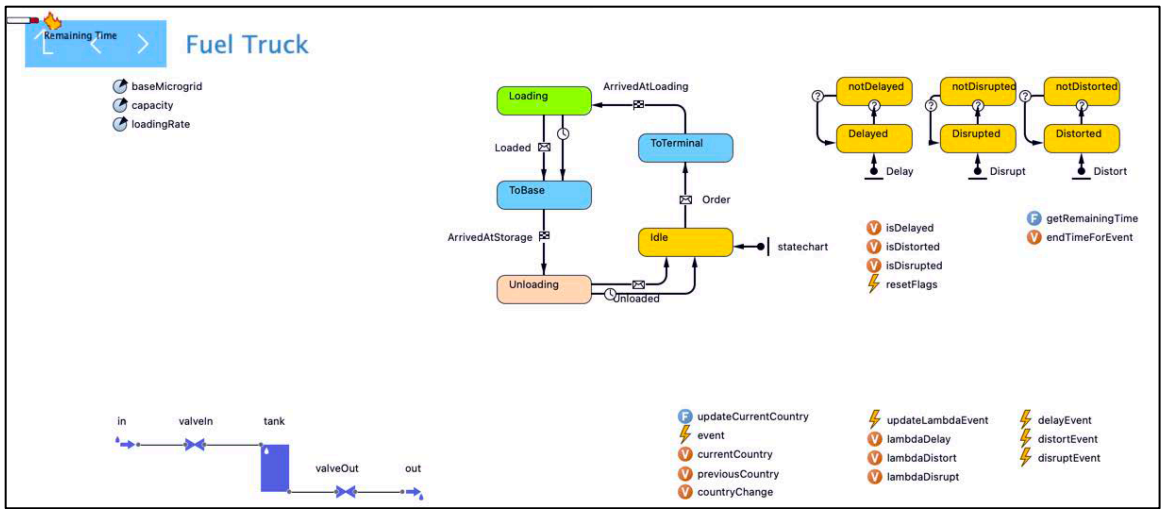


Figure 20. Fuel Truck Agent Sample Configurations

h. Air Tankers

Like Fuel Trucks, Air Tankers are also triggered by the Base Microgrid agents. Air Tankers, however, act only as emergency fuel supply if the Base Microgrid falls below its fuel threshold and does not receive fuel via its Fuel Truck (its primary route). The Air Tanker agents are modeled after a KC-130 aircraft designed to transport fuel Fields (Lockheed Martin 2015).

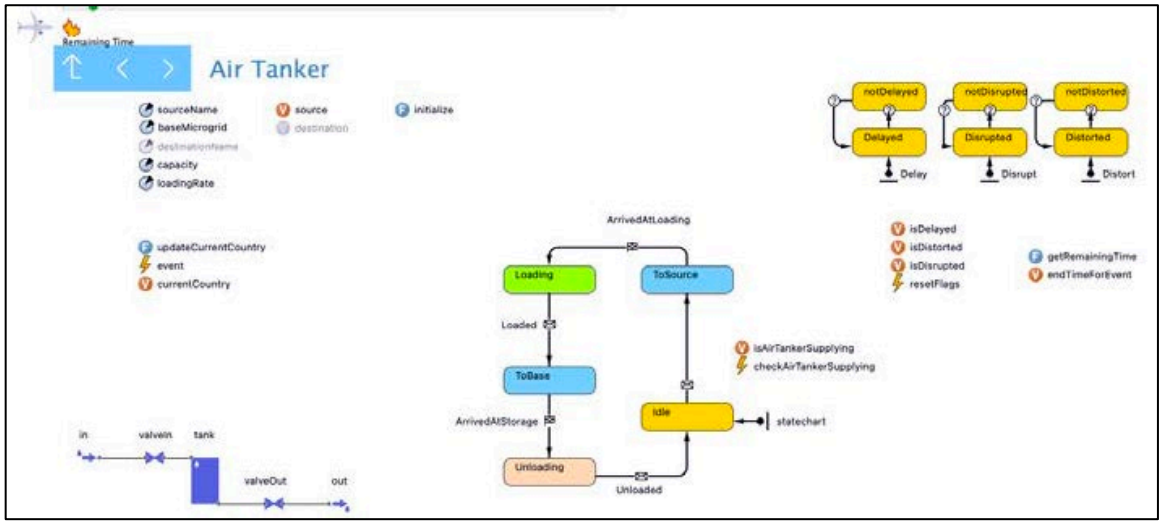


Figure 21. Air Tanker Agent Sample Configurations

2. Political Risk Assessment (Political Database Statistical Analysis)

The political risk assessment within this model quantifies the impact of terrorism-related risks on the petroleum supply chain, considering the various countries within the supply chain network. AnyLogic events query the GTD, assessing the frequency of terrorist attacks on specific targets and the consequent duration of their impacts on the supply chain. This assessment is managed by events and a statechart within AnyLogic, as depicted in Figure 22, which is applied universally across all agents in the model.

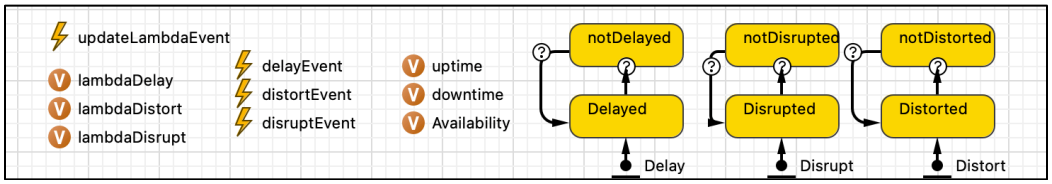


Figure 22. Base Microgrid Political Risk Assessment Events and Statechart

a. Estimating a Probability for Events

The probability of attacks on agents in the supply chain network is established by the "updateLambdaEvent" (see Figure 24), which queries the GTD for historical attacks within an agent's current location. The attacks are classified according to the framework by Talluri et al. (2013) into three categories—delays, disruptions, and distortions—each with distinct implications for the supply chain:

1. **Delays** affect the timing of operations, potentially leading to compounded downstream impacts.
2. **Disruptions** signify severe interruptions that may transform or halt the supply chain.
3. **Distortions** represent deviations from expected supply chain parameters but typically do not halt operations.

Table 3 matches the GTD attack categories with these risk types and provides justification for their potential impact on the supply chain.

Table 3. Applying Distortion, Delay, and Disruption times to GTD Target Types. Adapted from Talluri et al. (2013).

GTD Target	Risk Category	Justification
Educational Institution	Distortion (<3 days)	Unlikely to affect the oil supply chain directly but could cause social distortions.
Food or Water Supply		Indirectly affects manpower and resource allocation; could distort supply schedules.
Journalists & Media		Media coverage could distort public perception and indirectly affect supply chain decisions.
NGO		Activism could lead to distortions in policy affecting the oil supply chain.
Private Citizens & Property		Public opinion or protests could distort supply chain decisions.
Religious Figures/Institutions		Unlikely to affect the oil supply chain directly but could cause social distortions.
Tourists		Unlikely to affect the oil supply chain directly but could cause local or social distortions.
Abortion Related		Unlikely to affect the oil supply chain directly but could cause social distortions.
Utilities	Delay (3-7 days)	Utility issues at refineries or ports could delay oil processing or shipments.
Business		Corporate decisions or financial issues could delay oil production or shipment.
Transportation		Delays in transportation directly affect the timely delivery of oil.
Violent Political Party	Disruption (1-2 weeks)	Could lead to disruptions in the country's stability, affecting the oil supply chain.
Unknown		Uncertain risks could disrupt supply chains when the source or nature is not known.
Terrorists/Non-state Militia		Attacks could disrupt or halt the oil supply chain directly.
Airports and Aircraft		Critical for oil transportation; attacks could halt or divert shipments.
Police		Law enforcement is critical for protecting supply routes; attacks could disrupt this security.
Military		Directly related to the end user; attacks could disrupt the ability to receive oil.
Maritime		Attacks or piracy could disrupt oil shipments by sea.
Government (General)		Regulatory changes could disrupt the entire oil supply chain.
Government (Diplomatic)		Diplomatic incidents could result in sanctions or supply stoppages.
Telecommunication		Critical for coordinating oil supply; attacks could halt or misdirect shipments.

The model then employs a Poisson distribution with a mean rate parameter, lambda (λ), representing the expected number of delay events per year to simulate the random and independent occurrence of delay events over time. It does so by setting the Poisson process to ensure that:

1. Events are independent: The occurrence of one delay does not influence the timing of the next, which is based on the assumption terrorist activities are independent random variables.
2. The average rate is constant: λ is a fixed value, denoting the average frequency of delay events annually.
3. A single occurrence at a time: The model assumes that each event happens discretely over the simulation period.

For instance, with lambda set to 2, the model anticipates, on average, two delay events per year. This rate is derived from the Poisson probability mass function:

$$P(X = k) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

$P(X = k)$ represents the probability of observing k delay events in a year, with λ being the mean rate of occurrence and e denoting Euler's number (Keller 2017).

The Poisson distribution enables the accurate emulation of real-world scenarios where events, such as delays due to political risks, transpire randomly and independently over time.

The lambda values for each risk category are calculated as follows:

$$\lambda_{distortion} = \frac{Number\ of\ Distortion\ Attacks}{Total\ Years\ of\ Data}$$

$$\lambda_{delay} = \frac{Number\ of\ Delay\ Attacks}{Total\ Years\ of\ Data}$$

$$\lambda_{disruption} = \frac{Number\ of\ Disruption\ Attacks}{Total\ Years\ of\ Data}$$

For example, if Ghana experienced 200 ‘Distort,’ 100 ‘Delay’ type attacks, and 50 ‘Disrupt’ type attacks from 1970 to 2021, the lambda values would be:

$$\lambda_{distortion} = \frac{200}{2021 - 1970} = 4$$

$$\lambda_{delay} = \frac{100}{2021 - 1970} = 2$$

$$\lambda_{disruption} = \frac{50}{2021 - 1970} = 0.5$$

Therefore, any agent operating in Ghana will experience a distributed probability of 4 distortion attacks, two delay attacks, and 0.5 disruption attacks within any given simulation year.

These calculations are based on the GTD sample provided in Figure 23, highlighting the relevant queried columns “country” and “attack_type1_txt”.

eventid	iyear	imonth	iday	country_txt	region_txt	provstate	city	latitude	longitude	summary	attacktype1_txt
197000000001	1970	7	2	Dominican Republic	Central America & Caribbean	National	Santo Domingo	18.456792	-69.95116		Assassination
197000000002	1970	0	0	Mexico	North America	Federal	Mexico city	19.371887	-99.08662		Hostage Taking (Kidnapping)
197001000001	1970	1	0	Philippines	Southeast Asia	Tarlac	Unknown	15.478598	120.59974		Assassination
197001000002	1970	1	0	Greece	Western Europe	Attica	Athens	37.99749	23.762728		Bombing/Explosion
197001000003	1970	1	0	Japan	East Asia	Fukouka	Fukouka	33.580412	130.39636		Facility/Infrastructure Attack
197001010002	1970	1	1	United States	North America	Illinois	Cairo	37.005105	-89.17627	1/1/1970: Unknown African American assailants fired sever	Armed Assault
197001020001	1970	1	2	Uruguay	South America	Montevideo	Montevideo	-34.89115	-56.18721		Assassination
197001020002	1970	1	2	United States	North America	California	Oakland	37.791927	-122.2259	1/2/1970: Unknown perpetrators detonated explosives at t	Bombing/Explosion
197001020003	1970	1	2	United States	North America	Wisconsin	Madison	43.076592	-89.41249	1/2/1970: Karl Armstrong, a member of the New Years Gang	Facility/Infrastructure Attack
197001030001	1970	1	3	United States	North America	Wisconsin	Madison	43.07295	-89.38669	1/3/1970: Karl Armstrong, a member of the New Years Gang	Facility/Infrastructure Attack
197001050001	1970	1	1	United States	North America	Wisconsin	Baraboo	43.4685	-89.7443		Bombing/Explosion
197001060001	1970	1	6	United States	North America	Colorado	Denver	39.758968	-104.8763	1/6/1970: Unknown perpetrators threw a Molotov cocktail	Facility/Infrastructure Attack
197001080001	1970	1	8	Italy	Western Europe	Lazio	Rome	41.890961	12.490069		Hijacking
197001090001	1970	1	9	United States	North America	Michigan	Detroit	42.331685	-83.04792	1/9/1970: Unknown perpetrators set off a firebomb at the P	Facility/Infrastructure Attack
197001090002	1970	1	9	United States	North America	Puerto Rico	Rio Piedras	18.386932	-66.06113	1/9/1970: The Armed Commandos of Liberation claimed cr	Facility/Infrastructure Attack
197001100001	1970	1	10	East Germany (GDR)	Eastern Europe	Berlin	Berlin	52.50153	13.401851		Bombing/Explosion
197001110001	1970	1	11	Ethiopia	Sub-Saharan Africa	Unknown	Unknown				Unknown

Figure 23. GTD Sample Extract. Source: START (2022).

b. Assigning a Node

The model assigns a probability of attack based on an agent’s GIS location within the model. For instance, a fuel truck traversing Ghana is assigned a particular probability of experiencing an attack, which changes upon crossing the border into Nigeria. The fuel truck in Figure 23 will experience roughly ten delay attacks, 62 distortion attacks, and 35 disruption attacks within the simulation year. This number of attacks is high when modeled using a Poisson Distribution. However, it is used as an example because Nigeria has a comparatively high volume of terrorist attacks recorded in the GTD.

The model assumes that each agent in a given country will have the same probability of attack applied to it. For example, a fuel truck traversing a road in Ghana will have the same probability of attack applied to a fuel storage facility in Ghana. A more sophisticated analysis of the GTD could be applied in future work to capture more nuanced scenarios.

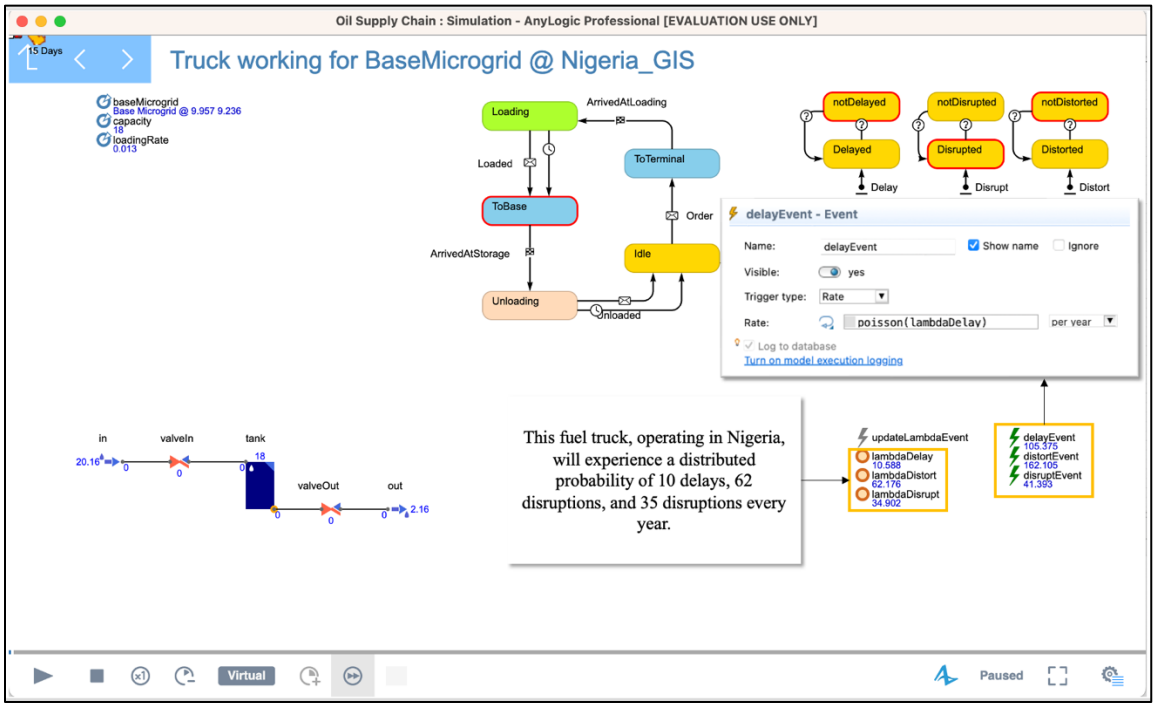


Figure 24. Assigning a Probability of Attack

The agents automatically propagate the effects of an attack to their downtrace agents. This is done by opening and closing each agent's fuel tank valves, which are triggered when an agent is attacked. The fuel truck in Figure 24 is experiencing a 15-day disruption. Its fuel tank valves will remain closed until the disruption ends.

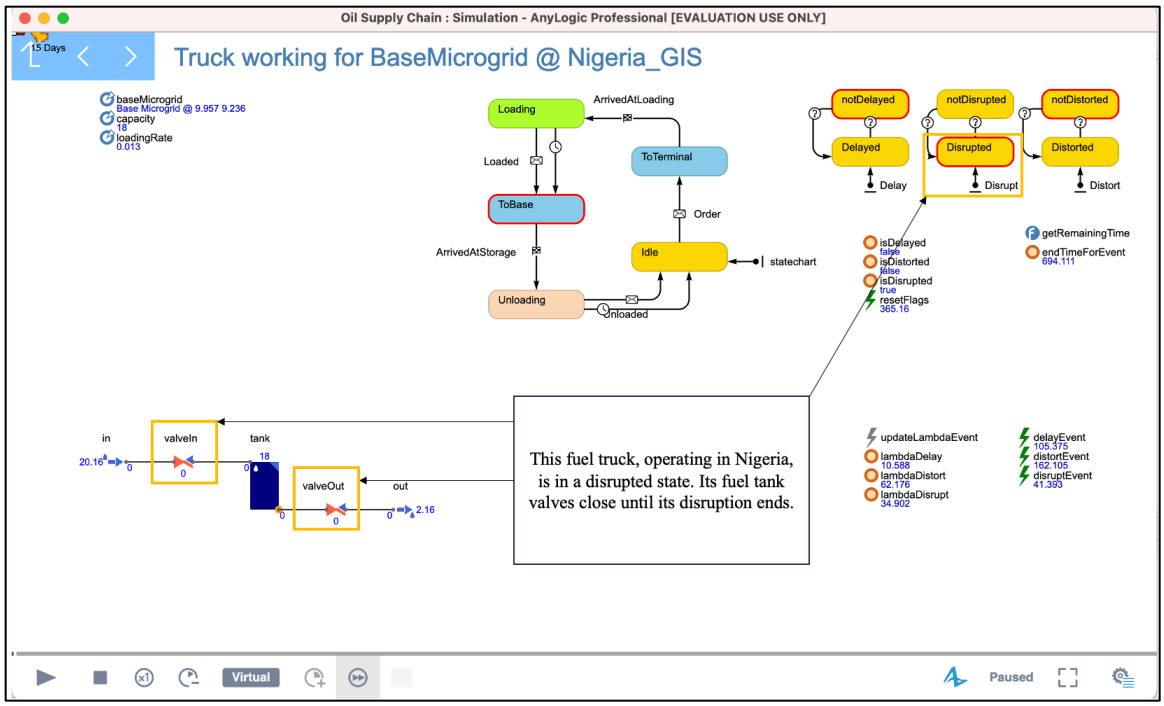


Figure 25. Simulating Supply Chain Interruption

The attacked fuel truck cannot deliver fuel to its subsequent agent, the base in Nigeria. Therefore, the probability that any given fuel truck is attacked implicitly assigns a probability of fuel supply interruption to its suppliers.

c. Assigning a Duration

If the agent is attacked, the event then assigns a time of interruption to the fuel in transit. The duration and node time of a distortion, delay, or disruption are adopted from the framework by Talluri et al. (2013), as presented in Table 3. If a node is "attacked," the AnyLogic event applies a random number between the associated duration upper and lower limits. For example, if a fuel truck is "delayed" in Ghana, it will stop fuel transit for a random duration between 3-7 days using:

$$Duration = Uniform\ (3,\ 7)$$

3. Microgrid Modeling (Process Modeling)

Military installations get their simulated power from components modeled in AnyLogic's process library, represented by a "microgrid'. The microgrid has distinct process blocks for each component to capture the dynamics between solar photovoltaic power, battery energy storage, and diesel generator power, as shown in Figure 26. The microgrid can sustain a 55-kWh load that depends on the time of day and depends on fuel delivery.

Based on the NPS Open-Source Microgrid Design Planning Tool, the model is designed with the same initial parameters and average energy demand (Giachetti et al. 2023). This approach serves two purposes: first, the Open-Source tool can be used to verify the simulation during development. Second, the 55-kWh load corresponds to the energy demand of an expeditionary Army Brigade staff element. In case of deployment to contingency locations in Africa, where redundant power sources are essential, a microgrid can provide the required backup.

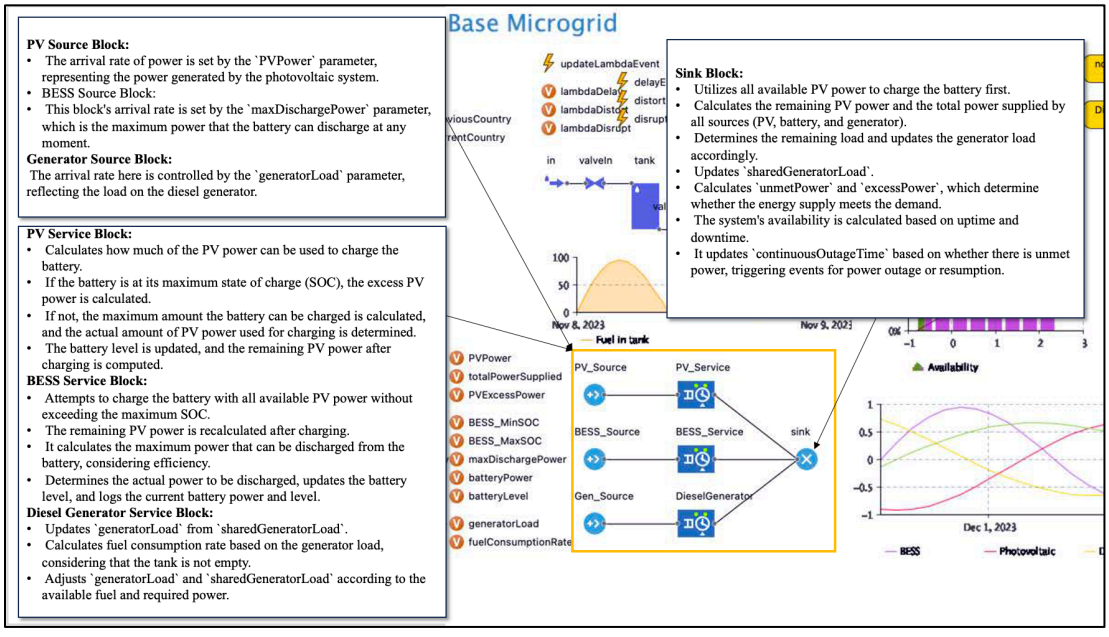


Figure 26. Microgrid Modeling

The Photovoltaic (PV) Source Block simulates solar power generation, with the "PVPower" parameter defining the rate of power arrival. The Battery Energy Storage System (BESS) Source Block indicates the maximum discharge capacity of the battery with the "maxDischargePower" parameter. The Diesel Generator Source Block simulates the generator's contribution to the power supply, with the "generatorLoad" parameter specifying its operational load. These components interact as a simplified microgrid system, where the diesel generator consumes fuel, the solar panels charge the battery system, and the sink block consumes the energy based on the energy demand. The system gives priority to battery energy first and then the generator.

a. Solar Photovoltaic Power

The PV Source Block captures the dynamics of solar power generation. The power input, dictated by the "PVPower" parameter, simulates the arrival of energy from sunlight in 12-hour increments. The PV Service Block reviews this power and determines how much can be used to charge the BESS. If the battery's SOC is at its maximum, the excess power is calculated and accounted for; otherwise, the simulation updates the battery level with the charge received and notes any remaining PV power. The PV panel parameters follow the generic sample on the Open-Source Microgrid Planning Tool, as shown in Figure 27.

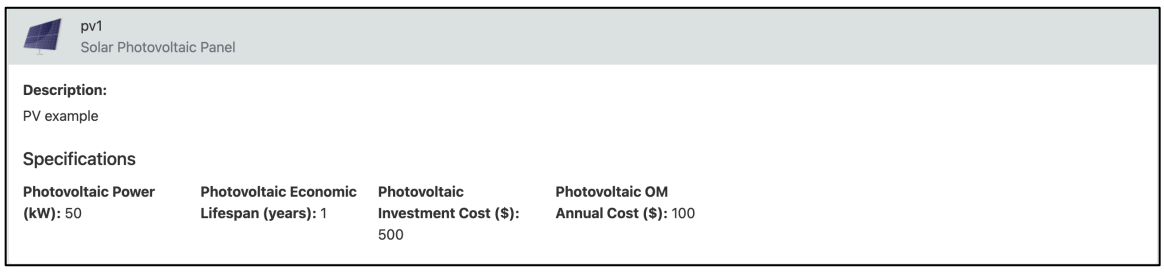


Figure 27. PV Power Parameters. Source: Giachetti et al. (2023).

b. Battery Energy Storage System (BESS)

The BESS Source Block, governed by the "maxDischargePower" parameter, represents the storage unit's capacity to provide power. It prioritizes using available PV

power for charging, ensuring the SOC does not exceed its limit. The BESS Service Block handles the discharge process, calculating the maximum and actual discharge power, reflecting the storage system's efficiency and the energy load's demands. The BESS parameters follow the generic sample BESS on the Open-Source Microgrid Planning Tool, as shown in Figure 28.

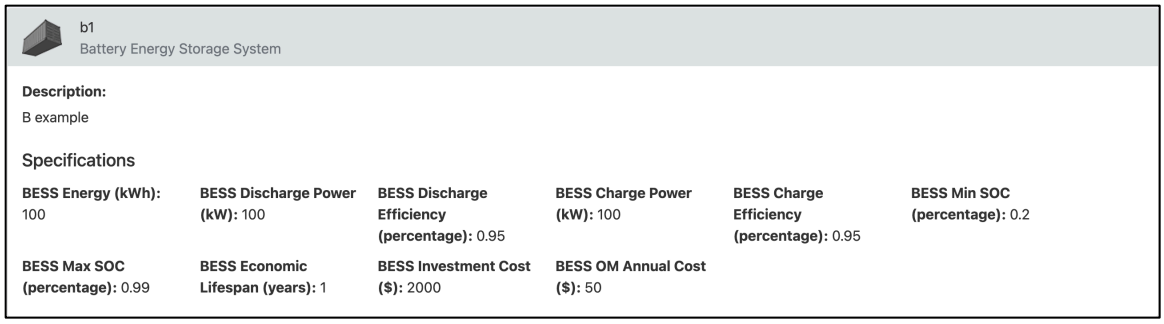


Figure 28. BESS Parameters. Source: Giachetti et al. (2023).

c. Diesel Generator

The Diesel Generator Source Block simulates additional power supply when renewable sources are insufficient. The "generatorLoad" parameter controls the load on the generator. The simulation adjusts this load in the Diesel Generator Service Block based on fuel availability and demand. The parameters detailed in Figure 28, such as the generator's power output, load, maximum load, and consumption rate, are integrated into the simulation to provide a realistic portrayal of the generator's performance and fuel efficiency.

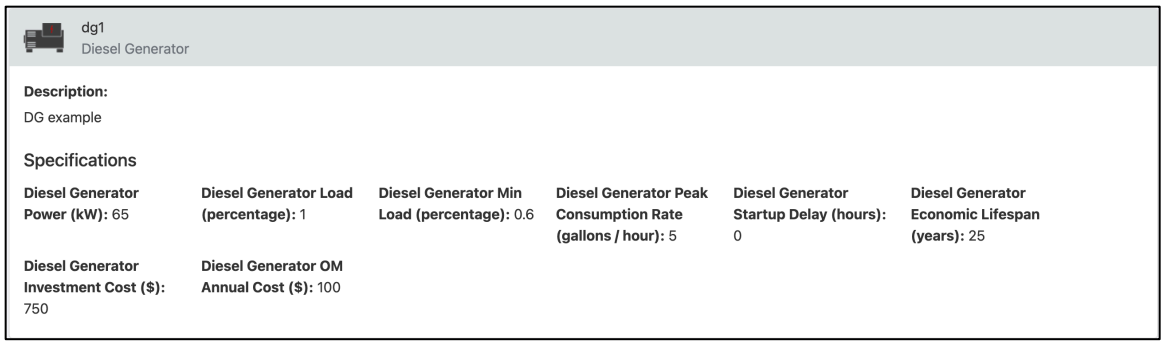


Figure 29. Diesel Generator Parameters. Source: Giachetti et al. (2023).

d. Microgrid Controller

The "sink" process block depicted in Figure 30 represents a simplified microgrid controller. The Sink Block initially utilizes all available PV power to charge the battery, then recalculates any excess PV power, and finally evaluates the total power supplied from all sources in relation to the energy load. It updates the generator load to respond to the remaining power demand, considering the fuel availability. The simulation monitors "unmetPower" and "excessPower" to gauge if the energy supply aligns with demand, contributing to system availability assessments and initiating suitable actions for power outages or resumption based on the power balance.

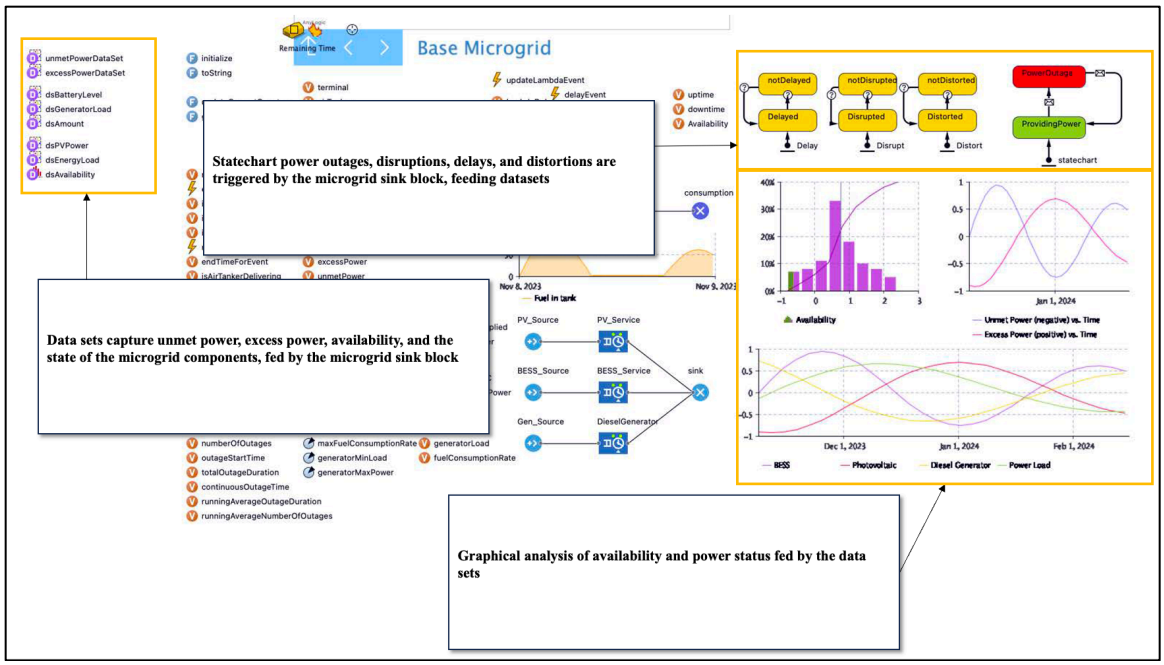


Figure 30. Microgrid Controller Simulation

4. Model Outputs

The model generates a histogram of the Availability, mean downtime, and number of power outages, as shown in Figure 31.

The simulation also collects the total disruption, delay, and distortion times, fuel outage instances and durations, and total supply chain flow to databases.

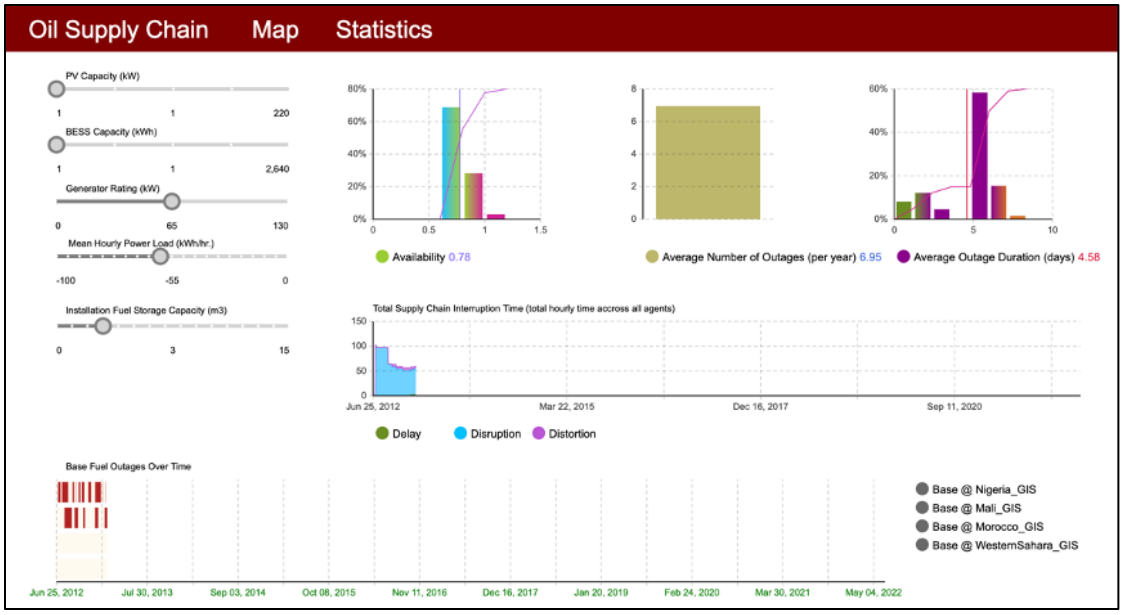


Figure 31. Sample Model Outputs

IV. DATA ANALYSIS

Data is gathered from AnyLogic Main and Optimization Experiments to determine the minimum resources required to achieve maximum power availability for a base over ten years. A Main Experiment provides the metrics displayed in the average availability, power outage number, and duration for all military installations. An Optimization Experiment is a factorial experiment to determine the best parameter values to achieve maximum availability. A total of five experiments are run.

1. **Main Experiment (Baseline):** The first experiment is conducted using the initial parameters from Table 2 and with no political disruptions. The installations in this first baseline experiment rely solely on diesel generators for power and are intended to demonstrate the base has stable energy in the absence of disruptions.
2. **Optimization Experiment 1 (Status Quo):** The first optimization experiment is conducted for the status quo, where all bases in the simulation rely on diesel-powered generators, and there are disruptions in the supply chain. This experiment determines the optimal fuel storage and generator capacity for the highest availability in the face of supply chain disruptions. These parameters, modified from the baseline, are later used in a main experiment for further analysis.
3. **Optimization Experiment 2 (Microgrid):** The second optimization experiment is conducted with a microgrid that includes diesel generators, solar power, and a battery storage system. This experiment determines the optimal combination of solar, battery, diesel power, and fuel capacity required to achieve the highest availability. These parameters, modified from the baseline, are later used in a main experiment for further analysis.
4. **Main Experiment (Status Quo):** This experiment presents the availability, number, and duration of power outages in military installations that rely solely on diesel power generation based on Optimization Experiment 1 parameters.

5. **Main Experiment (Microgrid):** This experiment shows the availability, number, and duration of power outages in military installations that have solar power and battery storage systems in addition to diesel power generation, based on the parameters of Optimization Experiment 2.

The optimization experiments are run using a factorial design. This experiment design determines the optimal parameter values for the DoD's R_C threshold of 0.99 or 99% availability. Each factor is tested at low and high levels, and the parameter values are determined based on operational constraints and objectives. Finally, the optimized values for Experiment 1 and Experiment 2 are run in the model's primary simulation for more detailed data analysis of the microgrids' availability, duration, and number of power outages over ten years.

A. MAIN EXPERIMENT (BASELINE)

The baseline experiment, pictured in Figure 32, shows a 100% availability over ten years, with zero power outages for military installations using only a 65-kW rated diesel generator with a seven-day fuel reserve under no political disruptions. This experiment indicates that the fuel supply chain works as intended in the absence of supply chain attacks.

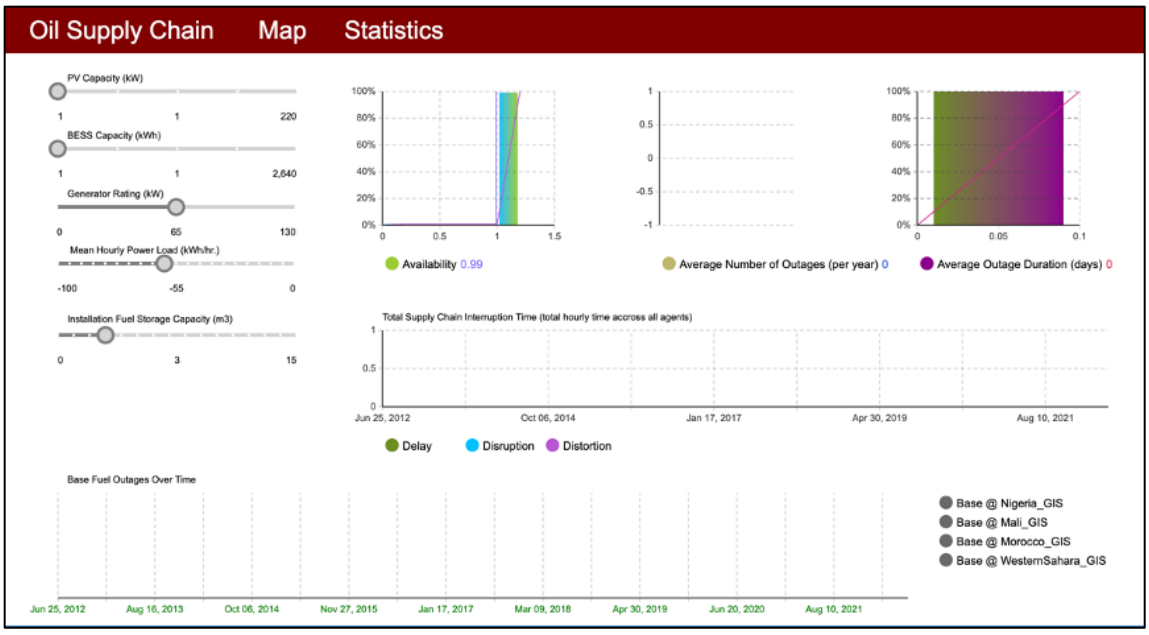


Figure 32. Main Experiment (Baseline)

B. OPTIMIZATION EXPERIMENT 1 (STATUS QUO)

Optimization Experiment 1 determines the optimal fuel storage capacity and generator power output to achieve above 99% availability. There is no battery or PV power for this experiment. The parameter maximum and minimum values are pictured in Figure 33.

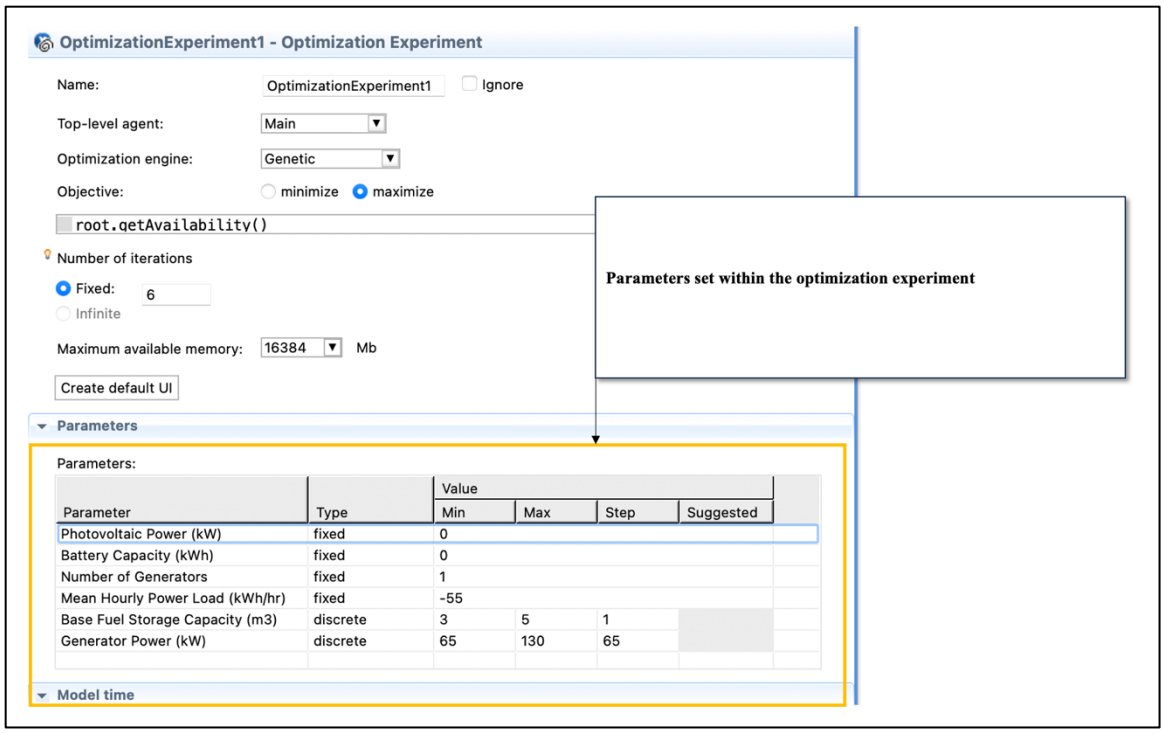


Figure 33. Optimization Experiment 1 (Status Quo) Parameters

1. Rationale for Parameter Selection

- **Diesel Generation Capacity:** The capacity of diesel generators is set relative to the Mean Hourly Power Load of 55-kWh/hr. The minimum parameter value of 65-kW is set to the minimum generator rating used in this simulation. The high capacity is set at twice the generator rating to accommodate peak demands or unexpected surges in usage.

- **Military Installation Fuel Storage Capacity:** The low fuel storage capacity is set to provide seven days of operation, as is standard for most installations in Africa. The high capacity extends this reserve to twelve days, providing a buffer against supply chain disruptions.

2. Optimization Results

Even with optimization, a 65-kW generator and fuel storage capacity of 5 cubic meters (roughly ten days of fuel reserve) do not meet the DoD's 99% availability threshold, as shown in Figure 34.

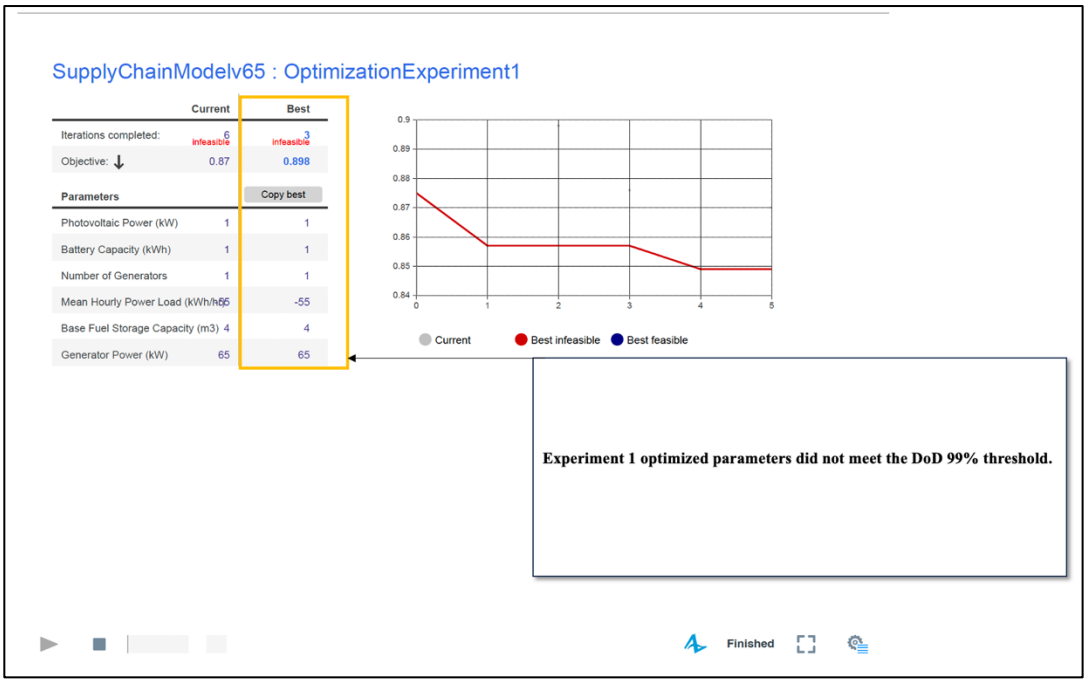


Figure 34. Optimization Experiment 1 (Status Quo) Results

A second simulation is carried out to determine the necessary fuel storage capacity for installations to ensure uninterrupted power in politically unstable areas. The simulation reveals that to meet the DoD threshold, the installations require 15 cubic meters of fuel to maintain a reserve of approximately 32 days, as illustrated in Figure 35.

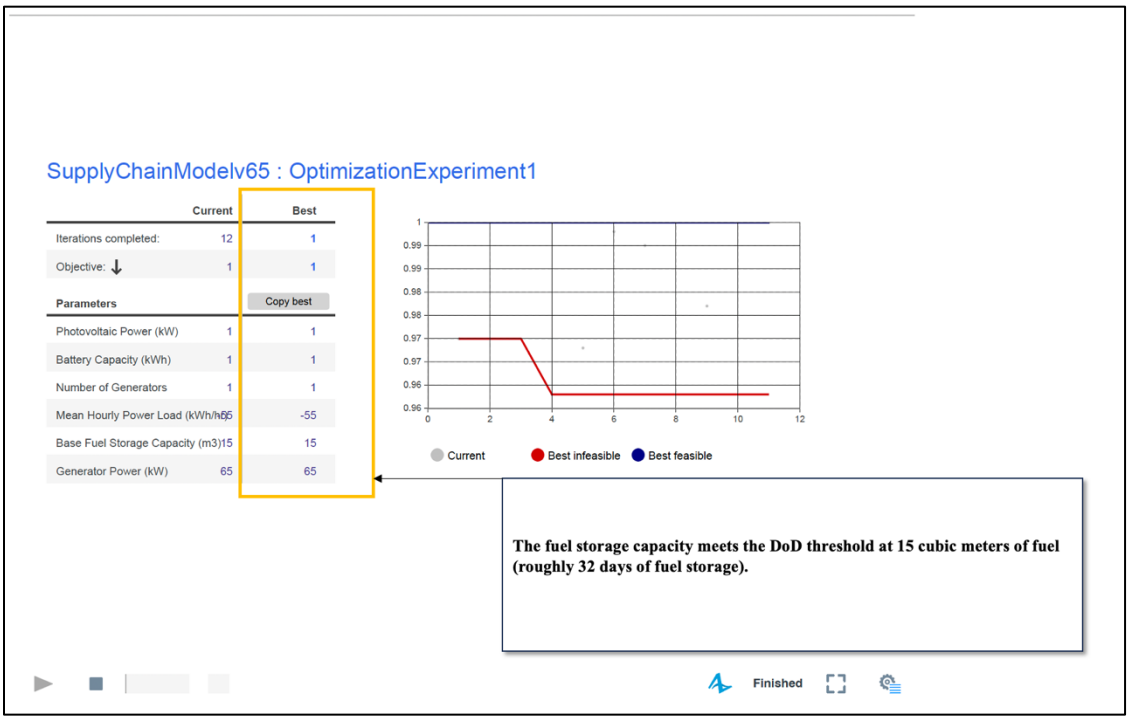


Figure 35. Optimization Experiment 1 (Expanded Fuel Storage Capacity)

C. OPTIMIZATION EXPERIMENT 2 (MICROGRID)

Optimization Experiment 2 determines the optimal capacity for generator, battery, PV power sources, and installation fuel storage to achieve above 99% availability. The parameter maximum and minimum values are pictured in Figure 36.

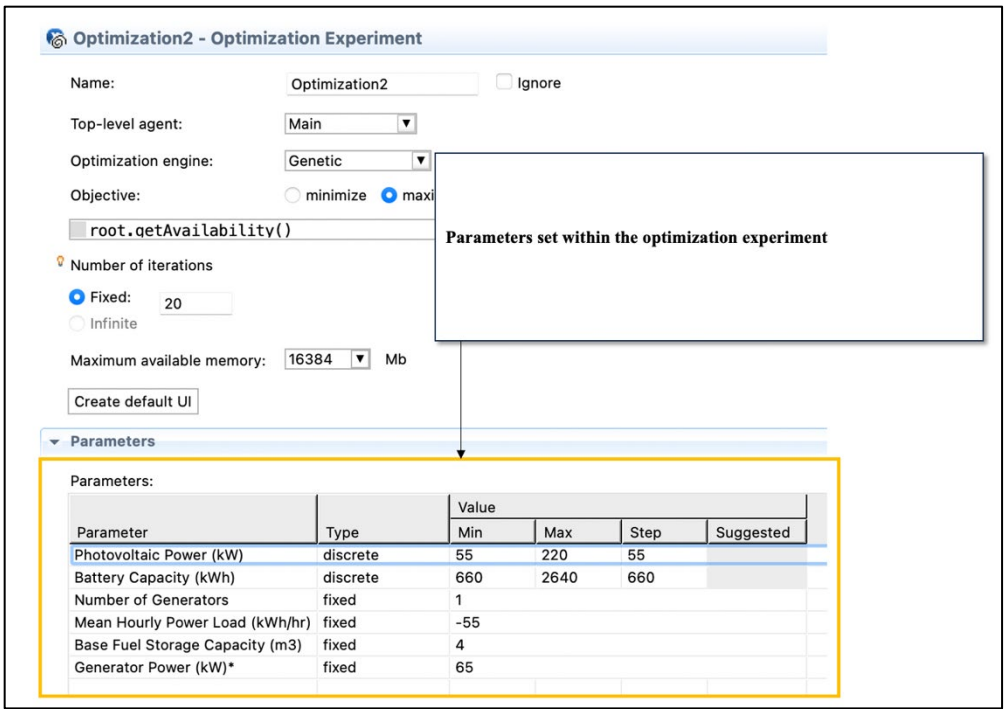


Figure 36. Optimization Experiment 2 (Microgrid) Parameters

1. Rationale for Parameter Selection

- **Diesel Generation Capacity:** The capacity of diesel generators is set relative to the Mean Hourly Power Load of 55-kWh/hr. The parameter value of 65-kW is set to the minimum generator rating used in this simulation.

- **Battery Capacity:** The capacity of battery storage is determined based on the daily energy demand to ensure that it can effectively balance fluctuations in daily consumption. The minimum capacity of 660-kWh is sufficient to meet the 55-kWh/hr. base power load for twelve hours, while the maximum capacity of 2,640 kWh can supply the base with power for two consecutive days.

- **Military Installation Fuel Storage Capacity:** The fuel storage capacity is set to provide seven days of operation, as is standard for most installations in Africa.

- **PV Size:** The PV low level ensures that the panels can recharge the batteries within 48 daylight hours, or four full days. The high-level simulates charging in 12 hours or one day of sunlight.

2. Optimization Results

According to Figure 37, a generator with a rating of 65-kW and a fuel storage capacity of 3 cubic meters can provide approximately seven days of fuel reserve, which is supplemented by a 660-kWh battery and a 165-kW PV system. This system successfully meets the DoD's 99% availability threshold.

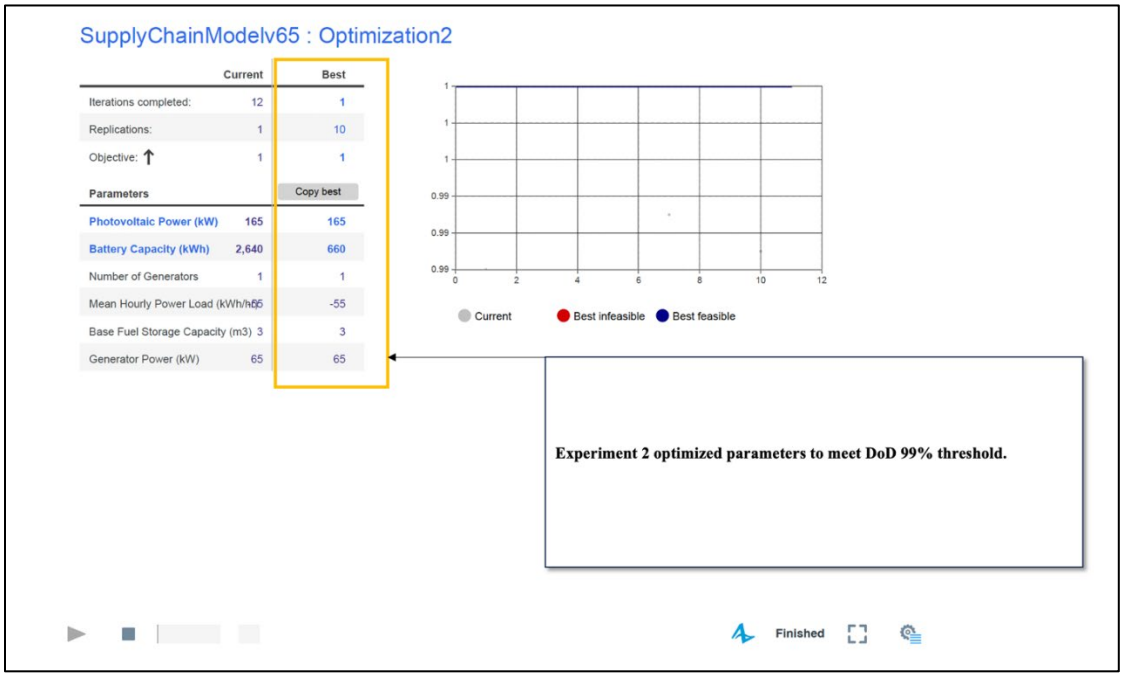


Figure 37. Optimization Experiment 2 (Microgrid) Results

D. MAIN EXPERIMENT (STATUS QUO)

The military installation operating solely on diesel generators and shows an average availability of 82%, with an average of seven eight-day interruptions to fuel delivery per year when subjected to supply chain attacks, as shown in Figure 38. The simulation applies roughly 70 hours per year of disruption time across the supply chain across all supply chain

nodes. This availability falls significantly below the DoD standard of 99%. Optimization Experiment 1's optimized parameters were used for this experiment, with a generator rated at 65-kW and a fuel storage capacity of 7 days contingency. It is expected that the experiment will not meet 99% availability, as the optimization experiment could not bring parameters above 88% without a 32-day contingency of fuel, which is unrealistic for most military installations.

Figure 38. Main Experiment (Status Quo)

It is important to note that this experiment simulates military installations in different regions, including Morocco, Western Sahara, Mali, and Nigeria, each facing unique political climates and risks. The simulation shows that despite having seven days of contingency fuel, Nigeria and Mali, the two most politically volatile regions, experience the most power outages. Nigeria's availability is 62%, experiencing the predominance of fuel outages due to disruptions, as shown in Figures 38 and 39.

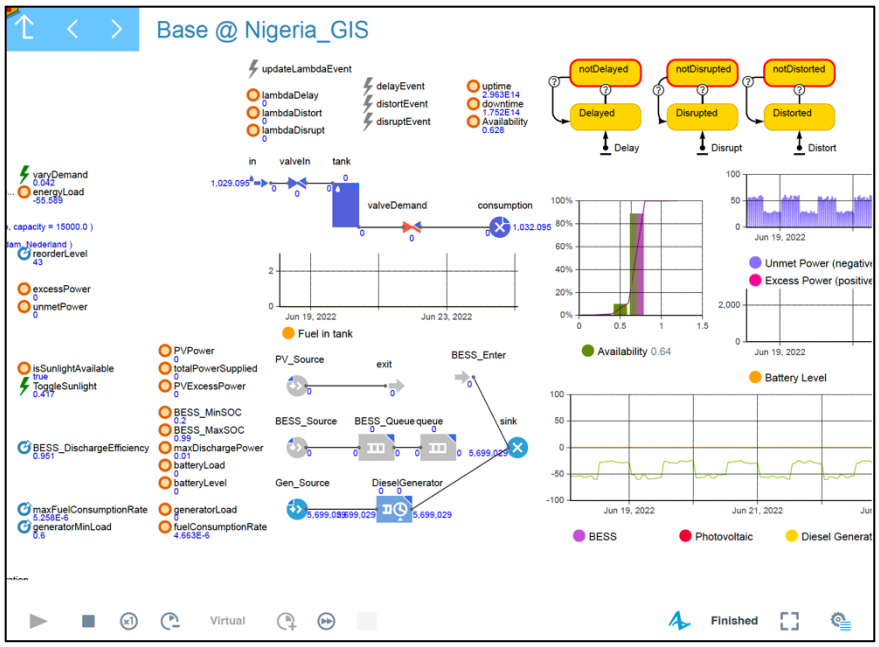


Figure 39. Main Experiment (Nigeria, Microgrid)

E. MAIN EXPERIMENT (MICROGRID)

Installing solar power and battery storage systems in military installations increases availability by over 10% and reduces outage duration to a negligible amount, as demonstrated in Figure 40. Both the Status Quo and Microgrid simulations experience roughly 70 hours of supply chain disruptions yearly across all agents, as they are conducted using the same seed value. Optimized parameters from Optimization Experiment 2, consisting of 165-kW of solar power and a 660-kWh battery, are used in the simulation.

Although the microgrid experiment results in 100% availability, it does not account for various factors that can impact solar power, such as weather conditions and maintenance. Further refinement in simulation is necessary to incorporate these factors and improve the accuracy of the microgrid design. Incorporating direct factors such as weather fluctuations can facilitate a more precise estimation of the optimal microgrid design.

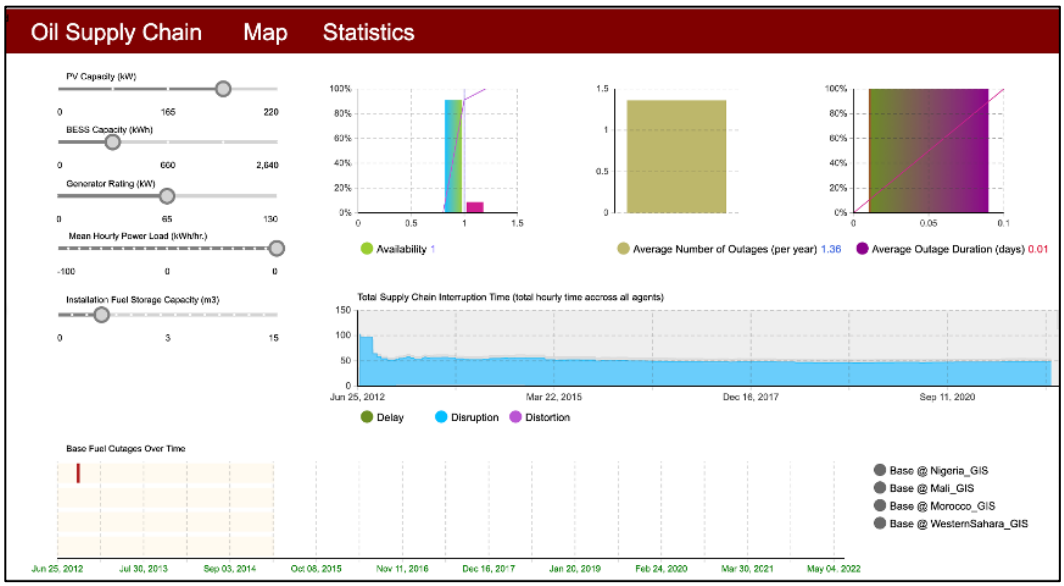


Figure 40. Main Experiment (Microgrid)

F. DATA ANALYSIS SUMMARY

The Data Analysis chapter of the thesis presents findings from AnyLogic experiments that assess power availability at military bases over a decade, as summarized in Table 4. The initial experiment indicates diesel generators maintain 100% availability without political interference. However, when factoring in supply chain attacks, optimization fails to achieve the 99% availability standard set by the DoD until fuel storage is increased to 32 days, which is unfeasible for many installations. When integrating a microgrid with solar and battery storage, however, the improved setup achieves 100% availability, which, while it does not account for factors that could interrupt solar power supply, is a marked improvement over a fuel-reliant installation.

Table 4. Experiment Results Summary

	Main Experiment (Baseline)	Optimization Experiment 1 (Status-Quo)	Optimization Experiment 1 (Status-Quo) Expanded Parameters	Optimization Experiment 2 (Microgrid)	Main Experiment (Status-Quo)	Main Experiment (Microgrid)
	Diesel generators only, no political disruptions	*Finds the optimum fuel storage and diesel generation parameters for maximum availability with political disruptions.*	*Finds the optimum fuel storage and diesel generation parameters for maximum availability with political disruptions. Expanded parameters, as the first experiment was well below 99% availability.*	*Finds the optimum solar power, battery storage, fuel storage, and diesel generation parameters for maximum availability with political disruptions.*	*Availability with diesel only, post-optimization parameters*	*Availability with diesel, solar, battery, post-optimization parameters*
Parameters	*Initial parameters from Table 2*	*Optimization Range*	*Optimization Range*	*Optimization Range*	*Best parameters from Optimization Experiment 1*	*Best parameters from Optimization Experiment 2*
	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65	Generator Rating (kW): 65
	Fuel Storage (days): 7	Fuel Storage (days): 7-10	Fuel Storage (days): 10-32	Fuel Storage (days): 7	Fuel Storage (days): 7	Fuel Storage (days): 7
	Battery (kWh): 0	Battery (kWh): 0	Battery (kWh): 0	Battery (kWh): 660-2840	Battery (kWh): 0	Battery (kWh): 660
	PV (kW): 0	PV (kW): 0	PV (kW): 0	PV (kW): 55-220	PV (kW): 0	PV (kW): 165
Results	*Met availability threshold with no supply chain interruptions.*	*Unable to meet 99% between 7-10 days fuel*	*Unable to meet 99% with up to 32 days of fuel*	*Met 99% availability threshold with optimized parameters.*	*82% availability, below DoD standard even with optimized parameters.*	*Over 10% increase in availability with a microgrid when compared to a diesel-generator only installation.*
	Applied SCN Disruptions: 0	Applied SCN Disruptions: 70/yr	Applied SCN Disruptions: 70/yr.	Applied SCN Disruptions: 70/yr.	Applied SCN Disruptions: 70/yr.	Applied SCN Disruptions: 70/yr.
	Availability: 100%	Availability: 89%	Availability: 100%	Availability: 100%	Availability: 82%	Availability: 100%
	#Outages / yr.: 0	Generator Rating (kW): 65	Generator Rating (kW): 65	Battery (kWh): 660	#Outages / yr.: 7	#Outages / yr.: 1.36
	Avg outage duration (day) : 0	Fuel Storage (days): 10	Fuel Storage (days): 32	PV (kW): 165	Avg outage duration (days): 8	Avg outage duration (days): 0.01

V. CONCLUSION

A. DISCUSSION

This thesis develops a methodology to incorporate political risk into the energy security and resilience framework for Department of Defense (DoD) microgrids in Africa, using AnyLogic software for simulation. It assesses how political disruptions, like terrorism and protests, affect the energy resilience of military installations. Through simulation experiments, it evaluates the impact of such disruptions on the power availability of microgrids and the fuel supply chain's robustness. The study demonstrates that adding photovoltaic and battery systems can significantly improve power availability.

The simulation shows that adding PV and battery power to military installations in politically unstable regions can increase their power availability by an average of 10%. However, accommodating the necessary 660-kWh battery and 165-kW PV system requires considerable space. Specifically, the 18-kW PV would need roughly one-quarter acre for installation, considering the space for an efficient setup and the area needed for safety setbacks for the battery system, which is approximately one-fourth of a football field (Solar Optimum 2023; 2023; NAVFAC 2023). This extent of space, along with the requisite maintenance and security measures, may not be feasible in the challenging terrains of Africa. The alternative, which involves increasing the base fuel contingency from a seven-day supply to a 32-day supply, also significantly increases real estate costs. Despite this, it remains a critical option as political events can severely disrupt fuel supply chains, impacting military operations. Hence, ongoing scenario analysis is essential for informed decision-making in energy security within politically volatile regions.

B. FUTURE WORK

This thesis acknowledges the inherent limitations and simplifications of simulation. The political risk assessment in this approach relies on a Poisson distribution, which may not accurately reflect fluctuations in political activity over time or across regions. One can utilize more sophisticated risk analysis, such as time-based or decision-tree of political data to illuminate seasonal or variable-dependent trends affecting supply chain dynamics.

Enriching the model with databases such as the Uppsala Conflict Data Program can also capture a broader spectrum of political events, extending to non-violent protests.

Secondly, the supply chain model is limited in scope. It assumes that all agents of a particular type operate under the same parameters and only considers a finite number of suppliers and consumers. For instance, the model assumes that the military installation in Nigeria has the same fuel storage capacity as the agent in Morocco, even though Morocco experiences fewer power outages. The development of a more detailed supply chain model would provide insights into the resilience of specific regions or scenarios more accurately.

Lastly, incorporating a more sophisticated microgrid can better inform trade-space analysis and incorporate the impacts of variables besides supply chains. A model that captures the impacted critical infrastructure of a military installation, such as operations centers or medical facilities, will better inform decision-makers about the severity and importance of energy resilience in susceptible areas. Also, research into the sizing of microgrid components can help provide an energy-savvy solution to austere regions that cannot accommodate large or maintenance-intensive components. With AnyLogic's ability to import and export Java code, existing microgrid models can be adapted to perform specific analyses.

C. SUMMARY

In conclusion, this thesis presents an advancement in integrating political risk into the broader context of energy security and military resilience, particularly for Department of Defense (DoD) microgrids in politically volatile regions. By developing an integrated model that assesses political risks, simulates fuel supply chain logistics, and evaluates microgrid systems using AnyLogic software, this research offers a quantitative foundation for strategic decision-making in energy planning. The utilization of AnyLogic, in conjunction with insights from the Global Terrorism Database, enables a comprehensive simulation of disruptions over a ten-year period, focusing on power outages and availability in DoD microgrids. The findings are revealing: a diesel generator-powered military installation in Africa, when supplemented with photovoltaic panels and battery storage systems, can achieve over a 10% increase in power availability and a significant reduction

in both the number and duration of power outages, even under politically unstable conditions. This is in stark contrast to installations relying solely on diesel generators. While the implementation of photovoltaic panels and battery storage systems is contingent upon factors like space, maintenance, and funding, these results underscore the necessity for resilient and robust energy solutions in volatile regions. This thesis provides a quantitative basis for decision-making and highlights areas where further research can refine and enhance the integration of political risk into energy security strategies.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. ANYLOGIC DOCUMENTATION

Model: SupplyChainModelv66

Description: The model supply chain simulates the delivery of oil to military installations in West Africa. The model starts with the transportation of crude oil from the Netherlands to Ghana's port storage through tankers. Then, the crude oil is transported through pipelines to refineries for processing into fuel, which is then piped to terminal storages and delivered to the installations through trucks. Demand is modeled stochastically, varying around a specified mean energy demand. The installations follow a simple inventory policy, reordering fuel from the nearest terminal once levels dip below a set threshold, while port storages lack a defined inventory policy with tankers continuously supplying crude oil. The model represents storages, refineries, military installations, pipelines, trucks, and tankers as distinct agents, and the configurations are GIS-driven to enable the simulation of regional "attacks" to SCNs. The model aims to evaluate the projected 10-year fuel disruption to inform microgrid planning for the specified military bases.

Although simplistic and hypothetical in its geographic and numeric values, the model underscores the supply route from the Netherlands to Ghana, then to the bases by truck, and identifies targeted terrorist attacks on roads and vendors as potential risks.

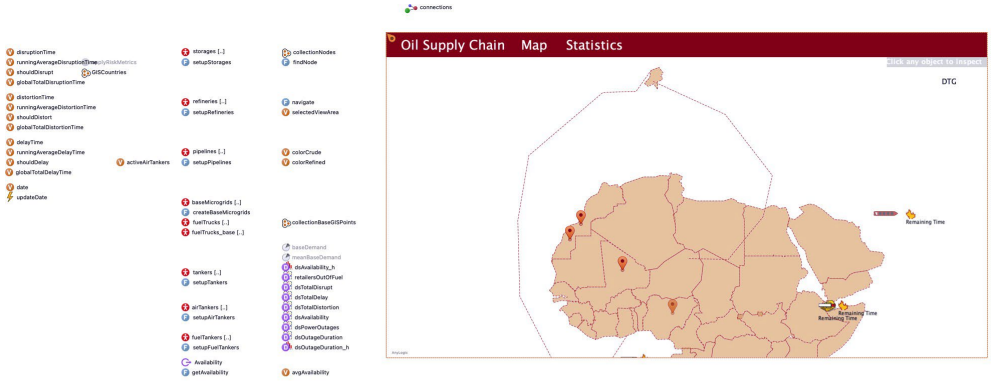
Name	Value
General	
Model time units	hours
System Dynamics solver	
Differention Equations Method	Euler
Algebraic Equations Method	Modified Newton
Mixed Equations Method	RK45+Newton
Absolute accuracy	1.0E-5
Time accuracy	1.0E-5
Relative accuracy	1.0E-5
Fixed time step	0.001
Advanced	
Java package name	oil_supply_chain_gis
File Name	/Users/abigailstaffnik/Models/SupplyChainModelv66/SupplyChainModelv66.alp
Description	
Description	The model supply chain simulates the delivery of oil to military installations in West Africa. The model starts with the transportation of crude oil from the Netherlands to Ghana's port storage through tankers. Then, the crude oil is transported through pipelines to refineries for processing into fuel, which is then piped to terminal storages and delivered to the installations through trucks. Demand is modeled stochastically, varying around a specified mean energy demand. The installations follow a simple inventory policy, reordering fuel from the nearest terminal once levels dip below a set threshold, while port storages lack a defined inventory policy with tankers continuously supplying crude oil. The model represents storages, refineries, military installations, pipelines, trucks, and tankers as distinct agents, and the configurations are GIS-driven to enable the simulation of regional "attacks" to SCNs. The model aims to evaluate the projected 10-year fuel disruption to inform microgrid planning for the specified military bases. Although simplistic and hypothetical in its geographic and numeric values, the model underscores the supply route from the Netherlands to Ghana, then to the bases by truck, and identifies targeted terrorist attacks on roads and vendors as potential risks.

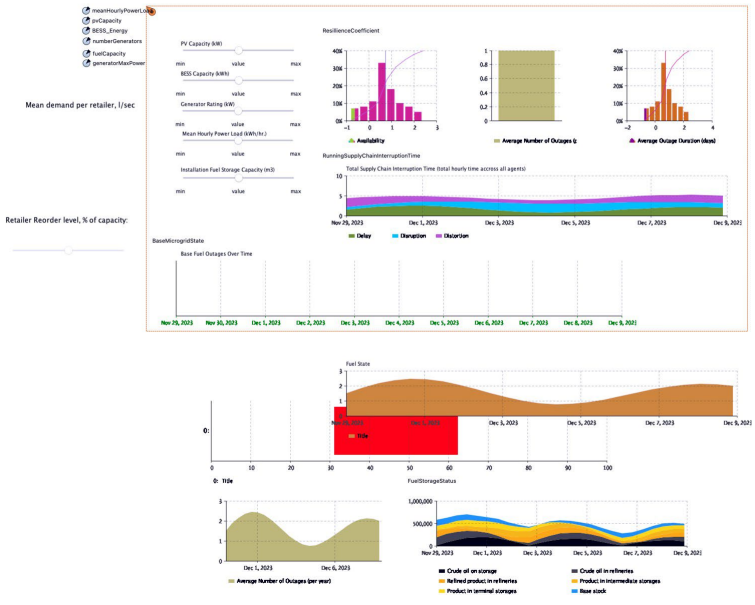
Agent Type: Main

Name	Value
Agent actions	

Startup code	// Call other setup functions
	setupStorages();
	setupRefineries();
	setupPipelines();
	createBaseMicrogrids();
	setupTankers();
	setupAirTankers();
	setupFuelTankers();

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(10 : MPS)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false
Space and network	
Enable steps	false
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false





Parameter: pvCapacity

Name	Value
General	
Array	false
Default value	165
Type	double
Show at runtime	true
Show name	true
Value editor	

Label	Photovoltaic Power (kW)

Name	Value
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: BESS_Energy

Name	Value
General	
Array	false
Default value	660
Type	double
Show at runtime	true
Show name	true
Value editor	
Label	Battery Capacity (kWh)
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: numberGenerators

Name	Value
General	
Array	false
Default value	1
Type	double
Show at runtime	true
Show name	true
Value editor	
Label	Number of Generators
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: meanHourlyPowerLoad

Name	Value
General	
Array	false
Default value	(-55 : PER_HOUR)
Unit	per hour
Show at runtime	true
Show name	true

Value editor	
Label	Mean Hourly Power Load (kWh/hr)
Editor control	Unit editor

Name	Value
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: fuelCapacity

Name	Value
General	
Array	false
Default value	(3 : CUBIC_METER)
Unit	cubic meters
Show at runtime	true
Show name	true
Value editor	
Label	Base Fuel Storage Capacity (m3)
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: generatorMaxPower

Name	Value
General	
Array	false
Default value	65*numberGenerators
Type	double
Show at runtime	true
Show name	true
Value editor	
Label	Generator Power (kW)
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Function: setupStorages

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	for (Storage storage : storages) {
	collectionNodes.add(storage);
	}

Advanced	
Access type	default
System dynamics units	false

Function: setupRefineries

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	for(Refinery refinery : refineries) {
	collectionNodes.add(refinery);
	}

Advanced	
Access type	default
System dynamics units	false

Function: setupPipelines

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	for(Pipeline pipe : pipelines) {
	pipe.initialize();
	}

Advanced	
Access type	default
System dynamics units	false

Function: findNode

Name	Value
General	
Return type	PipelineNode
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	

Body	PipelineNode result = null;
	for(PipelineNode node : collectionNodes) {
	if(node.locationName.equals(name)) {
	result = node;
	}
	}
	return result;

Advanced	
Access type	default
System dynamics units	false

Arguments:

Name	Type
name	String

Function: setupTankers

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	for(Tanker tanker : tankers) {
	tanker.initialize();
	}

Advanced	
Access type	default
System dynamics units	false

Function: navigate

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	selectedViewArea = viewArea;
	viewArea.navigateTo();
	groupMainMenu.setPos(viewArea.getX(), viewArea.getY());

Advanced	
Access type	default
System dynamics units	false

Arguments:

Name	Type
viewArea	ViewArea

Function: setupAirTankers

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true

Function body	

Body	for(AirTanker airTanker : airTankers) {
	  airTanker.initialize();
	}

Advanced	
Access type	default
System dynamics units	false

Function: createBaseMicrogrids

Name	Value

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	{
	  int airTankerIndex = 0;
	  int totalAirTankers = airTankers.size(); // Replace 'airTankers' with the name of your AirTanker population
	
	  for(GISPoint loc : collectionBaseGISPoints) {
	  // Setup baseMicrogrid
	  BaseMicrogrid baseMicrogrid = add_baseMicrogrids();
	  baseMicrogrid.initialize(loc);
	  traceln("Initializing BaseMicrogrid at location: " + loc);
	
	  // Setup truck for baseMicrogrid
	  FuelTruck truck = add_fuelTrucks_base();
	  truck.set_baseMicrogrid(baseMicrogrid);
	  baseMicrogrid.fuelTruck = truck;
	
	  // Assign an existing airTanker to baseMicrogrid
	  AirTanker assignedAirTanker = airTankers.get(airTankerIndex); // Replace 'airTankers' with the name of your AirTanker population
	  assignedAirTanker.baseMicrogrid = baseMicrogrid; // Directly setting the baseMicrogrid field of AirTanker
	  baseMicrogrid.airTanker = assignedAirTanker;
	
	  // Debug print
	  traceln("Fuel truck for BaseMicrogrid is: " + baseMicrogrid.fuelTruck);
	  traceln("AirTanker for BaseMicrogrid is: " + baseMicrogrid.airTanker);
	
	  // Add chart item
	  BaseMicrogridState.addDataSet(baseMicrogrid.dsAmount, "Base @ " + baseMicrogrid.myGISPoint.getName());
	
	  // Update airTankerIndex for next iteration
	  airTankerIndex = (airTankerIndex + 1) % totalAirTankers;
	  }
	}

Advanced	
Access type	default
System dynamics units	false

Function: setupFuelTankers

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	for(FuelTanker fuelTanker : fuelTankers) {
	fuelTanker.initialize();
	}

Advanced	
Access type	default
System dynamics units	false

Function: getAvailability

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Availability;
Advanced	
Access type	public
System dynamics units	false

Event: updateDate

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Assuming your date parameter is named “currentModelDate”
	date = date();

Variable: selectedViewArea

Name	Value
General	
Initial value	viewMap
Type	ViewArea
Show at runtime	true
Show name	true

Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: colorCrude

Name	Value
General	
Initial value	dimGray
Type	Color
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: colorRefined

Name	Value
General	
Initial value	orange
Type	Color
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: shouldDisrupt

Name	Value
General	
Initial value	true
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: shouldDistort

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: shouldDelay

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: globalTotalDisruptionTime

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: globalTotalDistortionTime

Name	Value
General	
Type	double
Show at runtime	true
Show name	true

Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: globalTotalDelayTime

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: activeAirTankers

Name	Value
General	
Initial value	new ArrayList<AirTanker>()
Type	ArrayList
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: avgAvailability

Name	Value
General	
Initial value	baseMicrogrids.Availability()
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: runningAverageDelayTime

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: runningAverageDisruptionTime

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: runningAverageDistortionTime

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: date

Name	Value
General	
Type	Date
Show at runtime	true
Show name	true
Advanced	

Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: disruptionTime

Name	Value
General	
Initial value	uniform(7 * day(), 21 * day());
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: distortionTime

Name	Value
General	

Name	Value
Initial value	uniform(1 * day(), 3 * day());
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: delayTime

Name	Value
General	
Initial value	uniform(3 * day(), 7 * day());
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Collection: collectionNodes

Name	Value
General	
Initial contents	{}
Initial contents	{}
Element class	PipelineNode
Collection class	ArrayList
Show at runtime	true
Show name	true
Advanced	
Access type	public
Save in snapshot	true

Collection: collectionBaseGISPoints

Name	Value
General	
Initial contents	{ Nigeria_GIS, Mali_GIS, Morocco_GIS, WesternSahara_GIS }
Initial contents	{ Nigeria_GIS, Mali_GIS, Morocco_GIS, WesternSahara_GIS }
Element class	GISPoint
Collection class	ArrayList
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Save in snapshot	true

Collection: GISCountries

Name	Value
General	
Initial contents	{ Algeria, Benin, BurkinaFaso, Cameroon, CentralAfricanRepublic, Chad, Congo, DemocraticRepublicoftheCongo, Djibouti, Egypt, Ethiopia, Gabon, Ghana, Guinea, Yemen, Uganda, Tunisia, Togo, Tanzania, Sudan, SouthSudan, Soomaaliyaالصومال, SierraLeone, Senegal, SaudiArabia, Niger, Nederlands, Morocco, Mauritania, Mali, Libya, Liberia, Kenya, IvoryCoast, Kenya, IvoryCoast, Nigeria }
Initial contents	{ Algeria, Benin, BurkinaFaso, Cameroon, CentralAfricanRepublic, Chad, Congo, DemocraticRepublicoftheCongo, Djibouti, Egypt, Ethiopia, Gabon, Ghana, Guinea, Yemen, Uganda, Tunisia, Togo, Tanzania, Sudan, SouthSudan, Soomaaliyaالصومال, SierraLeone, Senegal, SaudiArabia, Niger, Nederlands, Morocco, Mauritania, Mali, Libya, Liberia, Kenya, IvoryCoast, Kenya, IvoryCoast, Nigeria }
Element class	GISRegion
Collection class	ArrayList
Show at runtime	true
Show name	true
Advanced	

Access type	public
Save in snapshot	true

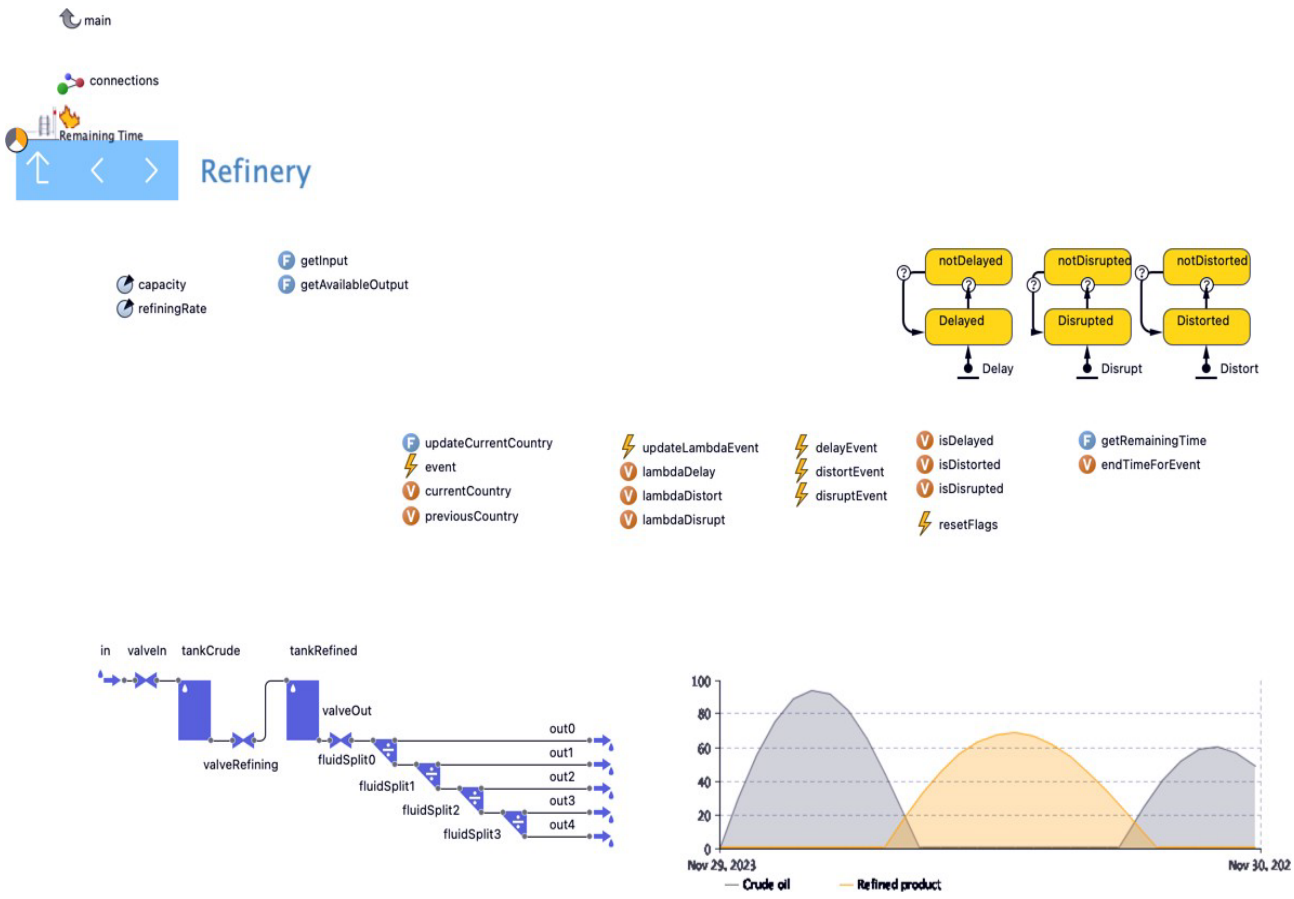
Link to agents: connections

Name	Value
General	
Show at runtime	false
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: Refinery

Name	Value
Agent actions	
Startup code	setLocation(main.map.searchFirst(locationName));
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(10 : MPS)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false
Advanced Java	
Generic	false

Name	Value
Advanced	
Extends other agent	ClassReference: oil_supply_chain_gis.PipelineNode (Resolved: true)
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false



Parameter: refiningRate

Name	Value
General	
Array	false
Default value	(2.17 : CUBIC_METER_PER_SECOND)
Unit	cubic meters / s
Show at runtime	true
Show name	true
Value editor	
Label	Refining rate, m3/sec
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: capacity

Name	Value
General	
Array	false
Default value	(164237710.23 : CUBIC_METER)
Unit	cubic meters

Show at runtime	true
Show name	true
Value editor	
Label	Capacity, m3
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Function: getInput

Name	Value
General	
Return type	FluidEnter
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return in;
Advanced	
Access type	default
System dynamics units	false

Function: getAvailableOutput

Name	Value
General	
Return type	FluidExit
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	

Body	if(! out0.isConnected()) return out0;
	if(! out1.isConnected()) return out1;
	if(! out2.isConnected()) return out2;
	if(! out3.isConnected()) return out3;
	if(! out4.isConnected()) return out4;
	error(“No available outputs at refinery”);
	return null;

Advanced	
Access type	default
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true

Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn't go negative
Advanced	
Access type	default
System dynamics units	false

Function: updateCurrentCountry

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = "Unknown";
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it's now part of BaseMicrogrid
	if (region.contains(x, y)) {
	newCountry = region.getTitle();
	break;
	}
	}
	
	// Check if the country has changed
	if (!newCountry.equals(currentCountry)) {
	currentCountry = newCountry;
	previousCountry = currentCountry;
	// Trigger condition for updateLambdaEvent should now evaluate to true
	}

Advanced	
Access type	public
System dynamics units	false

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout

Name	Value
Show at runtime	true
Show name	true
Action	

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Check if there is no disruption, distortion, or delay
	if (!isDelayed && !isDistorted && !isDisrupted) {
	  valveIn.open(); // Open the valveIn
	  valveOut.open(); // Open the valveOut
	} else {
	  valveIn.close(); // Close the valveIn
	  valveOut.close(); // Close the valveOut
	}

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	updateCurrentCountry(); // Update the current country based on the agent's location

Event: delayEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDelay) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDistort) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: disruptEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDisrupt) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: updateLambdaEvent

Name	Value
General	
Logging	true

EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties

Name	Value
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Loop through all instances of Refinery
	for (Refinery refinery : main.refineries) { // Assuming 'refineries' is the collection of Refinery agents
	String currentCountry = refinery.currentCountry;
	
	// Define the total number of years for which you have data
	double totalYears = 2021 – 1970;
	
	// Query for lambdaDelay
	double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Utilities," "Business," "Transportation"))
	.count();
	double lambdaDelay = countryDelayAttacks / totalYears;
	refinery.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Violent Political Party," "Unknown," "Terrorists/Non-state Militia," "Airports and Aircraft," "Police," "Military," "Maritime," "Government (General)," "Government (Diplomatic)," "Telecommunication"))
	.count();
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	refinery.lambdaDisrupt = lambdaDisrupt;
	
	// Query for lambdaDistort
	double countryDistortAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Educational Institution," "Food or Water Supply," "Journalists & Media," "NGO," "Private Citizens & Property," "Religious Figures/Institutions," "Tourists," "Abortion Related"))
	.count();
	double lambdaDistort = countryDistortAttacks / totalYears;

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Initial value	“null”

Type	String
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: previousCountry

Name	Value
General	
Initial value	currentCountry

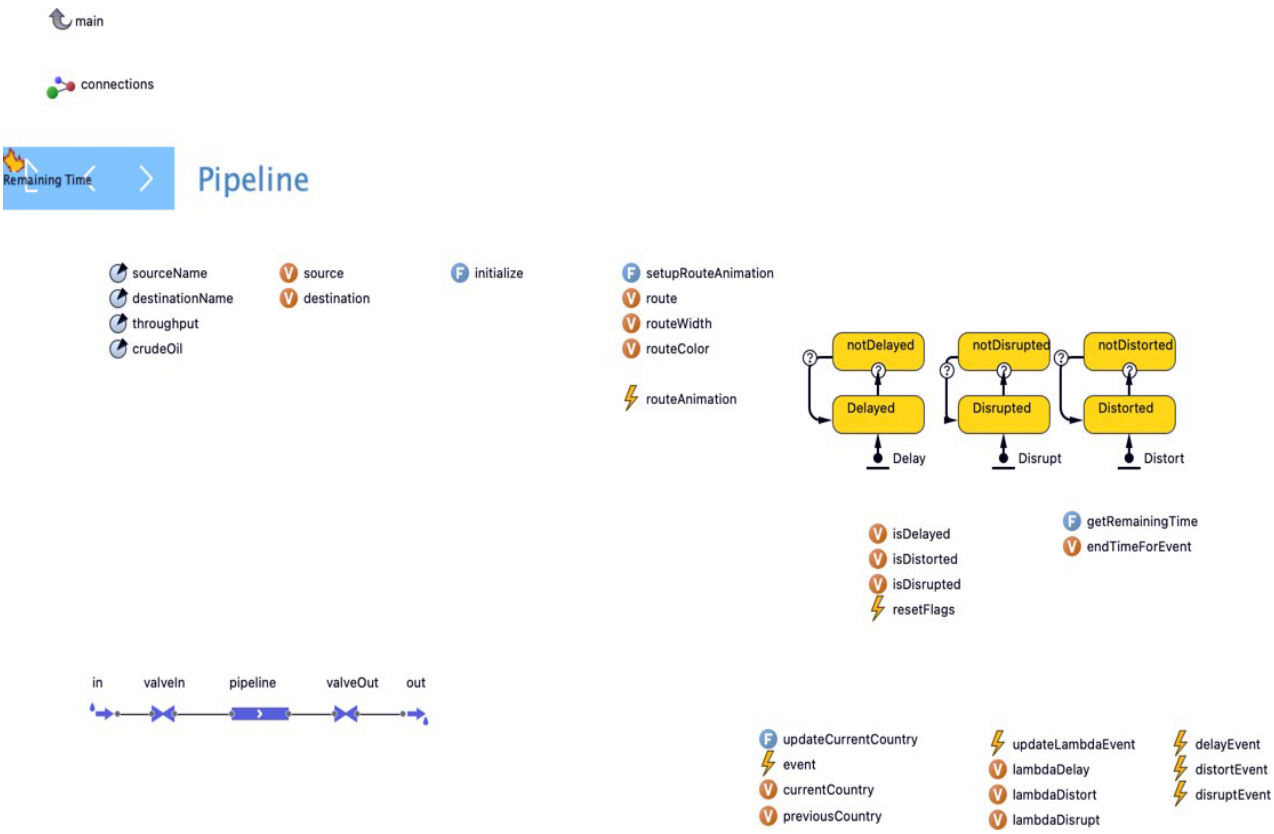
Name	Value
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: Pipeline

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(10 : MPS)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false
Space and network	
Space Type	GIS
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false



Parameter: throughput

Name	Value
General	
Array	false
Default value	0.0004
Type	double
Show at runtime	true
Show name	true
Value editor	
Label	Throughput, m3/sec
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: crudeOil

Name	Value
General	
Array	false
Default value	false
Type	boolean
Show at runtime	true
Show name	true

Value editor	

Name	Value
Label	Transfers
Editor control	Radio Button
Predefined Parameter Values	[Crude oil – Predefined Parameter Value, Refined products – Predefined Parameter Value]
Advanced	
System dynamics units	false
Save in snapshot	true

Predefined Parameter Values:

Name	Value
Crude oil	true
Refined products	false

Parameter: sourceName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Source
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: destinationName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Destination
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Function: setupRouteAnimation

Name	Value

General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true

Name	Value
Function body	

Body	route = new GISRoute(
	main.map,
	new GISMarkupSegmentLine(
	source.getLatitude(),
	source.getLongitude(),
	destination.getLatitude(),
	destination.getLongitude()
	)
	);
	route.setLineStyle(LINE_STYLE_SOLID);
	routeColor = crudeOil ? main.colorCrude : main.colorRefined;
	route.setLineColor(semiTransparent(gray));
	routeWidth = throughput / 30;
	route.setLineWidth(routeWidth);
	routeAnimation.reset();

Advanced	
Access type	private
System dynamics units	false

Function: initialize

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	source = main.findNode(sourceName);
	destination = main.findNode(destinationName);
	source.getAvailableOutput().connect(in);
	out.connect(destination.getInput());
	setupRouteAnimation();

Advanced	
Access type	default
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn't go negative

Advanced	
Access type	default
System dynamics units	false

Function: updateCurrentCountry

Name	Value

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	if (region.contains(x, y)) {
	newCountry = region.getTitle();
	break;
	}
	}
	
	// Check if the country has changed
	if (!newCountry.equals(currentCountry)) {
	currentCountry = newCountry;
	previousCountry = currentCountry;
	// Trigger condition for updateLambdaEvent should now evaluate to true
	}

Advanced	
Access type	public
System dynamics units	false

Event: routeAnimation

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	route.setLineWidth(routeWidth + 1 + sin(time()));

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Name	Value

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Check if there is no disruption, distortion, or delay
	if (!isDelayed && !isDistorted && !isDisrupted) {
	valveIn.open(); // Open the valveIn
	valveOut.open(); // Open the valveOut
	} else {
	valveIn.close(); // Close the valveIn
	valveOut.close(); // Close the valveOut
	}

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	updateCurrentCountry(); // Update the current country based on the agent's location

Event: delayEvent

Name	Value
General	
Logging	true
Rate	(lambdaDelay : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(lambdaDistort : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: disruptEvent

Name	Value
General	
Logging	true
Rate	(lambdaDisrupt : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	valveIn.close(); // Close if any of these conditions are true
	valveOut.close();
	} else {
	valveIn.open(); // Open otherwise
	valveOut.open();
	}

Event: updateLambdaEvent

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout

Name	Value
Show at runtime	true
Show name	true
Action	

Action	// Loop through all instances of Pipeline
	for (Pipeline pipeline : main.pipelines) { // Assuming 'pipelines' is the collection of Pipeline agents
	String currentCountry = pipeline.currentCountry;
	
	// Define the total number of years for which you have data
	double totalYears = 2021 – 1970;
	
	// Query for lambdaDelay
	double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Utilities," "Business," "Transportation"))
	.count();
	double lambdaDelay = countryDelayAttacks / totalYears;
	pipeline.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Violent Political Party," "Unknown," "Terrorists/Non-state Militia," "Airports and Aircraft," "Police," "Military," "Maritime," "Government (General)," "Government (Diplomatic)," "Telecommunication"))
	.count();
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	pipeline.lambdaDisrupt = lambdaDisrupt;
	
	// Query for lambdaDistort
	double countryDistortAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Educational Institution," "Food or Water Supply," "Journalists & Media," "NGO," "Private Citizens & Property," "Religious Figures/Institutions," "Tourists," "Abortion Related"))
	.count();
	double lambdaDistort = countryDistortAttacks / totalYears;

Variable: route

Name	Value
General	
Type	GISRoute
Show at runtime	true
Show name	true
Advanced	
Access type	private
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: routeWidth

Name	Value

Name	Value
General	

Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	private
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: routeColor

Name	Value
General	
Type	Color
Show at runtime	true
Show name	true
Advanced	
Access type	private
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: destination

Name	Value
General	
Type	PipelineNode
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: source

Name	Value
General	
Type	PipelineNode
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true

Advanced

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Initial value	“null”
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	

Name	Value
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: previousCountry

Name	Value
General	
Initial value	currentCountry
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Link to agents: connections

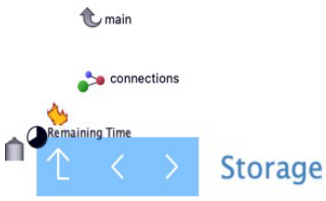
Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

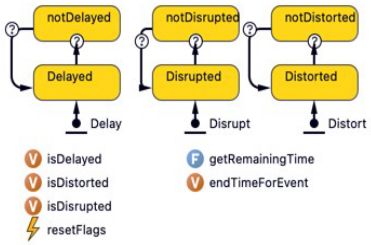
Agent Type: Storage

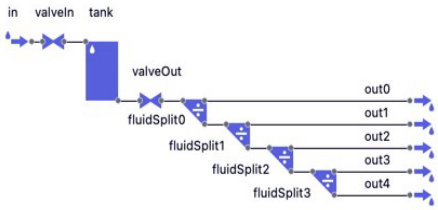
Name	Value
Agent actions	
Startup code	setLocation(main.map.searchFirst(locationName)); plot.setColor(0 , crudeOil ? dimGray : orange);
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	

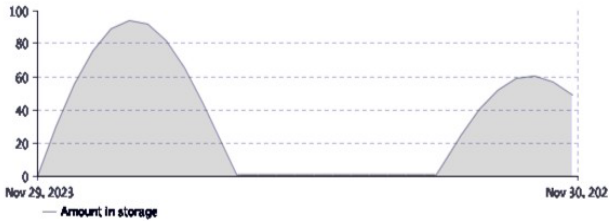
Speed	(10 : MPS)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false
Advanced Java	
Generic	false
Advanced	

Name	Value
Extends other agent	ClassReference: oil_supply_chain_gis.PipelineNode (Resolved: true)
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false









Parameter: crudeOil

Name	Value
General	
Array	false
Default value	true
Type	boolean
Show at runtime	true
Show name	true

Value editor	
Label	Contains
Editor control	Radio Button
Predefined Parameter Values	[Crude oil – Predefined Parameter Value, Refined products – Predefined Parameter Value]
Advanced	
System dynamics units	false
Save in snapshot	true

Predefined Parameter Values:

Name	Value
Crude oil	true
Refined products	false

Parameter: terminal

Name	Value
General	
Array	false
Default value	false
Type	boolean
Show at runtime	true
Show name	true
Value editor	
Label	Terminal storage
Editor control	Check Button
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: capacity

Name	Value
General	
Array	false
Default value	(50000 : CUBIC_METER)
Unit	cubic meters
Show at runtime	true
Show name	true
Value editor	
Label	Capacity, m3
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Function: getInput

Name	Value
General	
Return type	FluidEnter
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return in;
Advanced	
Access type	default
System dynamics units	false

Function: getAvailableOutput

Name	Value
General	
Return type	FluidExit
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	

Body	if(! out0.isConnected()) return out0;
	if(! out1.isConnected()) return out1;
	if(! out2.isConnected()) return out2;
	if(! out3.isConnected()) return out3;
	if(! out4.isConnected()) return out4;
	error(“No available outputs at storage”);
	return null;

Advanced	
Access type	default
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn’t go negative
Advanced	
Access type	default
System dynamics units	false

Name	Value

Function: updateCurrentCountry

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	if (region.contains(x, y)) {
	newCountry = region.getTitle();
	break;
	}
	}
	
	// Check if the country has changed
	if (!newCountry.equals(currentCountry)) {
	currentCountry = newCountry;
	previousCountry = currentCountry;
	// Trigger condition for updateLambdaEvent should now evaluate to true
	}

Advanced	
Access type	public
System dynamics units	false

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Check if there is no disruption, distortion, or delay
	if (!isDelayed && !isDistorted && !isDisrupted) {
	  valveIn.open(); // Open the valveIn
	  valveOut.open(); // Open the valveOut
	} else {
	  valveIn.close(); // Close the valveIn
	  valveOut.close(); // Close the valveOut
	}

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	updateCurrentCountry(); // Update the current country based on the agent's location

Event: delayEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDelay) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDistort) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: disruptEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDisrupt) : PER_YEAR)
Trigger type	Rate

Name	Value
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: updateLambdaEvent

Name	Value

General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Loop through all instances of Storage
	for (Storage storage : main.storages) { // collection of Storage agents
	String currentCountry = storage.currentCountry;
	
	// Define the total number of years for which you have data
	double totalYears = 2021 – 1970;
	
	// Query for lambdaDelay
	double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in(“Utilities,” “Business,” “Transportation”))
	.count();
	double lambdaDelay = countryDelayAttacks / totalYears;
	storage.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in(“Violent Political Party,” “Unknown,” “Terrorists/Non-state Militia,” “Airports and Aircraft,” “Police,” “Military,” “Maritime,” “Government (General),” “Government (Diplomatic),” “Telecommunication”))
	.count();
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	storage.lambdaDisrupt = lambdaDisrupt;
	
	// Query for lambdaDistort
	double countryDistortAttacks = selectFrom(db_1970_2020_db)

Name	Value
	(“Educational Institution”, “Food or Water Supply”, “Journalists & Media”, “NGO”, “Private Citizens & Property”, “Religious Figures/Institutions”, “Tourists”, “Abortion Related”)) .count(); double lambdaDistort = countryDistortAttacks / totalYears;

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	

Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	

Initial value	“null”
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true

Name	Value
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true

System dynamics units	false

Variable: previousCountry

Name	Value
General	
Initial value	currentCountry
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

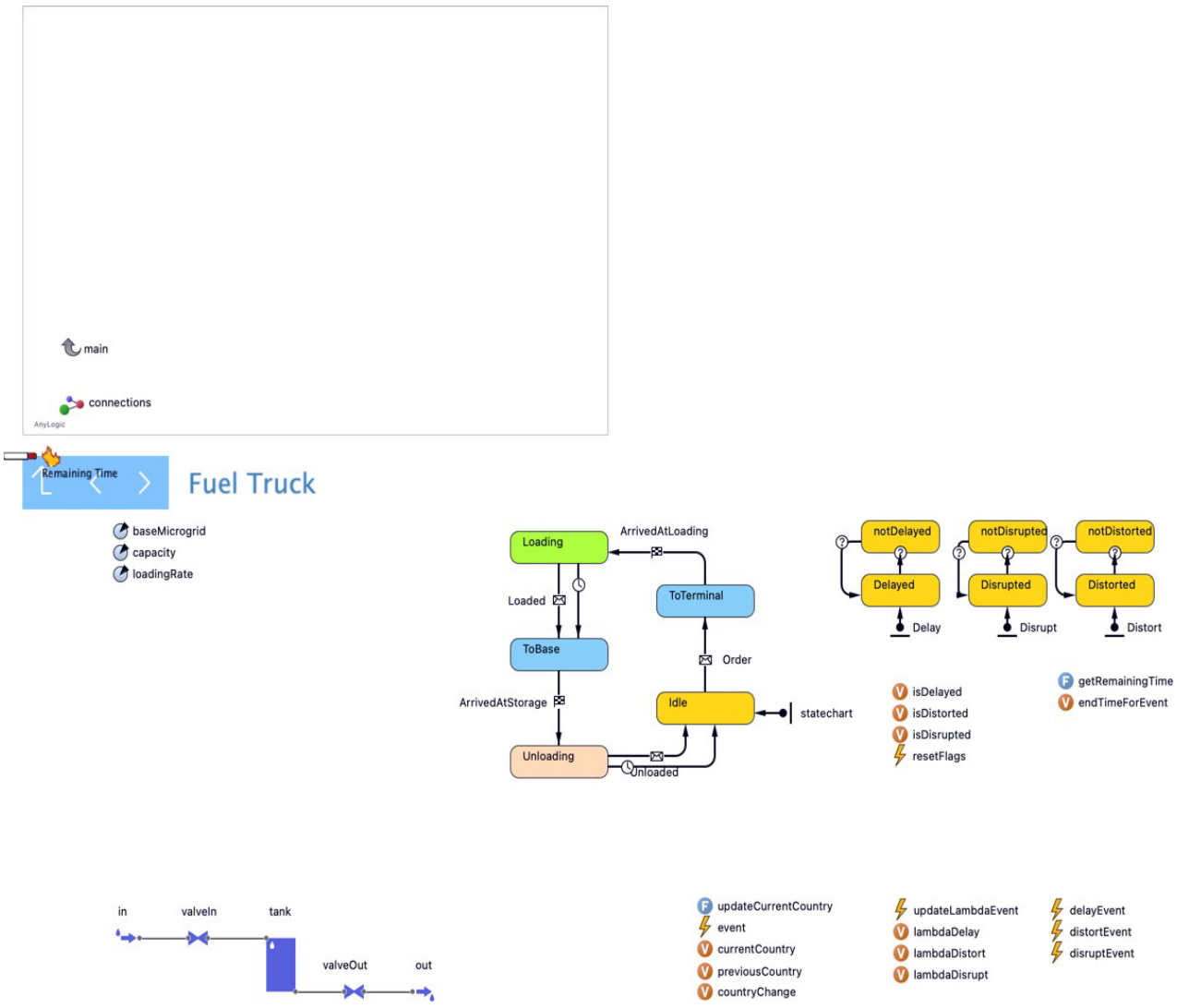
Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: FuelTruck

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(10 : MPH)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false

Name	Value
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false



Parameter: capacity

Name	Value
General	
Array	false
Default value	(18 : CUBIC_METER)
Unit	cubic meters
Show at runtime	true
Show name	true

Name	Value
Value editor	
Label	Capacity
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: loadingRate

Name	Value
General	
Array	false
Default value	(0.0126 : CUBIC_METER_PER_SECOND)
Unit	cubic meters / s
Show at runtime	true
Show name	true
Value editor	
Label	Load / unload rate, m3/sec
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: baseMicrogrid

Name	Value
General	
Array	false
Type	BaseMicrogrid
Show at runtime	true
Show name	true
Value editor	
Label	baseMicrogrid
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true
On change value	setLocation(baseMicrogrid);

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn't go negative
Advanced	
Access type	default

Name	Value
System dynamics units	false

Function: updateCurrentCountry

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	if (region.contains(x, y)) {
	newCountry = region.getTitle();
	break;
	}
	}
	
	// Check if the country has changed
	if (!newCountry.equals(currentCountry)) {
	previousCountry = currentCountry; // Save the old country
	currentCountry = newCountry; // Update to the new country
	
	// Check if the previous country doesn’t match the new current country
	if (!previousCountry.equals(currentCountry)) {
	countryChange = true; // Set countryChange to true as the country has changed
	} else {
	countryChange = false; // Set countryChange to false as the country remains the same
	}
	} else {
	countryChange = false; // Set countryChange to false as the country remains the same
	}

Advanced	
Access type	public
System dynamics units	false

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once

Name	Value
Trigger type	Timeout
Show at runtime	true

Show name	true
Action	

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Check if there is no disruption, distortion, or delay
	if (!isDelayed && !isDistorted && !isDisrupted) {
	  valveIn.open(); // Open the valveIn
	  valveOut.open(); // Open the valveOut
	} else {
	  valveIn.close(); // Close the valveIn
	  valveOut.close(); // Close the valveOut
	}

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	  if (region.contains(x, y)) {
	  newCountry = region.getTitle();
	  break;
	  }
	}
	
	// Check if the country has changed
	if (!newCountry.equals(currentCountry)) {
	  // Save the old country and Check if the previous country doesn’t match the new current country
	  if (!newCountry.equals(previousCountry)) {
	  countryChange = false; // Set countryChange to true as the country has changed
	  } else {
	  countryChange = true; // Set countryChange to false as the country remains the same
	  }
	
	  previousCountry = currentCountry; // Save the old country
	  currentCountry = newCountry; // Update to the new country
	} else {
	  countryChange = true; // Set countryChange to false as the country remains the same
	}

Event: delayEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDelay) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(lambdaDistort : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: disruptEvent

Name	Value
General	

Logging	true

Name	Value
Rate	(lambdaDisrupt : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	valveIn.close(); // Close if any of these conditions are true
	valveOut.close();
	} else {
	valveIn.open(); // Open otherwise
	valveOut.open();
	}

Event: updateLambdaEvent

Name	Value
General	
Logging	true
Condition	countryChange
Trigger type	Condition
Show at runtime	true
Show name	true
Action	

Action	// Loop through all instances of fuelTruck
	for (FuelTruck fuelTruck : main.fuelTrucks_base) { // Assuming 'fuelTrucks' is the collection of FuelTruck agents
	String currentCountry = fuelTruck.currentCountry;
	
	// Define the total number of years for which you have data
	double totalYears = 2021 – 1970;
	
	// Query for lambdaDelay
	double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Utilities," "Business," "Transportation"))
	.count();
	double lambdaDelay = countryDelayAttacks / totalYears;
	fuelTruck.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Violent Political Party," "Unknown," "Terrorists/Non-state Militia," "Airports and Aircraft," "Police," "Military," "Maritime," "Government (General)," "Government (Diplomatic)," "Telecommunication"))
	.count();
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	fuelTruck.lambdaDisrupt = lambdaDisrupt;
	
	// Query for lambdaDistort

Name	Value
	(currentCountry), db_1970_2020_db.targtype1_txt.in("Educational Institution", "Food or Water Supply", "Journalists & Media", "NGO", "Private Citizens & Property", "Religious Figures/Institutions", "Tourists", "Abortion Related"))

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Initial value	“null”
Type	String
Show at runtime	true
Show name	true

Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Name	Value
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: previousCountry

Name	Value
General	
Initial value	“”
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: countryChange

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Link to agents: connections

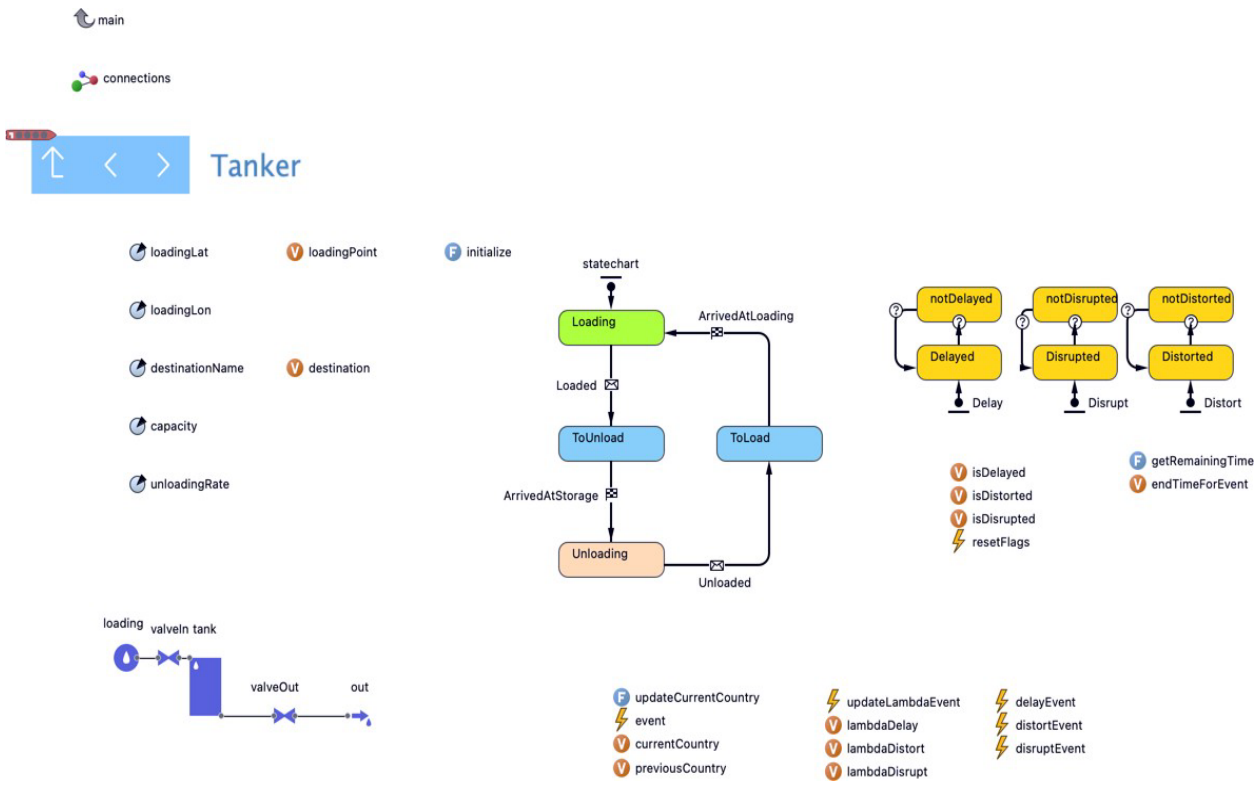
Name	Value

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: Tanker

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(13 : KN)
Rotate animation towards movement	true

Rotate vertically as well (along Z-axis)	false
Space and network	
Space Type	Continuous
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false



Parameter: capacity

Name	Value
General	
Array	false
Default value	(5000 : CUBIC_METER)
Unit	cubic meters
Show at runtime	true
Show name	true
Value editor	
Label	Capacity, m3
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: unloadingRate

Name	Value
General	
Array	false
Default value	(0.1 : CUBIC_METER_PER_SECOND)
Unit	cubic meters / s
Show at runtime	true
Show name	true
Value editor	
Label	Unloading rate, m3/sec
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: loadingLat

Name	Value
General	
Array	false
Type	double
Show at runtime	true
Show name	true
Value editor	
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: loadingLon

Name	Value
General	

Name	Value
Array	false
Type	double
Show at runtime	true
Show name	true
Value editor	
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Name	Value

Parameter: destinationName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Destination city
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Function: initialize

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	loadingPoint = new GISPoint(main.map, loadingLat, loadingLon);
	setLocation(loadingPoint);
	destination = main.findNode(destinationName);

Advanced	
Access type	default
System dynamics units	false

Function: updateCurrentCountry

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS

	space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	if (region.contains(x, y)) {
	newCountry = region.getTitle();
	break;
	}
	}
	
	// Check if the country has changed
	if (!newCountry.equals(currentCountry)) {
	currentCountry = newCountry;
	previousCountry = currentCountry;
	// Trigger condition for updateLambdaEvent should now evaluate to true
	}

Advanced	
Access type	public
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn’t go negative
Advanced	
Access type	default
System dynamics units	false

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	updateCurrentCountry(); // Update the current country based on the agent’s location

Event: delayEvent

Name	Value
General	
Logging	true
Rate	(lambdaDelay : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	valveIn.close(); // Close if any of these conditions are true
	valveOut.close();
	} else {
	valveIn.open(); // Open otherwise
	valveOut.open();
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(lambdaDistort : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	valveIn.close(); // Close if any of these conditions are true
	valveOut.close();
	} else {
	valveIn.open(); // Open otherwise
	valveOut.open();
	}

Event: disruptEvent

Name	Value
General	

Logging	true
Rate	(lambdaDisrupt : PER_YEAR)

Name	Value
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	valveIn.close(); // Close if any of these conditions are true
	valveOut.close();
	} else {
	valveIn.open(); // Open otherwise
	valveOut.open();
	}

Event: updateLambdaEvent

Name	Value
General	
Logging	true
Condition	!currentCountry.equals(previousCountry)
Trigger type	Condition
Show at runtime	true
Show name	true
Action	

Action	// Loop through all instances of Tanker
	for (Tanker tanker : main.tankers) { // Assuming 'tankers' is the collection of Tanker agents
	String currentCountry = tanker.currentCountry;
	
	// Define the total number of years for which you have data
	double totalYears = 2021 – 1970;
	
	// Query for lambdaDelay
	double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Utilities," "Business," "Transportation"))
	.count();
	double lambdaDelay = countryDelayAttacks / totalYears;
	tanker.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Violent Political Party," "Unknown," "Terrorists/Non-state Militia," "Airports and Aircraft," "Police," "Military," "Maritime," "Government (General)," "Government (Diplomatic)," "Telecommunication"))
	.count();
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	tanker.lambdaDisrupt = lambdaDisrupt;
	
	// Query for lambdaDistort
	double countryDistortAttacks = selectFrom(db_1970_2020_db)

Name	Value
	("Educational Institution", "Food or Water Supply", "Journalists & Media", "NGO", "Private Citizens & Property", "Religious Figures/Institutions", "Tourists", "Abortion Related")) .count(); double lambdaDistort = countryDistortAttacks / totalYears;

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Check if there is no disruption, distortion, or delay
	if (!isDelayed && !isDistorted && !isDisrupted) {
	    valveIn.open(); // Open the valveIn
	    valveOut.open(); // Open the valveOut
	} else {
	    valveIn.close(); // Close the valveIn
	    valveOut.close(); // Close the valveOut
	}

Variable: destination

Name	Value
General	
Type	PipelineNode
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: loadingPoint

Name	Value
General	
Type	GISPoint
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Initial value	“null”
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public

Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	

Name	Value
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: previousCountry

Name	Value
General	

Initial value	currentCountry
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true

Name	Value
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	

Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: PipelineNode

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(10 : MPS)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false

Name	Value
Space and network	
Space Type	Continuous
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties

Limit the number of data samples	false

Parameter: locationName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Location
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Function: getInput

Name	Value
General	
Return type	FluidEnter
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return null;
Advanced	
Access type	default
System dynamics units	false

Function: getAvailableOutput

Name	Value
General	
Return type	FluidExit
Return type:	Returns value
Show at runtime	true
Show name	true

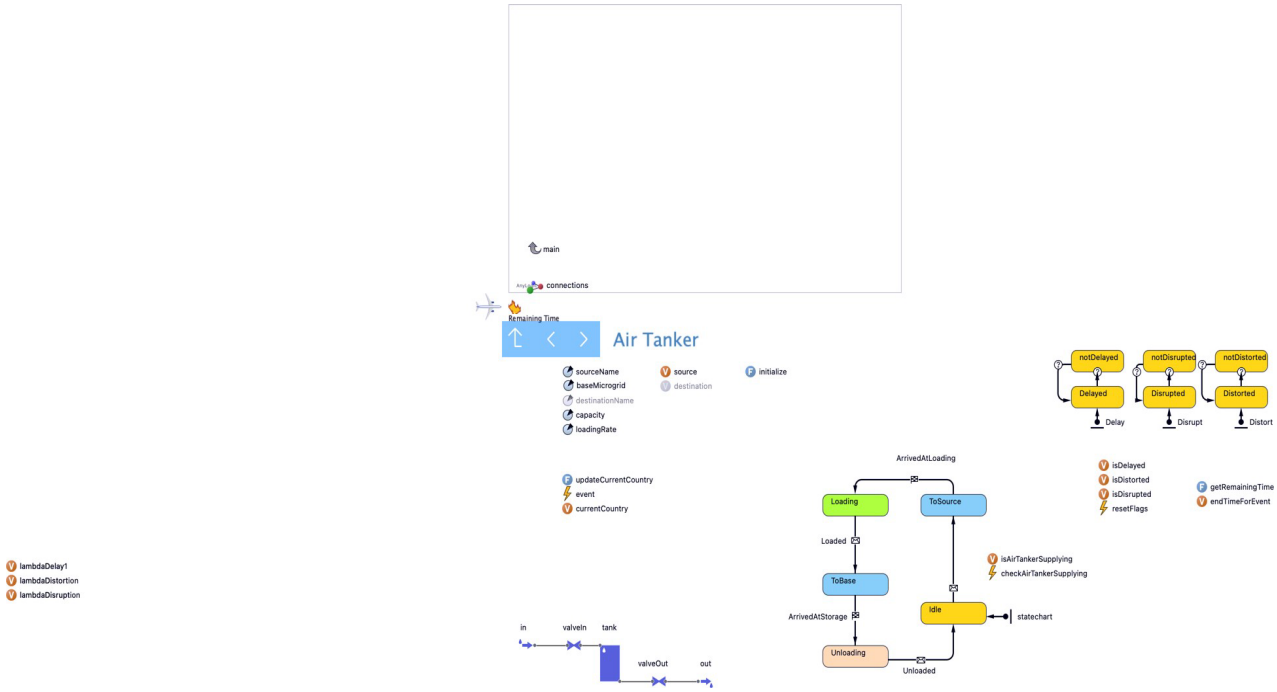
Function body	
Body	return null;
Advanced	
Access type	default
System dynamics units	false

Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: AirTanker

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(605 : KPH)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false



Parameter: capacity

Name	Value
General	
Array	false
Default value	(32.2 : CUBIC_METER)
Unit	cubic meters
Show at runtime	true
Show name	true
Value editor	
Label	Capacity, m3
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: loadingRate

Name	Value
General	
Array	false
Default value	(0.038 : CUBIC_METER_PER_SECOND)
Unit	cubic meters / s
Show at runtime	true
Show name	true
Value editor	
Label	Unloading rate, m3/sec
Editor control	Unit editor

Advanced	
System dynamics units	false

Name	Value
Save in snapshot	true

Parameter: sourceName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Source
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: baseMicrogrid

Name	Value
General	
Array	false
Type	BaseMicrogrid
Show at runtime	true
Show name	true
Value editor	
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true
On change value	setLocation(baseMicrogrid);

Function: initialize

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	
Body	source = main.findNode(sourceName); setLocation(source);
Advanced	

Access type	default
System dynamics units	false

Function: updateCurrentCountry

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it's now part of BaseMicrogrid
	if (region.contains(x, y)) {
	this.currentCountry = region.getTitle();
	return;
	}
	}
	this.currentCountry = "Unknown";

Advanced	
Access type	public
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn't go negative
Advanced	
Access type	default
System dynamics units	false

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout

Show at runtime	true
Show name	true
Action	
Action	updateCurrentCountry(); // Update the current country based on the agent's location

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event

Event: checkAirTankerSupplying

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	if (isAirTankerSupplying) {
	  if (baseMicrogrid != null) {
	  baseMicrogrid.valveIn.open();
	  traceln("AirTanker is supplying. ValveIn at baseMicrogrid is open.");
	  } else {
	  traceln("BaseMicrogrid is null. Cannot open valveIn.");
	  }
	} else {
	  traceln("AirTanker is not supplying. No action taken.");
	}

Variable: source

Name	Value
General	
Type	PipelineNode
Show at runtime	true
Show name	true

Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Name	Value
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isAirTankerSupplying

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true

Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistortion

Name	Value
General	
Type	double
Show at runtime	true

Name	Value
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisruption

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay1

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

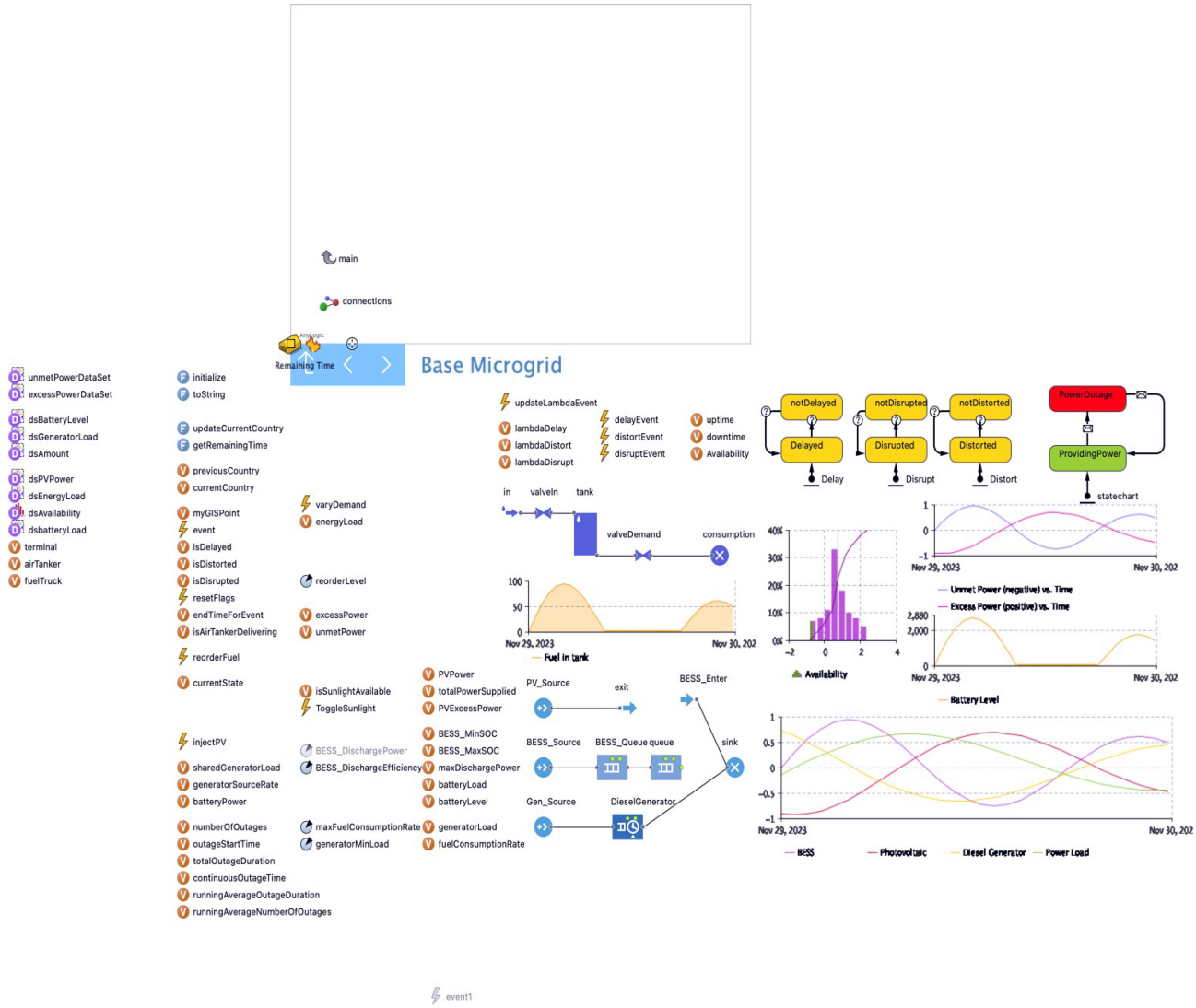
Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: BaseMicrogrid

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(10 : MPS)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	true
Advanced Java	

Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties
Limit the number of data samples	false



Parameter: reorderLevel

Name	Value
General	
Array	false
Default value	43
Type	int
Show at runtime	true
Show name	true
Value editor	

Label	Reorder level, % of capacity
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: maxFuelConsumptionRate

Name	Value
General	

Name	Value
Array	false
Default value	(5.25752e-6 : CUBIC_METER_PER_SECOND)
Unit	cubic meters / s
Show at runtime	true
Show name	true
Value editor	
Label	Max Fuel Consumption Rate gal/hr
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: BESS_DischargeEfficiency

Name	Value
General	
Array	false
Default value	.95100
Type	double
Show at runtime	true
Show name	true
Value editor	
Label	BESS_DischargeEfficiency (kW)
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: generatorMinLoad

Name	Value
General	
Array	false
Default value	.6
Type	double
Show at runtime	true
Show name	true

Value editor	
Label	Min Load (%)
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Function: initialize

Name	Value
General	
Return type:	Just action (returns nothing)

Name	Value
Show at runtime	true
Show name	true
Function body	

Body	{
	setLocation(location);
	terminal = getNearestAgent(main.storages.findAll(s -> s.terminal));
	myGISPoint = location; // Store the GISPoint
	}

Advanced	
Access type	default
System dynamics units	false

Arguments:

Name	Type
location	GISPoint

Function: toString

Name	Value
General	
Return type	String
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return “Base Microgrid @ “ + format(getLatitude()) + “ “ + format(getLongitude());
Advanced	
Access type	public
System dynamics units	false

Function: updateCurrentCountry

Name	Value
General	

Name	Value
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	if (region.contains(x, y)) {
	newCountry = region.getTitle();
	break;
	}
	}
	
	// Check if the country has changed
	
	if (!newCountry.equals(currentCountry)) {
	currentCountry = newCountry;
	previousCountry = currentCountry;
	// Trigger condition for updateLambdaEvent should now evaluate to true
	}

Advanced	
Access type	public
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn’t go negative
Advanced	
Access type	default
System dynamics units	false

Event: varyDemand

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties

Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Debug log to ensure that the event is being triggered
	//traceln("VaryDemand event triggered.");
	
	// Define daytime and nighttime demand multipliers
	double dayMinMultiplier = 0.9; // 90% of meanEnergyLoad for daytime
	double dayMaxMultiplier = 1.1; // 110% of meanEnergyLoad for daytime
	double nightMinMultiplier = 0.45; // 45% of meanEnergyLoad for nighttime
	double nightMaxMultiplier = 0.55; // 55% of meanEnergyLoad for nighttime
	
	// Declare a multiplier variable to hold the final multiplier value
	double multiplier;
	
	// Debug log to check the value of isSunlightAvailable
	//traceln("Is sunlight available? " + isSunlightAvailable);

Name	Value

	// Use the isSunlightAvailable variable to determine if it's daytime or nighttime
	
	if (isSunlightAvailable) {
	
	// Daytime
	
	multiplier = uniform(dayMinMultiplier, dayMaxMultiplier);
	
	} else {
	
	// Nighttime
	
	multiplier = uniform(nightMinMultiplier, nightMaxMultiplier);
	
	}
	
	// Calculate the new energyLoad value based on meanEnergyLoad and the multiplier

Event: event

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	updateCurrentCountry(); // Update the current country based on the agent's location

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Resetting flags
	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Logic for controlling valveIn
	if (!(isDelayed || isDistorted || isDisrupted)) {
	valveIn.open();
	}
	traceln("Flags reset and valveIn control logic executed.");

Name	Value

Event: ToggleSunlight

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Toggling sunlight availability
	isSunlightAvailable = !isSunlightAvailable;
	
	// Logic for controlling pvPower
	if (isSunlightAvailable) {
	PVPower = main.pvCapacity;
	} else {
	PVPower = 0;
	}
	
	//traceln("Sunlight toggled. Current status: " + (isSunlightAvailable ? "Available" : "Not Available"));

Event: delayEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDelay) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Logging information
	traceln("Inside delayEvent: lambdaDelay = " + lambdaDelay);
	
	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	
	// Calculate the running average delay time in days based on the simulation time elapsed in years
	if (time(YEAR) > 0) {
	main.runningAverageDelayTime = (main.globalTotalDelayTime / 24) / time(YEAR);
	} else {
	main.runningAverageDelayTime = (main.globalTotalDelayTime / 24);
	}
	
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn
	if (isDelayed || isDistorted || isDisrupted) {
	if (airTanker.isAirTankerSupplying) {

Name	Value

	valveIn.open(); // Always open if airTanker is supplying
	
	} else {
	
	valveIn.close(); // Close if no supply from airTanker
	
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDistort) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Logging information
	traceln("Inside distortEvent: lambdaDistort = " + lambdaDistort);
	
	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	
	// Avoid division by zero if time() returns 0 (unlikely but safe to check)
	if (time(YEAR) > 0) {
	  // Calculate the running average distortion time in days based on the simulation time elapsed in years
	  main.runningAverageDistortionTime = (main.globalTotalDistortionTime / 24) / time(YEAR);
	} else {
	  // If time() is 0, just use the distortion time in days
	  main.runningAverageDistortionTime = main.globalTotalDistortionTime / 24;
	}
	
	// Restart the flags based on the distortion time
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn
	if (isDelayed || isDistorted || isDisrupted) {
	  if (airTanker.isAirTankerSupplying) {
	  valveIn.open(); // Always open if airTanker is supplying
	  } else {
	  valveIn.close(); // Close if no supply from airTanker
	  }
	} else {
	  valveIn.open(); // Open otherwise
	}

Event: disruptEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDisrupt) : PER_YEAR)
Trigger type	Rate
Show at runtime	true

Name	Value
Show name	true
Action	

Action	// Logging information
	traceln("Inside disruptEvent: lambdaDisrupt = " + lambdaDisrupt);
	
	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	
	// Avoid division by zero if time() returns 0 (unlikely but safe to check)
	if (time(YEAR) > 0) {
	  // Calculate the running average disruption time in days based on the simulation time elapsed in years
	  main.runningAverageDisruptionTime = (main.globalTotalDisruptionTime / 24) / time(YEAR);
	} else {
	  // If time() is 0, just use the disruption time in days
	  main.runningAverageDisruptionTime = main.globalTotalDisruptionTime / 24;
	}
	
	// Restart the flags based on the disruption time
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn
	if (isDelayed || isDistorted || isDisrupted) {
	  if (airTanker.isAirTankerSupplying) {
	  valveIn.open(); // Always open if airTanker is supplying
	  } else {
	  valveIn.close(); // Close if no supply from airTanker
	  }
	} else {
	  valveIn.open(); // Open otherwise
	}

Event: updateLambdaEvent

Name	Value
General	
Logging	true
Condition	currentCountry.equals(previousCountry)
Trigger type	Condition
Show at runtime	true
Show name	true
Action	

Action	//traceln("Update Lambda Event Triggered");
	
	// Loop through all instances of BaseMicrogrid
	for (BaseMicrogrid baseMicrogrid : main.baseMicrogrids) {
	  String currentCountry = baseMicrogrid.currentCountry;
	  // traceln("Processing country: " + currentCountry);
	
	  // Define the total number of years for which you have data
	  double totalYears = 2021 – 1970;
	
	  // Query for lambdaDelay
	  double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Utilities," "Business," "Transportation"))
	  .count();
	  //traceln("countryDelayAttacks: " + countryDelayAttacks);

Name	Value

	baseMicrogrid.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Violent Political Party", "Unknown", "Terrorists/Non-state Militia", "Airports and Aircraft", "Police", "Military", "Maritime", "Government (General)", "Government (Diplomatic)", "Telecommunication"))
	.count();
	
	//traceln("countryDisruptAttacks: " + countryDisruptAttacks);
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	baseMicrogrid.lambdaDisrupt = lambdaDisrupt;

Event: reorderFuel

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	// Calculate the reorder level
	double reorderAmount = (reorderLevel / 100.0) * main.fuelCapacity;
	
	if (tank.amount() < reorderAmount) {
	  if (isDelayed || isDistorted || isDisrupted) {
	  if (main.activeAirTankers.size() < 2) { // Check if less than 2 AirTankers are active
	  if (airTanker != null) { // Make sure airTanker is not null
	  main.activeAirTankers.add(airTanker); // Add this airTanker to the list of active airTankers
	  send("GET_FUEL," airTanker);
	  } else {
	  // Log or handle the case when airTanker is null
	  }
	  } else {
	  // New logic to allow all agents to proceed with fuel trucks
	  send("GET_FUEL," fuelTruck);
	  }
	  } else {
	  send("GET_FUEL," fuelTruck);
	  }
	}
	valveDemand.close();

Event: injectPV

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	PV_Source.inject((int) Math.round(PVPower));

Variable: fuelTruck

Name	Value
General	
Type	FuelTruck
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: terminal

Name	Value
General	
Type	Storage
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDelayed

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public

Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	false
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Initial value	““
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: airTanker

Name	Value

General	
Type	AirTanker
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: myGISPoint

Name	Value
General	
Type	GISPoint
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: uptime

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: batteryLevel

Name	Value
General	
Initial value	main.BESS_Energy
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public

Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isSunlightAvailable

Name	Value

Name	Value
General	
Initial value	true
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: totalPowerSupplied

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: excessPower

Name	Value
General	
Initial value	PVExcessPower
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: PVPower

Name	Value
General	
Initial value	main.pvCapacity
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Name	Value
Save in snapshot	true
System dynamics units	false

Variable: batteryLoad

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: generatorLoad

Name	Value
General	
Initial value	main.generatorMaxPower
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: BESS_MinSOC

Name	Value
General	
Initial value	.2

Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: BESS_MaxSOC

Name	Value
General	

Name	Value
Initial value	.99
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: fuelConsumptionRate

Name	Value
General	
Initial value	maxFuelConsumptionRate
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: PVExcessPower

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Save in snapshot	true
System dynamics units	false

Variable: sharedGeneratorLoad

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: energyLoad

Name	Value
General	
Initial value	main.meanHourlyPowerLoad
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: generatorSourceRate

Name	Value
General	
Initial value	main.generatorMaxPower
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: maxDischargePower

Name	Value
General	
Initial value	.01

Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentState

Name	Value
General	
Type	String
Show at runtime	true
Show name	true
Advanced	

Name	Value
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: downtime

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true

System dynamics units	false

Variable: Availability

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isAirTankerDelivering

Name	Value
General	

Name	Value
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: unmetPower

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true

Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: outageStartTime

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: totalOutageDuration

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: numberOfOutages

Name	Value
General	
Initial value	0
Type	int
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Name	Value
Save in snapshot	true
System dynamics units	false

Variable: continuousOutageTime

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: previousCountry

Name	Value
General	
Initial value	currentCountry

Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: runningAverageOutageDuration

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: runningAverageNumberOfOutages

Name	Value
General	
Type	double

Name	Value
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: batteryPower

Name	Value
General	
Initial value	0
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true

System dynamics units	false

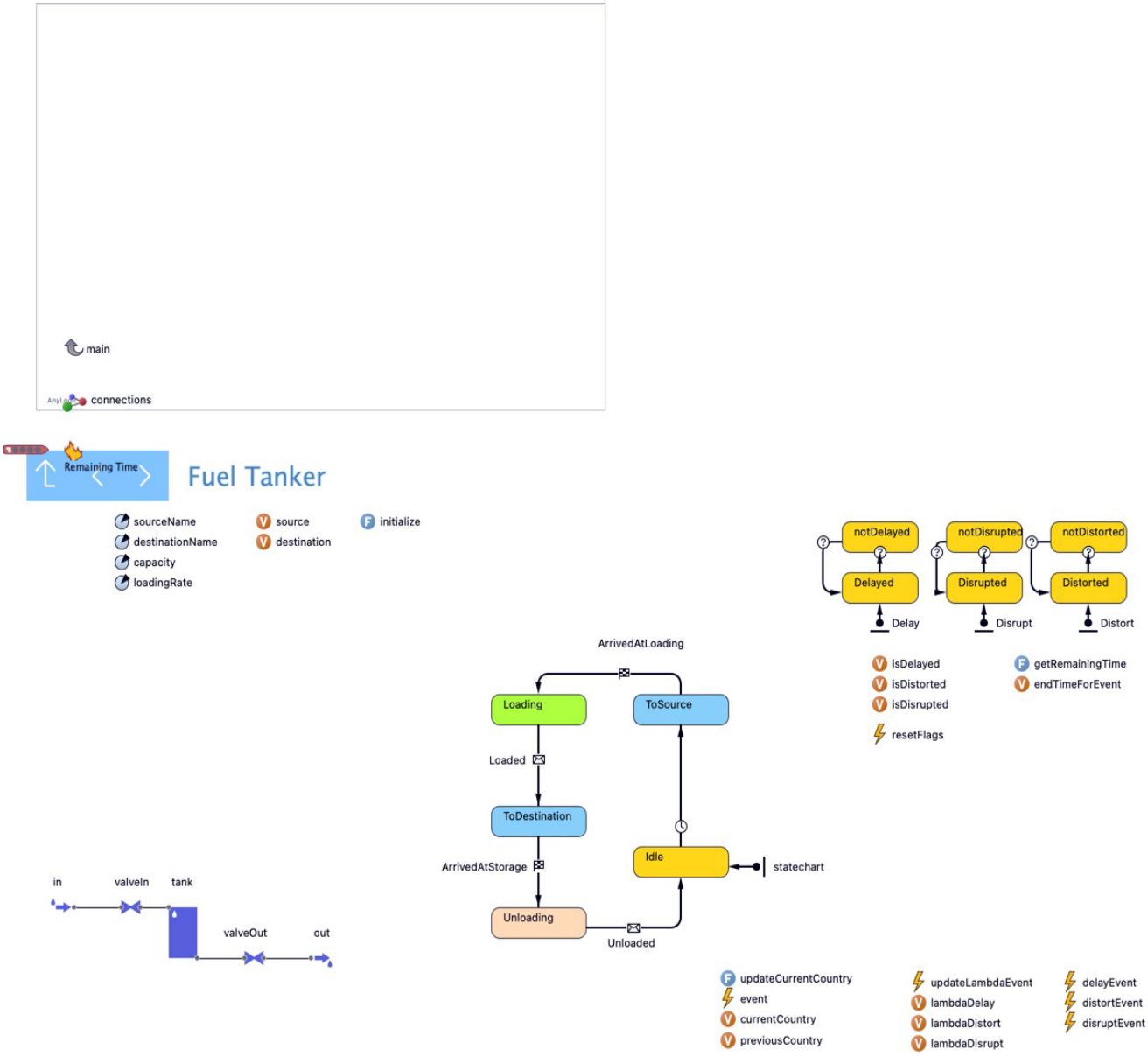
Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Agent Type: FuelTanker

Name	Value
Agent in flowcharts	
Use in flowcharts as	Agent
Dimensions and movement	
Speed	(100 : KN)
Rotate animation towards movement	true
Rotate vertically as well (along Z-axis)	false
Advanced Java	
Generic	false
Advanced	
Logging	true
Auto-create datasets	true
AOC_DATASETS_UPDATE_TIME_PROPERTIES	- Recurring Event Properties

Name	Value
Limit the number of data samples	false



Parameter: capacity

Name	Value
General	
Array	false
Default value	(3000 : CUBIC_METER)
Unit	cubic meters
Show at runtime	true
Show name	true
Value editor	
Label	Capacity, m3
Editor control	Unit editor
Advanced	

Name	Value
System dynamics units	false
Save in snapshot	true

Parameter: loadingRate

Name	Value
General	
Array	false
Default value	(.1 : CUBIC_METER_PER_SECOND)
Unit	cubic meters / s
Show at runtime	true
Show name	true
Value editor	
Label	Unloading rate, m3/sec
Editor control	Unit editor
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: destinationName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Destination city
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Parameter: sourceName

Name	Value
General	
Array	false
Type	String
Show at runtime	true
Show name	true
Value editor	
Label	Source
Editor control	Text
Advanced	
System dynamics units	false
Save in snapshot	true

Function: initialize

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Body	source = main.findNode(sourceName);
	setLocation(source);
	destination = main.findNode(destinationName);

Advanced	
Access type	default
System dynamics units	false

Function: getRemainingTime

Name	Value
General	
Return type	double
Return type:	Returns value
Show at runtime	true
Show name	true
Function body	
Body	return Math.max(0, endTimeForEvent – time()); // Ensure it doesn't go negative
Advanced	
Access type	default
System dynamics units	false

Function: updateCurrentCountry

Name	Value
General	
Return type:	Just action (returns nothing)
Show at runtime	true
Show name	true
Function body	

Name	Value

Body	double x = this.getX(); // Get the X coordinate of the agent in GIS space
	double y = this.getY(); // Get the Y coordinate of the agent in GIS space
	
	String newCountry = “Unknown”;
	
	for (GISRegion region : main.GISCountries) { // Directly using GISCountries since it’s now part of BaseMicrogrid
	  if (region.contains(x, y)) {
	  newCountry = region.getTitle();
	  break;
	  }
	}
	
	// Check if the country has changed
	
	if (!newCountry.equals(currentCountry)) {
	  currentCountry = newCountry;
	  previousCountry = currentCountry;
	  // Trigger condition for updateLambdaEvent should now evaluate to true
	}

Advanced	
Access type	public
System dynamics units	false

Event: resetFlags

Name	Value
General	
Logging	true
EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Occurs once
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	

Action	isDelayed = false;
	isDistorted = false;
	isDisrupted = false;
	endTimeForEvent = 0; // Reset the end time for the event
	
	// Check if there is no disruption, distortion, or delay
	if (!isDelayed && !isDistorted && !isDisrupted) {
	  valveIn.open(); // Open the valveIn
	  valveOut.open(); // Open the valveOut
	} else {
	  valveIn.close(); // Close the valveIn
	  valveOut.close(); // Close the valveOut
	}

Event: event

Name	Value
General	
Logging	true

EVENT_TIMEOUT_PROPERTIES	- Recurring Event Properties
Mode	Cyclic
Trigger type	Timeout
Show at runtime	true
Show name	true
Action	
Action	updateCurrentCountry(); // Update the current country based on the agent's location

Event: delayEvent

Name	Value
General	

Name	Value
Logging	true
Rate	(poisson(lambdaDelay) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for delayEvent
	isDelayed = true;
	endTimeForEvent = time() + main.delayTime;
	main.globalTotalDelayTime += main.delayTime;
	resetFlags.restart(main.delayTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	valveIn.close(); // Close if any of these conditions are true
	valveOut.close();
	} else {
	valveIn.open(); // Open otherwise
	valveOut.open();
	}

Event: distortEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDistort) : PER_YEAR)
Trigger type	Rate
Show at runtime	true
Show name	true
Action	

Action	// Your existing logic for distortEvent
	isDistorted = true;
	endTimeForEvent = time() + main.distortionTime;
	main.globalTotalDistortionTime += main.distortionTime;
	resetFlags.restart(main.distortionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: disruptEvent

Name	Value
General	
Logging	true
Rate	(poisson(lambdaDisrupt) : PER_YEAR)
Trigger type	Rate
Show at runtime	true

Name	Value
Show name	true
Action	

Action	// Your existing logic for disruptEvent
	isDisrupted = true;
	endTimeForEvent = time() + main.disruptionTime;
	main.globalTotalDisruptionTime += main.disruptionTime;
	resetFlags.restart(main.disruptionTime);
	
	// Logic for controlling valveIn and valveOut
	if (isDelayed || isDistorted || isDisrupted) {
	  valveIn.close(); // Close if any of these conditions are true
	  valveOut.close();
	} else {
	  valveIn.open(); // Open otherwise
	  valveOut.open();
	}

Event: updateLambdaEvent

Name	Value
General	
Logging	true
Condition	!currentCountry.equals(previousCountry)
Trigger type	Condition
Show at runtime	true
Show name	true
Action	

Action	// Loop through all instances of FuelTanker
	for (FuelTanker fuelTanker : main.fuelTankers) { // Assuming 'fuelTankers' is the collection of FuelTanker agents
	String currentCountry = fuelTanker.currentCountry;
	
	// Define the total number of years for which you have data
	double totalYears = 2021 – 1970;
	
	// Query for lambdaDelay
	double countryDelayAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Utilities," "Business," "Transportation"))
	.count();
	double lambdaDelay = countryDelayAttacks / totalYears;
	fuelTanker.lambdaDelay = lambdaDelay;
	
	// Query for lambdaDisrupt
	double countryDisruptAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Violent Political Party," "Unknown," "Terrorists/Non-state Militia," "Airports and Aircraft," "Police," "Military," "Maritime," "Government (General)," "Government (Diplomatic)," "Telecommunication"))
	.count();
	double lambdaDisrupt = countryDisruptAttacks / totalYears;
	fuelTanker.lambdaDisrupt = lambdaDisrupt;
	
	// Query for lambdaDistort
	double countryDistortAttacks = selectFrom(db_1970_2020_db)
	
	.where(db_1970_2020_db.country_txt.eq(currentCountry),
	
	db_1970_2020_db.targtype1_txt.in("Educational Institution," "Food

Name	Value

	.count();
	
	double lambdaDistort = countryDistortAttacks / totalYears;
	fuelTanker.lambdaDistort = lambdaDistort;

Variable: destination

Name	Value
General	
Type	PipelineNode
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: source

Name	Value
General	

Type	PipelineNode
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDelayed

Name	Value
General	
Initial value	true
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDistorted

Name	Value
General	
Initial value	true

Name	Value
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: isDisrupted

Name	Value
General	
Initial value	true
Type	boolean
Show at runtime	true
Show name	true
Advanced	
Access type	public

Constant	false
Save in snapshot	true
System dynamics units	false

Variable: endTimeForEvent

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: currentCountry

Name	Value
General	
Initial value	“null”
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDelay

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDistort

Name	Value
General	
Type	double

Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: lambdaDisrupt

Name	Value
General	
Type	double
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false
Save in snapshot	true
System dynamics units	false

Variable: previousCountry

Name	Value
General	
Initial value	currentCountry
Type	String
Show at runtime	true
Show name	true
Advanced	
Access type	public
Constant	false

Name	Value
Save in snapshot	true
System dynamics units	false

Link to agents: connections

Name	Value
General	
Show at runtime	true
Show name	true
Communication	
Message type	Object
Animation	
Draw line	false

Database: Database

Name	Value
General	
Database shutdown compacr	false
Import	
List of tables import data settings	[– Import Settings, – Import Settings, – Import Settings, – Import Settings, – Import Settings, – Import Settings]
Export	
Auto export	false
Log	
Logging	false

Database Table: fueltrucks

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
source_name	VARCHAR(16777216)	-	-	
destination_name	VARCHAR(16777216)	-	-	
throughput	VARCHAR(16777216)	-	-	
crudeoil	VARCHAR(16777216)	-	-	

Database Table: db_1970_2020_db

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
eventid	DOUBLE	-	-	
iyear	INTEGER	-	-	

null	null	null	null	null
imonth	INTEGER	-	-	
iday	INTEGER	-	-	
approxdate	VARCHAR(16777216)	-	-	
extended	INTEGER	-	-	
resolution	TIMESTAMP	-	-	
country	INTEGER	-	-	
country_txt	VARCHAR(16777216)	-	-	
region	INTEGER	-	-	
region_txt	VARCHAR(16777216)	-	-	
provstate	VARCHAR(16777216)	-	-	
city	VARCHAR(16777216)	-	-	

latitude	DOUBLE	-	-	
longitude	DOUBLE	-	-	
specificity	INTEGER	-	-	
vicinity	INTEGER	-	-	
location	VARCHAR(16777216)	-	-	
summary	VARCHAR(16777216)	-	-	
crit1	INTEGER	-	-	
crit2	INTEGER	-	-	
crit3	INTEGER	-	-	
doubtterr	INTEGER	-	-	
alternative	INTEGER	-	-	
alternative_txt	VARCHAR(16777216)	-	-	
multiple	INTEGER	-	-	
success	INTEGER	-	-	
suicide	INTEGER	-	-	
attacktype1	INTEGER	-	-	
attacktype1_txt	VARCHAR(16777216)	-	-	
attacktype2	INTEGER	-	-	
attacktype2_txt	VARCHAR(16777216)	-	-	
attacktype3	INTEGER	-	-	
attacktype3_txt	VARCHAR(16777216)	-	-	
targtype1	INTEGER	-	-	
targtype1_txt	VARCHAR(16777216)	-	-	
targsubtype1	INTEGER	-	-	
targsubtype1_txt	VARCHAR(16777216)	-	-	
corp1	VARCHAR(16777216)	-	-	
target1	VARCHAR(16777216)	-	-	
natlty1	INTEGER	-	-	
natlty1_txt	VARCHAR(16777216)	-	-	
targtype2	INTEGER	-	-	
targtype2_txt	VARCHAR(16777216)	-	-	
targsubtype2	INTEGER	-	-	
targsubtype2_txt	VARCHAR(16777216)	-	-	
corp2	VARCHAR(16777216)	-	-	
target2	VARCHAR(16777216)	-	-	
natlty2	INTEGER	-	-	
natlty2_txt	VARCHAR(16777216)	-	-	

null	null	null	null	null
targtype3	INTEGER	-	-	
targtype3_txt	VARCHAR(16777216)	-	-	
targsubtype3	INTEGER	-	-	
targsubtype3_txt	VARCHAR(16777216)	-	-	
corp3	VARCHAR(16777216)	-	-	
target3	VARCHAR(16777216)	-	-	
natlty3	INTEGER	-	-	
natlty3_txt	VARCHAR(16777216)	-	-	

gname	VARCHAR(16777216)	-	-	
gsubname	VARCHAR(16777216)	-	-	
gname2	VARCHAR(16777216)	-	-	
gsubname2	VARCHAR(16777216)	-	-	
gname3	VARCHAR(16777216)	-	-	
gsubname3	VARCHAR(16777216)	-	-	
motive	VARCHAR(16777216)	-	-	
guncertain1	INTEGER	-	-	
guncertain2	INTEGER	-	-	
guncertain3	INTEGER	-	-	
individual	INTEGER	-	-	
nperps	INTEGER	-	-	
nperpcap	DOUBLE	-	-	
claimed	INTEGER	-	-	
claimmode	INTEGER	-	-	
claimmode_txt	VARCHAR(16777216)	-	-	
claim2	INTEGER	-	-	
claimmode2	INTEGER	-	-	
claimmode2_txt	VARCHAR(16777216)	-	-	
claim3	INTEGER	-	-	
claimmode3	INTEGER	-	-	
claimmode3_txt	VARCHAR(16777216)	-	-	
compclaim	INTEGER	-	-	
weaptype1	INTEGER	-	-	
weaptype1_txt	VARCHAR(16777216)	-	-	
weapsubtype1	INTEGER	-	-	
weapsubtype1_txt	VARCHAR(16777216)	-	-	
weaptype2	INTEGER	-	-	
weaptype2_txt	VARCHAR(16777216)	-	-	
weapsubtype2	INTEGER	-	-	
weapsubtype2_txt	VARCHAR(16777216)	-	-	
weaptype3	INTEGER	-	-	
weaptype3_txt	VARCHAR(16777216)	-	-	
weapsubtype3	INTEGER	-	-	
weapsubtype3_txt	VARCHAR(16777216)	-	-	
weaptype4	INTEGER	-	-	
weaptype4_txt	VARCHAR(16777216)	-	-	
weapsubtype4	INTEGER	-	-	
weapsubtype4_txt	VARCHAR(16777216)	-	-	
weapdetail	VARCHAR(16777216)	-	-	

null	null	null	null	null
nkill	INTEGER	-	-	
nkillus	INTEGER	-	-	
nkillter	INTEGER	-	-	
nwound	INTEGER	-	-	
nwoundus	INTEGER	-	-	

nwoundte	INTEGER	-	-	
property	INTEGER	-	-	
propextent	INTEGER	-	-	
propextent_txt	VARCHAR(16777216)	-	-	
propvalue	DOUBLE	-	-	
propcomment	VARCHAR(16777216)	-	-	
ishostkid	INTEGER	-	-	
nhostkid	INTEGER	-	-	
nhostkidus	INTEGER	-	-	
nhours	DOUBLE	-	-	
ndays	INTEGER	-	-	
divert	VARCHAR(16777216)	-	-	
kidhijcountry	VARCHAR(16777216)	-	-	
ransom	INTEGER	-	-	
ransomamt	DOUBLE	-	-	
ransomamtus	DOUBLE	-	-	
ransompaid	DOUBLE	-	-	
ransompaidus	INTEGER	-	-	
ransomnote	VARCHAR(16777216)	-	-	
hostkidoutcome	INTEGER	-	-	
hostkidoutcome_txt	VARCHAR(16777216)	-	-	
nreleased	INTEGER	-	-	
addnotes	VARCHAR(16777216)	-	-	
scite1	VARCHAR(16777216)	-	-	
scite2	VARCHAR(16777216)	-	-	
scite3	VARCHAR(16777216)	-	-	
dbsource	VARCHAR(16777216)	-	-	
int_log_db	INTEGER	-	-	
int_ideo	INTEGER	-	-	
int_misc	INTEGER	-	-	
int_any	INTEGER	-	-	
related	VARCHAR(16777216)	-	-	

Database Table: db_2021_db

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
eventid	DOUBLE	-	-	
iyear	INTEGER	-	-	

null	null	null	null	null
imonth	INTEGER	-	-	
iday	INTEGER	-	-	
approxdate	VARCHAR(16777216)	-	-	

extended	INTEGER	-	-	
resolution	TIMESTAMP	-	-	
country	INTEGER	-	-	
country_txt	VARCHAR(16777216)	-	-	
region	INTEGER	-	-	
region_txt	VARCHAR(16777216)	-	-	
provstate	VARCHAR(16777216)	-	-	
city	VARCHAR(16777216)	-	-	
latitude	DOUBLE	-	-	
longitude	DOUBLE	-	-	
specificity	INTEGER	-	-	
vicinity	INTEGER	-	-	
location	VARCHAR(16777216)	-	-	
summary	VARCHAR(16777216)	-	-	
crit1	INTEGER	-	-	
crit2	INTEGER	-	-	
crit3	INTEGER	-	-	
doubtterr	INTEGER	-	-	
alternative	INTEGER	-	-	
alternative_txt	VARCHAR(16777216)	-	-	
multiple	INTEGER	-	-	
success	INTEGER	-	-	
suicide	INTEGER	-	-	
attacktype1	INTEGER	-	-	
attacktype1_txt	VARCHAR(16777216)	-	-	
attacktype2	INTEGER	-	-	
attacktype2_txt	VARCHAR(16777216)	-	-	
attacktype3	INTEGER	-	-	
attacktype3_txt	VARCHAR(16777216)	-	-	
targtype1	INTEGER	-	-	
targtype1_txt	VARCHAR(16777216)	-	-	
targsubtype1	INTEGER	-	-	
targsubtype1_txt	VARCHAR(16777216)	-	-	
corp1	VARCHAR(16777216)	-	-	
target1	VARCHAR(16777216)	-	-	
natlty1	INTEGER	-	-	
natlty1_txt	VARCHAR(16777216)	-	-	
targtype2	INTEGER	-	-	
targtype2_txt	VARCHAR(16777216)	-	-	
targsubtype2	INTEGER	-	-	
targsubtype2_txt	VARCHAR(16777216)	-	-	
corp2	VARCHAR(16777216)	-	-	
target2	VARCHAR(16777216)	-	-	
natlty2	INTEGER	-	-	
natlty2_txt	VARCHAR(16777216)	-	-	

null	null	null	null	null

targtype3	INTEGER	-	-	
targtype3_txt	VARCHAR(16777216)	-	-	
targsubtype3	INTEGER	-	-	
targsubtype3_txt	VARCHAR(16777216)	-	-	
corp3	VARCHAR(16777216)	-	-	
target3	VARCHAR(16777216)	-	-	
natlty3	INTEGER	-	-	
natlty3_txt	VARCHAR(16777216)	-	-	
gname	VARCHAR(16777216)	-	-	
gsubname	VARCHAR(16777216)	-	-	
gname2	VARCHAR(16777216)	-	-	
gsubname2	VARCHAR(16777216)	-	-	
gname3	VARCHAR(16777216)	-	-	
gsubname3	VARCHAR(16777216)	-	-	
motive	VARCHAR(16777216)	-	-	
guncertain1	INTEGER	-	-	
guncertain2	INTEGER	-	-	
guncertain3	INTEGER	-	-	
individual	INTEGER	-	-	
nperps	INTEGER	-	-	
nperpcap	INTEGER	-	-	
claimed	INTEGER	-	-	
claimmode	INTEGER	-	-	
claimmode_txt	VARCHAR(16777216)	-	-	
claim2	INTEGER	-	-	
claimmode2	INTEGER	-	-	
claimmode2_txt	VARCHAR(16777216)	-	-	
claim3	INTEGER	-	-	
claimmode3	INTEGER	-	-	
claimmode3_txt	VARCHAR(16777216)	-	-	
compclaim	INTEGER	-	-	
weaptype1	INTEGER	-	-	
weaptype1_txt	VARCHAR(16777216)	-	-	
weapsubtype1	INTEGER	-	-	
weapsubtype1_txt	VARCHAR(16777216)	-	-	
weaptype2	INTEGER	-	-	
weaptype2_txt	VARCHAR(16777216)	-	-	
weapsubtype2	INTEGER	-	-	
weapsubtype2_txt	VARCHAR(16777216)	-	-	
weaptype3	INTEGER	-	-	
weaptype3_txt	VARCHAR(16777216)	-	-	
weapsubtype3	INTEGER	-	-	
weapsubtype3_txt	VARCHAR(16777216)	-	-	
weaptype4	VARCHAR(16777216)	-	-	
weaptype4_txt	VARCHAR(16777216)	-	-	
weapsubtype4	VARCHAR(16777216)	-	-	
weapsubtype4_txt	VARCHAR(16777216)	-	-	
weapdetail	VARCHAR(16777216)	-	-	

null	null	null	null	null
nkill	INTEGER	-	-	
nkillus	INTEGER	-	-	
nkillter	INTEGER	-	-	
nwound	INTEGER	-	-	
nwoundus	INTEGER	-	-	
nwoundte	INTEGER	-	-	
property	INTEGER	-	-	
propextent	INTEGER	-	-	
propextent_txt	VARCHAR(16777216)	-	-	
propvalue	DOUBLE	-	-	
propcomment	VARCHAR(16777216)	-	-	
ishostkid	INTEGER	-	-	
nhostkid	INTEGER	-	-	
nhostkidus	INTEGER	-	-	
nhours	DOUBLE	-	-	
ndays	INTEGER	-	-	
divert	VARCHAR(16777216)	-	-	
kidhijcountry	VARCHAR(16777216)	-	-	
ransom	INTEGER	-	-	
ransomamt	DOUBLE	-	-	
ransomamtus	INTEGER	-	-	
ransompaid	DOUBLE	-	-	
ransompaidus	INTEGER	-	-	
ransomnote	VARCHAR(16777216)	-	-	
hostkidoutcome	INTEGER	-	-	
hostkidoutcome_txt	VARCHAR(16777216)	-	-	
nreleased	INTEGER	-	-	
addnotes	VARCHAR(16777216)	-	-	
scite1	VARCHAR(16777216)	-	-	
scite2	VARCHAR(16777216)	-	-	
scite3	VARCHAR(16777216)	-	-	
dbsource	VARCHAR(16777216)	-	-	
int_log_db	INTEGER	-	-	
int_ideo	INTEGER	-	-	
int_misc	INTEGER	-	-	
int_any	INTEGER	-	-	
related	VARCHAR(16777216)	-	-	

Database Table: pipelines

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null

source_name	VARCHAR(16777216)	-	-	
destination_name	VARCHAR(16777216)	-	-	

null	null	null	null	null
throughput	INTEGER	-	-	
crudeoil	BOOLEAN	-	-	

Database Table: storages

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
location_name	VARCHAR(16777216)	-	-	
capacity	INTEGER	-	-	
crudeoil	BOOLEAN	-	-	
terminal	BOOLEAN	-	-	

Database Table: tankers

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
loading_lat	DOUBLE	-	-	
loading_lon	DOUBLE	-	-	
destination_name	VARCHAR(16777216)	-	-	

Database Table: refineries

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
location_name	VARCHAR(16777216)	-	-	
capacity	INTEGER	-	-	
refining_rate	DOUBLE	-	-	

Database Table: airtankers

Name	Value
Advanced	

Cached database table	false

null	null	null	null	null
route_id	VARCHAR(16777216)	-	-	

null	null	null	null	null
source_name	VARCHAR(16777216)	-	-	
destination_name	VARCHAR(16777216)	-	-	
throughput	INTEGER	-	-	
crudeoil	BOOLEAN	-	-	
number	INTEGER	-	-	

Database Table: fueltankers

Name	Value
Advanced	
Cached database table	false

null	null	null	null	null
route_id	INTEGER	-	-	
source_name	VARCHAR(16777216)	-	-	
destination_name	VARCHAR(16777216)	-	-	
throughput	INTEGER	-	-	
crudeoil	BOOLEAN	-	-	

LIST OF REFERENCES

Anglani, Norma, Giovanna Oriti, Ruth Fish, and Douglas L. Van Bossuyt. 2022. "Design and Optimization Strategy to Size Resilient Stand-Alone Hybrid Microgrids in Various Climatic Conditions." *IEEE Open Journal of Industry Applications* 3: 237–46. https://doi.org/10.1109/OJIA.2022.3201161.

Anuat, Edward, Douglas L. Van Bossuyt, and Anthony Pollman. 2021. "Energy Resilience Impact of Supply Chain Network Disruption to Military Microgrids." *Infrastructures* 7(1): 4. https://doi.org/10.3390/infrastructures7010004.

"AnyLogic: Simulation Modeling Software Tools & Solutions for Business." 2023. https://www.anylogic.com/.

Asadu, Chinedu. 2022. "Frustration Grows in Nigeria at Continuing Fuel Shortage." *AP News*, 2022. https://apnews.com/article/business-africa-nigeria-west-africa-abuja-49e14190da69170a3120ec25f968034b.

Benverren, Fortune. 2022. "DLA Energy Support to USAFRICOM." Brief presented at the AFRICOM J435 DLA Energy Meeting, October 18. https://dod365.sharepoint-mil.us/:p:/r/sites/AFRICOM-J435/_layouts/15/Doc.aspx?sourcedoc=%7b89bcd1fc-ae73-46d0-b08d-bf92c0c72b43%7d&file=dla%20ea%20-%20africom%20acj4%20briefing%20slides%20-%2014%20oct%20-%20notes%20deleted.pptx&action=edit&mobileredirect=true&DefaultItemOpen=1.

Chenoweth, Erica, Jonathan Pinckney, and Orion A. Lewis. 2019. "NAVCO 3.0 Dataset." Harvard Dataverse. https://doi.org/10.7910/dvn/innyeo.

Clauset, Aaron, and Ryan Woodard. 2012. "Estimating the Historical and Future Probabilities of Large Terrorist Events." https://doi.org/10.48550/ARXIV.1209.0089.

Defense Logistics Agency. 2023. "Bulk Petroleum Services." Defense Logistics Agency The Nation's Combat Logistics Support Agency. October 19, 2023. https://www.dla.mil/Energy/Services/Bulk-Petroleum-Services/.

Department of Defense. 2019. "Lead Inspector General Report to Congress: East Africa and North and West Africa Counterterrorism Operations." December 31, 2019. https://media.defense.gov/2020/Feb/21/2002252793/-1/-1/1/lead%20ig%20east%20africa%20and%20north%20and%20west%20africa%20counterterrorism%20operations.pdf.

———. 2020. “Metrics and Standards for Energy Resilience.” https://www.acq.osd.mil/eie/Downloads/IE/Metrics%20and%20Standards%20for%20Energy%20Resilience%2020%20May%202021.pdf.

Department of Peace and Conflict Research. 2021. “UCDP – Uppsala Conflict Data Program.” 2021. https://ucdp.uu.se/.

“DOD Instruction 5200.44 (Change 3): Protection of Mission Critical Functions to Achieve Trust Systems and Networks (TSN).” 2018. Department of Defense: Washington, DC, USA. https://www.esd.whs.mil/portals/54/documents/dd/issuances/dodi/520044p.pdf.

Ezell, Barry Charles, Steven P. Bennett, Detlof Von Winterfeldt, John Sokolowski, and Andrew J. Collins. 2010. “Probabilistic Risk Analysis and Terrorism Risk.” *Risk Analysis* 30 (4): 575–89. https://doi.org/10.1111/j.1539-6924.2010.01401.x.

Faisal, Mohd. Nishat, D. K. Banwet, and Ravi Shankar. 2007. “Management of Risk in Supply Chains: SCOR Approach and Analytic Network Process.” *Supply Chain Forum: An International Journal* 8 (2): 66–79. https://doi.org/10.1080/16258312.2007.11517183.

Fan, Yiyi, and Mark Stevenson. 2018. “A Review of Supply Chain Risk Management: Definition, Theory, and Research Agenda.” *International Journal of Physical Distribution & Logistics Management* 48 (3): 205–30. https://doi.org/10.1108/IJPDLM-01-2017-0043.

Genova, Ann, and Toyin Falola. 2003. “Oil in Nigeria: A Bibliographical Reconnaissance.” *History in Africa* 30: 133–56. https://doi.org/10.1017/S0361541300003181.

Giachetti, Ron, Giovanna Oriti, Daniel Reich, Susan M. Sanchez, and Douglas Van Bossuyt. 2023. “Open-Source Microgrid Design Planning.” Open-Source Microgrid Design Planning. March 2023. https://microgrid.nsetti.nps.edu/.

Giachetti, Ronald E., Douglas L. Van Bossuyt, William W. Anderson, and Giovanna Oriti. 2022. “Resilience and Cost Trade Space for Microgrids on Islands.” *IEEE Systems Journal* 16 (3): 3939–49. https://doi.org/10.1109/JSYST.2021.3103831.

Giroux, Jennifer. 2009. “Targeting Energy Infrastructure: Examining the Terrorist Threat in North Africa and Its Broader Implications.” *Elcano Newsletter*, no. 53 (February): 10 p.

Huamaní, Enrique Lee, Alva Mantari, and Avid Roman-Gonzalez. 2020. “Machine Learning Techniques to Visualize and Predict Terrorist Attacks Worldwide Using the Global Terrorism Database.” *International Journal of Advanced Computer Science and Applications* 11 (4). https://doi.org/10.14569/IJACSA.2020.0110474.

Hunkuyi, Magaji Lsa, and Hope Abah Emmanuel. 2021. "Nigeria: 93 Checkpoints – How Extortions, Multiple Taxes Force Transporters Off Taraba, Benue Roads." *Daily Trust*, February 11, 2021, sec. News. https://allafrica.com/stories/202102110172.html.

Inman, Ronald. 2017. "Refinery to Flight." *Defense Logistics Agency*, November 1, 2017. https://www.dla.mil/About-DLA/News/Energy/Article/1370566/refinery-to-flight/https%3A%2F%2Fwww.dla.mil%2FAbout-DLA%2FNews%2FNews-Article-View%2FArticle%2F1370566%2Frefinery-to-flight%2F.

Keller, Gerald. 2017. *Statistics for Management and Economics*. 11th ed. Boston: Cengage Learning.

Kennedy-Darling, Julia, Nick Hoyt, Kyle Murao, and Allison Ross. 2008. "The Energy Crisis of Nigeria An Overview and Implications for the Future," June.

Kreisher, Otto. 2010. "Move That Gas." *Air & Space Forces Magazine*, September. https://www.airandspaceforces.com/article/0910gas/.

Lambrechts, Derica, and Lars B. Blomquist. 2017. "Political–Security Risk in the Oil and Gas Industry: The Impact of Terrorism on Risk Management and Mitigation." *Journal of Risk Research* 20 (10): 1320–37. https://doi.org/10.1080/13669877.2016.1153502.

Lockheed Martin. 2015. "Leading Aerospace and Defense." https://www.lockheedmartin.com/en-us/index.html.

McCarthy, Rory. 2022. "Transgressive Protest after a Democratic Transition: The Kamour Campaign in Tunisia." *Social Movement Studies* 21 (6): 798–815. https://doi.org/10.1080/14742837.2021.1967128.

NAVFAC. 2023. "FC 2–000-05N Facility Planning Criteria for Navy and Marine Corps Shore Installations | WBDG – Whole Building Design Guide." 2023. https://wbdg.org/ffc/dod/unified-facilities-criteria-ufc/fc-2-000-05n.

Njanji, Susan. 2021. "South Africa to Deploy 25,000 Troops to Curb Unrest over Food, Fuel Shortages." *The Times of Israel*, July 15, 2021. https://www.timesofisrael.com/south-africa-seeks-to-deploy-25k-troops-to-curb-unrest-over-food-fuel-shortages/.

Port of Rotterdam. 2023. "The 5 Oil Refineries in Rotterdam | Port of Rotterdam." 2023. https://www.portofrotterdam.com/en/setting/industry-port/refining-and-chemicals/oil-refineries.

Reece, Beth. 2019. "Supply Problems – DLA Solutions." *Defense Logistics Agency*, September 1, 2019. https://www.dla.mil/About-DLA/News/News-Article-View/Article/1963753/supply-problems-dla-solutions/https%3A%2F%2Fwww.dla.mil%2FAbout-DLA%2FNews%2FNews-Article-View%2FArticle%2F1963753%2Fsupply-problems-dla-solutions%2F.

Rossetti, Manuel D., and Juliana Bright. 2018. "Bulk Petroleum Supply Chain Simulation Modeling." In *2018 Winter Simulation Conference (WSC)*, 3060–71. Gothenburg, Sweden: IEEE. https://doi.org/10.1109/WSC.2018.8632343.

Shawn, Terry. 2015. "DLA Energy's Pacific Pivot Begins in the Americas." *Defense Logistics Agency*, June 24, 2015. https://www.dla.mil/About-DLA/News/Energy/Article/617985/dla-energys-pacific-pivot-begins-in-the-americas/https%3A%2F%2Fwww.dla.mil%2FAbout-DLA%2FNews%2FNews-Article-View%2FArticle%2F617985%2Fdla-energys-pacific-pivot-begins-in-the-americas%2F.

Smith, Irene. 2023. "DLA Energy Europe & Africa Supports NATO Pipeline." *Defense Logistics Agency*, March 6, 2023. https://www.dla.mil/About-DLA/News/News-Article-View/Article/3320172/dla-energy-europe-africa-supports-nato-pipeline/https%3A%2F%2Fwww.dla.mil%2FAbout-DLA%2FNews%2FNews-Article-View%2FArticle%2F3320172%2Fdla-energy-europe-africa-supports-nato-pipeline%2F.

Solar Optimum. 2023. "What Are Tesla Powerwall Dimensions and Space Requirements?" *Solar Optimum* (blog). July 17, 2023. https://solaroptimum.com/blog/2023/07/17/what-are-tesla-powerwall-dimensions-and-space-requirements/.

Staff, FreightWaves. 2020. "How Many Gallons of Fuel Does a Container Ship Carry?" FreightWaves. January 16, 2020. https://www.freightwaves.com/news/how-many-gallons-of-fuel-does-a-container-ship-carry.

START (National Consortium for the Study of Terrorism and Responses to Terrorism). 2022. "Global Terrorism Database 1970 – 2020." https://www.start.umd.edu/gtd/.

Tema Oil Terminal. 2023. "Tema Oil Terminal – A Bulk Fuel Storage and Distribution Terminal – Home." 2023. https://totgh.com/.

Ton, Dan T., and Merrill A. Smith. 2012. "The U.S. Department of Energy's Microgrid Initiative." *The Electricity Journal* 25 (8): 84–94. https://doi.org/10.1016/j.tej.2012.09.013.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California