**Athena User's Guide**

Athena Regional Stability Simulation, V6.2

September, 2014

William H. Duquette

David R. Hanks

Jet Propulsion Laboratory

*Copyright 2008-2014, by the California Institute of Technology. ALL RIGHTS RESERVED. United States Government Sponsorship acknowledged. Any commercial use must be negotiated with the Office of Technology Transfer at the California Institute of Technology.*

*This software is subject to U.S. export control laws and regulations and has been classified as EAR99. By accepting this software, the user agrees to comply with all applicable U.S. export laws and regulations. User has the responsibility to obtain export licenses, or other export authority as may be required before exporting such information to foreign countries or providing access to foreign persons.*

**Table of Contents**

Part I: Overview 10

1. Introduction 12

1.1 Overview of This Document 12

1.2 Other Documents 12

1.3 Changes for Athena 6 13

1.3.1 Changes in Athena 6.1 13

1.3.1.1 PMESII-PT 13

1.3.1.2 Mark II Strategy Engine 13

1.3.1.3 Goods Production Infrastructure 14

1.3.1.4 Economics Model Changes 14

1.3.1.5 Expression and Script Syntax Checking 14

1.3.1.6 Scenario Export Scripts 15

1.3.2 Changes in Athena 6.2 15

1.3.2.1 Belief System Upgrade 15

1.3.2.2 Web Map Service Support 15

1.3.2.3 Environmental Situations are now Abstract Situations 15

1.3.2.4 New Abstract Situations 15

1.3.2.5 Abstract Events 15

1.3.2.6 Intel Ingestion Wizard 16

1.3.2.7 Neighborhood Ingestion Wizard 16

1.3.2.8 Athena Attrition Model Removed 16

2. Athena Model Overview 17

2.1 The Simulation and its Objects 17

2.2 The Modeling Areas 18

2.3 Basic Concepts 20

3. The Time Area 21

4. The Physical Area 22

4.1 The Playbox 22

4.1.1 Neighborhoods 22

4.1.2 Neighborhood Proximity 23

4.1.3 Local vs. Non-Local Neighborhoods 23

4.2 Groups 23

4.3 Deployment 24

4.4 Activity Assignment 24

4.5 Units 24

4.6 Security 24

4.7 Coverage 26

4.8 Activity Situations 27

4.9 Abstract Situations 27

4.9.1 Abstract Situations and URAM 27

4.10 Essential Non-Infrastructure (ENI) Services 28

4.10.1 The Notion of a Service 28

4.10.2 Measurement of ENI Services 29

4.10.3 Required Level of ENI Services 29

4.10.4 Effects of ENI Services 30

4.10.5 Services vs. Abstract Situations 30

5. The Political Area 31

5.1 Actors 31

5.2 Strategies: Blocks, Tactics, and Conditions 32

5.2.1 Assets 32

5.2.2 Conditions 33

5.2.3 Tactics 33

5.2.4 Blocks 33

5.2.5 Strategy Execution 34

5.2.6 Strategy Execution On Lock 34

5.2.7 Other Block Options 34

5.2.8 What the Actor Knows, and When He Knows It 35

5.2.9 Expenditures 35

5.3 Vertical Relationships 35

5.3.1 Force and Organization Groups 36

5.3.2 Civilian Groups 36

5.4 Support, Influence, and Control 37

5.4.1 Support 37

5.4.2 Influence 38

5.4.3 Control 38

5.4.4 When Control Shifts 39

6. The Military Area 40

6.1 Force Groups 40

6.2 Force Activities 40

6.3 Magic Attrition 41

7. The Economics Area 42

7.1 Dollars 42

7.2 Sectors 42

7.3 Shape vs. Size 43

7.4 Economic Outputs 44

7.5 Neighborhood Aggregation/Disaggregation 44

7.5.1 Neighborhood Aggregation 45

7.5.2 Neighborhood Disaggregation 45

7.6 Ways to Affect the Economy 45

7.7 Effects of the Economy 46

8. The Social Area 48

8.1 Civilian and Organization Groups 48

8.1.1 Civilian Groups 48

8.1.2 Organization Groups 48

8.2 Demographics 49

8.2.1 Base Population 49

8.2.2 Current Population 49

8.2.3 Subsistence Agriculture 50

8.2.4 Consumers and Workers 50

8.2.5 Demographic Situations 50

8.2.5.1 Consumption of Goods 50

8.2.5.2 Unemployment 51

8.3 Belief Systems and Affinities 51

8.3.1 Beliefs and Topics 51

8.3.2 Affinity 52

8.3.3 Playbox Commonality 53

8.3.4 Entity Commonality 53

8.4 Attitudes 54

8.4.1 Unified Regional Attitude Model (URAM) 54

8.4.1.1 URAM Attitude Curves 54

8.4.1.2 Persistent and Transient Inputs 55

8.5 Horizontal Relationships 55

8.6 Satisfaction Levels 56

8.6.1 The Four Concerns 56

8.6.2 Saliencies 57

8.6.3 Group Mood 57

8.7 The Driver Assessment Model (DAM) 57

8.7.1 Direct and Indirect Effects 57

8.8 Magic Changes to Attitudes 58

8.8.1 Complex User-defined Role-based Situations and Events (CURSEs) 58

8.8.2 Roles in CURSEs 58

9. The Infrastructure Area 60

9.1 Communications Asset Packages (CAPs) 60

9.1.1 CAP Coverage 60

9.1.2 Access to CAPs 61

9.1.3 CAP Costs 61

9.2 Goods Production Infrastructure 61

9.2.1 Scenario Preparation 62

9.2.2 Levels of Repair 63

9.2.3 Building New Plants 63

9.2.4 Auto-Maintenance 63

9.2.5 Total Production Capacity and Unemployment 64

10. The Information Area 65

10.1 Cooperation Levels 65

10.2 Information Operations Messages (IOMs) 66

10.2.1 IOM Description 66

10.2.2 Semantic Hooks 66

10.2.3 IOM Payloads 67

10.2.4 Broadcasting an IOM 67

10.2.5 Effectiveness of IOM Payloads 68

Part II: Using Athena 69

11. Installation 70

11.1 Starting Athena 70

11.2 Athena Documentation 70

11.3 Multiple Versions of Athena 70

12. Using the Athena Application 71

12.1 Athena Scenario Files 71

12.2 Athena Workflow 71

12.3 Scenario Mode vs. Simulation Mode 72

12.4 Viewing Athena Results 72

12.5 The Significant Events Log 72

12.6 The Debugging Log 73

12.7 Background Errors 73

12.8 Athena Scripting 73

12.8.1 Executive Commands 73

12.8.2 The Athena Command Line 74

12.8.3 Executive Scripts 74

12.8.3.1 Internal vs. External Scripts 74

12.8.3.2 Procedures vs. Scripts 74

12.8.3.3 Resetting the Executive 76

12.8.4 Scenario Scripts 76

12.9 Batch Mode 77

12.9.1 Invoking Athena in Batch Mode 77

12.9.2 Simulation Control 78

12.9.3 Simulation Results 78

12.10 Athena Experiments 78

12.10.1 The Base Scenario 79

12.10.2 Experiment Cases and Case Parameters 79

12.10.3 The Experiment Database 79

12.10.4 Experimental Results and Case Outcomes 79

12.10.5 Workflow 80

12.11 Athena PBS (Portable Batch System) 81

12.11.1 Terms 81

12.11.2 Running with Athena PBS 82

12.11.3 Monitoring the Cluster 83

12.11.4 Results 84

13. The Athena User Interface 85

13.1 The Main Window 85

13.2 The Menu System 86

13.2.1 The File Menu 86

13.2.2 The Edit Menu 86

13.2.3 The View Menu 86

13.2.4 The Bookmarks Menu 87

13.2.5 The Orders Menu 87

13.2.6 The Wizards Menu 87

13.2.7 The Help Menu 87

13.3 The Main Toolbar 88

13.3.1 The Scenario Mode Toolbar 88

13.3.2 The Simulation Mode Toolbar 88

13.4 The Command Line 89

13.5 The Detail Browser 90

13.5.1 Detail Browser Sidebar 91

13.5.2 Detail Browser Search Box 91

13.5.3 Context Menu 91

13.5.4 Useful URLs 91

13.6 The Strategy Browser 93

13.6.1 The Agent List 93

13.6.2 The Block List 93

13.6.3 The Block Overview Tab 95

13.6.4 The Block Condition Tab 95

13.6.5 The Block Tactic Tab 96

13.7 The Physical Tab 98

13.7.1 The Physical/Map Tab 98

13.7.1.1 The Map Proper 98

13.7.1.2 Map Reference Strings 99

13.7.1.3 Neighborhood Polygons 99

13.7.1.4 Unit and Abstract Situation Icons 99

13.7.1.5 Map Browser: Main Toolbar 100

13.7.1.6 The Secondary Toolbar 101

13.7.1.7 Context Menus 101

13.7.2 The Physical/Neighborhoods Tab 102

13.7.3 The Physical/Proximities Tab 102

13.7.4 The Physical/Absits Tab 102

13.7.5 The Physical/CURSEs Tab 102

13.7.6 The Physical/Security Tab 103

13.7.7 The Physical/Activity Tab 103

13.7.8 The Physical/NbhoodDemog Tab 103

13.8 The Political Tab 104

13.8.1 The Political/Actors Tab 104

13.8.2 The Political/VRel Tab 104

13.9 The Military Tab 105

13.9.1 The Military/Force Groups Tab 105

13.10 The Economics Tab 106

13.10.1 The Economics/SAM Tab 107

13.10.2 The Economics/Overview Tab 107

13.10.3 The Economics/Expenditures Tab 107

13.10.4 The Economics/Population Tab 107

13.10.5 The Economics/Groups Tab 107

13.11 The Social Tab 108

13.11.1 The Social/Beliefs Tab 108

13.11.1.1 The Topics Pane 109

13.11.1.2 The Entities Tree 109

13.11.1.3 The Beliefs Pane 110

13.11.1.4 The Affinities Pane 110

13.11.2 The Social/CivGroups Tab 111

13.11.3 The Social/OrgGroups Tab 111

13.11.4 The Social/Satisfaction Tab 111

13.11.5 The Social/HRel Tab 111

13.11.6 The Social/Demog Tab 111

13.12 The Infrastructure Tab 112

13.12.1 The Infrastructure/CAPs Tab 112

13.12.2 The Infrastructure/GOODS Plants Tab 112

13.13 The Information Tab 114

13.13.1 The Information/Cooperation Tab 114

13.13.2 The Information/NbhoodCoop Tab 114

13.13.3 The Information/Semantic Hooks Tab 114

13.13.4 The Information/Info Ops Messages Tab 114

13.14 The Time Tab 116

13.14.1 The Time/Significant Events Tab 116

13.14.2 The Time/Orders Tab 116

13.14.3 The Time/Log Tab 117

13.15 The Bookmarks Tab 118

13.16 The Scripts Tab 119

13.17 Athena Wizards 119

13.17.1 The Neighborhood Ingestion Wizard 120

13.17.2 The Intel Ingestion Wizard 120

14. Creating an Athena Scenario 121

14.1 The Actors 121

14.1.1 Sources of Actor Income 121

14.1.2 Use of Political Support 122

14.2 The Map 122

14.3 The Neighborhoods 122

14.4 Neighborhood Proximities 123

14.5 Civilian Groups 124

14.6 Force Groups 124

14.7 Organization Groups 125

14.8 Belief Systems 125

14.8.1 Define the Topics 125

14.8.2 Define the Beliefs 126

14.8.3 Compute the Affinities 126

14.8.4 Adjust the Affinities 126

14.9 Horizontal Relationships 127

14.10 Vertical Relationships 127

14.11 Group Satisfaction 128

14.12 Group Cooperation 128

14.13 Abstract Situations 128

14.14 GOODS Production Infrastructure 129

14.15 The Economic Model 129

14.15.1 The *goods* sector 130

14.15.2 The *black* sector 130

14.15.3 The *pop* sector 130

14.15.4 The *actors* sector 130

14.15.5 The *region* sector 130

14.15.6 The *world* sector 131

14.15.7 Other Inputs 131

14.15.8 Calibration 131

14.16 Information Operations Campaigns 132

14.17 CURSEs 132

14.18 Strategies 132

14.18.1 The Role of an Actor’s Strategy 133

14.18.2 Order Matters 133

14.19 The Model Parameter Database 133

14.19.1 Model Parameters 134

14.19.2 Browsing the Model Parameter Database 134

14.19.3 Parameter Help 134

14.19.4 Editing Model Parameters 135

14.19.5 Model Parameter Files 135

14.20 Rebasing the Scenario 135

Part III: Athena Cookbook 136

15. Recipes 137

15.1 Disabling the Economic Model 137

15.2 Comparing Satisfaction Changes to the Real World 137

Part IV: Reference 139

16. Athena Objects 140

16.1 Tactics, Conditions, Injects, and Payloads 140

16.2 Neighborhoods 141

16.3 Actors 142

16.4 Civilian Groups 145

16.5 Force Groups 146

16.6 Organization Groups 147

16.7 Communications Asset Packages 148

16.8 Information Operations Messages 149

16.9 Semantic Hooks 150

16.10 GOODS Production Infrastructure 151

16.11 Complex User-defined Role-based Situations and Events (CURSEs) 152

16.12 Abstract Situations 153

16.12.1 Operations on Abstract Situations 153

16.12.2 Attributes of Abstract Situations 154

16.13 Horizontal Relationships 155

16.14 Vertical Relationships 156

16.15 Satisfaction Levels 157

16.16 Cooperation Levels 158

17. Tactic Types 159

17.1 ABSIT 160

17.2 ACCIDENT 161

17.3 ASSIGN 162

17.3.1 Activities 164

17.4 ATTRIT 165

17.5 BROADCAST 166

17.6 BUILD 167

17.7 CURSE 168

17.8 DAMAGE 169

17.9 DEMO 170

17.10 DEMOB 171

17.11 DEPLOY 172

17.12 DEPOSIT 174

17.13 EXECUTIVE 175

17.14 EXPLOSION 176

17.15 FLOW 177

17.16 FUND 178

17.17 FUNDENI 180

17.18 GRANT 182

17.19 MAINTAIN 183

17.20 MOBILIZE 184

17.21 RIOT 185

17.22 SIGEVENT 186

17.23 SPEND 187

17.24 STANCE 189

17.25 SUPPORT 190

17.26 VIOLENCE 191

17.27 WITHDRAW 192

18. Condition Types 194

18.1 COMPARE 195

18.2 CONTROL 196

18.3 EXPR 197

19. CURSE Inject Types 198

19.1 COOP Inject 199

19.2 HREL Inject 200

19.3 SAT Inject 201

19.4 VREL Inject 202

20. Payload Types 203

20.1 COOP Payload 204

20.2 HREL Payload 205

20.3 SAT Payload 206

20.4 VREL Payload 207

21. Gofers 208

21.1 ACTORS Gofer 208

21.2 CIVGROUPS Gofer 208

21.3 FRCGROUPS Gofer 208

21.4 GROUPS Gofer 208

21.5 NBHOODS Gofer 208

21.6 NUMBER Gofer 208

22. Simulation Orders 209

23. Glossary 210

24. Acronyms 229

25. Scenario Checklist 231

# Part I: Overview

Athena’s Objective and Approach

Athena's objective is to help our country's decision makers anticipate the likely medium-term consequences, both planned and unplanned, of potential courses of action that use our national power in troubled regions.

Athena’s approach anticipates high-order results by combining many models of first-order effects. Modeling is comprehensive; modeling areas include political, economics, attitudes, information, demographics, and physical phenomena. Where possible, models are based on scientifically established quantitative relationships and data. Where no established scientific models are available, models are based on common sense and expert opinion. Incorporation of suggestions and feedback from expert users is a key component.

## Introduction

This document presents the models and software of the Athena 6 Regional Stability Simulation from the user’s point of view. Users are advised to read this document before moving on to the other Athena documents.

The Athena simulation is a decision support tool designed to allow a skilled analyst to consider the intended and unintended consequences of various courses of action that might be taken during Stability & Recovery Operations. Athena contains models descended from the Joint Non-kinetic Effects Model (JNEM), and includes many new models and other changes. In addition, where JNEM is a federated simulation, Athena is a stand-alone single-user application.

The intent of Athena’s models is first to capture and make explicit a wide variety of first order causal links, each of which makes sense on the face of it, and secondly to present the second and third order consequences of events while preserving the causal chain.

Everyone is familiar with the story of the six blind men and the elephant. The goal of Athena is to model each of the elephant’s parts, and to link them together so that the man who has the elephant by the tail is sure to get thwacked by the elephant’s trunk (not to mention everything in between) and so must pay attention to the entire elephant.

### Overview of This Document

This document covers four major topics.

* Part I (Sections 1 through 10) describes the Athena models and philosophy at a conceptual level; those interested in more detail can see the low-level model descriptions in the *Athena Analyst’s Guide* and other documents.
* Part II (Sections 11 through 14) describes the Athena application itself: the parts of the application, how to enter scenario data, how to run the simulation, and how to find the results.
* Part III (Section 15) contains a cookbook of how to make use of Athena’s models and inputs for particular problems.
* Part IV (Sections 16 through 25) contains reference information, including a complete glossary of terms. Further details can be found in the Athena application’s on-line help.

### Other Documents

In addition to this user’s guide, Athena is delivered with the following documents:

*Athena Analyst’s Guide*

This document contains a detailed, low-level description of the models used in Athena; it serves as the specification document for the implementation of the models in the Athena code. Consult it when you need to know more about the models than is contained in this user’s guide.

*Mars Analyst’s Guide*

Mars is an infrastructure layer that is shared with the Joint Non-kinetic Effects Model (JNEM). The *Mars Analyst’s Guide* can be thought of as an appendix to the *Athena Analyst’s Guide* that describes the models that are implemented in the Mars code base, including the Unified Regional Attitude Model (URAM) and the Mars Affinity Model (MAM). The former tracks direct and indirect effects on civilian attitudes; the latter models belief systems and the resulting affinities between actors and groups.

*Athena Rules Document*

This document describes the events and situations (drivers) that affect civilian attitudes in more detail than does this user’s guide, and also details each of the Driver Assessment Model (DAM) rule sets that assess the attitude change caused by the drivers.

*Athena On-line Help*

The Athena application includes extensive on-line help; see the **Help** menu in the application’s main menu bar. The most detailed reference information for the Athena software is found here.

When Athena is installed on Microsoft Windows, these documents (except for the on-line help) are available from the Athena folder on the Start Menu. Alternatively, go to the Athena application directory and open “docs\index.html” in a web browser. The documents can also be obtained directly from the Athena Project; contact [William.H.Duquette@jpl.nasa.gov](mailto:William.H.Duquette@jpl.nasa.gov).

### Changes for Athena 6

Athena 6.2 contains many significant changes over Athena 5.

#### Changes in Athena 6.1

##### PMESII-PT

The Athena user interface is now designed around the PMESII-PT paradigm.[[1]](#footnote-1)

##### Mark II Strategy Engine

The biggest change in Athena 6 is the introduction of the Mark II Strategy Engine, which has major implications for Athena usage.

* A strategy is now a list of one or more prioritized *blocks*, rather than a list of *tactics*.
* Each block can contain:
  + Any number of *tactics*
  + Any number of *conditions*
* Most metadata associated with tactics in Athena 5 (e.g., the "on-lock" flag) is now associated with blocks.
* Time constraints are now an explicit part of the block; the Athena 5 AFTER, AT, BEFORE, and DURING conditions are no longer required or available.
* A block is eligible for execution only if its time constraints and all attached conditions are met.
* Given that a block is eligible, the tactics in the block will execute only if there are sufficient resources to execute the entire block.

This engine naturally implies a new Strategy Browser, which is much easier to use than the Mark I Strategy Browser; and in particular **it is possible to cut, copy, and paste blocks, tactics, and conditions from one strategy to another**.

In addition:

* The tactics have been improved and made more flexible.
* The set of conditions has been streamlined; the new **COMPARE** condition allows the direct comparison of many different model outputs. (Mark I conditions allowed the comparison of a particular output with a particular number).

##### Goods Production Infrastructure

In the past, Athena allocated production of economic goods and services (the **goods** sector) to neighborhoods in proportion to the size of each neighborhood's work force. Athena 6 models **goods** production infrastructure explicitly, allowing actors to own, build, and maintain production "plants", and disaggregates unemployment to neighborhoods according to the distribution of plants. See Section 9.2 for more.

##### Economics Model Changes

Athena 6 contains a number of Economics model changes related to the new model of **goods** production infrastructure. In particular:

* The capacity of the **goods** sector is computed from the amount of goods production infrastructure in neighborhoods.
* Geographic Unemployment (GU), which is the number of workers unwilling to travel to available jobs in neighborhoods they deem “too far away”, is computed by the demographics model and further constrains the capacity of the workforce in the **pop** sector.

##### Expression and Script Syntax Checking

The EXPR condition dialog and the Scripts Editor now provide for detailed syntax checking of Tcl expressions and scripts, displaying the error details to the user right where the code is being edited (i.e., in the Scripts Editor and in the EXPR condition's edit dialog).

Not all errors can be caught before the scenario is run, but many significant ones are now found immediately.

##### Scenario Export Scripts

Athena has always had the capability to export a scenario as a script of Athena orders, using exactly the same orders that were used to build it via the GUI. By running the script at the Athena command line interface, Athena rebuilds the scenario from scratch just as it was built to begin with.

Athena 6 retains this capability, but now includes the ability to export a minimal order script, a script that contains the minimum number of orders required to rebuild the scenario, arranged in a logical sequence. Minimal order scripts are frequently much shorter, and easier to follow.

#### Changes in Athena 6.2

##### Belief System Upgrade

The Athena Belief System model has been significantly upgraded. The user interface is much easier to use, and individual belief systems can now be shared by multiple groups or actors. In particular, similar groups in different neighborhoods can now easily share the same belief system.

##### Web Map Service Support

The Web Map Service (WMS) is an open interface for retrieving map images across the network. Athena now supports this web API, and can retrieve maps from any available WMS server.

##### Environmental Situations are now Abstract Situations

Athena 1.1 inherited JNEM's model of *abstract situations*, or *absits*; in Athena these were renamed *environmental situations*, or *ensits*. During Athena 6.2 development it was pointed out that *ensit* is a military term meaning "enemy situation", and so Athena's *ensits* are now *absits* once again.

##### New Abstract Situations

There are two new abstract situation types, DROUGHT and TRAFFIC, added to support the Intel Ingestion Wizard.

##### Abstract Events

Athena now has a notion of "abstract events". An abstract event is a one-time event that triggers a related attitude rule set. It is triggered by a related SYSTEM agent tactic. Athena has five abstract event types, ACCIDENT, DEMO, EXPLOSION, RIOT, and VIOLENCE, all defined to support the Intel Ingestion Wizard.

##### Intel Ingestion Wizard

Athena's new Intel Ingestion Wizard is a proof-of-principle and the first instance of Athena's new wizard framework. The idea of a wizard is a GUI for automating some portion of Athena operation, and particularly scenario preparation. The Intel Ingestion Wizard ingests intel messages in TIGR format at the user's direction. The user can then triage the TIGR messages, and associate them with particular abstract situation and abstract event types; customize the resulting situations and events; and ingest them into the scenario as tactics in the SYSTEM agent's strategy. In this way, the events described by the intel messages will become events and situations in the scenario, and will play out over time as the simulation runs.

##### Neighborhood Ingestion Wizard

Athena's new Neighborhood Ingestion Wizard is the second instance of Athena's new wizard framework. It takes as input one or more Google KML files containing polygons representing political boundaries (country, province, district, etc.) and one additional file that shows the semantic relations between polygons (i.e., this is a province of that country). Given these, the wizard displays the available polygons in a hierarchy, and allows the user to pick and choose those to include in a scenario. Then, the wizard will add neighborhoods to the scenario for each of the selected polygons.

##### Athena Attrition Model Removed

The Athena Attrition Model, which first appeared in Athena 1.1, has been removed (with the exception of magic attrition). Its model of uniformed forces hunting non-uniformed forces has been shown to not adequately reflect real world tactics. In the future, we hope to add an improved (and more general) attrition model.

## Athena Model Overview

Athena is a collection of many models that involve the relations and interactions between several kinds of simulation object. This section gives a brief overview of the most important kinds of simulation object, and of the major modeling areas. The various kinds of simulation object are documented in detail in Section 16; the modeling areas and the models they contain are described more fully in Sections 3 through 10 and are documented in detail in the *Athena Analyst’s Guide*, *Mars Analyst’s Guide*, and *Athena Rules* documents.

### The Simulation and its Objects

At its highest level, Athena models the actions taken by significant decision makers, called *actors*, within a region of interest, called the *playbox*. The playbox is divided into sub-regions, called *neighborhoods*. Each neighborhood is inhabited by some number of civilians, who are divided into *civilian groups*, and who form both the consumers and the labor force in the *local economy*.

The actors attempt to control, aid, or otherwise influence the civilians in the neighborhoods by means of their actions, which are called *tactics*. Executing tactics requires *assets*, of which there are several kinds: money, personnel, *communications asset packages* (CAPs), and **goods** production infrastructure (*plants*). Each actor has an income. Actors own personnel in the form of army troops, police forces, humanitarian relief organizations, and so forth. These are called *force groups* or *organization groups*, depending on the nature of the group. Actors may also own and use communications assets (i.e., television stations, newspapers, and web sites). By means of *conditions*, an actor can determine when and under what circumstances tactics are used. The prioritized set of an actor’s tactics with their attached conditions is called the actor’s *strategy*.

The civilian groups have *attitudes*: satisfaction or dissatisfaction with respect to particular *concerns*, and a willingness or unwillingness to *cooperate* (i.e., share information with) members of force groups. Further all groups have *horizontal relationships* with each other and *vertical relationships* with each actor. Relationships derive from compatible or incompatible belief systems, but are also affected by the actors' actions, and by conditions in the neighborhoods. These attitudes vary over time in response to the events and situations that occur in the simulation, including those triggered by the actors’ actions.

Groups can *support* actors to a greater or lesser degree. Support is based upon vertical relationships, but is also affected by actors’ actions and by conditions in the civilians’ neighborhoods. Actors may use the support they receive, or lend it to other actors. An actor with sufficient support in a neighborhood is said to have *influence* in the neighborhood; and an actor with sufficient influence may *control* the neighborhood.

The *economy* underlies all of this; it determines actors' incomes, as well as the level of civilian employment and consumption.

Thus, we have the following feedback loop:



The actors’ actions determine the situation on the ground (including the economy), which in turn affects civilian attitudes. This in turn affects civilian support for the actors, which can cause a change in which actor controls each neighborhood. That political situation then drives the actors’ actions. There are, of course, smaller feedback loops within this one; for example, an actor can increase his support in a neighborhood by moving a significant military force into that neighborhood. The force supports him, possibly enough so that he gains control, bypassing (at least temporarily) civilian attitudes altogether. If the force is properly trained and deployed, it can increase civilian security in the neighborhood, which might further boost the actor's support. Similarly, there is a feedback loop from the situation on the ground back to the actors’ actions—even politicians look at more than just the latest polls. But at a high level, this is the dynamic that drives Athena execution.

The primary outputs at the end of an Athena run are these:

* Which actor is in control of each neighborhood
* The stability and security of each neighborhood
* The civilian groups’ attitudes about the state of the playbox
* The resources and decisions required by each actor to bring about this end state.

Thus, Athena can be used to both analyze an existing political situation and assess the results of various courses of action designed to change it, from the point of view of any or all of the actors involved.

### The Modeling Areas

Athena consists of a great many models; for purposes of discussion they fall loosely into seven broad areas. The borders between these areas are often fuzzy, and some of the Athena models straddle them. The seven areas are as follows:

**The *Time* Area**

Athena is a time-step simulation. (Section 3)

**The *Physical* Area**

Literally, where things are on the ground, what they are doing, and what is happening to them as a result. (Section 4)

**The *Political* Area**

This area deals with actors and their strategies (goals, prioritized tactics, and attached conditions) along with the determination of support, influence, and neighborhood control. It includes the vertical relationships of the resident civilians with the actors. (Section 5)

**The *Military* Area**

Actors can own military, paramilitary, and criminal forces, and use them to achieve their ends in the playbox. (Section 6)

**The *Economics* Area**

This area deals with the local economy, relating changes in population, security and production to the gross domestic product (GDP) and the unemployment rate, and determining actors’ incomes. (Section 7)

**The *Social* Area**

This area concerns the civilian residents of the playbox, including their beliefs, attitudes, and demographics. Attitudes include their relationships with each other and their satisfaction with various aspects of their environment. Demographics include where the civilians live and how many of them there are, along with the computation of such population statistics as the labor force and the number of consumers in the local economy. (Section 8)

**The *Infrastructure* Area**

This area deals with the infrastructure in the playbox. At present, this includes production infrastructure for the economy’s **goods** sector (e.g., manufacturing plants) and communications asset packages. (Section 9)

**The *Information* Area**

This area deals with information flow in the playbox, and especially the effect of information flow and information campaigns on attitudes and politics. It also includes the flow of intelligence from civilians to the various force groups in the playbox, as indicated by the cooperation attitude. (Section 10)

Note that these areas align with the PMESII-PT paradigm. The models in each of these areas are inter-related: every area depends on inputs and outputs from the other areas, as shown in the following figure. It is necessary to track all of them to get a complete view of the simulation.



Each of these areas and the models contained within it are described in the referenced sections; for full details on the models, see the *Athena Analyst’s Guide*, *Mars Analyst’s Guide*, and *Athena Rules* documents.

### Basic Concepts

The following concepts are fundamental to understanding Athena. They are defined in more detail in the following sections.

The region of interest is broken up into sub-regions called *neighborhoods*, which can be of any size. Neighborhoods contain people and infrastructure. The collection of all neighborhoods in the scenario is called the *playbox*. (Section 4.1)

*Actors* are the significant decision makers in the playbox. They own assets, take action in neighborhoods, and contest for the control of neighborhoods. (Section 5.1)

The people in the playbox are divided into *groups*, of which there are three kinds: *civilian groups*, *force groups*, and *organization groups*, e.g., neighborhood residents, the U.S. Army, and the Red Cross. (Sections 6.1 and 8.1)

## The Time Area

Athena is a time-step simulation with a time-step of one week; time is measured in integer weeks, also known as ticks. Time is reported to the user in both ticks and calendar weeks; the user may associate tick 0 in simulation time with any given calendar week, i.e., the 20th week of 2014.

By convention, the simulation begins at tick 0. The user can set the starting tick to any desired integer value. When rebasing a scenario, for example, the starting tick will usually be the same as the simulation time at which the scenario was rebased.[[2]](#footnote-2)

Colloquially, we will refer to the simulation time at simulation start as "week 0" or "time 0".

## The Physical Area

The Physical modeling area is literally concerned with what is happening on the ground: where people are, and what they are doing, and the results of their actions. As such, it has a tendency to overlap with all of the other areas. It includes:

* The breakdown of the playbox into neighborhoods and the relationships among the neighborhoods.
* Where civilian, force, and organization personnel are located
* The activities they are performing, including combat activities and law enforcement
* Neighborhood security levels
* Activity coverage
* Abstract situations
* Essential Non-Infrastructure (ENI) services
* Consumption of goods by the civilian population

Events and situations occurring in the Physical area affect attitudes, and the demographics and production capacity of the playbox and hence the economy as well. Actors base their decisions on the state of affairs in the playbox.

### The Playbox

The playbox is the geographic area in which the simulation takes place. It is modeled as a collection of polygonal regions called *neighborhoods*, which are laid out on a map as an aid to visualization. Athena takes little note of neighborhood geography or geographic distances; the analyst explicitly associates the other simulation entities with neighborhoods, rather than with particular points on the map. A neighborhood's content is assumed to be homogeneous across the neighborhood.

#### Neighborhoods

Almost everything that happens in Athena takes place in the context of a neighborhood. Civilian groups reside in neighborhoods; force and organization group personnel are deployed to neighborhoods; infrastructure exists in neighborhoods; personnel and actors act in neighborhoods; and many output statistics are computed by neighborhood.

Neighborhoods can be of any size, from villages to portions of a city (neighborhoods in the proper sense of the word) to entire cities, counties, districts, provinces, countries, or groups of countries. Neighborhoods can nest, i.e., a neighborhood representing a city can be placed on top of a neighborhood representing a province.[[3]](#footnote-3) In practice, however, regions are usually province or district-sized, and rarely overlap.

Each neighborhood can be more or less urbanized, and can contain more or less of the playbox’s economic production capacity.

#### Neighborhood Proximity

Although Athena doesn’t concern itself with geographic distances, it is clear that socially some neighborhoods are more closely related than others. Because this is a social distance, not a physical distance, we model it directly rather than deriving it from the geography.

We call this social distance *neighborhood proximity*; it is defined as the distance between two neighborhoods from the point of view of the residents of the first neighborhood. More specifically, we say that with respect to neighborhood A, neighborhood B is near, far, or remote. The degree to which A is affected by events in B tapers off with distance, and is zero if B is remote.

Proximity need not be symmetric. If neighborhood A contains popular destinations, residents of B might consider it to be nearby, even though residents of A consider B to be far away.

#### Local vs. Non-Local Neighborhoods

Athena assumes that the neighborhoods that make up the playbox are more or less contiguous and have a single more or less unified economy. Sometimes, however, it can be convenient to include neighborhoods in the scenario that are outside the economy. Pakistan is greatly affected by the decisions made by actors in India, for example; in a scenario involving the inner workings of Pakistan, it might be desirable to include India as a neighborhood while excluding India from the modeled economy (except as part of the "rest of the world"). Thus, we can mark neighborhoods as *local* (participants in the local economy) or *non-local* (excluded from the local economy).

### Groups

Athena represents masses of human beings as *groups*. There are three kinds: *civilian groups*, *force groups*, and *organization groups*, described in Sections 8.1.1, 6.1, and 8.1.2 respectively. All people in the playbox belong to one or another group.

### Deployment

Every person in the playbox has to be somewhere, i.e., has to be located in some neighborhood. Civilians are simply located in their home neighborhoods; force and organization group personnel need to be *deployed* to particular neighborhoods.

Deployment is determined by the tactics chosen by the actors, as is the number of group personnel available for deployment; see Section 5.2. The number of group personnel initially in the playbox is a scenario input.

Troops are deployed by their owning actors during strategy execution, and remain in place throughout the week until the next strategy execution.

### Activity Assignment

Force and organization group personnel deployed to a neighborhood can be assigned activities in that neighborhood: patrolling, guarding, law enforcement, various kinds of humanitarian relief, and so forth. See Section 17.1.1 for a complete list of activities. These activities affect the attitudes of the civilian population. Activities are assigned by the actors during strategy execution, and take place during the following week.

Force and organization group activities have security requirements; a body of troops might be tasked to do humanitarian relief of some kind, but their efforts will be of no avail if they have insufficient security in the neighborhood (Section 4.6). See the *Athena Analyst’s Guide* and the model parameter database (Section 14.19) for the security requirements.

### Units

All deployed personnel, and all civilian personnel, are placed in *units*. The name derives from the classic military term; in Athena it simply means a collection of personnel belonging to the same group and assigned the same activity. Units have no distinctive or long-running identity; they are created during strategy execution and represent the location and activity of group personnel over the following week. If personnel from force group A are deployed to neighborhood B, and assigned various activities, then A will have at least one unit for each activity, plus an additional unit for those personnel that remain unassigned.

Units are useful for visualization; and many of the models in the Physical Area operate on units.

### Security

A neighborhood can be a safe or unsafe place to be for the people within it—and to a great extent that depends on whom they are and who is in the neighborhood with them. Athena computes a measure called the *security* of each group in each neighborhood.

Both depend on the personnel in the neighborhood and nearby neighborhoods, and on their relationships with each other (see Section 8.5 for more on relationships).

First, each group in the neighborhood (whether civilian, force, or organization) can project a certain amount of force, given the kind of group it is and the number of personnel present. This is its power to defend itself. Civilian groups project minimal force per person, given that civilian groups include the very old and the very young, and many adults who are not inclined to project force. Force groups exist to project force, and do it much better; their effectiveness depends on the kind of force they are. Regular military projects the most force per person. Of organization groups, only contractors project any force (private security guards)—more than civilians but less than any force group. (See Section 4.2 for more on the different kinds of group.)

But a group’s ability to defend itself does not depend solely on its own force—they may have friends to help them. Athena totals up the force available to each group, including friends in the same neighborhood and (to a lesser degree) friends in nearby neighborhoods; and similarly it totals up the force available to the group’s enemies in the same neighborhood and (to a lesser degree) in nearby neighborhoods.

Every civilian group has its criminals, who are assumed to be the enemies of everyone. This *background criminal activity* can be suppressed by law enforcement, thus decreasing volatility and improving everyone’s security.

The *volatility* of a neighborhood ranges from 0 to 100, and is a measure of how dangerous the neighborhood is to a random passerby given the degree of enmity present in the neighborhood, i.e., how likely a person is to get caught up in random violence that does not directly concern him. Volatility includes the effects of background criminal activity in the civilian population; this can be suppressed by law enforcement.

Danger to a group comes from its enemies and from the kind of random violence measured by volatility. We capture this as the *security* of the group in the neighborhood. Security is an abstract measure ranging from -100 to 100. For actual use we convert it to a qualitative measure (high, medium, low, or none), or to a multiplicative factor using a Z-curve function.[[4]](#footnote-4)

In Athena, *volatility* is primarily a component of *security*, whereas *security* affects many things, including:

* Whether force and organization groups can carry out particular activities in a neighborhood.
* The degree to which groups can actively support the actors of their choice.

The addition of a military force to a neighborhood can greatly change the security of all groups in the neighborhood. In previous versions of Athena, the effect of force group personnel on neighborhood security depended on the relationships of the force group to the other groups present in the neighborhood. Thus, deploying a large force to a neighborhood to do peacekeeping would decrease civilian security if the group’s relationship to the residents was negative, even if the intent was peacekeeping.

However, the owning actor can also specify the *stance* of a force group toward other groups; the group can have a positive stance toward groups with which it has a negative relationship. When security is computed, the force group’s effective relationship is the designated stance, as modified by the group’s training level (some force groups are better at overcoming their relationships than others). In addition, some activities (e.g., patrolling and law enforcement) have a greater effect on security than others.

### Coverage

*Coverage* is a measure, from 0.0 to 1.0, of the fraction of a neighborhood or group affected by some situation. The notion of coverage is used in a number of places in Athena:

* Abstract situations are assigned a coverage fraction when they are created. (Section 4.9)
* Coverage is computed for the presence of a military force deployed in a neighborhood.
* Coverage is computed for activities assigned to groups of all kinds. (Section 4.4)

For abstract situations, the coverage is simply an input. For presence and activities, it is a function defined by the number of troops required to achieve two-thirds coverage given the size of the population. For presence, for example, presence coverage is two-thirds when there are 25 troops present for each 1000 people in the civilian population. Coverage drops to zero when there are no troops present or engaged in the activity, and increases asymptotically to 1.0 as troops are added above the two-thirds mark.

Assigned activities usually have a security requirement. If there are 25 troops per 1000 people assigned to do the **MEDICAL** activity, but the security of those troops is low, they cannot carry out the activity effectively and hence the coverage of that activity by those troops is 0.0.

The coverage fraction is used as a multiplier in the relevant rule set in the Driver Assessment Model (Section 8.7); and presence coverage is used in a variety of places, most notably in the Athena Attrition Model (Section 6.3).

### Activity Situations

When a group is conducting an activity of a particular type in a neighborhood with coverage greater than 0.0, we have what we call an *activity situation*,[[5]](#footnote-5) or “actsit”. Activity situations are created when activity coverage exceeds 0.0, and are destroyed when activity coverage returns to 0.0. So long as the situation persists it will have effects on civilian attitudes as determined by the relevant rule set in the Driver Assessment Model; see Section 8.7and the *Athena Rules* document.

Force and organization group activities can mitigate particular abstract situations. For example, the **MEDICAL** activity will have a greater effect on civilian attitudes when there is a **DISEASE** situation in the neighborhood.

### Abstract Situations

*Abstract situations*, or “absits,” represent problems in a neighborhood’s environment that adversely affect the resident civilians, e.g., power outages and food shortages; see the *Athena Rules Document* for the complete list of situation types.

Abstract situations are usually created by the analyst, or by actors using the **ABSIT** tactic. An absit will typically have a big negative effect on satisfaction on inception, a negative effect so long as the situation persists, and possibly a positive effect on Autonomy when the situation is resolved, provided that is resolved by the locals themselves.

The duration of an absit can be set when the absit is created; it can also be resolved explicitly by the analyst or by an actor using the **EXECUTIVE** tactic. Each absit also has a coverage fraction, nominally 1.0, which can be reduced to decrease the magnitude of the absit’s effects.

The effect of certain abstract situations can be mitigated by appropriate force and organization group activities, as indicated in the *Athena Rules* document.

In the future, it is likely that many of the existing absit types (e.g., power outages) will be replaced by service-oriented models like the current Essential Non-Infrastructure (ENI) Services model (Section 4.10), which is more suited to the Athena time frame.

#### Abstract Situations and URAM

Absit attitude effects are structured as follows:

* There is a transient negative effect each week the situation exists.
* The transient negative effect is usually larger the first week; this is the inception penalty.
* On resolution, the transient effects cease; and there is a transient plus to Autonomy that week if the situation was resolved by the locals.

### Essential Non-Infrastructure (ENI) Services

Essential Non-Infrastructure (ENI) Services are services provided to civilians in a neighborhood by an actor. They are defined as services which do not require substantial infrastructure[[6]](#footnote-6) to provide but which cause hardship when they are absent. Provision of services is controlled by the actor’s strategy, and can be targeted to specific groups in the neighborhood, ignoring others.

#### The Notion of a Service

A service is something provided to the civilians (possibly by their own efforts, as enabled or supported by actors) that has a level that can increase or decrease over time. Examples are electrical power, postal service, communications, water supply, the court system and other governmental services, and the like (though not law enforcement, as that’s an assigned activity). We call the level of service for a particular service the *LOS*. For any given service there are four specific levels of service that are of interest:

**The Actual Level of Service (ALOS)**

How much of the service is the group actually receiving at the present time?

**The Required Level of Service (RLOS)**

How much of the service does the group need to live without significant hardship?

**The Expected Level of Service (ELOS)**

How much of the service is the group accustomed to getting?

**The Saturation Level of Service (SLOS)**

What’s the level of the service which saturates the demand? Once the civilians have all they want, they don’t care if more is available.

The units appropriate for measuring a particular the level of a particular service will vary from service to service.

The expected level of service slowly approaches the actual level of service over time; in other words, the civilians eventually become accustomed to whatever level of service they receive. Expectations rise more quickly than they fall: we become accustomed to good things more quickly than we become resigned to bad things.

For example, in most of America ELOS for the power supply simply *is* the SLOS. Most of us have all the power we are willing to buy. If the power is out, we are immediately unhappy, and it would take us quite a while to get used to power provided on a regular but intermittent schedule; but when the power goes back on, we get used to it again with great rapidity.

Civilian attitudes improve when the ALOS is greater than expected (though not more than the SLOS), and worsen when the ALOS is less than expected, and especially if it is less than required.

There are four cases of particular interest:

* Case R–: Service is less than required
* Case E–: Service is less than expected
* Case E: Service meets expectations
* Case E+: Service is better than expected

Case R– trumps all of the others; if the ALOS is less than required then expectations don't matter as much as the reality of hardship. Also, note that case E+ can only occur if the expected level of service is less than saturation.

At the present time we have used this paradigm only for ENI services; we may make use of it for infrastructure-based services in the future.

#### Measurement of ENI Services

Actors provide ENI services to groups in neighborhoods by spending money on them using the **FUNDENI** tactic. No infrastructure is required, by definition; and we assume that every dollar spent translates (not necessarily linearly) into service provided.

For convenience, we measure the provision of ENI services to a group in a neighborhood as a fraction of the saturation level of service (SLOS) for that group: 0.0 implies no service, and 1.0 implies the saturation level of service. Then, we specify the saturation level of service by the per capita funding required to achieve it. If actors provide funding for more than the saturation level of service, the ALOS will be greater than 1.0.

#### Required Level of ENI Services

The required level of ENI services is set in the model parameter database as a fraction of the saturation level of service, according to the urbanization level of the neighborhood.

#### Effects of ENI Services

The current level of ENI services affects two things:

* Civilian satisfaction levels; see Section 8.6.
* The vertical relationships of civilian groups with actors; see Section 5.3.

In each case, the fundamental questions are whether the civilians are receiving the required level of service; and if so, whether they are receiving more or less service than they expect.

In terms of the vertical relationships, it also matters whether or not the actor providing the service has control of the group’s neighborhood; see Section 5.4.3 (and the *Athena Analyst’s Guide*) for details about neighborhood control, and the *Athena Rules Document* for the attitude effects.

#### Services vs. Abstract Situations

At present, the service paradigm is used only for essential non-infrastructure services. In the future, we hope to apply it to infrastructure-bases services such as the electrical power grid and the water supply, instead of the existing abstract situation paradigm (Section 4.9).

Consider the power grid. The absit paradigm implicitly assumes that the ALOS is normally at its expected value and that it drops down to 0.0 when problems occur. Horrors ensue until the problem is resolved, at which point the service returns to its previously expected level.

In a long-run scenario, however, it is quite possible that the power service may be substandard (though not zero) for quite long periods of time. Power for 12 hours a day is much better than no power at all; and after a few weeks’ time, the civilians will begin to adjust to it (and the attitude effects will cease). If power then dropped to 4 hours a day, they would then again react negatively; but if it returned to 24 hours a day they would react positively.

Consequently, we hope to replace many of the existing absit types with service-oriented models as development continues.

## The Political Area

The Political area deals with actors and their strategies (goals, tactics, and attached conditions) along with the determination of support, influence, and neighborhood control. As described above in Section 2, the interplay of actor’s strategies being executed over time is the engine that makes Athena run.

### Actors

*Actors* are significant decision makers in the playbox. Depending on the needs of the scenario, an actor may be an individual, a committee, a small group, a ruling body, or an entire country. The essential point is that an actor has:

* Goals it wishes to achieve
* A strategy for achieving them
* Assets to use to achieve them.

Actors may have *income*, which derives from the economy, or a *budget*, which derives from foreign sources. Cash spent by budget actors enters the local economy. Money can be spent in a variety of ways. Actors can own force and organization groups whose personnel they can use, and may also have access to communications media and other infrastructure assets.

Actors can have different domains—military, political, economic, cultural—and although these domains are implicit in the actor’s assets and strategies rather than explicit in the model, they are no less important for all that.

Most actors participate in the political process, seeking control of neighborhoods or supporting those who do. Organization groups, which usually do not play politics, are often owned by a “pseudo-actor”, an actor that supports no one politically and exists only to hold the strategy that determines organization group behavior.

There is one additional pseudo-actor, or *agent*, the **SYSTEM** agent. The **SYSTEM** agent is not an actor, and cannot be involved in politics; however, it does have a strategy, which allows the analyst to set up various events and situations to occur when conditions in the simulation are right. In particular, the **SYSTEM** agent is responsible for the movement of civilians around the playbox, natural disasters, and a variety of other things. The **SYSTEM** agent also owns all **goods** production infrastructure[[7]](#footnote-7) by default.

### Strategies: Blocks, Tactics, and Conditions

As described in Section 5.1, the actors are the significant decision makers in the playbox. Actors have goals that they wish to achieve, and *assets* to use to achieve them; they execute *tactics* using their assets to achieve these goals; and they decide which tactics to use, and whether or not their goals have been achieved, through the use of *conditions*. Tactics and conditions are grouped together in *blocks*.

This section will define all of these terms, and how they relate to each other. There are many different kinds of tactic, and many different kinds of condition; these are documented in Sections 17and 18 in the reference portion of this user’s guide, and in the on-line help.

#### Assets

Actors can have several kinds of assets: personnel, broadcast communications assets, goods production infrastructure (plants) and cash.

An actor’s personnel are simply the members of the force and organization groups owned by the actor; these groups are assigned to actors as part of scenario preparation. By executing the appropriate tactics, described in Section 17, the actor can mobilize, demobilize, deploy, and assign activities to personnel, as described in Sections 4.3 and 4.4.

Next, each actor may own zero or more *communications asset packages* (CAPs) (Section 9.1), e.g., television stations, which may be used to broadcast *information operations messages* (IOMs) to affect civilian attitudes (Section 10.2). In addition, an actor may make any of his CAPs available to some or all of the other actors.

Finally, an actor can own, build, and maintain **goods** production infrastructure, thereby affecting both local unemployment and the state of the economy at large.

Each actor begins the scenario with two pots of money, his *cash-on-hand*, money that is available to be spent, and his *cash-reserve*, money that is being saved for use at a later time. Tactics that cost money are paid for from the cash-on-hand; leftover cash rolls over to the next week. The actor can use tactics to move money between his cash-on-hand and his cash-reserve.

In addition, each actor has a weekly income. If the Economics model is enabled, this income comes from particular sectors of the economy, and grows or shrinks as the relevant sectors grow or shrink. (If the Economics model is disabled, income is fixed at its initial setting.) The income flows into the actor’s cash-on-hand just before strategy execution each week.

Finally, the actor can receive funds from other actors. When actor A funds actor B, he does so by executing the **FUND** tactic, which transfers some quantity of A’s *cash-on-hand* to B. The money becomes available to B the following week.

Naturally, an actor must spend money to support and make use of his assets.

#### Conditions

Actors are decision makers, and so it is necessary to allow the analyst to express in the simulation the *conditions* that lead the actors to make one decision or another. A condition is a Boolean predicate, true if some particular state of affairs exists in the simulation, and false otherwise. Athena defines a variety of types of condition; for example,

* Does actor A control neighborhood N?
* Is actor A’s *cash-reserve* greater than $1,000,000?
* Is group G’s mood less than -40.0?

The complete list is documented in Section 18 and in the on-line help. In addition, the analyst can define complex conditions using Boolean expressions via the **EXPR** condition.

#### Tactics

A *tactic* is an action that the actor can choose to take in the playbox. He may deploy troops to neighborhoods, assign troops to do particular activities, fund essential services, support other actors, and so forth. There are many different kinds of tactic; each kind is defined in detail in Section 17.

A tactic may have a cost in dollars, personnel, or other assets, depending on the tactic type and its parameters. Dollars spent on a tactic are consumed. Personnel used by a tactic are unavailable for use by other tactics during the same week. Any number of IOMs may be sent via a CAP during a particular week, but the actor must either own the CAP or have been granted access to it; and sending IOMs will usually also cost money. If the required assets are not available, the tactic cannot be executed.

Note that it is quite possible for an actor to take actions that are counter-productive given his stated goals. The Political model is not intended to compute optimal courses-of-action for the actors; rather, it is intended to allow the analyst to model the decision makers in the playbox, along with their limitations and prejudices, and track the consequences of the actions they actually take.

#### Blocks

A strategy is a prioritized list of *blocks*; each block represents some set of actions that the actor wishes to take under certain circumstances. The block consists of:

* An *intent*: the analyst’s brief description of what the block is for.
* A set of time constraints; the block can only be executed when all time constraints are met.
* Zero or more conditions; the block can only be executed if all conditions are met.
* One or more tactics; the block will execute only if there are sufficient resources for all tactics in the block.

#### Strategy Execution

At the beginning of each week, every actor executes his strategy for the following week given the current state of affairs. This process is called strategy execution, and it is fairly straightforward:

* The actor considers each of his blocks in order from highest priority to lowest priority.[[8]](#footnote-8)
  + If any of the block’s time constraints are not met, the block is skipped.
  + If any of the blocks conditions are not met, the block is skipped.
  + If the actor has insufficient assets remaining, the block is skipped.
  + Otherwise, all tactics in the block are executed.

Note that “executing the tactic” does not mean that all of the tactic’s work is done immediately. Sometimes executing the tactic simply registers the actor’s intent, which then plays out in the simulation over the following week. When the actor executes a tactic, it is as though the actor has just given the relevant orders to his subordinates.

#### Strategy Execution On Lock

The actor’s strategies are also used to establish the state of the simulation at week 0, on scenario lock (e.g., the initial troop deployments). (Only certain tactic types take effect on lock.) To make a block execute on lock, set its “On Lock?” flag. Note that conditions are ignored when executing strategies on lock.

#### Other Block Options

In addition to its time constraints and “On Lock?” flag, blocks have the following optional settings:

* The “Once?” flag. If this flag is set, the block will disable itself at the end of its first successful execution. This might be used to implement a surge, for example, when a new body of troops enters the playbox via the **MOBILIZE** tactic.
* By default, all attached conditions must be met for the block to execute. If desired, the block can execute so long as at least one of the attached conditions is met.
* By default, there must be sufficient resources for all tactics in the block or the block will not execute. If desired, Athena will execute as many of the block’s tactics as possible, in priority order, given available resources.

#### What the Actor Knows, and When He Knows It

Athena assumes that all of the actors execute their strategies simultaneously, without any collusion or comparing of notes. Thus, as the actor works through the process of executing his strategy, he knows two things:

* The state of affairs before he started.
* The decisions he has already made during the current execution process.

For example, he knows how much cash he has left given previous expenditures during the current strategy execution process.

The relevant conditions are defined according to this scheme, following this basic rule: when actor A is querying his own assets and decisions, he sees their current values as of that point in strategy execution, but when he is querying the assets of and decisions of others he sees the information as of the beginning of strategy execution. See the description of each condition for more details.

#### Expenditures

Just as an actor's income flows from the Economics model into the Political model, so too do the actor's expenditures flow back into the Economics model. For each tactic that involves an expenditure of money, there is a model parameter econ.shares.*tactic*.*sector* that determines what share of the actor's expenditures on a given *tactic* flow to each *sector*.

### Vertical Relationships

Every group has a vertical relationship with every actor that represents how it feels about that actor. Vertical relationships are unidirectional—that is, we measure each group’s relationship to each actor, but not the actor’s relationship to each group. Actors are what they do, and what they do is determined by their strategies, not by their affinities.

Vertical relationships are measured from -1.0 to 1.0, a range which is often expressed qualitatively:

* **Supports**
* **Likes**
* **Is Indifferent To**
* **Dislikes**
* **Opposes**

Vertical relationships play a major role in the Political Area, for they are the basis for computing each actor’s support and influence and for the determination of neighborhood control (Section 5.4).[[9]](#footnote-9)

Vertical relationships are tracked by URAM, and the vertical relationship effects are handled by the Driver Assessment Model (DAM) rule sets like the other attitude effects. See Section 8.7.

The natural level of group *g'*s relationship with actor *a* is *g'*s affinity for actor *a* based on their belief systems (see Section 8.3). Over time, and in the absence of other drivers, the relationship will regress back to this natural level.

#### Force and Organization Groups

Every actor has an affinity for every other actor, and every force group (Section 6.1) and organization group (Section 8.1.2) is owned by an actor. A force or organization group’s vertical relationship with its owner is nominally 1.0, and its vertical relationship with any other actor is nominally its owner’s affinity for that actor.

The vertical relationships of force and organization groups are constant by default, and will vary only due to magic attitude inputs (see Section 8.8). Such magic inputs can affect the support of the groups for the relevant actors, which may then affect the actors’ influence on and control of the relevant neighborhoods; see Section 5.4.

Note that the vertical relationship between a force or organization group and its owning actor has no effect on the group’s obedience to the actor’s orders. Athena does not model mutiny.

#### Civilian Groups

The vertical relationship of a civilian group (Section 5.3) with an actor is rather more complicated, as it can vary dynamically as the simulation runs. It is based on the affinity of the group for the actor, but this is adjusted by several factors:

* Whether the actor is or is not in control of the group’s neighborhood
* How the group’s mood (see Section 8.6.3) has changed
* Whether and to what extent the actor is providing Essential Non-Infrastructure (ENI) services to the group

The baseline for assessing these factors is the state of the neighborhood at the start of the simulation; or, later on, the state when control of the neighborhood last shifted—an actor newly in control is judged on the state of affairs on his watch, rather than his predecessor’s.

The number of rule sets that affect vertical relationships is increasing over time; see the *Athena Rules* document for the complete set.

### Support, Influence, and Control

We say that an actor “controls” a neighborhood when the residents of the neighborhood hold him responsible for dealing with the neighborhood’s problems. The actor in control is blamed when things go poorly, and given credit when things go well; these things affect the vertical relationships between the residents and the actor (Section 5.3).

As actors make decisions and the civilians respond, the political model determines who is in control, and when control has shifted from one actor to another.

The basic paradigm is as follows: actors receive *support* from the people in the neighborhood (including force group personnel). An actor with sufficient support is said to have *influence* in the neighborhood. Depending on the relative influence values, the actor with the most influence is usually the one said to be “in control”.

#### Support

Group G (of whatever type) is said to support actor A directly in neighborhood N if both of the following conditions hold:

* G’s vertical relationship with A is strong enough (nominally, greater than 0.2)
* G’s security in N is high enough.[[10]](#footnote-10)

In essence, group G has to *want* to support A, and has to have sufficient freedom of movement to be *able* to support A. Support is 0 if G does not wish to support A, and increases from there with increases in population, vertical relationship, and security.

Actor A’s direct support in N is the sum of his support from all groups in N; the result is a number from 0.0 to 1.0, and can be thought of as the fraction of the neighborhood that supports A. Note that support is not a zero-sum game: group G can support any number of actors to differing degrees.

Actor A can use this support in three ways:

* Actor A can try to gain influence in N
* Actor A can support some other actor in N
* Actor A can choose not to use his support at all, thereby bowing out of the political process all together. (Actors that own organization groups often do this.)

Support received from other actors is called derived support; and actor A’s total support is simply the sum of the two. Note that because of derived support, an actor can find himself with enough influence to be in control of the neighborhood even if he chooses not to use his own direct support.

The analyst specifies, as part of the scenario, each actor’s choice of whom to support by default: himself, some other particular actor, or no one. Then, the actor may change who he supports in each neighborhood from week to week using the **SUPPORT** tactic.

#### Influence

Actors have influence in a neighborhood in proportion to their support relative to other actors. If only one actor has support in the neighborhood, only that actor can have any influence. Note that influence requires at least a minimal amount of support, nominally 0.1; this prevents a negligible force (e.g., a Red Cross team) from moving into a neighborhood that’s in a state of chaos and finding themselves in control despite having almost no support.

Influence is computed by taking the set of actors who meet the minimum support requirement and normalizing their total support figures. Thus, an actor’s influence is a number between 0.0 and 1.0; and influence *is* a zero-sum game. An actor cannot increase his own influence without decreasing the influence of the other actors.

#### Control

As part of scenario preparation, the analyst specifies which actor is in control of each neighborhood; this includes the possibility that *no* actor is in control of the neighborhood at the start of the scenario. Then, at the end of each week Athena looks to see which actor (if any) will be in control for the following week.

Suppose that actor A was in control of neighborhood N at the end of the previous week:

* If actor A now has more influence than any other actor, then A remains in control of N.
  + If A has less than the threshold amount of influence (nominally 0.5) in neighborhood N, then actor A still remains in control of N though in a rather precarious position.
* If some actor B now has more influence than any other actor, and more than the threshold amount of influence, then B is now in control; **control has shifted**.
* If at least one actor now has more influence than A but no actor has more than the threshold amount, then no actor has control of the neighborhood; **control has shifted**.

If no actor was in control of N, the rules are simpler:

* If some actor B now has more than the threshold amount of influence, then B is now in control; **control has shifted**.

To put it in other words, the actor in control loses it if he is seen to be dominated by some other actor; but that other actor might not be powerful enough to take control himself. Control of the neighborhood is then up for grabs.

#### When Control Shifts

When control does shift to a different actor, or to no actor, the civilians respond to the change:

* There is a new political situation, resulting in a persistent shift in vertical relationships (Section 5.3).
* The change affects the attitudes of the civilian groups that are resident in the neighborhood.
* See the **CONTROL** rule set in the *Athena Rules* document for details.

## The Military Area

### Force Groups

Military forces and activities are represented in the scenario by *force groups*.

A *force group* is an organized group of personnel intended to project and use force in a neighborhood. Force groups belong to actors, and do their bidding according to the actor’s strategies. For example, an actor can increase his support in a neighborhood by moving his troops into that neighborhood. There are five kinds:

* Regular military
* Irregular military (e.g., militias)
* Paramilitary (e.g., SWAT teams, militarized police forces)
* Police
* Criminal

Each force type has a different force multiplier; regular military troops, for example, are much better at projecting and using force than normal police.

Force groups can carry out military activities of various kinds, and can also participate in humanitarian relief efforts and law enforcement (Sections 4.4).

Each force group starts the scenario with a certain number of troops, which can increase and decrease as the simulation runs. At present, force groups cannot recruit from the civilian population, though this is an area we hope to pursue in the future.[[11]](#footnote-11)

Force groups do not have belief systems of their own; rather, they behave as if they inherit them from their owning actors.

Every force group has a training level; the more highly trained, the better the force group will perform at particular activities (e.g., law enforcement).

### Force Activities

An actor can assign force group personnel deployed in a neighborhood to perform a variety of *force activities* in that neighborhood: patrolling, guarding, law enforcement, various kinds of humanitarian relief, and so forth. (See Section 17.1.1 for a complete list of activities). These activities affect the attitudes of the civilian population. Activities are assigned by the actors during strategy execution, and take place during the following week.

Force group activities have security requirements; a body of troops might be tasked to do humanitarian relief of some kind, but their efforts will be of no avail if they have insufficient security in the neighborhood (Section 4.6). See the *Athena Analyst’s Guide* and the model parameter database (Section 14.19) for the security requirements.

### Magic Attrition

Athena does not contain an explicit model of combat attrition, or of terror bombings, assassinations of political figures, or deaths due to collateral damage. And yet, these kinds of deaths occur. For this reason Athena provides the ability to do “magic attrition,” which can be initiated by the analyst, or by an actor or the **SYSTEM** agent using the **EXECUTIVE** tactic. Magic attrition can affect members of any group; and in particular, the attitude effects of civilian casualties are assessed by the Driver Assessment Model.

Note that magic attrition should not be used for civilian deaths due to natural disasters, epidemics, or other causes that do not involve combat. For those kinds of things, the attitude effects are quite different and should be handled by either abstract situations (Section 4.9) or CURSEs (Section 8.8).

## The Economics Area

The Athena Economics model models the economy of the region of interest, which can be an entire country, a portion of a country, or several small countries taken together. We refer to this as the *local economy*. The core of the model is a six-sector Computable General Equilibrium (CGE) model, a system of non-linear equations solved using the Gauss-Seidel algorithm. The CGE is described in full detail in the *Athena Analyst’s Guide*; this section will give an overview, and discuss how the Economic model relates to the rest of Athena.

### Dollars

Athena represents money in “dollars”; however, no attempt is made to tie the value of an Athena “dollar” to that of a real dollar. If the inputs to Athena's economics model are tied to some real currency, then the outputs will be as well; and if not, not. Further, Athena does not currently model inflation, either real or nominal.

### Sectors

The CGE partitions the local economy into six sectors: **goods**, **black**, **pop**, **actors**, **region**, and **world**.

**The “goods” sector**

The **goods** sector includes all production of goods and services in the local economy. The unit of production is the *goods basket*, abbreviated “goodsBKT”, a notional basket of goods and services nominally costing $1.

**The “black” sector**

The **black** sector includes the international black market in drugs and guns, as well as human trafficking, in so far as this trade takes place in the playbox. The unit of production is the metric ton, or tonne, of whatever product the black market is engaged in trading. The maximum amount of product produced is constrained by the amount of feedstock that is available. For example, the production of opium depends on the amount of poppies. The price per unit and the cost of production feedstock are both exogenous to the model, unaffected by activities in the playbox; however, the amount of production can be changed via the EXECUTIVE tactic, resulting in changes in actors’ incomes.

**The “pop” sector**

The **pop** sector includes all labor by the workers in the local region, and all consumption by the population of the local region. The unit of production is the work-year of an *average worker*. Just as the goods basket represents a notional bundle of goods and services, the average worker represents a notional bundle of skills and kinds of work.

**The “actors” sector**

The **actors** sector includes all income to actors from the other sectors of the economy, and all expenditures by the actors. Income to the actors from other sectors is usually expressed in terms of a tax-like payment from the other sectors.

Actors can receive income from the **black** sector either as a tax-like payment (e.g., tolls on movement of material) or by direct participation in the profits. If no actor gets any share of the profits, all **black** profits are assumed to go to the **world** sector.

Actors can also siphon money from foreign aid intended for the **region** sector in the form of graft.

Actors' expenditures are determined by each actor’s overhead and tactics. Overhead in Athena is modeled as implicit payments by actors to adequately sustain a certain status quo without having to explicitly allocate money for it, and is expressed as a percentage of income. Any money left over after overhead is paid is available to be used for tactics that require money. An actor is not required to allocate any income to overhead.

**The “region” sector**

The **region** sector represents everything else in the local economy that isn’t covered by the first two sectors. The **region** can be a part of a country, a number of countries, or something in between depending on the needs of the analyst. The sector supplies production capacity to the **black** and **goods** sectors and collects tax-like revenues in return. This sector also receives that part of foreign aid that is not extracted by actors in the form of graft.

**The “world”** sector

The **world** includes the entire rest of the world outside the playbox. It includes normal imports and exports, as well as all foreign aid and remittances flowing into the playbox, and all other monetary flows into and out of the playbox. The world sector has no product but it does supply imported feed stocks for use in the **black** market.

Each sector produces a certain amount of output; measured in dollars, this output is called the *revenue* of the sector. In the process of producing its output, the sector consumes the output of some or all of the other six sectors; measured in dollars, this vector is called the sector’s *expenses*.

### Shape vs. Size

We distinguish between the *size* of the economy, which can be roughly thought of as total revenues, and the *shape* of the economy, or the proportion of revenues across the sectors. The size of the economy is driven by consumption: increase the number of consumers, or the amount consumed by each, and the economy must increase in size. Decrease the amount of consumption, and the economy must shrink. But as the economy increases and decreases in size, its basic shape remains the same, because the basic industries and technologies in use remain the same.

The size of the economy is primarily driven by the number of consumers, which comes from the Demographic model (Section 5). The shape is determined by the following inputs:

* The base wage: the average wage for one work-year, in dollars.
* The average consumption of goods by each consumer each year, in goods baskets.
* How the participants in the economy allocate their income to their expenses.

Setting these parameters is called *calibrating the Economic model*; the process is described in Section 14.15.

### Economic Outputs

The Economic model is run once a week. Each week, it produces the following outputs for each pair of sectors *i* and *j*.

* The quantity demanded: the number of units of sector *j*’s output purchased by sector *i*, only for sectors that demand a product (**goods**, **black**, and **pop**).
* The expense: the dollars spent by sector *i* to buy the quantity demanded of *j*’s output.

Then, for each sector:

* The quantity supplied: the output of the sector, in the sector’s units of production, for those sectors that have product (**goods**, **black**, and **pop**).
* The revenue: the output of the sector in dollars.
* The expense: the dollars spent on the ingredients by the sector.

And then, for the local economy as a whole:

* The unemployment rate, as a percentage of the size of the work force.
* The gross domestic product (GDP), in dollars: the total revenue of the economy, excluding the **world** sector but including subsistence agriculture, i.e., the “size” of the local economy.
* The average per-capita demand for goods.

### Neighborhood Aggregation/Disaggregation

The CGE covers the entire local economy; but Athena generally deals with happenings in particular neighborhoods. Thus, we need to aggregate neighborhood inputs for use by the CGE, and disaggregate CGE outputs down to the neighborhood level.

#### Neighborhood Aggregation

The relevant figures are the number of workers, the number of consumers, and the production capacity of each neighborhood. The Demographics model outputs the first two, as described in Section 5. The Infrastructure model determines the production capacity of each neighborhood as follows:

* Each neighborhood has a “production capacity factor”, or PCF. The PCF represents the neighborhood’s production at time 0.
* The size of the economy is solved at time 0.
* The quantity supplied by the **goods** sector represents the production of the playbox at time 0.
* We allocate that quantity supplied to the neighborhoods in proportion to each agent’s shares of infrastructure, the PCF, and the initial average repair level of goods production infrastructure yielding the production of each neighborhood by agent.
* We assume that each neighborhood is producing at its capacity at time 0.

The total production capacity of the economy is then the sum of the production capacities of the neighborhoods; this limits the size of the economy in the same way that the size of the labor force does.

#### Neighborhood Disaggregation

It would seem reasonable to disaggregate a number of outputs to the neighborhood level, such as the wages paid. This rarely turns out to be useful, however, as the only basis for doing so is usually the population or labor force size, which was used to do the aggregation to begin with.

Consequently, we currently disaggregate only the Unemployment Rate, resulting in the number of unemployed workers per neighborhood. Factors influencing the disaggregation of unemployment are the production capacity of neighborhoods, the amount of labor in each neighborhood, and the proximity of neighborhoods that have labor to those that have goods production infrastructure, and thus jobs. This turns out to be interesting because the size of the labor force as a fraction of the population can vary from neighborhood to neighborhood, due to the presence or absence of subsistence agriculture groups (Section 8.2.3). Also, given the maximum neighborhood proximity workers are willing to travel to find work, we see that the number of unemployed workers as computed by the disaggregation algorithm in the demographics model may be higher than that as computed by the labor constrained view in the economics model. This difference is what we call Geographic Unemployment and further impacts the constraint of labor on the fully constrained view of the economy. We then use the number of unemployed workers relative to the neighborhood population to drive the Unemployment situation (Section 8.2.5.2).

### Ways to Affect the Economy

In Athena 6, there are a number of ways to affect the economy:

* Civilian casualties can decrease the number of consumers and workers.
* Population growth rates can increase or decrease the number of consumers and workers.
* When a civilian group’s security decreases, workers and shoppers can stay at home out of fear, thus reducing the effective size of the labor force and the amount of consumption.
* By their choice of tactics, actors can increase or decrease their expenditures, or choose to take money out of circulation.
* EXECUTIVE tactics can be employed to change input parameters to the CGE, such as changing the price or amount of feedstock available to the black market.
* Each neighborhood’s Production Capacity Factor can be increased or decreased, reflecting the amount of goods production infrastructure initially allocated to neighborhoods.
* Actors can increase or decrease the average repair level of goods production plants thereby increasing or decreasing the goods production capacity in neighborhoods.
* Actors can build new goods production plants thereby increasing the production capacity of neighborhoods.
* The SYSTEM agent can set the average repair level of goods production infrastructure thereby increasing or decreasing production capacity in neighborhoods.
* The maximum neighborhood proximity workers are willing to travel can be set to change how much labor in neighborhoods is available to those neighborhoods that have jobs.
* When displaced from their land, the subsistence population willy-nilly become consumers, and might or might not contribute to the labor force.

### Effects of the Economy

Economic outputs affect the economy in the following ways:

* Civilian attitudes are affected directly by the unemployment rate, as described in Section 8.2.5.2.
* Actors receive their income from the economy, and their expenditures on their tactics flow back into the economy. An actor can employ all his tactics only if he has enough money to do so. Thus, as the economy grows or shrinks an actor’s ability to act may increase or decrease.
* The labor-constrained number of jobs is an input to the unemployment disaggregation algorithm in the demographics model.
* The quantity of goods produced determines the total consumption of goods by the civilian groups, which in turn has satisfaction effects.

## The Social Area

The Social Area concerns the residents of the playbox, their demographics, their beliefs and attitudes, their well-being, their contributions to the local economy, and so forth. This is the largest part of Athena, with many different sub-models.

### Civilian and Organization Groups

Athena represents masses of human beings as *groups*. There are three kinds: *civilian groups*, representing the residents of the neighborhoods in the playbox, *force groups* (described in Section 6.1), representing military and other forces in the playbox, and *organization groups*, which include NGOs (non-governmental organizations) and other similar groups. Every group has a demeanor, **apathetic**, **average**, or **aggressive**, that indicates its propensity for violence.

#### Civilian Groups

The population of the playbox is broken down into a number of *civilian groups*. Each civilian group resides in a particular neighborhood, and has a *belief system* that gives it its identity (Section 8.1.1) and determines how it relates to other groups and actors. It also has greater or lesser *satisfaction* with the current state of affairs, and a greater or lesser *cooperation* with (willingness to give information to) the actors’ various force groups (Sections 8.6 and 10.1). Civilian groups support actors to a greater or lesser degree; ultimately, any successful actor must either derive his power in a neighborhood from the resident civilians, or expect to keep a significant body of troops in place.

In JNEM, and in earlier versions of Athena, it has been usual to think of civilian groups in ethnic terms, as clans, tribes, or related groupings of such. At the beginning of the Athena project we expected each group to be a stovepipe containing individuals from cradle to grave. Since Athena 5, however, we define civilian groups to be collections of people living in the same area who share a belief system. When an ethnic grouping contains significant divisions, e.g., a “generation gap”, it may be prudent to split the ethnic grouping into multiple groups. Similarly, if a particular belief system cuts across ethnic groups, it may be wise to treat it as a single group.

It is possible to *flow* civilian population from one group to another; such population flows can represent seasonal migration, displacement of persons, or conversion of belief (i.e., individuals can move from one group to one whose belief system better represents their views). Civilian groups can be *empty*, i.e., can have zero-population; such empty groups are usually place-holders for migration or displacement that may occur during the course of the simulation.

#### Organization Groups

Organization groups are similar to force groups, but have some mission other than the projection and use of force. There are three kinds:

* Non-Governmental Organizations (NGOs), e.g., Doctors Without Borders
* Inter-Governmental Organizations (IGOs), e.g., UNESCO
* Contractors, e.g., Halliburton

Like force groups, members of organization groups can be assigned activities in neighborhoods, though they are limited to humanitarian relief activities of various kinds. In principle, organization groups can support the political goals of actors by their presence in one neighborhood or another; however, organization groups are usually owned by pseudo-actors who are not politically active.

### Demographics

The Demographics model is concerned with how many civilians there are and where they live. The Demographics model proper is responsible for determining the current population (by group, neighborhood, and playbox), as well as the size of the labor force, the number of consumers in the local economy, and similar population statistics. In addition, the Demographic Situation (demsit) model determines the effects of unemployment and consumption (or lack of consumption) of goods on each civilian group, which in turn drives attitude change. The Demographics model disaggregates unemployment based on the number of jobs demanded by the Economic model, neighborhood infrastructure capacity and neighborhood proximity.

#### Base Population

The population is divided into civilian groups (Section 8.1.1); each civilian group resides in a neighborhood. At time 0, each civilian group has an initial or *base* population. The base population of the neighborhood is simply the total across the civilian groups, and the base population of the playbox is simply the total across the neighborhoods. Note that we also track the total population of local neighborhoods (Section 4.1.3), because that figures into the Economics Area.

#### Current Population

The *current population* of civilian groups and of neighborhoods can change over time. Athena has always modeled deaths due to civilian collateral damage (Section 6.3). The Demographics model tracks attrition to date, subtracting it from the current population.

In addition, Athena has a simple model of population growth or decline. Each group has a rate of population change, expressed as a yearly percentage, e.g., group G grows at a rate of 1% per year. This rate is applied each week, resulting in a slow, steady growth or decline in a group's population.

#### Subsistence Agriculture

Civilian personnel can support themselves by *subsistence agriculture* or by participating in the local economy. In Athena there is a hard line between the two: any given group does one or the other. People groups who live by a mixture of the two should be modeled as two distinct groups in Athena.

Because subsistence personnel do not (by definition) participate in the local economy, they are neither consumers nor members of the labor force. As a result, they are not directly affected by shortages of jobs or of consumer goods. If subsistence personnel are displaced from their homes, however, they should also be displaced from the normal livelihood and so necessarily enter the economy as consumers (and possibly also as workers).

#### Consumers and Workers

All members of non-subsistence groups are presumed to be *consumers* in the local economy;[[12]](#footnote-12) the total number of consumers drives the size of the economy.

In each non-subsistence group, only a percentage of the consumers (nominally 60%) are members of the *labor force*. This percentage can be set on a group by group basis, and will typically be lower (and possibly even 0) for groups that represent populations that are displaced from their homes.

#### Demographic Situations

*Demographic situations* are situations detected by the Demographic model that affect the attitudes of the civilians. Athena defines two demographic situations, or *demsits*: the Consumption of Goods situation and the Unemployment situation.

##### Consumption of Goods

Goods produced by the economy are allocated to groups in proportion to their contribution to the labor force—workers have the money to buy goods. A group's consumption of goods affects the group's attitude according to two factors:

* Whether the actual level of consumption is more or less than expected.
* The fraction of the group that is living below the poverty line.

The poverty line is determined by applying a Lorenz curve given the Gini coefficient for the playbox.

See the CONSUMP rule set in the *Athena Rules Document* for more details.

##### Unemployment

The unemployment rate is computed for the entire playbox by the Economics model and affects civilian groups in proportion to the number of workers in each group. High unemployment affects civilians indirectly, by the presence of unemployed workers in the neighborhood. See the UNEMP rule set in the *Athena Rules* document for specifics.

Note that the effects of economic hardship due to unemployment are handled by the Consumption of Goods demsit.

### Belief Systems and Affinities

Every civilian group and actor has a *belief system*, a statement of the ideas and issues that are important to the group or actor, along with how important they are and how the group or actor feels about those who disagree. For a civilian group, the group’s belief system is the source of the group’s identity and the basis for its relations with all other groups. An actor’s belief system may indeed reflect the actor’s deeply held beliefs, or it may be a construct intended to garner support from the civilians.

Given the belief systems of two entities, A and B, we compute the *affinity* of A with B, and of B with A. The affinity is a number from 1.0 down to -1.0 that indicates whether A supports or opposes the same things as B. Note that affinity need not be symmetric.

All horizontal and vertical relationships in Athena are ultimately based on affinities, and hence on belief systems.

Belief systems and affinities are computed by the Mars Affinity Model (MAM), which is documented in the *Mars Analyst’s Guide*.

#### Beliefs and Topics

A belief system consists of an entity’s beliefs about one or more topics. A topic is some value, principle, or issue about which there is some disagreement in the playbox. In Pakistan, for example, Islam is a significant fault line between the Pakistani citizens and the United States. Topics are chosen by the analyst; there is no default set.

A belief is described by two values, the entity’s *position* for or against the topic of interest, sometimes known as the *zeal*, and the entity’s *emphasis* on agreement or disagreement with that position. The former indicates how much the entity cares, and the latter determines how it feels about those who agree or disagree.

The position and emphasis are usually entered qualitatively, using the following values. For position, the entity may be:

* **Passionately For (P+)**
* **Strongly For (S+)**
* **Weakly For (W+)**
* **Ambivalent (A)**
* **Weakly Against (W-)**
* **Strongly Against (S-)**
* **Passionately Against (P-)**

The entity may put its emphasis on agreement or disagreement, as follows:

* **Agreement—Strong**
* **Agreement**
* **Neither**
* **Disagreement**
* **Disagreement—Strong**
* **Disagreement—Extreme**

If the emphasis is on agreement, the entity will tend to have a higher affinity with those entities with whom it agrees on this topic, while to some extent disregarding disagreements. If the emphasis is on disagreement, the entity will tend to have a lower affinity with those with whom it disagrees on this topic, while to some extent disregarding agreements.

Note that strong beliefs do not thereby imply a propensity to use violent action (though they may engender violent action in those groups that are so inclined). Propensity to violence is driven by a group’s *demeanor*; see Section 8.1. For example, I might be passionately in favor of chocolate ice cream, and greatly dislike anyone who passionately dislikes chocolate ice cream without being inclined to take violent action against the chocolate ice cream haters. It depends on my demeanor.

Levity aside, a strong position and emphasis on a topic *does* indicate some willingness to take action. It simply does not indicate a propensity for violence.

#### Affinity

The affinity between two entities is computed by comparing their beliefs on each topic, and tallying the effects of their agreements and disagreements given their positions and emphases.

As stated above, affinity is a number from -1.0 to 1.0 that is used as the basis for computing relationships.

Beliefs are the basis for affinity, and are also used in the semantic hooks of information operations messages (see Section 10.2). Because some beliefs with no significant effect on affinity can be used to make a message palatable to a particular group, Athena adds an *affinity flag* to each topic; if it is true, each entity’s beliefs on that topic will be used when computing affinities, and otherwise not.

Affinity is computed when the scenario is locked and simulation begins, and is constant thereafter. In future versions it might be allowed it to vary.

#### Playbox Commonality

*Playbox commonality* is the degree of belief that is generally common to all groups in the playbox, to the degree that they are part of the dominant culture.

Selecting the relevant set of topics for a given playbox is more of an art than a science, and by the nature of things the tendency is to accentuate the negative—it is simply easier to identify fault lines than significant areas of agreement. When commonality is ignored, however, the computed affinities tend to indicate that all parties concerned hate each other with a deep and abiding hatred.

The Mars Affinity Model handles this using the *playbox commonality* slider, a numeric factor nominally set to 1.0. We assume that the entities in the playbox have significant commonality, and in fact generally agree on about as many things as they disagree. More specifically, for each topic entered by the analyst, we add an implicit pseudo-topic of general agreement.

The number of these pseudo-topics is determined by the slider. If it is set to 1.0, there are as many pseudo-topics as real topics. If it is set to 2.0, there are twice as many; if it is set to 0.5, there are half as many. Moving the slider up and down will move affinities up and down accordingly.

#### Entity Commonality

The *entity commonality* is the extent to which the entity participates in the general consensus indicated by the playbox commonality.

The playbox commonality lets you establish the degree of belief that is generally common to all groups in the playbox. But in fact, different entities will share in that common pool of belief to different extents, and foreign groups and actors will often have significantly different beliefs. Consequently, each group and actor has an *entity commonality* slider: a number from 0.0 to 1.0. If the entity commonality is 1.0, the entity shares the general consensus completely; if 0.0, not at all.

Decreasing an entity’s entity commonality will tend to decrease the entity’s affinity with other entities.

### Attitudes

The *Attitudes Model* deals with the attitudes of the people in the playbox, and particularly:

* The horizontal relationships between groups
* The vertical relationships between groups and actors
* The satisfaction of the civilian groups with respect to various concerns
* The cooperation (i.e., willingness to share information) of the civilian groups with respect to the force groups.
* Assessment of the effects of events and situations in the other models on the attitudes.

Note that the term “attitudes” properly applies to horizontal relationships, vertical relationships, satisfaction levels, and cooperation levels, all of which are managed by the Unified Regional Attitude Model (URAM) (see Section 8.4.1 and the *Mars Analyst’s Guide*). However, the belief systems from which the relationships are derived are also attitudes in a wider sense.

Vertical relationships are discussed in the Political Area; see Section 5.3. Cooperation levels are discussed in the Information Area; see Section 10.1.

#### Unified Regional Attitude Model (URAM)

Athena's attitudes are tracked by the Unified Regional Attitude Model (URAM). URAM is described in full in the *Mars Analyst’s Guide*; this section explains some of the basic URAM concepts.

##### URAM Attitude Curves

A URAM attitude level is a curve that increases and decreases over time. In particular, the value of the curve at time *t* is a *baseline level* plus the transient effects of current attitude drivers. If all transient effects ceased, the *actual level* would drop back to the baseline level.

The baseline level of the attitude curve is recomputed at each time step, based on the transient effects present in the previous time step, and possibly also on the *natural level* of the curve.

Consider a civilian group’s SFT (Safety) satisfaction (see Section 8.6.1). The group has a certain baseline level of SFT; in the absence of drivers that cause the group to feel more or less safe, this will also be the group’s actual level of SFT. Now, suppose that activities go on that make the group to feel less safe. These will cause transient effects that will make the group’s actual level of SFT be lower than the baseline. If the transient effects continue week after week, the baseline level itself will decrease, resulting in an even lower actual level.

Suppose these activities cease. The transient effects will cease immediately, and the group’s actual level of SFT will immediately increase…but not to where it was before. The baseline level is still lower than it was. However, there is a natural level of SFT; it depends on the group’s security at time *t* (see Section 4.6). The more secure the group really is, the higher its natural level of SFT, and the less secure the lower. In the absence of other drivers, the group’s baseline level of SFT will naturally regress back to the natural level over time.

##### Persistent and Transient Inputs

Attitude drivers can cause persistent and transient inputs to URAM. A transient input causes the kind of transient effects described in the previous section. Thus, a POWEROUT situation will decrease QOL by a nominal –15 points during each week that the situation continues; but this is not cumulative. It is simply a scaled –15 points taken from the baseline level during the current week. When the situation is resolved, those –15 points go away; QOL returns to the baseline level.

Persistent inputs, by contrast, directly affect the baseline level of an attitude, changing it more or less permanently. Persistent inputs are used for one-time events, such as civilian casualties.

### Horizontal Relationships

A horizontal relationship is a representation of how a group feels about another group. Every group, whether civilian, force, or organization, has a horizontal relationship with every other group. The relationship is represented as a number for –1.0 to 1.0, where 1.0 indicates that the groups are bosom friends (every group automatically has a relationship of 1.0 with itself) and –1.0 indicates that the groups are the bitterest of enemies.

These relationships are attitudes that can be affected by various attitude drivers.[[13]](#footnote-13) The horizontal relationship of one group with another is initially the affinity of the first group for the second, and will regress to that affinity over time. Force and organization groups do not have belief systems, but they are owned by actors that do; and for the purpose of computing horizontal relationships we simply presume that they inherit the belief systems of their owners.

This horizontal relationship is one of the most basic concepts in Athena, and has significant effects across the entire simulation. For example, horizontal relationships are taken into account in URAM when computing the spread of indirect satisfaction and cooperation effects across the playbox.

The natural level of group *f'*s relationship with group *g* is *f'*s affinity for group *g* based on their belief systems (see Section 8.3). Over time, and in the absence of other drivers, the relationship will regress back to this natural level.

The initial, affinity-based horizontal relationships can be overridden by an analyst’s preferred value during scenario preparation.

### Satisfaction Levels

Every civilian group has a sense of satisfaction or dissatisfaction with the state of affairs in the playbox. Satisfaction in this sense is not a feeling, *per se*, though we often use the language of feelings and talk about the group’s “mood” or say that the group “likes” or “dislikes” some event or situation. Rather, dissatisfaction is the will to change the current state of affairs, and satisfaction is the will to preserve the current state of affairs.

Satisfaction is measured as a number from 100.0 to –100.0, where 100.0 is perfectly satisfied and –100.0 is utterly dissatisfied.

The initial satisfaction levels are set by the analyst during scenario preparation; once simulation begins they vary depending on the events and situations that occur (Section 8.7).

#### The Four Concerns

We measure satisfaction on four axes, called the four *concerns*, also known as the four *soft factors*:[[14]](#footnote-14)

**Autonomy (AUT):** Is the group satisfied with their government? AUT measures how a group feels about its governance. This includes how it feels about the actors that govern it, its ability to participate in the political system, and whether or not it has its preferred system of governance.

**Safety (SFT):** Do members of the group fear for their lives, either from hostile attack or from collateral damage from force activities? This fear includes environmental concerns such as life-threatening disease, starvation, and dying of thirst.

**Culture (CUL):** Does the group feel that its culture and religion, including cultural and religious sites and artifacts, are respected or denigrated?

**Quality of Life (QOL):** Is the group satisfied with their surroundings? QOL measures how a group feels about its prosperity, living standards, and economy. This includes the physical plants that provide services, including water, power, public transportation, commercial markets, hospitals, communications, etc., and those things associated with these services, such as sanitation, health, education, employment, food, clothing, and shelter.

The AUT, CUL, and QOL satisfaction levels do not regress back to a natural level; the initial values are set by the analyst, and baseline changes persist indefinitely.

#### Saliencies

Each group has a *saliency* for each concern: a number from 1.0 to 0.0 that indicates how important that concern is to the group, where 1.0 is crucial and 0.0 is negligible. Saliency is the basis for comparing satisfaction levels across groups and concerns, e.g., for computing *mood*.

Saliency is distinct from satisfaction magnitude. Suppose a group has a saliency of 1.0 for QOL and 0.5 for AUT, but the group’s QOL is at 10.0 and its AUT is at –100.0. At that time, then, the group cares much more about its AUT than its QOL, even though QOL has the greater saliency. When computing mood, however, QOL will count twice as much as AUT.

#### Group Mood

A group’s *mood* is the average of its four satisfaction levels, weighted by the group’s saliency for each concern. Mood is a convenient summary statistic, and is an input to a number of other models.

### The Driver Assessment Model (DAM)

The *Driver Assessment Model* (DAM) is responsible for assessing the satisfaction, cooperation, and relationship effects of the various events and situations that occur in the simulation, e.g., civilian casualties (Section 6.3) and activity situations (Section 4.8). Collectively, these events and situations are called *drivers*. Athena contains a rule set for each kind of driver; the rule set assesses the driver, and gives *persistent inputs* and *transient inputs* (see 8.4.1.2) to URAM, which tracks the effects as they play out over time. These rule sets are documented in detail in the *Athena Rules* document.

#### Direct and Indirect Effects

Every attitude input targets a specific curve: group A’s satisfaction with QOL, or group B’s cooperation with group C. This is called the direct effect. But for satisfaction and cooperation inputs, other groups are affected as well. If group A takes casualties, for example, this affects not only A’s satisfaction but also the satisfaction of the other groups in the neighborhood. This is called an indirect effect, and it is usually modified by the relationship between the groups.

Indirect effects can occur in the same neighborhood, in nearby neighborhoods, and in far-away neighborhoods (see Section 4.1.2 for a discussion of neighborhood proximity). The spread of indirect effects is determined by the input’s *here factor*, *near factor*, and *far factor*, each of which is a simple multiplier of 0.0 to 1.0 applied to the magnitude of the indirect effect in the same neighborhood (here factor), in nearby neighborhoods (near factor) and in far-away neighborhoods (far factor). These factors are also commonly referred to as *s*, *p*, and *q*, respectively. The here factor, *s*, is 1.0 in all of the built-in rule sets, but may be set to other values by CURSEs (Section 8.8). The near and far factors, *p* and *q*, are set in the model parameter database for all of the built-in rule sets; see the *Athena Rules* document for the default values, or query the model parameter database in the Athena application (Section 14.19).

Horizontal and vertical relationship inputs do not have indirect effects.

### Magic Changes to Attitudes

The Driver Assessment Model’s rule sets cover a wide variety of events and situations, but they don’t cover everything; and in particular, they don’t cover one-of-a-kind situations and events like the assassination of a government leader, the World Trade Center attack or the outbreak of a deadly virus. Athena provides two ways for the analyst to create rule sets “on the fly”: Complex User-defined Role-based Situations and Events (CURSEs), and Magic Attitude Drivers (MADs).[[15]](#footnote-15) See Section 13.7.5 for a description of the user interface for CURSES.

#### Complex User-defined Role-based Situations and Events (CURSEs)

*Complex User-defined Role-based Situations and Events* (CURSEs) consist of a description, such as “Virus Outbreak”, and any number of *attitude injects* which cause attitudes to change due to the CURSE. CURSES are defined as part of scenario preparation, and can be magically injected into the simulation using the CURSE tactic. CURSEs can affect all four kinds of attitude with both persistent and transient effects.

CURSEs are defined in terms of *roles* rather than specific groups or actors: the roles are mapped to specific groups or actors using the CURSE tactic. A single "Flood" CURSE can be applied to any number of distinct neighborhoods by creating multiple tactics that reference the CURSE and specify different targets for the CURSE's roles.

#### Roles in CURSEs

When creating CURSE injects, *roles* are used to define who is affected by the CURSE. A *role* is similar to a variable and is defined when an attitude inject is added to a CURSE. There are four different types of roles: generic group roles, civilian group roles, force group roles and actor roles. A role’s name is defined by the user, but Athena knows the type of role based on the type of inject desired (e.g. injects to affect vertical relationships require actor roles and generic group roles).

When a CURSE tactic is added to the SYSTEM agent’s strategy, each *role* must be associated with a rule to retrieve the appropriate groups or actors in the scenario that should be affected by the CURSE. Each rule is selected from a list of available rules that depend on the type of role. For example, if a CURSE contains an inject that should affect cooperation then that inject will need two rules defined: one to retrieve affected civilian group(s) and one to retrieve affected force group(s). For the civilian group role the rule could be “All civilian groups in neighborhoods N1, N2 and N3” where N1, N2 and N3 are neighborhoods defined in the scenario. Similarly, for the force group role the rule could be “Force groups owned by A1” where A1 is some actor in the scenario that owns one or more force groups. When the CURSE tactic actually executes during runtime, the rules associated with the roles are evaluated to determine the actual groups or actors that should be affected by the CURSE. This way, the analyst need not concern himself with particular groups or actors, the rule will do this work for him. The rules available to retrieve groups and actors are predefined in Athena.

## The Infrastructure Area

Infrastructure consists of physical assets located in neighborhoods that provide some benefit to the residents of those neighborhoods. Examples include the power grid, the water supply system, means of communications, and the means of production of goods and services. Athena 6 explicitly models two kinds of infrastructure: *Communications Asset Packages* (CAPs) and ***goods*** *production infrastructure*. These are discussed in the following sections.

### Communications Asset Packages (CAPs)

In the real world, there are many different kinds of communications asset: television stations, radio stations, telephone and cell phone networks, the Internet, and so forth. We refer to these assets as *communications asset packages*, or CAPs. Athena's model of CAPs focuses narrowly on broadcast communications, without worrying about the construction or maintenance of the infrastructure underlying the capability.[[16]](#footnote-16)

Every CAP is owned by some actor, and reaches some portion of the civilian population.

#### CAP Coverage

The reach of the CAP is governed by three parameters: *capacity*, *neighborhood coverage*, and *group penetration*.

**Capacity.** The capacity of a CAP is a number from 0.0 to 1.0. Reducing the capacity reduces the number of people reached by the CAP’s broadcasts in proportion. A CAP’s capacity can be adjusted as the simulation runs via the EXECUTIVE tactic and the CAP:CAPACITY order, as a proxy for damage, repair, and interdiction by other actors.

**Neighborhood Coverage.** Traditional broadcast media have a coverage area based on the location of their antennas and the strength of their transmitters. Even Internet-based media have coverage areas; not every locale has broadband internet access. The neighborhood coverage of a CAP is specified as the fraction of each neighborhood in which the CAP’s broadcast is available. Thus, a particular television station might be visible in all of neighborhood N1 but in only half of neighborhood N2.

**Group Penetration.** Just because a particular broadcast is available in a neighborhood doesn’t mean that it will reach all of the civilians in that neighborhood. The group penetration of a CAP is the fraction of each group that will receive broadcasts that are available to them.

Thus, for each civilian group the fraction of the group that receives a broadcast is the product of the CAP’s capacity, the coverage of the group’s neighborhood, and the penetration of the group. For example, group G1 lives in neighborhood N1. CAP K1’s capacity is 1.0; its coverage of N1 is 0.8; and its penetration of G1 is 0.5. Any broadcast sent via K1 will thus be seen by 1.0⨯0.8⨯0.5 = 0.4 or 40% of the members of group G1.

Thus, the choice of CAP can make a big difference in the effectiveness of IOMs; the actor must select CAPs that reach the civilian groups he is targeting, while doing his best to avoid the groups he is not.

#### Access to CAPs

By default, only the actor who owns a CAP may send IOMs via that CAP. The owner may use the GRANT tactic to grant access to a CAP to some or all of the other actors.

In reality, an actor might seize control of a CAP. The analyst can handle this by having the owner grant access to the actor seizing control.

#### CAP Costs

Given the wide variety of CAPs in the real world, and the vastly different economics of running a television station vs. a newspaper vs. a website, it is difficult to come up with a one-size-fits-all cost model that makes sense. In Athena we have chosen to err on the side of simplicity. Each CAP has a broadcast cost in dollars per week; in order to send an IOM via the CAP during a particular week, the actor must pay the broadcast cost. This cost is established by the analyst during scenario preparation.

### Goods Production Infrastructure

The **goods** sector is that part of the economy that provides all non-black-market goods and services. Producing goods and services requires infrastructure ranging from power plants and factories to stores and other consumer businesses.

Prior to Athena 6, the infrastructure required by the **goods** sector was purely implicit. The **goods** production capacity of the local region was ascribed to the local neighborhoods in proportion with their populations, the assumption being that the work was done where the people lived. There was no real way to damage or increase production capacity as part of an intended course of action.

Athena 6 adds a model of **goods** production infrastructure that allows production capacity to vary from one neighborhood to another according the actors' decisions.

The basic unit of production is the *plant*, which can produce a certain number of **goodsBKTs** (goods baskets) per week. An actor may own any number of plants; may build new plants over time; and must maintain the plants he owns or they will degrade, decreasing production capacity.

Although the notion is certainly that of a manufacturing plant, for Athena the *plant* is simply a convenient unit of production, with its accompanying infrastructure. Plants take time and money to build, and do not come "on-line" until they are complete.

#### Scenario Preparation

**Total Production Capacity.** The analyst specifies the total production capacity of the local region as one of the Economics model's inputs (section 14.15.1). By default, all production capacity is given to the SYSTEM agent in the form of plants, in proportion to the population in each neighborhood. These SYSTEM-owned plants do not degrade over time and need not be maintained. Thus, in the absence of other user input, Athena 6 effectively handles **goods** production capacity as in previous versions: the infrastructure is modeled, but unobtrusively so.

**Neighborhood Production Capacity.** In a given scenario, it may be that production capacity isn't distributed in proportion to population: an urban neighborhood might have proportionally more of it than a remote neighborhood, with workers coming from nearby suburbs to fill the jobs. The analyst can adjust the amount of initial production capacity in each neighborhood by setting the neighborhood's *Production Capacity Factor*, or *PCF*. The PCF is a multiplier, initially 1.0, which determines how many shares of production capacity the neighborhood receives per capita. To give a neighborhood twice as much production capacity per capita as other neighborhoods, set its PCF to 2.0.

**Ownership of Production Infrastructure.** As the next step, the initial ownership of production infrastructure in each neighborhood can be given to various actors. Suppose that in neighborhood N1, actor A owns 50% of the production infrastructure, actor B owns 25%, and the remaining 25% is owned by actors too small to be included in the scenario. The analyst can indicate this by giving breaking N1's production infrastructure into four shares, and giving two shares to actor A, one share to actor B, and one share to the SYSTEM agent.

**Calculation of Plants.** Thus, the user must specify these particulars:

* Total production **goods** production capacity for the local economy, measured in goodsBKTs; or, equivalently, in dollars, since at time 0 one goodsBKT costs one dollar.
* Each neighborhood's fraction of that total capacity, expressed as a PCF. This may take some trial and error, but there are tools to help.
* Each actor's fraction of ownership for the production capacity of each neighborhood, *to the extent that the user wishes to break it down*. Any unclaimed production capacity is given to the SYSTEM agent.
* The initial average level of repair by neighborhood and actor, expressed as a percentage between 0% and 100%.

Given these inputs, Athena will compute the total number of "plants" needed to just exactly match the goods sector capacity at time 0 in each neighborhood and assign them to actors or to the SYSTEM agent.

#### Levels of Repair

An actor's plants in a given neighborhood have an average level of repair ranging from 100% down to 0%. The capacity of the actor's plants in the neighborhood is based on the level of repair, with a 100% level of repair providing full capacity and 0% producing zero capacity.

By default, actor-owned plants degrade over time: the level of repair will drop each week, ultimately reaching zero, unless the owning actor chooses to maintain his plants using the MAINTAIN tactic. Maintenance costs money, and an actor may choose to maintain his plants at any desired level of repair given available cash.

#### Building New Plants

An actor may choose to construct new plants using the BUILD tactic. A plant requires multiple weeks to build[[17]](#footnote-17), and only so much work can be done on a single plant in a given week. Each time the BUILD tactic executes, it will begin or continue construction on the plants nearest completion for which work can still be done that week. Construction is done on best efforts basis given the resource constraints of the actor doing the construction.

#### Auto-Maintenance

Each actor has an *auto-maintain* flag. If the flag is true, maintenance of that actor's plants takes place automatically, just as it does for the SYSTEM agent. The actor need not spend any of his income on maintenance; it is assumed that required maintenance costs are implicit in the Economic model's CGE, and that the relevant funds are not included in the actor's income.

Thus, the analyst can choose between three broad approaches for modeling **goods** production infrastructure in a particular neighborhood:

* He may leave it all in the hands of the SYSTEM agent.
* He may assign it to specific actors, but allow maintenance to be automatic.
* He may assign it to specific actors, but require explicit maintenance.

Which approach the analyst chooses will depend on the courses of action being studied.

#### Total Production Capacity and Unemployment

As the simulation runs, the total production capacity in each neighborhood each week depends on the number of plants and their levels of repair. Thus, if an actor doesn't maintain his plants the production capacity of the neighborhood will drop; whereas if he builds new plants or raises the average level of repair of existing plants, the production capacity of the neighborhood will increase. Employment is based on the distribution of production capacity: workers are presumed to travel to where the work is (within reason). Hence, unemployment will tend to be higher in neighborhoods with a higher population and lower production capacity.

## The Information Area

Information, its use, and its spread are crucial in the kinds of scenarios Athena has been designed to address. In principle, the Information Area includes the following:

* Command and control of troops and organizations
* The spread of information within the civilian population and across the playbox
* Intelligence received by force groups (and hence by actors) from the civilian population
* Information operations: propaganda and other media techniques intended to affect the perceptions of the people in the playbox, so as to:
  + Increase support for an actor
  + Decrease support for an actor
  + Skew the information available to an actor, thus affecting the actor's decisions
  + Change other civilian attitudes

In particular, Athena models information in the following ways:

* URAM tracks the cooperation (willingness to give information) of civilian groups with force groups.
* URAM includes a simple model of information flow across the playbox as it applies to satisfaction and cooperation effects.

An actor may conduct information campaigns by sending *information operations messages* (IOMs) to the civilians via broadcast *communications asset packages* (CAPs); these messages can affect civilian attitudes in a number of ways. CAPs are described in Section 9.1; IOMs are discussed below, and in full detail in the *Athena Analyst’s Guide*.

### Cooperation Levels

Cooperation is the measure of how much information a civilian group, whether willing or coerced, will provide to a force group. Each civilian group is said to have a *cooperation level* with respect to each force group. The cooperation level is a number from 0 to 100, and represents the probability that a member of the civilian group will give information to a member of the force group.

Like satisfaction levels, cooperation levels are initialized during scenario preparation and vary thereafter based on the events and situations (drivers) that occur (Section 8.7).

Note that having a high level of cooperation with a force group does not imply that the civilian group will overtly aid the force group in any way. It might or might not.

Athena also computes neighborhood cooperation levels: the average cooperation level of the groups in the neighborhood with the various force groups.

Cooperation and neighborhood cooperation levels are useful summary statistics; at present, they directly affect only the attrition model (Section 6.3).

The natural level of civilian group *f'*s cooperation with force group *g* is simply the initial cooperation level as set by the analyst. Over time, and in the absence of other drivers, the relationship will regress back to this natural level.

### Information Operations Messages (IOMs)

An actor may send an *information operations message* (IOM) via a CAP in order to affect civilian attitudes. The choice of CAP determines the set of civilians who will receive and respond to the IOM.

An Athena IOM is a highly abstract creature. A real-world IOM can be a smoothly produced piece of work with significant informational and emotional content. Athena is not able to parse and understand even the informational content, let alone the choice of colors, graphics, music, timing, and so forth in a typical radio, television or magazine spot. It is up to the analyst to truly analyze the message and its effects into terms that Athena can do something with. Nor is Athena able to determine or model how often an IOM should be sent during the week. It is assumed that those responsible for broadcasting the IOM will do the appropriate thing given the context.

An IOM consists of the following components: a *description*, a *semantic hook*, and one or more *payloads*.

#### IOM Description

The IOM’s description is a brief, English-language statement of the message, entered as an aid to the analyst. Athena displays the description, but does not attempt to parse it.

#### Semantic Hooks

An IOM attempts to get its message across by appealing to the beliefs of the targeted civilians. This appeal is performed by the IOM’s *semantic hook*, which references the belief systems described in Section 8.3.

A semantic hook is simply a set of positions on some subset of the belief system topics; it is possible to define topics purely for use in semantic hooks.

Such a set of positions can be more or less compatible with any given civilian group’s belief system; we compute this in the same way we compute affinity. Thus, a group will be more responsive to an IOM whose hook corresponds well to the group’s beliefs. This is called the *resonance* of the IOM with the civilian group.

#### IOM Payloads

An IOM payload is a desired effect on civilian attitudes specified by the analyst. An IOM can have one or more payloads. Athena determines (based on the choice of CAP, the perceived source of the IOM, the semantic hook, and the payloads themselves) the actual magnitude of the desired effect.

Every IOM payload affects the civilian groups *f* covered by the chosen CAP. There are four kinds of payload:

**Cooperation Payload (COOP).** The COOP payload affects civilian group *f’*s cooperation with some specific force group *g*.

**Horizontal Relationship Payload (HREL).** The HREL payload affects civilian group *f’*s horizontal relationship with some specific group *g*, of any group type.

**Satisfaction Payload (SAT).** The SAT payload affects civilian group *f’*s satisfaction with some specific concern *c*.

**Vertical Relationship Payload (VREL).** The VREL payload affects group *f’*s vertical relationship with some specific actor *a*.

The analyst gives each payload a nominal magnitude, rather like the magnitudes in the rule sets in the *Athena Rules* document. The analyst chooses the magnitude in light of the putative quality and aptness of the real-world IOM and the team producing it.

Note that this magnitude is only nominal. It will be adjusted (and usually decreased) by the CAP’s coverage, the resonance of the semantic hook, and the regard of the receiving group for the actor perceived as the source of the IOM. The adjusted magnitude will ultimately be given to URAM, where it will be interpreted in light of the current levels of the relevant attitude curves.

#### Broadcasting an IOM

An IOM is broadcast via the BROADCAST tactic, which has the following parameters:

* The CAP by which the IOM will be broadcast.
* The perceived Source.
* The IOM to broadcast.
* The production cost.

**Perceived Source.** The perceived source is the actor who will be seen as the source of the IOM, the actor who is trying to influence public opinion. This might be the same as the true source, the actor who executes the BROADCAST tactic; it might be some other actor; or there might be no perceived source, as when an IOM is seen as straight reporting.

The true source might very well wish to hide its identity, and make it look like the IOM is coming from some other source. Athena does not attempt to adjudicate this. Rather, it is up to the analyst to decide which actor (if any) will be perceived as the source, whether this perception accords with the true source’s wishes or not.

**Production Cost.** The cost of broadcasting an IOM is assumed to have two components: the production cost and the transmission cost. The transmission cost is a function of the chosen CAP; the CAP’s cost per message per week is part of the CAP’s definition. However, the cost of producing the message for transmission is a function both of the message itself and of the medium by which it is broadcast, and can’t be defined as simply a part of either CAP or the IOM. Thus, we allow the analyst to enter it here.

Note that the magnitude of the production cost affects only the resources consumed by the broadcast, not how effective the broadcast is. That is determined by the payload magnitudes.

#### Effectiveness of IOM Payloads

The actual effectiveness of an IOM payload given its magnitude depends on a number of factors: the resonance of the semantic hook with the receiving group’s belief system, the relationship between the group and the perceived source, and the usual input scaling performed by URAM. See the *Athena Analyst’s Guide* for the details.

# Part II: Using Athena

## Installation

Athena requires 32-bit Windows, and specifically Windows 7, and is known to operate on 64-bit Windows 7. It is believed to operate on Windows Vista, and may also work on Windows XP.

To install Athena, first acquire the installation wizard from the TRADOC G-2 Intelligence Support Activity (TRISA) at Ft. Leavenworth, Kansas; at time of writing, the contact is Robert Crowson, at [Robert.E.Crowson2.civ@us.army.mil](mailto:Robert.E.Crowson2.civ@us.army.mil). Save the installer to your desktop and execute it in the usual way; the wizard will lead you through the installation process.

### Starting Athena

At the end of the process Athena will be available from the Start menu, and optionally as an icon on your desktop. In addition, you can start Athena with a particular scenario by double-clicking on the scenario file.

### Athena Documentation

The Athena documentation, including this User’s Guide, is available on the Windows Start menu. In addition, the Athena application has extensive on-line help; select any help topic from the Help menu.

### Multiple Versions of Athena

Multiple versions of the Athena software can coexist on one machine, and will have distinct entries on the Windows Start menu. However, there are two caveats.

Double-clicking a scenario file will always invoke the most recently installed version of Athena. This will usually be what you want. If it is not, invoke the correct version, and use its File/Open menu item to open the scenario.

Second, the scenario file’s schema frequently changes during Athena development. If you are running development builds (e.g., 7.0.x) be aware that a scenario created with one version might not work with a later version, and that you might have to take extra steps to convert it. Release builds (e.g., successive releases of version 6.1.*x*) do not have this problem. When installing a new version, always look at the Release Notes (available with the Athena documentation on the Windows Start menu); they will tell you of any version incompatibilities.

## Using the Athena Application

This section gives a top-level overview of the Athena application and how to use it.

### Athena Scenario Files

Athena is a document-centric application, like a word processor or spreadsheet application. In Athena’s case, the documents are called *scenario files*. Each scenario file includes the data required by a single Athena scenario: neighborhoods, actors, groups, strategies, and other inputs. If saved after the Athena simulation has run forward in time, the scenario file will also include the results.

Athena provides the usual **File/New**, **File/Open**, **File/Save**, and **File/Save As** menu items; you can also invoke Athena with a specific scenario file by double-clicking the scenario file in the Windows Explorer (but see Section 11.3 if you have multiple versions of Athena installed).

On Windows, Athena defaults to saving scenario files in your Documents\Athena folder. However, you may save them anywhere you like.

### Athena Workflow

Athena operates in several distinct states; what you can do varies from state to state. The states are shown in the following diagram:



When you create a new scenario, Athena is in the *Scenario Preparation* state, called **Prep** for short. In this state you create the scenario, adding neighborhoods, actors, groups, strategies, and so forth. When you’ve defined the scenario to your liking, you *lock* the scenario, entering the **Paused** state. In this state the simulation is initialized, but time is not advancing. You can examine the initial state of the simulation, and then either return to the **Prep** state or tell Athena to run time forward.

While time is advancing, Athena will be in the **Running** state; in this state, most user input is blocked. After advancing time the appointed amount, Athena will return to the **Paused** state, where you can examine the results. You can then examine the results and either direct Athena to continue advancing time or unlock the scenario, returning the **Prep** state (and losing any simulation results). Once there, you can revise the scenario and start over.

Note that if you save the scenario while Athena is **Paused**, all simulation results will be saved in the scenario file, and will be available when you re-open the file.

### Scenario Mode vs. Simulation Mode

In the **Prep** state you can edit the scenario, but you can’t advance simulation time. In the **Paused** state, you can advance simulation time, but you can’t edit the scenario. The data and tools required are different in these two states, and Athena’s user interface changes to match. In *Scenario Mode*, Athena displays the tools you need to create and edit the scenario. When you lock the scenario, Athena switches to *Simulation Mode*, and adds the tools you need to control simulation time and view the results.

### Viewing Athena Results

Athena provides a great many tools for browsing the Athena results interactively; these tools will be described in Section 13. The most important of these tools is the Detail Browser, which provides a web-browser-like view of all of the scenario data.

It is also possible for other programs to access Athena results. The Athena scenario file is an SQLite database file[[18]](#footnote-18); SQLite bindings are available for most programming languages. You can browse Athena’s SQL database schema by entering the address “my://rdb” into Athena’s **Detail Browser**'s address bar (Section 13.5).

### The Significant Events Log

As Athena advances simulation time, it creates a log of significant simulation events: tactics executed, deployments, civilian casualties, and so forth. Errors in **EXECUTIVE** tactics and **EXPR** conditions also appear in the significant events log.

The significant events log is displayed on the **Time/Significant Events** tab. In addition, the actor, neighborhood, and group pages all end with a list of the significant events relevant to that entity from the last time advance.

### The Debugging Log

As Athena runs, it creates a low-level log of its activities. This log is primarily intended for use in debugging the application and its models, and so it is not usually visible in the GUI. The information contained in the debugging log may be useful to the analyst at times, however. To view the debugging log, select **View/Scrolling Log** from the Athena menu. This will make the **Time/Log** tab visible. See Section 13.14.3 for more on the log browser.

### Background Errors

When Athena encounters an unexpected run-time error, it does not usually halt execution. Instead, the error and a complete stack trace are written to the *debugging log*, and the user is notified. This is called a *background error*, or *bgerror* (pronounced “B-G-error”) for short.

It is usually wise to save the scenario under a new name after a bgerror, and then restart the application.

Background errors should be reported to the development team. The report should include the following items, insofar as this is possible given security considerations:

* The bgerror and stack trace from the debugging log. This is the text with the orange background.
* The complete text of the log file, if possible. Select **detail** on the **Log** pull-down on the **Time/Log** tab's toolbar before cutting and pasting the log text.
* A description of what was happening immediately before the bgerror occurred (if known).
* A detailed procedure for reproducing the error, if known.

### Athena Scripting

Athena provides significant support for scripting via the Athena *Executive*, a scripting language processor based on the TCL programming language.[[19]](#footnote-19)

#### Executive Commands

The Athena Executive extends the TCL programming language with a wide variety of *executive commands*; these commands can create and revise neighborhoods, actors, groups, and other simulation entities, lock and unlock the scenario, advance time, and so forth. The complete list of executive commands is given (with full documentation) in the Athena application’s on-line help; select “Executive Commands” from the Help menu.

The most important executive command is the send command, which is used to send *orders* to Athena. See Section 22 for more about Athena orders.

#### The Athena Command Line

Executive commands can be entered interactively by the user at the Athena command line. The command line is not displayed by default; to make it available, select “Command Line” from the View menu.

#### Executive Scripts

The Athena Executive is built around a TCL interpreter, to which it adds Athena's executive commands. The full capabilities of TCL as a programming language (conditionals, loops, and procedures) are available to the Athena user, who can actually write short programs for execution by the Executive. These programs are called *scripts*.

##### Internal vs. External Scripts

Prior to Athena 5, it was necessary to write scripts using an external program (e.g., NotePad) and save them on the disk as text files with a “.tcl” extension (e.g., “myscript.tcl”). The script files were then placed in the same directory as the scenario (“.adb”) file, and called from the Executive command line or by an **EXECUTIVE** tactic (see Section 17.12) using the call command. External scripts are a nuisance, as they can easily become separated from the scenario file to which they belong.

While external script files can still be used, Athena 5 added a new capability: executive scripts that are stored as part of the scenario file. These *internal scripts* can do all of the same things the external scripts can do, but are edited from within Athena itself, and cannot be accidentally separated from the scenario.

Internal scripts can be created within Athena; and external scripts can be imported into the scenario, thus becoming internal scripts. Internal scripts can be called by name using the call command just like external scripts. In addition, internal scripts can be made *auto-executing*; such a script is executed automatically when the scenario is loaded. Auto-executing scripts are useful for defining procedures, as described in the next section.

##### Procedures vs. Scripts

Prior to Athena 5, executive scripts were used as a kind of subroutine: you put a sequence of commands into a file, and invoked it as a subroutine using the call command. The TCL language has a much better feature called a *procedure*, which is used to define new commands. Once defined, these commands can be used just like any other executive command.

Procedures were little used prior to Athena 5, because they had to be defined by scripts and there was no way of making sure that any particular script would be executed before the procedure would be called. Since Athena 5, however, "auto-executing" internal scripts can be used to define procedures that will always be present when the scenario is loaded. They can then be used from the Executive command line, in **EXECUTIVE** tactics, and so forth

TCL procedures are defined using the proc command:

proc hello {} {

return "Hello, world!"

}

Procedures can take arguments:

proc setpop {pop} {

send CIVGROUP:UPDATE -g CG1 -basepop 10000

}

For example, suppose an **EXECUTIVE** tactic is to be used to inject magic attitude inputs[[20]](#footnote-20) due to a flood in some neighborhood. In Athena 4, you might have used an external script that looks something like this:

send MAD:CREATE ...

send MAD:SAT ...

send MAD:SAT ...

Now, you can import the script or create a new script in the Scripts Editor (see Section 13.16) and use it just the same. But you might also convert it to a procedure called flood:

proc flood {} {

send MAD:CREATE ...

send MAD:SAT ...

send MAD:SAT ...

}

And then you might get creative. For example, instead of creating a flood in just one particular neighborhood, you might want to be able to create a flood in any neighborhood, and have it affect whoever lives there:

proc flood {nbhood} {

send MAD:CREATE ...

foreach g [civgroup residentInN $nbhood] {

send MAD:SAT ...

send MAD:SAT ...

}

}

Then, the command executed by the EXECUTIVE tactic can simply be

flood N1

to flood neighborhood N1.

##### Resetting the Executive

On reset the Executive is restored to its pristine state, minus any variables or procedures or other changes the user might have made from the command-line; and then any auto-executing internal scripts are executed in sequence, thus defining any procedures that are meant to be part of this particular scenario.

The Executive is reset at the following times:

* When a new scenario is created (e.g., by selecting **File/New Scenario** from the menu).
* When an existing scenario is opened (e.g., by selecting **File/Open Scenario** from the menu).
* As part of preparing to run an experimental case (see Section 12.10).
* On demand, by entering the reset command at the command-line or by pressing the "Reset" button in the Scripts Editor.

#### Scenario Scripts

Although the user interface hides this from the user to a great extent, an Athena scenario is built by sending a sequence of simulation orders to Athena. These orders create and update neighborhoods, groups, and so on. A script can send orders using the send executive command, and so scripts can be used to create entire scenarios.

Athena supports the exporting of scenarios as scripts in two different ways. First, Athena can create a "replay" script, a script that simply replays the series of orders that led to its current state. Replay scripts can get quite large, especially if the current scenario was created by starting with a previously existing scenario.

The second, newer, way is to create a minimal scenario script, a script that contains the minimal set of orders required to rebuild the scenario just as it is, arranged in a logical sequence. These minimal scripts execute more quickly, and are easier to read and edit.

To export a scenario as a minimal scenario script, select File/Export Scenario As... from the main menu, or use the export command at the Athena Command line:

> export myscenario.tcl

>

This will save the scenario script as myscenario.tcl in the same directory as the scenario file. The resulting script can then be executed in the usual way to recreate the existing scenario, or edited by hand to create a slightly different scenario. This provides a convenient way to load an existing scenario into a new version of Athena when using development builds (Section 11): export it from the old version, make any necessary changes by hand, and load it into the new version.

To create a replay script, use the export -history command at the Athena Command line:

> export –history myscenario.tcl

>

Naturally, scenario scripts can also be created by hand or by other software; this can be a convenient way to load data from other databases into Athena. It is simply necessary to use the proper sequence of Athena orders. See Section 22 and the application’s on-line help for documentation on the individual orders.

### Batch Mode

Athena can be invoked in *batch mode* from the Windows command line or by other programs. In batch mode Athena has no user interface; instead it initializes itself, runs an external script, and then shuts down. Typically the script will advance time and save the scenario under another name. The results can then be acquired by querying the saved scenario file as an SQLite database.

#### Invoking Athena in Batch Mode

To invoke Athena in batch mode, use the following command line (question marks denote optional arguments):

athena.exe –batch –script myscript.tcl ?myscenario.adb?

In batch mode:

* No GUI is created.
* A scenario file is loaded, if one is given on the command line. If it includes auto-executing internal scripts, they will be executed when the scenario is opened. Thus, the batch script may use any procedures defined by these scripts.
* A script is executed if the –script option is given.
* If results are to be saved, the script must explicitly call the save command.
* Athena exits.

If Athena runs with no errors, it will terminate with exit code 0. If any error occurs, details about the error will be written to the file “error.log” in the user’s working directory, and Athena will terminate with exit code 1.

#### Simulation Control

In batch mode, the simulation is controlled by a script of executive commands specified using the -script option. Documentation for the Athena executive commands is in the Athena help; to view it, invoke Athena interactively, and select Help/Executive Commands from the menu bar.

The following executive commands will be most useful; see the Help for full details, including command syntax:

* advance: Advances simulation time.
* call: Calls another script.
* export: Exports the current scenario as a script.
* load: Loads a scenario file.
* lock: Locks the scenario so that time can advance.
* new: Creates a new, empty scenario.
* save: Saves the scenario, including any simulation results.
* send: Sends an Athena order.
* unlock: Unlocks the scenario, so that basic scenario data can be changed.

#### Simulation Results

To process simulation results, use the save executive command to save a scenario file after advancing time; this file will contain the results, along with all of the other scenario data. The results can be viewed in two ways:

* First, you can open the file in Athena and browse it interactively.
* Second, the scenario file is an SQLite database; any SQLite client can open it and execute queries. You can browse the schema by running Athena interactively and entering "my://rdb" into the Detail Browser's address bar.

### Athena Experiments

Athena 5 added a new capability to run experiments: to make a set of related runs, called *cases*, and accumulate the results into a single file, called an *Athena Experiment Database* (AXDB). Athena experiments are defined using the axdb family of executive commands, which are documented in full in the on-line help. This section describes the concepts and the basic workflow for creating and running an experiment.

#### The Base Scenario

Every experiment is built on top of a *base scenario*: a standard Athena scenario file that contains all of the baseline inputs. Before running an experiment, the analyst should create and thoroughly sanity-check the base scenario.

#### Experiment Cases and Case Parameters

An experiment consists of a (possibly very large) number of *cases*, each of which is a slight modification of the base scenario. Cases are defined by assigning values to *case parameters*.

For example, consider the capital city of Elitia, which might be destabilized by an influx of Peons from rural Peonia. The government of Elitia is considering deploying troops to keep the peace. The two relevant case parameters are the number of rural Peons in the capital city, and the number of troops deployed to keep the peace in the city.

Case parameters are translated into scenario inputs by means of *case parameter scripts*.

#### The Experiment Database

The experiment's data, including its case parameters, case parameter scripts, actual cases, and all computed results, are stored in an Athena Experiment Database (AXDB) file. Usually an executive script file is used to define and run the experiment.

#### Experimental Results and Case Outcomes

At present, the experimental results are the time series (e.g., mood) stored in Athena's hist\_\* tables. As each case is run, all of these time series are copied into the AXDB file, and tagged with the case ID. Thus, at the end of the set of runs the analyst has a single SQLite database file with all of the results, for analysis using any desired method.

In addition, each case has an "outcome", a value that indicates whether the case ran to completion or not. The outcome values are:

* **OK** — The case ran to completion with no obvious problems.
* **FAILURE** — There was a sanity check failure on-lock or at some point during the run.
* **ERROR** — There was an unexpected software error during the run.

In the latter two cases, Athena saves context in the AXDB that might help figure out what happened.

#### Workflow

This section gives an overview of how to use to the axdb family of executive commands to define and run an experiment. See the on-line help for full documentation of each command.

These steps can be done from the Athena command-line, but it is usually best to write an external executive script to create and run the experiment. Then, the AXDB can be loaded into Athena and queried interactively.

First, create and sanity check the base scenario.

Next, use axdb create to create a new experiment file. By default, it will be created in the same directory as the scenario file:

axdb create myexperiment.axdb

Next, define the case parameters. For each case parameter you need a name, a human-readable description, and a short executive script that translated parameter values into scenario inputs:

axdb parm define ppop "Population of PEONU in the CITY" {

send CIVGROUP:UPDATE -g PEONU -basepop $ppop

}

Here the parameter is called "ppop". The parameter script can access the parameter's value as a TCL variable, as shown in the send command. We might similarly define a parameter called troops, that is the number of army troops to be deployed to keep the peace. Note that case parameter scripts can make use of scenario-specific executive commands defined in the scenario's stored executive scripts; see Section 12.8.3.1.

Next, define the cases. Each case consists of values for some number of case parameters. For example, suppose we wish to vary the ppop parameter from 10,000 to 100,000 by increments of 10,000, and the number of troops from 1,000 to 5,000 by increments of 1,000. Our experiment definition script might look like this:

for {set ppop 10000} {$ppop <= 100000} {incr ppop 10000} {

for {set troops 1000} {$troops <= 5000} {incr troops 1000} {

axdb case add ppop $ppop troops $troops

}

}

There are a number of axdb commands for interactively examining the defined cases. For example, axdb parm list lists all of the defined case parameters, and axdb case list lists all of the defined cases, with their outcomes if appropriate.

Next, run the cases. You can run some or all of the cases using axdb run; for example, to run cases 10 through 19 for 52 weeks, use this command:

axdb run -start 10 -end 19 -weeks 52

Athena will run the ten cases, saving the results in the AXDB, and leaving you with the unlocked base scenario.

While preparing the base scenario or the experiment, it is often convenient to run a single case. There are two ways to do this. First, the axdb prepare command will take the base scenario and add the scenario inputs for a specific case, leaving the scenario unlocked. You can then examine the changes, and lock and run the scenario yourself. Second, the axdb runcase command will prepare the case and run it for a specified number of weeks, leaving Athena in simulation mode with the time advanced.

Once a case or cases have been run, you can examine the results from within Athena by opening the experiment file and using a variety of commands; see the on-line help for the full list. The two most useful are axdb select and axdb csvfile. The former allows you to do arbitrary SQL SELECTs on the data in the experiment database, displaying the results in tabular form; the latter is similar but saves the results to disk in CSV (comma-separated value) notation, for easy import into Excel and other analysis programs.

See my://axdb in the Detail Browser for the AXDB's SQL schema.

### Athena PBS (Portable Batch System)

Athena 5 adds a new application for invoking multiple instances of Athena in parallel on a cluster of compute nodes. This application allows users to import base scenarios along with test cases in an Athena eXperiment Database (AXDB) and run each instance with a specified number of test cases on a Red Hat Linux cluster that has the Portable Batch System (PBS) installed. The application is responsible for dispatching test cases to compute nodes based upon the number of tests to run and the number of nodes requested. The application is also responsible for coalescing results from the cases run into a single database that can then be queried to examine and export history of all test cases run.

#### Terms

Some terminology:

*head node*: the CPU on which the Athena PBS application is run. Jobs are dispatched to compute nodes from the head node. Normally, no jobs run on the head node.

*compute node*: a single CPU on which a job is run.

*cluster*: a collection of CPUs linked together with PBS. Includes head node and compute nodes.

*job*: an execution of Athena in batch mode on a compute node to run one or more test cases.

*test case*: a single test as defined in an imported AXDB.

#### Running with Athena PBS

Athena PBS is invoked by issuing the following command on the command line:

athena pbs

This command launches the Athena PBS user interface on the head node. The application is used to import an Athena (.adb) baseline scenario and Athena eXperiment Database (AXDB) containing the test cases. It is presumed that the user has done all the upfront work needed to prepare these using the Athena application to create a baseline scenario (Section 14) and test cases (Section 12.10) and is ready to invoke instances of Athena to run on the PBS cluster. Note that if this application is invoked on a system that is anything other than a Red Hat Linux system with PBS installed an appropriate message is displayed and the application halts.

After the user has imported a baseline scenario and the AXDB, the user can optionally select an output directory to be used for results from running all requested test cases. If the user selects an AXDB that contains both test cases and test case results, the results are deleted from the database and the test cases outcomes are reset. By default, the application writes results to the users home directory. After these selections have been made, the application is ready to dispatch instances of Athena to the cluster, each running the minimum number of test cases in order to minimize total run time and still run all test cases requested. Athena PBS can be dispatched to any of the queues available on the PBS cluster subject to the following constraints:

|  |  |  |
| --- | --- | --- |
| **Expected Run Time** | **Queue Name** | **Max Nodes Available** |
| Up to 3 hours | shortq | 128 |
| Up to 12 hours | mediumq | 128 |
| Up to 24 hours | longq | 64 |
| Up to 48 hours | verylongq | 64 |

After the PBS queue has been selected, Athena PBS automatically picks the largest number of nodes that results in the fewest number of test cases per node. The number of nodes and number of tests run can be changed before dispatching the test cases to the compute nodes if the default selections are not desired. This may be desirable if, for example, some other user of the cluster has already taken a chunk of nodes and choosing fewer nodes would mean the jobs would start immediately rather than waiting for the busy nodes to become free.

When the run button is pressed Athena PBS dispatches the jobs to the compute nodes based upon the selections made: number of tests, number of nodes, etc… using a PBS job array. PBS assigns the job array a unique 6-digit ID, which is appended to the name of host on which PBS is running. Then, each of the jobs in the job array has a sequential digit appended to the hostname/ID pair. For instance if the job array ID is 123456 on the host “myserver”, then job number 7 in the job array appears as

myserver.123456[7]

in the status window. In a nutshell, a job array is set of related jobs that run in parallel without any inter-job communication. Each job, although related to other jobs by a base scenario, is independent of all other jobs during job execution. A complete discussion of PBS job arrays is beyond the scope of this document.

After the requisite test cases are dispatched to the compute nodes, the Athena PBS GUI provides a status window to monitor the state of each of the jobs as they run.

#### Monitoring the Cluster

The PBS system provides commands that can be invoked at the command line to monitor the status of jobs on compute nodes. Athena PBS makes uses of these commands to monitor the jobs requested using its interface. The current status of jobs is displayed in the status window as jobs execute. The following information is provided:

|  |  |
| --- | --- |
| **Job Id** | The unique ID of the job assigned by PBS. |
| **Name** | The name of the job, limited to 16 characters - it is the first 16 characters of the AXDB selected. |
| **User** | The name of the user logged into the cluster. |
| **Time Use** | The amount of time elapsed for each job requested. |
| **State** | The state of the job, see the state table below for a description of each state. |
| **Queue** | The name of the queue requested. |

Using the stop button that looks like a red ‘X’ will stop a requested job. If a job is stopped before it is completed, no results are returned from the compute nodes. In order to get results, a job must be run to completion.

The possible states for a job:

|  |  |
| --- | --- |
| **State** | **Description** |
| B | The job array has started. |
| E | Job is exiting after having run. |
| H | Job is held. A user or administrator that has privilege does this. The job stays held until released. |
| Q | Job is queued, eligible to be run. |
| R | Job is running. |
| S | Job is suspended. This can happen if a higher priority job needs resources used by this job. |
| T | Job is in transition (Athena PBS cannot do this) |
| U | Job is suspended due to workstation becoming busy. |
| W | Job is waiting for its requested execution time to be reached. |

Note: the most common job states encountered are Q, R and E. The B state will only be seen on the first node listed in the Athena PBS status window.

#### Results

After the requested test cases run to completion, Athena PBS coalesces results into a single AXDB file on the head node that can then be queried for test results. The AXDB containing the results is written to the selected output directory, or the home directory if no output directory was specified. While results are being written the Athena PBS status window indicates from which node results are being written and how many nodes remain to be written. Depending on the size of the baseline scenario and the length of the run, the process of coalescing results can take a significant amount of time. The output AXDB filename uses the following convention: the name of the input AXDB appended with the PBS array ID. So, for example, if the input AXDB is called myexp.axdb and the PBS job array ID is 123456 then the resulting AXDB is called:

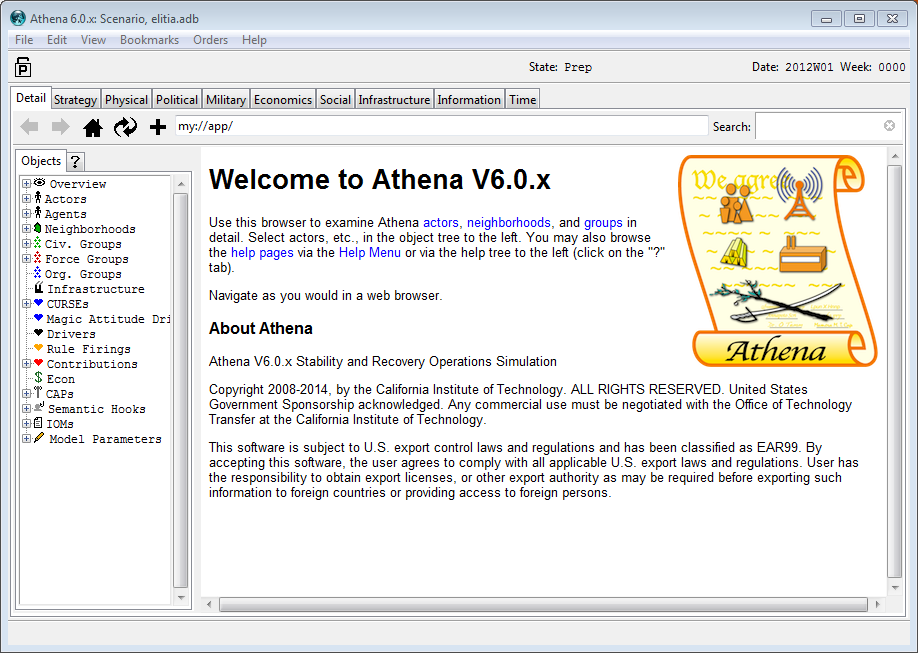
myexp\_123456.axdb

This file can be opened by the Athena GUI and results inspected. See sections 12.9.3 and 12.10.4 for more information about extracting results from an AXDB.

## The Athena User Interface

This section describes the parts of the Athena user interface, and how to make use of them.

### The Main Window



The Athena main window consists of the following components:

* A traditional menu bar (Section 13.2)
* The application toolbar, whose content changes depending on the application mode, Scenario or Simulation (Section 13.3)
* The main body of the window, which contains many tabbed data panes organized according to the PMESII-PT paradigm, most with sub-tabs of their own. These tabs are all described in detail in the on-line help; select **Help/Application Tabs** from the menu.
* The command line, where executive commands may be entered interactively. The command line is hidden by default. (Sections 13.4 and 12.8.1)
* The status line, where Athena displays transient messages.

These components are discussed in detail in the following sections.

### The Menu System

The menu items in Athena menu system are documented in detail in the on-line help. This section gives an overview of the top-level menus.

#### The File Menu

From the **File** menu you can create new scenarios, open existing scenarios, save the scenario, and so forth. In addition, you can import maps (Section 13.7.1), import and export the model parameter database (Section 14.19), export the current scenario as an order script (Section 12.8.4) and save the Command Line’s scrollback buffer to disk as a text file.

#### The Edit Menu

The **Edit** menu has the usual Undo/Redo, Cut/Copy/Paste, and Select All menu items.

The Undo/Redo capability operates on the level of individual simulation orders as executed interactively via the GUI. For example, if you use the Update Civilian Group order to change a civilian group’s base population and demeanor, you can undo the entire change. Athena maintains a deep undo stack; the stack is cleared when the scenario is saved, when the scenario is locked, and when simulation time starts to advance.

The Cut/Copy/Paste operations are available in any field in which text is edited. In addition, it is possible to select and copy text from the **Detail Browser**. In addition, it is possible to cut, copy, and paste blocks, tactics, and conditions in the Strategy Browser (Section 13.6.) It is not possible to cut, copy, or paste other items of data, e.g., groups, actors, and neighborhoods.

#### The View Menu

The **View** menu allows the user to determine which application components are visible at any given time.

**View a Specific Tab:** The first section of the **View** menu lists all of the currently visible tabs and sub-tabs. You can have Athena go directly to a particular tab by selecting the corresponding menu item.

**Optional Components:** By default the Athena command line and the **Bookmarks**, **Scripts**, **Orders**, **Scripts**, and **Log** tabs are not displayed, nor is the Executive's command line interface pane. The user can make them visible by selecting the relevant item from the bottom of the **View** menu.

#### The Bookmarks Menu

The **Bookmarks** menu allows the user to save, manage, and access bookmarks to specific **Detail Browser** pages. Athena bookmarks are similar to web browser bookmarks. Bookmarks are saved as part of the scenario, and are not shared across scenarios.

#### The Orders Menu

The **Orders** menu contains submenus and menu items for almost all of the simulation orders supported by Athena. Selecting the menu item pops up the order dialog for the given order. If an order cannot be used in the current simulation state, the related menu item will be disabled. The orders themselves are documented in the on-line help; select **Help/Orders** from the menu.

Some orders are only available from other parts of the GUI; for example, it’s necessary to add a tactic to a block from in the **Strategy Browser**. Consequently, these orders do not appear on the **Orders** menu.

#### The Wizards Menu

Wizards are tools to help automate scenario creation, usually based on externally available data. Athena's wizards are accessed via the **Wizards** menu. The wizards themselves are described in Section 13.17.

#### The Help Menu

The **Help** menu is a convenient way to access Athena’s on-line help, which is displayed in the **Detail Browser**. You can also see an outline of all of the help topics by clicking on the “?” tab in the **Detail Browser**’s sidebar.

You can browse the on-line help and use the application at the same time by selecting **File/New Detail Browser…** from the menu. This will pop up a new window containing only a **Detail Browser**; then, select the help topic from the sidebar.

### The Main Toolbar

The content of the main toolbar changes depending on whether the application is in Scenario Mode or Simulation Mode (Section 12.3).

#### The Scenario Mode Toolbar

In Scenario Mode, the main toolbar contains the following controls:

| Control | Description |
| --- | --- |
|  | **Lock Scenario Preparation.** When pressed, this button locks the scenario. All sanity checks are done, and (if the scenario is "sane") the application enters Simulation Mode. |
|  | The toolbar always shows the current simulation state. |
|  | The toolbar always shows the current simulation time. In Scenario Mode, this is time 0, the simulation time we will have when the simulation is locked. The mapping from the simulation time in integer weeks to Julian week strings can be changed by selecting **Orders/Simulation/Set Start Date…** menu item, and the initial week number can be changed from 0 by selecting **Orders/Simulation/Set Start Tick...** menu item. |

#### The Simulation Mode Toolbar

In Simulation Mode, the main toolbar contains the following controls:

| Control | Description |
| --- | --- |
|  | **Unlock Scenario Preparation.** When pressed, this button unlocks the scenario, returning to Scenario Preparation. Any simulation results are lost. |
|  | **Run Simulation.** When pressed, the simulation begins to run, advancing time week-by-week for the interval indicated in the **Duration of Run** pull-down. |
|  | **Pause Simulation.** While time is advancing, the **Run Simulation** button turns into a pause button. When pressed, the simulation will pause. |
|  | **Duration of Run.** Determines the length of the next time-step. |
|  | The toolbar always shows the current simulation state. |
|  | The toolbar always shows the current simulation time. |

### The Command Line

The Athena Command Line allows the user to enter executive commands interactively; see Section 12.8.1 for more information about executive commands. For example, the **=** command can be used as a calculator:



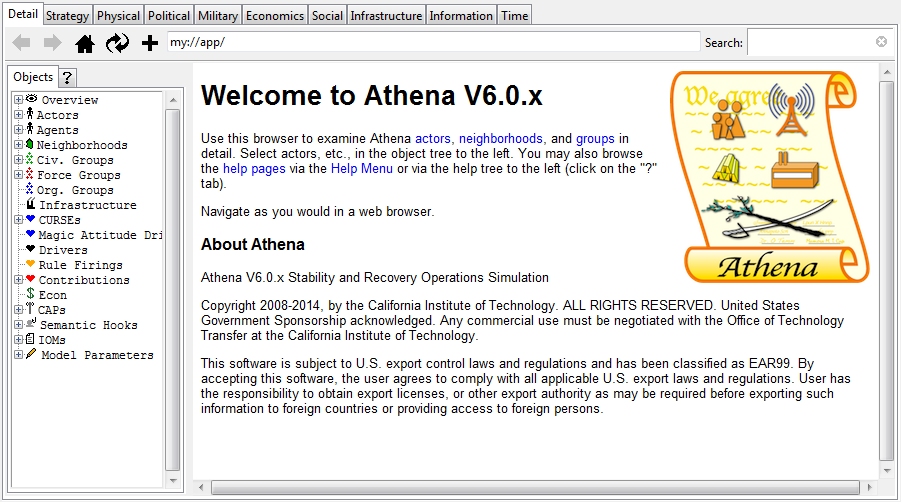
The Command Line is not visible initially; to make it visible, select **View/Command Line** from the menu.

While the cursor is in the Command Line, the up and down cursor keys can be used to step through and select previously entered commands.

By default, the Command Line retains 500 lines of output; this output can be browsed using the scrollbar, and can be saved to disk by selecting **File/Save CLI Scrollback Buffer…** from the menu.

### The Detail Browser

The **Detail Browser**, contained on the **Detail** tab in the main window, is the central part of the Athena user interface. It is the primary spot for viewing the scenario and the results.



In form, the **Detail Browser** looks much like a web browser. It displays pages of HTML-formatted data, each of which has its own URL. The user can navigate the data by clicking on links and by use of the back and forward buttons, and by entering specific URLs into the browser’s address bar. However, the **Detail Browser** does not implement true web access: it will only display data from within the application, not data from external web pages.

The **Browser** has several distinctive parts:

* The Sidebar, which provides quick links to many of the available pages.
* The toolbar, which provides the usual back, forward, home, and reload buttons, a "new bookmark" button, the address bar, and a search box.
* The data pane, in which data pages are displayed.

By selecting **File/New Detail Browser…** from the menu, the user can pop up a window containing only a **Detail Browser**; in this way it is possible to look at two pages at the same time, or to look at a page while using some other portion of the user interface.

#### Detail Browser Sidebar

The Sidebar has two tabs: the Objects tab and the “?” or Help tab. The Objects tab contains a tree of simulation objects and other topics for which pages exist. Selecting an entry in the tree displays the related page in the data pane. Note that the data available for an entry often depends on the state of the simulation.

The “?” tab includes a tree of all of the topics in the on-line help. Selecting a topic displays the help page in the browser.

#### Detail Browser Search Box

The Search Box provides an easy way to search the on-line help for information about a particular topic. Unfortunately, it does not search any of the rest of the pages available in the browser. To search for a topic, enter a string in the search box. The usual operators are available; for example, enclose a phrase in double-quotes to get an exact match.

#### Context Menu

Right-clicking on the data pane pops up the **Detail Browser**’s context menu, which contains the following items:

Save to Disk…

Saves the currently displayed page to disk as an HTML file.

View in System Web Browser…

Saves the currently displayed page to disk as an HTML file, and displays it in the system’s default web browser. This can be a convenient way to print a browser page. Note that some pages contain active application content that cannot be viewed in this way.

View Source…

Displays the HTML source in a pop-up window.

#### Useful URLs

Most of the pages of interest are included in the Sidebar. However, there are some URLs which are not, or which are otherwise useful to know.

my://app

This is the application’s home page.

my://help

This is the root of the on-line help tree.

my://rdb

This URL is the root of a tree of pages concerning the schema and content of Athena’s run-time database (RDB). If you wish to query the RDB or a scenario file directly using SQLite, this is a good place to start exploring.

Adding “/urlhelp” to any of these root URLs brings up a page that describes all of the URLs that can built from that root. For example, browsing “my://rdb/urlhelp” will tell you that you can type “my://rdb/schema/civgroups” into the address bar to browse the SQL schema of the “civgroups” table.

The URL of the current page is always displayed in the **Detail Browser**’s address bar; and when the mouse pointer is over a link, the linked URL will appear momentarily on the window’s status line.

### The Strategy Browser

The Strategy Browser is where actor strategies (Section 5.2) are created, edited, and viewed.



The browser has three major sections:

* The Agent list
* The Block list
* The Block detail pane, which has three subtabs:
  + The Block Overview tab
  + The Block Conditions tab
  + The Block Tactics tab

#### The Agent List

The Agent list includes the IDs of all defined actors plus the special **SYSTEM** agent. Select an agent from the list to see its strategy.

#### The Block List

The Block List displays a list of the strategy blocks in the selected agent's strategy, in priority order.



The pane displays the following data for each block:

| Column | Description |
| --- | --- |
| ID | The block's numeric ID. |
| Exec | The block's execution status icon (see below) |
| State | The block's state, one of **normal**, **disabled**, or **invalid**. A block is invalid if its sanity check has failed; all sanity check failures must be fixed before the scenario can be locked. A block can be disabled by the user; disabled blocks are ignored during simulation. (This is one way to temporarily fix a sanity check failure.) Otherwise, the block's state is normal. |
| Intent | The intent of this block, as entered by the user, i.e., "Normal troop deployments". Athena uses this field for display only. |
| At Time | The block's time constraint: during what time period is the block eligible to be executed? |
| On Lock? | Whether the block is to be executed on scenario lock, or not. |
| Once? | Whether the block is to be automatically disabled after its first successful execution, or not. |
| Last | The time at which the block last executed. |

The block's execution status icon will be one of the following after scenario lock:

| Icon | Description |
| --- | --- |
|  | The block was skipped over during execution for some non-model-related reason, it is disabled, it contains no tactics, or something of this sort. |
|  | The block's time constraints were not met. |
|  | The block's conditions were not met. |
|  | The actor had insufficient cash to execute all of the block's tactics. |
|  | The actor had insufficient personnel to execute all of the block's tactics. |
|  | The actor had insufficient of some other asset to execute all of the block's tactics. |
|  | An error occurred while attempting to obligate resources for the tactic. |
|  | The block executed successfully, but some tactic failed on execution because access was not granted to a CAP. (Section 9.1.2) |
|  | The block executed successfully. |

To add or delete blocks, and edit block metadata, use the Block List's controls:

| Control | Description |
| --- | --- |
|  | **Add Block.** Adds a new block and pops up the Update Block dialog. |
|  | **Edit.** Pops up the Update Block dialog. |
|  | **Toggle Block State.** Enables or disables the currently selected block. See the description of the "State" column, above. |
|  | **Set Priority.** These buttons are used to adjust a block's priority relative to the agent’s other blocks. They move the block to the top, up one slot, down one slot, and to the bottom, respectively. |
|  | **Sanity Check.** Forces a sanity check of all strategies. Tactics and conditions are guaranteed to be “sane” when they are created; but if groups or other entities are deleted after a tactic or condition that references them has been created, the tactic or condition is now in error. This is also checked automatically when the scenario is locked. |
|  | **Delete.** Deletes the currently selected block. |
|  | Sets or clears the selected block's "On Lock" flag. (See above.) |
|  | Sets or clears the selected block's "Once" flag. (See above.) |
|  | Pops up the Update Block dialog, so that the block's time constraints can be edited. |

#### The Block Overview Tab

This tab displays the content of the tab in narrative form, including the effects of all block metadata, the block's conditions, and the block's tactics. Any sanity check failures will be detailed here.

Look here to understand a block and its status.

#### The Block Condition Tab

The Block Condition tab lists all conditions attached to the selected block:



The pane displays, for each condition:

| Column | Description |
| --- | --- |
| ID | The condition's numeric ID. |
| Flag | The condition's flag icon (see below). |
| State | The condition's state, one of **normal**, **disabled**, or **invalid**. A condition is invalid if its sanity check has failed; all sanity check failures must be fixed before the scenario can be locked. A condition can be disabled by the user; disabled conditions are ignored during simulation. (This is one way to temporarily fix a sanity check failure.) Otherwise, the condition's state is normal. |
| Narrative | The condition's meaning in human-readable prose. |
| Type | The condition's type name. |

The flag status icons are as follows:

| Icon | Description |
| --- | --- |
|  | The condition was not assessed at the last strategy execution. |
|  | The condition was not met when the block was last assessed. |
|  | The condition was met when the strategy was last assessed*.* |

To create and edit conditions, use the controls on the Block Condition tab’s toolbar:

| Control | Description |
| --- | --- |
|  | Set the condition mode: "All of" or "Any of". |
|  | **Add Condition.** Pops up the Create Condition menu. Select the condition type from the menu to create a new condition. Once the condition is created, Athena pops up the Update Condition dialog. |
|  | **Edit.** Pops up the appropriate dialog for editing the currently selected condition. |
|  | **Toggle Condition State.** Enables or disables the currently selected condition. When a condition is disabled, it is as though it doesn’t exist. |
|  | **Delete.** Deletes the currently selected condition. |

#### The Block Tactic Tab

The Block Tactic tab shows the block's tactics in priority order, highest first.



The pane displays, for each tactic:

| Column | Description |
| --- | --- |
| ID | The tactic's numeric ID. |
| Exec | The tactic's execution status icon. Tactic and block status icons are the same; see Section 13.6.2. |
| State | The block's state, one of **normal**, **disabled**, or **invalid**. A block is invalid if its sanity check has failed; all sanity check failures must be fixed before the scenario can be locked. A block can be disabled by the user; disabled blocks are ignored during simulation. (This is one way to temporarily fix a sanity check failure.) Otherwise, the block's state is normal. |
| Narrative | The block's meaning in human-readable prose. |
| Type | The block's type name. |

To create and edit tactics, use the controls on the tab's toolbar:

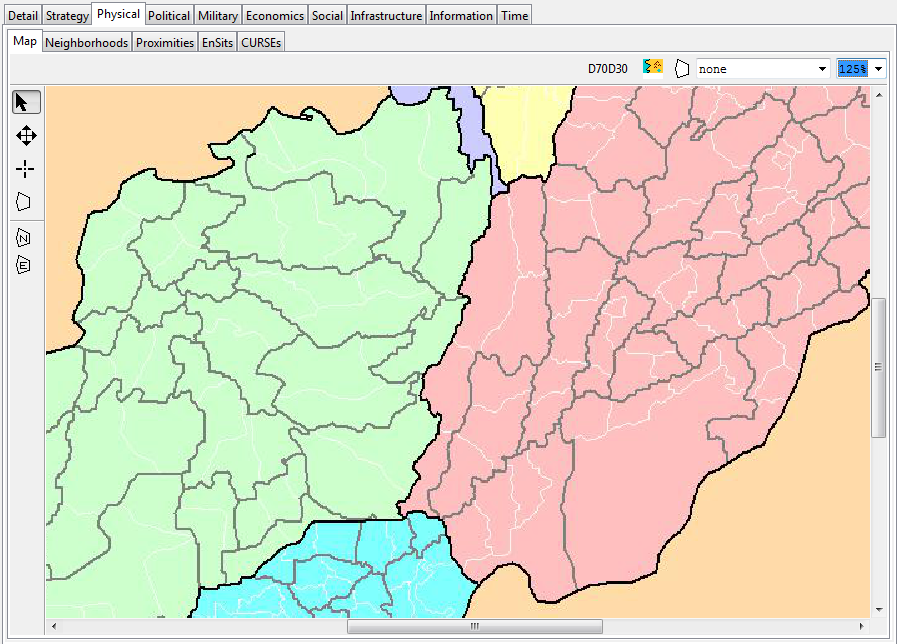
| Control | Description |
| --- | --- |
|  | Select the tactic execution mode from the pull down. If the mode is "**All tactics or none**" then the block will execute only if there are sufficient resources for all tactics in the block. If the mode is "**As many tactics as possible**" then the block's tactics will execute in priority order, until all tactics for which resources are available have executed. |
|  | **Add Tactic.** Pops up the Create Tactic menu. Select the desired tactic type to add to the block. Once the new tactic is created, the Update Tactic dialog is displayed. |
|  | **Edit.** Pops up the appropriate dialog for editing the currently selected tactic. |
|  | **Toggle Tactic State.** Enables or disables the currently selected tactic. When a tactic is disabled, it is as though it doesn’t exist. |
|  | **Set Priority.** These buttons are used to adjust a tactic’s priority relative to the block's other tactics. They move the tactic to the top, up one slot, down one slot, and to the bottom, respectively. Tactic priority only matters when the tactic mode is "**As many tactics as possible**". |
|  | **Delete.** Deletes the currently selected tactic. |

### The Physical Tab

The Physical tab has a number of sub-tabs that describe the physical environment: the map of the region, the neighborhoods into which it is divided, the proximities of those neighborhoods to one another, and so forth.

#### The Physical/Map Tab

The Map Browser, contained on the **Physical/Map** tab, is a tool for visualizing the playbox. It displays neighborhoods, units, and abstract situations; and the neighborhood polygons can be colored to display various neighborhood display variables.



##### The Map Proper

The main part of the Map Browser displays the actual map. By default, this is simply a blank canvas on which neighborhoods can be drawn. Alternatively, any desired map image can be imported as a map background by selecting **File/Import Map…** from the menu.

##### Map Reference Strings

Because Athena is designed to use any map image the analyst may have on hand, from a high-quality scan of a printed map to a snapshot of a map hand-drawn on a piece of paper, it cannot assume that maps are properly geo-referenced, and so cannot use geographic coordinates or grid reference strings.

Instead, Athena uses (X,Y) coordinates in the range 0 to 999 for both axes. The origin is at the upper left; positive X runs from left to right and positive Y from top to bottom. These coordinates are then mapped to the pixel coordinates of the map, with the map’s longer dimension mapping to the range 0 to 999 and its shorter dimension mapping to a proportionally smaller range. For example, the lower right corner of a map 1000 pixels wide by 500 pixels high would have numeric map coordinates (999,499).

These numeric coordinates are used internally; for input and output they are converted into *map reference strings*. A map reference string is a six-character string that represents a point on the *scenario*'s map. It has the form ***AnnAnn*** where *A* is a letter from "A" to "K" (excluding "I") and *n* is a digit, "0" to "9". Thus, the location (123,456) has reference string "B23E56".

##### Neighborhood Polygons

As part of creating a neighborhood, the user must supply a neighborhood polygon. The steps are as follows:

1. Go to the **Map** tab
2. Click on the **Create Neighborhood** tool, .
3. Enter the neighborhood information into the fields in the Create Neighborhood dialog.
4. Select the Reference Point field, then click on the neighborhood’s reference point on the map. The map reference string will appear in the field.
5. Select the Polygon field, and then draw the polygon on the map:
   1. The first time you click on the map you will start to draw a line.
   2. Click on the vertices of the desired polygon, proceeding around the edge of the neighborhood.
   3. The mouse pointer will automatically snap to the vertices of existing neighborhoods. This makes easy to create neighborhoods with shared borders.
   4. At the end, either click on the first point again, or simply double-click on the last point.
   5. The polygon coordinates will appear in the field.
6. To change a reference point or polygon, simply repeat the steps given above.

##### Unit and Abstract Situation Icons

The Map Browser displays unit and abstract situation icons. To see more information about an icon, select the Browse Tool, , and click on the icon. Information will appear on the status line.

Abstract situation icons indicate the state of the situation. If the situation has not yet had its attitude effects assessed (i.e., it is brand new, and time has not advanced), it will display with red text on a white background. If it has been assessed and is not resolved, it will display with red text on a yellow background. If it has been resolved, it will display with green text on a white background.

To move an icon, click-and-drag it to its new location. Unit icons can only be moved within their neighborhood. Abstract situation icons can be moved between neighborhoods provided that they have not yet been assessed; after that, they must remain in the same neighborhood.

##### Map Browser: Main Toolbar

The main toolbar controls determine how the map data is displayed:

| Control | Description |
| --- | --- |
|  | The map reference string under the mouse pointer. |
|  | **Extended Scroll Region.** Most map images are rectangular; Athena’s map coordinates describe a square region. By default the Map Browser displays only the map proper; if this button is latched, the Map Browser will scroll to the addressed region that is off the bottom or right side of the map image. This can be a useful place to draw remote neighborhoods, i.e., staging areas or communities in other parts of the world. |
|  | **Opaque Fill.** By default, neighborhood polygons are transparent. If this button is latched, they are filled with some color. The color can simply be white, or it can vary across the neighborhoods to visualize some neighborhood display variable. The variable to use is selected from the pull-down.  As shown, the neighborhoods will be colored according to the neighborhood mood, with positive satisfactions ranging from white to bright green and negative satisfactions ranging from white to bright red. |
|  | **Zoom factor.** Zoom in and out on the map by selecting the desired zoom factor. |

##### The Secondary Toolbar

The secondary map toolbar runs down the left side of the map, and contains controls for interacting with the map and its contents. The controls are as follows:

| Control | Description |
| --- | --- |
|  | **Browse Tool.** Use this to click on items on the map. |
|  | **Pan Tool.** With this tool selected, the user can scroll the map by clicking and dragging. |
|  | **Point Tool.** Use this tool to select map coordinates for entry into order dialogs. *Note: It is rarely necessary to select this tool explicitly, as Athena selects it automatically when the user edits a field requiring a map coordinate to be entered.* |
|  | **Draw Polygon Tool.** Use this tool to draw neighborhood polygons and enter their coordinates into order dialogs. *Note: It is rarely necessary to select this tool explicitly, as Athena selects it automatically when the user edits a field requiring a map coordinate to be entered.*  See Section 13.7.1.3 for information on how to draw neighborhood polygons. |
|  | **Create Neighborhood.** Pops up the Create Neighborhood dialog. |
|  | **Create Abstract Situation.** Pops up the Create Abstract Situation dialog. |

##### Context Menus

Neighborhood polygons and icons have context menus; access them by right-clicking the mouse on the polygon or icon.

The neighborhood context menu has the following options:

Browser Neighborhood Detail

Opens the neighborhood's detail page in the Detail Browser.

Create Abstract Situation

Pops up the Create Abstract Situation dialog with the right-clicked neighborhood filled in.

Bring Neighborhood to Front

Brings the polygon to the top of the neighborhood stacking order.

Send Neighborhood to Back

Sends the polygon to the back of the neighborhood stacking order.

The unit context menu has the following options:

Browse Group Detail

Opens the unit's group's detail page in the Detail Browser.

The abstract situation context menu has the following options:

Update Situation

Pops up the Update Abstract Situation order. This allows the user to modify an abstract situation, provided that it has not yet been assessed.

#### The Physical/Neighborhoods Tab

The **Physical/Neighborhoods** tab lists the neighborhoods in the playbox.



#### The Physical/Proximities Tab

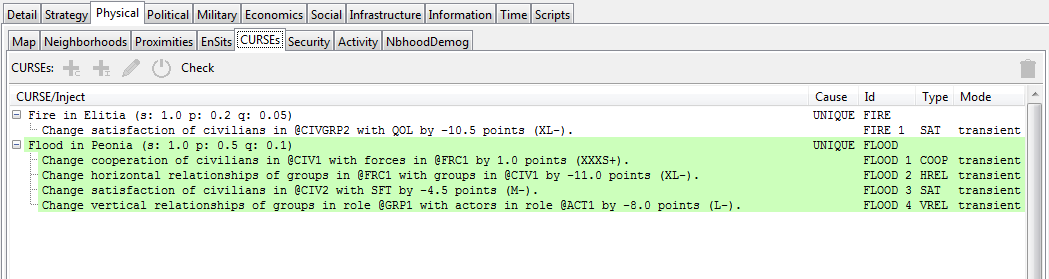
This tab states the proximity between pairs of neighborhoods. Note that proximity is a social measure as much as a physical measure; consequently, the proximity of neighborhood A with neighborhood B might be different than the proximity of neighborhood B with neighborhood A.

#### The Physical/Absits Tab

This tab displays the existing abstract situations, with their attributes. There are controls to create, edit, resolve, and delete situations.

#### The Physical/CURSEs Tab

This tab displays the attributes of all currently defined CURSEs and allows them and their associated injects to be created and edited. Through the use of “roles”, CURSEs are flexible way for the analyst to model the effects of some event or series of events that aren’t already explicitly modeled by Athena. See Section 8.8 for more on the use of CURSEs and roles.



#### The Physical/Security Tab

**Simulation Mode Only.** This tab displays the security of each group in each neighborhood, along with the components of security in the security computation. See Section 4.6.

#### The Physical/Activity Tab

**Simulation Mode Only.** This tab displays a list of all group activities being done in the neighborhoods of the playbox, including the coverage fractions, the effective personnel, and the resulting activity situation types. See Section 4.8.

#### The Physical/NbhoodDemog Tab

**Simulation Mode Only.** This tab displays the population statistics for each neighborhood, along with whether or not the neighborhood is part of the local economy.

### The Political Tab

The **Political** tab contains data relating to the Political part of the model: the actors, the relationships of the groups in the neighborhood with them, their resulting support and influence, and neighborhood control.

The **Strategy** tab (Section 13.6) can also be regarded as part of the Political area, but it

#### The Political/Actors Tab

This tab displays the actors defined in the playbox, with controls to create, edit, and delete them.

#### The Political/VRel Tab

This tab displays the vertical relationships between the groups and the actors in the playbox as computed from their beliefs. These computed values can be overridden in Scenario Prep Mode. In Simulation Mode, the tab displays them as they change over time.

### The Military Tab

The Military tab contains data relating to the use of military and paramilitary force in the playbox.

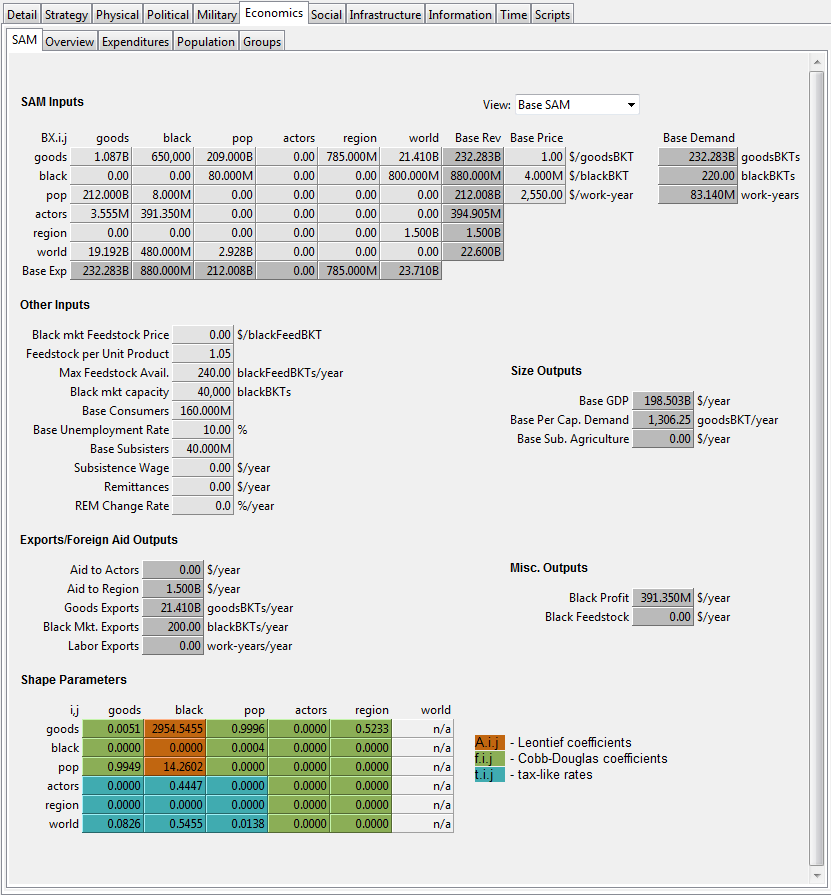
#### The Military/Force Groups Tab

This tab displays the force groups defined in the playbox, with controls to create, edit, and delete them. Force groups are deployed to neighborhoods and assigned to activities by the strategies of their owning actors.

### The Economics Tab

The **Economics** tab has several sub-tabs which display the economics of the playbox in various ways. The **Economics/SAM** tab is visible in both Simulation and Scenario Mode, the others are only visible in Simulation Mode.

Most of the sub-tabs have the same form: a tabular display of data. For example, the **Economics/Population** tab shows economic demographics by neighborhood. In this figure, however, a spreadsheet like tableau is shown:



For tabs that display rows of tabular data, note the filter box at the right-hand end of the toolbar. Type any string into the filter box to display only the matching rows. Click on the “sieve” icon in the filter box to pull down the filter options menu.

The on-line help goes into great detail about the data columns listed on each of the sub-tabs; select **Help/Application Tabs** from the menu, and drill down. This section will give a brief description of each sub-tab.

#### The Economics/SAM Tab

Displays the Social Accounting Matrix used to set up the money flows for the economy to be simulated.

#### The Economics/Overview Tab

This tab displays an overview of the playbox economy in the form of a spreadsheet.

#### The Economics/Expenditures Tab

This tab displays the expenditures of each actor by sector.

#### The Economics/Population Tab

This tab displays the neighborhood populations broken down into economic categories, along with unemployment statistics.

#### The Economics/Groups Tab

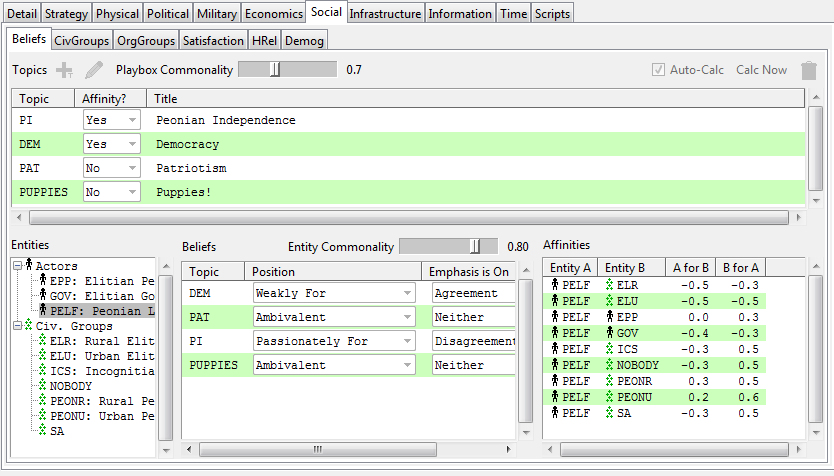
This tab displays the group populations broken down into economic categories, along with unemployment statistics.

### The Social Tab

The Social tab contains data relating to the residents of the playbox, their beliefs, their attitudes, and their demographics.

#### The Social/Beliefs Tab

The Belief System browser is found on the **Social/Beliefs** tab. It is used to create and edit the belief systems of the actors and civilian groups in the playbox (Section 8.3). This browser differs from most of the others in that you can change input values directly in the browser, without popping up a dialog box.



The browser has four major areas:

* The Topics Pane
* The Entities Tree
* The Beliefs Pane
* The Affinities Pane

Note that you can click and drag the border between any pair of these areas to make an area larger or smaller.

##### The Topics Pane

The Topics pane is where the user creates, edits, and views the set of topics about which actors and civilian groups have beliefs.



For each topic, the pane shows the topic ID, whether or not the topic is relevant, and the topic’s title. The relevance flag is simply a convenient way of enabling or disabling a topic while assembling belief systems, so as to see its effect. To edit a topic’s title, click on the title.

The Topic pane has the following controls on its toolbar:

| Control | Description |
| --- | --- |
|  | **Add Topic.** Pops up the Create Topic dialog. |
|  | **Edit Topic.** Pops up the Update Topic dialog, allowing the user to edit this topic. |
|  | **Playbox Commonality.** Allows the user to adjust the playbox commonality. |
|  | **Auto-Calc.** If the box is checked, Athena will automatically compute all affinities any time any part of a belief system is changed. For large scenarios, this quickly becomes annoying. |
|  | **Calculate Now.** Forces a recalculation of affinities. Use this after a collection of changes when Auto-Calc is unchecked. |
|  | **Delete Topic.** Deletes the currently selected topic. |

##### The Entities Tree

The Entities tree displays a tree of the actors and civilian groups in the scenario. Select an actor or group to see its belief system in the Beliefs pane and its affinities in the Affinities pane.



##### The Beliefs Pane

The Beliefs pane is where the user actually sets a particular actor or group’s beliefs.



It lists the topics in the same order as the Topics pane, and allows the user to select the entity’s position and emphasis on each by clicking on the pull-downs. The entity’s commonality with the playbox can be adjusted using the Entity Commonality slider on the pane’s toolbar.

##### The Affinities Pane

The Affinities pane shows the affinities of the entity selected in the Entities tree, both its affinities for the other entities, and their affinities for it. As shown, you can sort on any column by clicking on the column header.



#### The Social/CivGroups Tab

This tab displays the civilian groups defined in the playbox, with controls to create, edit, and delete them.

#### The Social/OrgGroups Tab

This tab displays the organization groups defined in the playbox, with controls to create, edit, and delete them.

#### The Social/Satisfaction Tab

This tab displays the baseline satisfaction levels and saliency for all groups and concerns, with controls to adjust or edit them. In Simulation Mode, it also displays the current satisfaction levels as they change over time.

#### The Social/HRel Tab

This tab displays the horizontal relationships (HRel) between the various pairs of groups in the playbox as computed from their beliefs. These computed values can be overridden in Scenario Prep Mode. In Simulation Mode it displays them as they change over time.

#### The Social/Demog Tab

This tab displays, for each group, the neighborhood in which the group resides, its base population as of time 0, its current population in the neighborhood, and the number of people killed in the simulation to date.

### The Infrastructure Tab

The Infrastructure tab shows data relating to various kinds of infrastructure in the playbox.

#### The Infrastructure/CAPs Tab

This tab shows the Communications Asset Packages (CAPs) currently present in the playbox:



It consists of three panes. The top pane lists the CAPs, along with their owning actors, current capacity, and message broadcast cost. The lower left pane shows the coverage area of each CAP in terms of a coverage fraction for each neighborhood; the lower right pane shows the penetration of each CAP into each civilian group, along with the final group coverage computed from the penetration, neighborhood coverage, and current capacity. See Section 9.1 for details.

**Note:** By default, the lower panes show only the neighborhoods with non-zero coverage and the groups with non-zero penetration. To add more neighborhoods use the "View:" pull down to select "All" rather than non-zero, and then give the desired neighborhoods non-zero coverage values. Groups can be added in the same way.

#### The Infrastructure/GOODS Plants Tab

This tab shows the ownership of GOODS production plants by the various agents in the playbox (See Section 9.2). If no ownership is specified for a neighborhood, the SYSTEM agent will own all GOODS production plants in that neighborhood.



### The Information Tab

The Information Tab displays data related to the flow and use of information in the playbox.

#### The Information/Cooperation Tab

Cooperation is a measure of the willingness of a civilian group to give information to a force group (see Section 10.1). This tab displays the initial cooperation levels for all civilian/force group pairs, with controls to adjust or edit them. In Simulation Mode, it displays the current cooperation levels as they change over time.

#### The Information/NbhoodCoop Tab

**Simulation Mode Only.** This tab displays the cooperation of each neighborhood with each force group, at time 0 and at the current time. This is simply the average cooperation across the groups in the neighborhood.

#### The Information/Semantic Hooks Tab

This tab displays the currently defined semantic hooks. A semantic hook is an appeal to the beliefs of a target civilian group; Information Operations Messages (IOMs) use semantic hooks to catch an audience:



For each hook, it displays the positions the hook takes on different belief system topics. See Section 10.2.2 for a discussion of semantic hooks.

#### The Information/Info Ops Messages Tab

This tab lists each of the currently defined IOMs, along with its payloads:



Use the controls to create and edit IOMs and their payloads. See Section 10.2 for more about IOMs.

### The Time Tab

This tab displays data relating to the progress of the simulation over time, i.e., the history of the simulation run. (Scheduling of future events is done via the actor's strategies; see Sections 5.2 and 13.6.)

#### The Time/Significant Events Tab

As Athena runs, it keeps a log "significant events", e.g., what tactics executed at which time. This tab displays this log, ordered from most recent to least recent. The user can also select a time range of interest.

Note that the Significant Events log can also be seen in the Detail Browser; in particular, each Actor, Neighborhood, and Group page in the Detail Browser displays the most recent events relating to the given entity.

#### The Time/Orders Tab

This tab displays the simulation orders sent by the user. This tab is usually hidden; to reveal it, select **View/Order History** from the menu.



Note the filter box at the right-hand end of the toolbar. Type any string into the filter box to display only the matching rows. Click on the “sieve” icon in the filter box to pull down the filter options menu.

The on-line help goes into great detail about the data columns listed on each of the sub-tabs; select **Help/Application Tabs** from the menu, and drill down. This section will give a brief description of each sub-tab.

This browser lists all of the orders sent interactively by the user—orders that might, in principle, be undone by selecting **Edit/Undo** from the menu. This set of orders can be exported as an executive script using the **export -history** command; this is a convenient way to capture the scenario input in script form.

#### The Time/Log Tab

In addition to the Significant Events log, Athena keeps a detailed, low-level debugging log. Because this information is primarily of use to the developer, the **Time/Log** tab is not displayed by default. To view it, select **View/Scrolling Log** from the menu.



The log browser is a powerful tool for searching and filtering logs so as to find the data you need; however, it is rarely needed by the end users. See the on-line help for specifics.

### The Bookmarks Tab

The Detail Browser (Section 13.5) displays Athena data using a web-browser-like interface, and as with a web browser the user can create bookmarks to important pages. The Bookmarks Tab displays the currently defined bookmarks, and allows the user to edit and reorder them. The Bookmarks Tab is hidden by default; to display it, select **View/Bookmarks** from the menu.



### The Scripts Tab

The **Scripts** tab contains the Scripts Editor, which is used to edit and run internally stored executive scripts. Scripts can be created, imported from disk, exported to disk, renamed, and executed. Stored scripts are part of the scenario, and are loaded and executed automatically by Athena. They are usually used to define TCL procedures (like the flood procedure shown here) for use in EXECUTIVE tactics, from the Executive's command line, and so on.

The Scripts tab is not displayed by default; select **View/Scripts** Editor from the menu to display it.



See Section 12.8 for more about stored scripts, and see the on-line help for details on the Script Editor's command.

**Note on saving scripts:** The "Save" button on the Script Editor's tool bar saves an edited script "into the scenario", i.e., your edited script becomes part of the scenario and will be saved and loaded with all of the rest of the scenario. It is still necessary to use the **File/Save** menu item to save the scenario as a whole. To put it another way, pressing the Script Editor's "Save" button is like pressing the "Send" button in the "Update Civilian Group" dialog. The desired change is made to the scenario, but the scenario still needs to be saved.

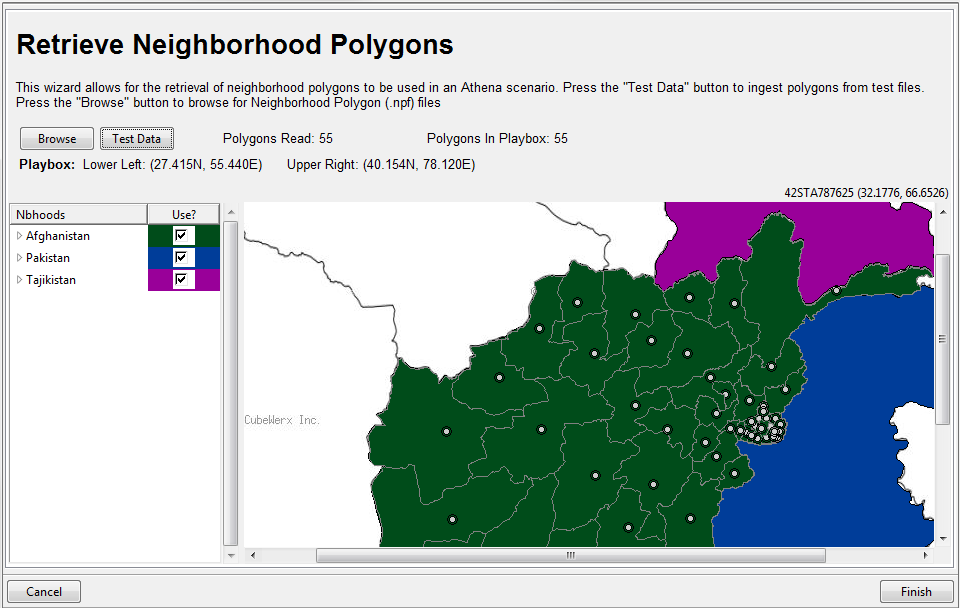
### Athena Wizards

Athena's "wizards" are tools to automate some part of scenario creation. Each wizard is a pop-up window that allows the user to load externally available data, massage it, and then add it to the scenario. This may involve sending dozens or even hundreds of orders on the user's behalf.

Athena 6.2 includes two wizards, the Neighborhood Ingestion Wizard and the Intel Ingestion Wizard. Both are found on the **Wizards** menu.

#### The Neighborhood Ingestion Wizard

The Neighborhood Ingestion wizard is used to retrieve Keyhole Markup Language (KML) polygon files from the local file system and to choose which of those polygons should be used in an Athena scenario as neighborhoods. The wizard assumes that a geo-registered map has already been imported into Athena and then uses that map as the background on which to lay down the polygons. If a geo-registered map has not yet been imported into Athena, the wizard will not launch and a message is displayed.

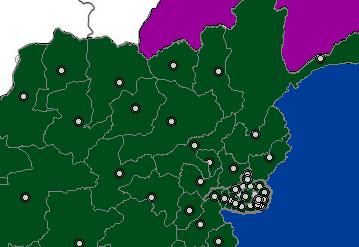


For this version of Athena, a JPL developed file format has been introduced that specifies hierarchical relationships between polygons. This is because KML in and of itself does not contain these kinds of semantics. Through the use of this file, a user can specify, for example, that certain polygons are provinces of a country polygon and are, therefore, children of that polygon. The wizard uses these relationships to display the ingested polygons in a hierarchical tree.

The “Browse” button can be used to browse the local disk for a Neighborhood Polygon File (.npf) to ingest the KML polygons for display. In this version, a “Test Data” button is included that will ingest canned polygons for Pakistan, Tajikistan and Afghanistan.

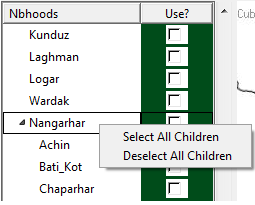
After the polygons are ingested, the polygon names are displayed on the left side of the wizard in a tree according to the parent-child relationships specified in the .npf file. The polygons themselves are displayed on the map on the right side of the wizard. The wizard also shows how many polygons were read and how many are actually inside the bounds of the playbox defined by the map.

By default, top level polygons in the hierarchy are automatically selected for output to Athena. Child polygons are displayed in the tree without the “Use?” checkbox selected and on the map with grey polygon boundaries and reference points. Selecting and deselecting children is done by either clicking the checkbox in the “Use?” column of the tree or by right clicking inside a polygon on the map.



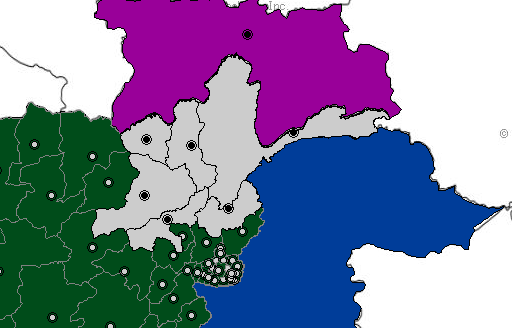
***Deselected Children with Grey Polygons***

For polygons that have children, right clicking on the polygon name in the tree pops up a menu that the user can use to select or deselect all children of that polygon. **Note**: Polygons that have all their children selected for output cannot be selected because the wizard assumes that the children completely cover the area of the Earth occupied by the parent and, thus, makes the parent redundant in the context of an Athena scenario.



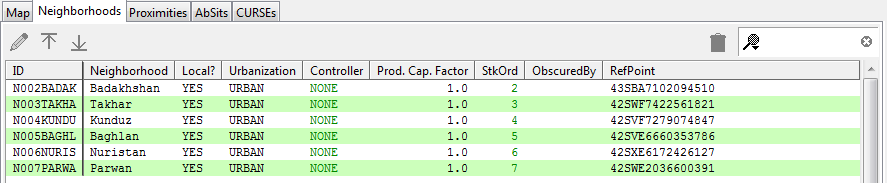
***Parent Polygon Popup Menu***

If a polygon read in by the wizard already exists in the Athena scenario it will show up in the wizard as light grey and cannot be deselected. It must be explicitly deleted by the user in order for it to appear in the wizard as a selectable polygon.



***Grey Polygons Already Existing in Scenario***

Once the user has completed choosing which polygons should be included as neighborhoods in a scenario, pressing the “Finish” button will cause the wizard to create the neighborhoods in Athena. The polygon name in the tree is used as the neighborhood long name in Athena and the short name (or neighborhood ID) is generated from the long name by appending the first few letters of the long name to an automatically generated ID of the form ‘Nnnn’ where ‘N’ is the literal capital N and nnn is a three digit number based on the short name of neighborhoods created from a prior use of the wizard, if they exist. The creation of neighborhoods can be undone and redone in the normal way if a mistake is made. Neighborhood data can be edited in Athena if the defaults used by the wizard are not appropriate and the wizard can be used multiple times as a scenario evolves.



***Neighborhoods Created by the Wizard***

#### The Intel Ingestion Wizard

TBD.

## Creating an Athena Scenario

This section proceeds step-by-step through the process of creating an Athena scenario. In particular, it is intended to identify all of the things to keep in mind in order to create a high-quality scenario. Section 25 contains this same process in the form of a check-list; it might be useful to print out this check-list when putting together new scenarios.

The steps need not necessarily be done in the order given here; for example, this section has the user create all neighborhoods before creating any groups, whereas one can certainly go back and forth.

**Note:** All of this work must be done while the scenario is unlocked.

### The Actors

The first step is usually to choose the actors in the playbox. Deciding who the actors are, and which ones you need in the scenario, requires a deep knowledge of the region of interest and is beyond the scope of this document. Here are some things to remember:

* An actor can represent a particular individual, or personify some larger group or entity. In Afghanistan, Hamid Karzai might be an actor. The Afghan government might be an actor. The USA might be an actor.
* Actors are essentially decision makers; they do not, by themselves, occupy neighborhoods or feed the hungry. They need boots (or shoes) on the ground to implement their decision. That is, they need to own force and organization groups who provide them with personnel.

To create an actor, go to the **Political/Actors** tab, and press the Add Actor button: . Enter the actor’s ID and attributes in the resulting dialog. See Section 16.3 or the on-line help for a description of the actor's attributes.

The primary decisions to make here concern the actor’s financial resources and the actor’s default use of his political support.

#### Sources of Actor Income

Actors get most of their income in the form of taxes or tax-like fees (tools, baksheesh), as a percentage of the revenue of the various economic sectors. They can also receive a share of black market profits. The "income" parameters determine the actor's income at time 0; thereafter, the actor's income will float as the economy grows and shrinks.

#### Use of Political Support

The actor's *support* attribute determines how the actor will use his political support by default. Does he seek control for himself? Does he work to support some other actor? Or does he remain aloof from the political process? The **SUPPORT** tactic is used to customize this basic position for particular neighborhoods. For example, a local actor might be concerned to control his home neighborhood, but have no political goals elsewhere. This actor might choose to support no one in general, and himself in his own neighborhood.

### The Map

The next step is usually to choose a map of the desired region. It isn’t necessary to use a map, as it is used purely for visualization and has no effect on the simulation results; instead, you can draw your neighborhoods on a blank canvas instead. The choice is yours.

If you choose to use one, your map should be stored as a PNG, GIF, or JPEG image file, saved at the resolution you’d usually prefer to look at. Simpler maps are generally more pleasant to use than highly detailed maps.

To use a map, select **File/Import Map …** from the Athena menu, and select the map file in the resulting dialog box. The imported map is saved in your scenario file; you do not need to keep the map file available once it has been imported.

### The Neighborhoods

Next, define your neighborhoods, remembering that neighborhoods are simply conveniently-sized patches of ground, where “conveniently” may vary from scenario to scenario. If your playbox should encompass a single city, the neighborhoods might be real neighborhoods, in the normal sense of the word. If your playbox encompasses an entire country, the neighborhoods might be districts or provinces.

The important thing is to put neighborhoods only where you need them. If you are concerned primarily with the north of the country, you do not need to model the south of the country in as much detail—or, possibly, at all.

In general, Athena is more concerned about neighborhood population than about the geographical size of a neighborhood. However, the size of your neighborhoods will have an effect on how you set the neighborhood proximities (Section 4.1.2).

To define a neighborhood, go to the **Map** tab and click the Create Neighborhood icon (). The **Create Neighborhood** dialog will appear. See Section 16.2 for a description of the neighborhood attributes, or click the question mark icon in the dialog for the on-line help.

To enter the reference point, click in the field on the dialog and then click on the point on the map; the map reference string will appear in the field. Similarly, to enter the polygon click in the Polygon field and click on the map; you will begin drawing the polygon. When the polygon is complete, the vertices will appear in the field.

To redraw a polygon, just repeat the process.

When you have the neighborhood defined as you like, press **Send & Close**.

The **Physical/Neighborhoods** tab lists all of the neighborhoods and their attributes. (Section 16.2.)

### Neighborhood Proximities

Athena needs to know how close together or how far apart neighborhoods are, from a social rather than geographic point of view. This is called the *proximity* of one neighborhood to another, and it is defined qualitatively using the values **HERE**, **NEAR**, **FAR**, and **REMOTE**. Note that it need not be symmetric: downtown might be **NEAR** to the people in the suburb, but the suburb might be **FAR** to the people who live downtown.

Proximity is used in two ways.

First, proximity governs the spread of satisfaction and cooperation effects across the playbox. When civilian casualties occur in neighborhood N1, for example, there may be effects on the people in N2 as well. These effects depend primarily on the horizontal relationship between the pairs of groups, but also on proximity: violence next door affects my sense of safety more than does violence across the country. Thus, indirect effects are highest **HERE**, in the same neighborhood; are lower in **NEAR** neighborhoods, lower still in **FAR** neighborhoods, and negligible in **REMOTE** neighborhoods.[[21]](#footnote-21)

Second, proximity plays into the Security model (Section 4.6): groups can call on their friends in **NEAR** neighborhoods.

Neighborhood size should affect your proximity settings as follows:

* If your neighborhoods are small, then neighborhood will more often be **NEAR** each other.
* If your neighborhoods are large, then neighborhoods will more often be **FAR** or **REMOTE**; few will be **NEAR**.

In addition, if most of your neighborhoods are large you might wish to set the force.proximity model parameter to 0. This parameter determines the extent to which a group is affected by the presence of friendly or enemy groups in **NEAR** neighborhoods.

To set neighborhood proximities, go to the **Physical/Proximities** tab. By default, every neighborhood is **HERE** to itself (this cannot be changed), and **REMOTE** from all other neighborhoods; to change the proximity, click on a row in the browser, and press the Edit button: .

### Civilian Groups

Next, the civilian groups must be defined and added to the neighborhoods. As with actors, the task of how best to divide a region’s population into distinct groups requires a deep knowledge of the region and is beyond the scope of this document. However, here are things to keep in mind:

* Belief systems are crucial. A group’s belief system gives it its identity. The group’s responses and reactions to other groups will depend on its relationships, and its relationships depend on its belief system.
* Each group resides in a single neighborhood. If you wish to have members of the same group reside in two neighborhoods, create two groups, one in each, and give them similar or identical belief systems.
* Every neighborhood needs at least one civilian group. If you’ve got no civilian group to put in a neighborhood, delete the neighborhood.
* You can create empty civilian groups (groups with zero-population) as place-holders to receive population via the **FLOW** tactic (Section 17.13).

To create a civilian group, go to the **Social/CivGroups** tab and press the Add Civilian Group button: . See Section 16.4 for a description of a civilian group’s attributes.

### Force Groups

Next, the force groups must be defined. Each such group is owned by an actor, and its personnel may be deployed and otherwise made use by that actor. Remember that force groups do not operate on their own; rather, they do what their owning actor tells them to do. There is a corollary to this: force groups do not have belief systems of their own, but instead each force group inherits the beliefs and affinities of its owning actor.

It is up to the analyst to know the important force groups currently operating in the playbox.

To define a force group, go to the **Military/Force Groups** tab, and press the Add Force Group button: . The most important decisions to make with regard to force groups are the force type, the deployment costs, and the initial number of personnel in the playbox. See Section 16.5 for a description of a force group’s attributes.

### Organization Groups

Like force groups, organization groups are assets belonging to actors. The primary differences between the two are that organization groups do not carry out military operations or project significant force.

To define an organization group, go to the **Social/OrgGroups** tab, and press the Add Organization Group button: . See Section 16.6 for a description of an organization group’s attributes. The most important decisions to make with regard to organization groups are the organization type, the deployment costs, and the initial number of personnel in the playbox.

Every organization group must belong to some actor. Humanitarian relief organizations are often apolitical (at least so far as the playbox is concerned); if there is no appropriate actor present in the playbox, then define a dummy actor, indicate that it supports no actor politically, and use its strategy to control such apolitical organizations. In such a case, the group’s deployment cost per person per week can usually be set to zero.

### Belief Systems

Next, belief systems must be defined for each of the actors and civilian groups. This can be a complex (and enlightening) process. To build a belief system, go to the Belief System browser on the **Social/Beliefs** tab. The subsequent steps are described in the following subsections.

Belief systems have two purposes. First, they are used to determine the affinity of groups for actors and other groups, and hence are the basis for all horizontal and vertical relationships. Second, they are used in the Information area (Section 10.2) to determine the resonance of information operations messages with civilian groups.

#### Define the Topics

First, consider the topics on which the residents of the playbox have strongly held beliefs. Some of these will be basic to the culture or cultures in the playbox (e.g., beliefs about the role of women) and some will be more topical (e.g., beliefs about the presence of U.S. forces in the region). It is useful to identify the cultural and political fault lines in the playbox, as these will often reveal the topics on which groups disagree.

Topics must be stated in absolute terms, so as to mean the same thing to all groups. “Public safety” might be a topic, but if so it must be understood as the safety of the public in general, not the safety merely of “my group”.

Note that affinities are computed from a subset of the topics, as determined by each topic's "Affinity" flag. The remaining topics and the beliefs associated with them are available for use in semantic hooks (see Section 10.2.2).

Topics are defined by pressing the “Add Topic” button in the top half of theBelief Systems browser, , and filling out the resulting dialog.

#### Define the Beliefs

The actors and civilian groups are shown in the “Entities” tree in the lower left part of the Belief Systems Browser. Select an actor or group to edit its beliefs.

Each belief consists of the selected entity’s position on a topic, a qualitative variable that ranges from **Passionately For** to **Passionately Against**, and an emphasis on agreement or disagreement with respect to that topic. A group that emphasizes disagreement strongly will be more likely to have strongly negative affinities with those groups that disagree with it on the topic.

#### Compute the Affinities

Once you have defined some or all of the beliefs, it is time to compute and examine the affinities. The lower right corner of theBelief Systems Browser shows the affinity of the selected entity with all other entities, and their affinities to him. Each affinity is expressed as a number from −1 to +1.

By default, affinities are computed automatically, whenever a topic is added or a belief is changed. For large scenarios with many actors and civilian groups, this can be quite slow. In this case, de-select the **Auto-Calc** checkbox, which is toward the top of the Belief Systems Browser; then, press the **Calc Now** button when you wish to compute the affinities. Note that affinities will be computed automatically on scenario lock, if necessary.

#### Adjust the Affinities

It is likely that the affinities will not look quite right at first. In this case, there are a number of things to try:

* If two groups known to be enemies have a positive affinity, look for additional topics that divide them and add them in.
* If two groups known to be friends have a negative affinity, look for additional topics on which they agree and add them in.
* Use the **Playbox Commonality** slider to adjust the degree of cultural commonality in the playbox. Increasing the playbox commonality is like adding topics of general agreement, and will tend to make all affinities increase; decreasing the playbox commonality is like removing such topics and will tend to make all affinities decrease.
* Use the **Entity Commonality** slider to decrease a particular entity’s participation in the common culture of the playbox; this is especially useful for foreign actors (e.g., the U.S.A.). Reducing the entity’s Entity Commonality is like reducing the entity’s agreement with the residents of the playbox on basic cultural topics, and will tend to decrease the entity’s affinity with other entities.

### Horizontal Relationships

Horizontal relationships are the relationships between groups, based on their affinities for each other; their affinities derive from their belief systems, where force and organization groups are assumed to inherit the beliefs of their owning actors. The initial horizontal relationships produced in this way can be seen on the **Social/HRel** tab.

In general, relationships should be adjusted by tweaking the belief systems of the relevant groups. Alternatively, the analyst may choose to override the affinity-based relationship with some preferred value. To do this, select the relevant relationship on the **Social/HRel** tab and press the Edit button, . Enter the desired value in the dialog box.

Note that what is being set is the *baseline relationship*: the relationship that the groups will have in the absence of any relevant attitude drivers. In Athena 6, only CURSEs, Information Operations Messages (IOMs), and Magic Attitude Drivers (MADs) will affect horizontal relationships.

### Vertical Relationships

Vertical relationships are the relationships between groups and actors, based on the affinity of the groups for the actor's belief systems; force and organization groups are assumed to inherit the beliefs of their owning actors, and always have a relationship of 1.0 with those actors. The initial vertical relationships produced in this way can be seen on the **Political/VRel** tab.

In general, relationships should be adjusted by tweaking the belief systems of the relevant groups or actors. Alternatively, the analyst may choose to override the initial affinity-based relationship with some preferred value. To do this, select the relevant relationship on the **Political/VRel** tab and press the Edit button, . Enter the desired value in the dialog box.

Note that what is being set is the *baseline relationship*: the relationship that a group will have with an actor in the absence of any relevant attitude drivers.

### Group Satisfaction

Next, it is necessary to state the initial satisfaction levels and saliencies for each concern of each civilian group. The four concerns are defined in Section 8.6.1.

By default, a civilian group is ambivalent (neither satisfied nor dissatisfied) about all four concerns, and gives all four the same importance (the saliency of each concern is 1.0). Neither of these defaults is realistic. Some set of circumstances obtains at time 0, and the civilians will have attitudes about it; and groups really do give more or less importance to different things. For example, a group from an honor-based culture might give Culture (CUL) a much higher saliency than Safety (SFT) or Quality of Life (QOL). Saliency is important because the group’s mood is their average satisfaction across their concerns as weighted by the saliency.

To edit a group’s satisfaction level, go to the **Social/Satisfaction** tab and select the concern in the browser. Press the Edit button, , and enter the desired changes in the resulting dialog. See Section 16.14 for a detailed description of the attributes of a satisfaction level.

Note that what is being set is the *baseline satisfaction level*: the level of satisfaction that a group will have with an actor in the absence of any relevant attitude drivers.

### Group Cooperation

Next, it is necessary to consider the initial levels of cooperation between the civilian groups and the force groups. Remember that cooperation has to do with the flow of intelligence between the groups; it can be thought of as the likelihood, 0% to 100% that a member of the civilian group will give intelligence information to a member of the force group. Cooperation defaults to 50%, which is a reasonable choice lacking other information, but an analyst with a knowledge of the groups in the playbox should do better.

Like satisfaction curves, cooperation curves have trends and thresholds; these can be used to make the curve regress back to some natural level in the absence of other drivers.

To edit a cooperation level, go to the **Information/Cooperation** tab and select the cooperation curve in the browser. Press the Edit button, , and enter the desired changes in the resulting dialog. See Section 16.15 for a detailed description of the attributes of a cooperation level.

Note that what is being set is the *baseline cooperation level*: the level of cooperation that will exist in the absence of any relevant attitude drivers.

### Abstract Situations

Abstract situations represent on-going problems in neighborhoods, e.g., an epidemic in this neighborhood or a power outage in that neighborhood. If any such exist at the start of the simulation, they should generally be created at this time.

To create an abstract situation, go the **Map** tab, and press the Create Abstract Situation button: . See Section 16.11 for a description of the attributes of abstract situations.

### GOODS Production Infrastructure

The capacity of the goods sector in the Economic model is explicitly modeled with GOODS production infrastructure in the form of “plants”. Shares of GOODS production infrastructure can be allocated to any number of actors in any of the neighborhoods. Alternatively, none can be allocated and the SYSTEM agent will own all plants in Athena. See Section 9.2 for a description of the GOODS production infrastructure.

To allocate shares of plants to an actor go to the Infrastrucure/GOODS Plants tab and press the “Add Plant Ownership” button: . See Section 16.10 for a description of the attributes of shares of GOODS production infrastructure.

### The Economic Model

The shape of the economy is determined from a base case Social Accounting Matrix (SAM). Inputs to the SAM include money flows between sectors, the number of consumers upon which the money flows are based, the price of a unit of production for the sectors that have product (goods, black and pop), remittances, black market feedstock price and availability. Since actor’s revenue and expenses are determined by their specified income sources and strategies, it is not necessary to provide money flows to and from that sector.

Cells that appear light green in color are editable and can be double clicked to edit. After editing is complete press the <Enter> key or <Tab> to the next cell. If there is an error in the entry (ie. it is not a number) then the editing session for that cell stays active and a valid entry must be made. To cancel editing, press the <Esc> key and the previous value is restored.

In the **SAM Inputs** matrix, the cell names are denoted *BX.i.j* where *i* is the row and *j* is the column. So, for example, the cell with the name *BX.goods.pop* is in the first row and third column. The actors row and column (cells *BX.actors.j* and *BX.i.actors*) are not editable because their values are determined by Athena based on the income and strategies for each actor.

The sections below describe filling in the SAM by considering each sectors expenditures on one of the other sectors and then editing cells row by row. There’s no reason that it couldn’t be filled in by considering each sectors revenues from each of the others and then filling it in column by column or some mix of both a sectors’ revenue and its expenditures.

Note that the entire economic model can be disabled by setting the parameter econ.disable to “yes”. If disabled, actors always get the income specified for them.

#### The *goods* sector

The goods sector produces one product, the goods basket (*goodsBKT*), which is a notional collection of goods that the population consumes. The goods basket nominally has a fixed base price of $1.00. Expenditures for goods baskets are supplied by filling in the cells across the goods sector row. Note that in most cases the single largest expenditure for goods will likely come from the *population* sector, since that’s who’s consuming most of the goods.

As each cell in the goods sector row is filled in the base revenue (*Base Rev*) is recomputed and displayed at the end of the row along with the base demand (which will be identical to the base revenue as long as the base price of a goods basket remains $1.00.)

#### The *black* sector

The black sector can be used to model the production or trafficking of any illicit product that makes sense for the region being analyzed. In some cases this may be illegal drugs, in other cases weapons or the trafficking of humans. Like the *goods* sector the *black* sector has a single product which has the unit of metric ton (tonne). The black sector, thus, must have a base price supplied for a unit of its product. Then, like the goods sector, expenditures of each sector on the black sector are supplied. Note that it does not make sense to have the black sector expend any of its money on itself nor have the goods sector use any black market product so the cells marked *BX.goods.black* and *BX.black.black* should always be zero. Base revenue and base demand are recomputed whenever an expenditure made on the black market is changed.

#### The *pop* sector

The population sector also has a product: a work-year. One work-year also has a price associated with it: the average wage of a typical worker in the region at hand. The population sector receives money in the form of expenditures by the other sectors that need labor. These are changed in the *pop* row which gives rise to the base revenue of the population sector and, based upon the average wage, the base demand in work-years. The base demand should be somewhat close to the expected number of people employed and the unemployment rate for the region.

#### The *actors* sector

The actors sector cannot be edited using the SAM. This is because actors’ revenue and expenditures are determined from the actors’ definitions and their strategies. See section 16.3 for definitions of actors’ income sources. Note that actors do not have a product; they only have revenues and expenditures.

#### The *region* sector

The region sector represents flows of money into and out of the region of interest that is not explicitly defined by Athena’s neighborhoods. The *region* should be confined to areas that are economically similar to those areas explicitly defined in Athena. For example, if the neighborhoods in Athena make up part of Pakistan, then the region could be the rest of Pakistan that is not defined by the neighborhoods. Once this region is defined then expenditures by the other sectors on this region can be added to the region row. It’s possible that these expenditures are all zero or that only *BX.region.world* is filled in, which has a special connotation: foreign aid to the region.

#### The *world* sector

The world sector comprises everything else not already defined by the other sectors and would represent expenditures made for imports and revenues for exports. Note that *BX.world.world* should always remain zero since expenditures made by the *world* for the *world* have no impact on the region under analysis.

#### Other Inputs

There are a few other inputs that Athena needs to be able to calibrate the computable general equilibrium (CGE) that represents the actual economy based on the demographics and actors defined in Athena. These inputs are found under the SAM matrix and include these items:

* Black market feedstock price – the price of the raw materials per tonne needed to manufacture the product made in the black market. For example, the cost of poppies to produce opium.
* Feedstock per unit of product – the amount feedstock required to produce one unit of product. If it takes 1.1 tonnes of poppies to produce 1 tonne of opium, this value is 1.1.
* Maximum feedstock available – the amount of feedstock per year that could possibly be available for the black market product being made.
* Black market capacity – the maximum possible output of the black market.
* Base consumers – the number of consumers upon which the money flows in the SAM is based. This should be approximately equal to the total population defined in Athena.
* Base unemployment rate – the unemployment rate in the playbox at time 0. Athena will calibrate the economic model using this.
* Base subsisters – the number of people engaged in subsistence agriculture. This should approximate the number of subsisters defined in Athena.
* Subsistence wage – the approximate value in $ per year of a person engaged in subsistence agriculture. This wage is used to compute the subsistence agriculture component of GDP. A good first approximation would be the poverty level in the region.
* Remittances – the amount of money per year sent into the local economy from outside.
* Remittance change rate – the per annum rate of change of remittances. This can be positive or negative.

#### Calibration

Once Athena is locked, the CGE is calibrated using the shape parameters computed from the SAM and the values taken from the *Other Inputs* section of the tab. The CGE is given initial values for its money flows that are just the values supplied for base case money flows in the SAM. It is also given the quantities of product demanded and supplied for those sectors that have product. The CGE is then analyzed and calibrated to the base GDP and unemployment rate defined on the SAM tab. This will result in Athena determining the wage in the population sector depending on the actual number of people working.

The number of consumers and workers may change as Athena runs; this affects the actor's incomes, which in turn affects how actors can spend their money, which then feeds back into other sectors. At each time step the CGE is solved to obtain equilibrium and produce the outputs that gauge the health of the economy.

### Information Operations Campaigns

Next, if the scenario includes information operations campaigns you will need to create some number of *communications asset packages* (CAPs) and assign them to the relevant actors, and then define one or *information operations messages* (IOMs) with their accompanying *semantic hooks*.

To create a CAP, go to the **Infrastructure/CAPs** tab and press the Add Comm. Asset Package button (). See Section 16.9 for a description of a CAP's attributes. The most important decisions to make are the CAP's neighborhood coverage and group penetration.

To create an IOM, go to the **Information/Info Ops Messages** tab and press the Add Info Ops Message button (). Assign a semantic hook, and attach the desired payloads. See Section 16.8 for a description of an IOM's attributes, and Section 19 for the details on the available payload types.

To create a semantic hook, first define the belief system topics required, and then go to the **Info/Semantics Hooks** tab and press the Add Semantic Hook button (). Create the hook, and then attach to positions on the topics of interest. See Secton 16.9 for a description of a hook's attributes.

### CURSEs

Most scenarios will involve magic attitude inputs for events and situations not included in Athena's DAM rules. Prior to Athena 5, the usual way to enter these in a scenario was via Magic Attitude Drivers (MADs) and the **EXECUTIVE** tactic. In Athena 5, the preferred way is to define CURSE objects. Each CURSE has a name and some number of attitude injects, each defined in terms of role names rather than group and actor names. Then, the **CURSE** tactic can be added to the SYSTEM agent's strategy; the tactic maps groups and actors to roles, and when executed causes the injects to occur.

### Strategies

The final step is to define a strategy for each actor. The subtleties of defining a realistic strategy are beyond the scope of this section; instead, it will give an overview of the process and note a few general principles. See Sections 17 and 18 for documentation of the available tactics and conditions, and Part III of this document for specifics of how to accomplish particular things.

Strategies are edited in the Strategy Browser, which is found on the **Strategy** tab. Select the actor from the list on the left, and then add blocks, tactics, and conditions using the toolbar buttons.

#### The Role of an Actor’s Strategy

An actor’s strategy has two roles. First, the actor’s strategy models his decision-making process: it specifies which tactics the actor will choose to execute under varying circumstances. This is the way we usually think of strategies.

But second, and less obviously, it can also model simulation events that directly affect the actor and his assets that are beyond the actor’s control. For example, suppose the analyst believes that Actor A’s troops will desert him if he loses control of the neighborhood that represents the capital city. This can be handled straightforwardly: Actor A’s strategy can demobilize his troops (using the **DEMOB** tactic) if the **CONTROL** condition indicates that he no longer controls the relevant neighborhood. This is something that happens to Actor A rather than something that Actor A chooses, but his strategy can make it happen.

Tactics are also used on scenario lock to set up the initial conditions for the simulation. See Section 5.2.6.

#### Order Matters

The order in which blocks are listed in the actor’s strategy matters greatly, because he will attempt to execute them in that order. If assets are consumed before the end of the list is reached, lower priority blocks might not be executed. If blocks near the top of the list change current circumstances, the conditions on lower priority blocks might not be met.

### The Model Parameter Database

Athena defines a vast number of user-customizable parameters. These *model parameters* are used to calibrate models, to enable or disable features, and (in a few rare cases) as scenario inputs. Although it is not usually necessary for analysts to change these parameters, it is often useful to examine their values; for example, the model descriptions in the *Athena Analyst’s Guide* frequently refer to model parameters by name. The complete set of model parameters is referred to as the *model parameter database*.

This section gives a general overview of the model parameter database and its contents, as well as how to learn more about it, browse its contents, and make changes.

#### Model Parameters

A model parameter is essentially a named variable with a specific data type and a default value.

For example, the force.proximity parameter determines the extent to which nearby neighborhoods contribute to a group’s force in a neighborhood, which in turn affects security and volatility. Its value must be a number no less than 0.0; its default value is 0.1.

Parameter names are hierarchical. For example, all parameters used by the force and security model have names beginning with “force.”; we will sometimes refer to the whole set as the “force.\*” parameters.

The naming hierarchy can have many levels. The “dam.BADFOOD.nearFactor” parameter, for instance, gives the “near factor”, or “p” value for the Driver Assessment Model’s **BADFOOD** rule set. The parameters that apply to this rule set all have names like “dam.BADFOOD.\*”.

#### Browsing the Model Parameter Database

To browse the contents of the model parameter database, go to the Detail Browser, and select “Model Parameters” or one of its children from the Objects tree in the sidebar. This allows you to see either all model parameters or one of the top-level subsets (e.g., “force.\*”); for each parameter you will see its name, its default value, and its current value. If the current value differs from the default, it will be shown in red.

The parameter names link to the on-line help; click on a name to read more about the parameter.

If desired, you can further limit the list of displayed parameters by entering a wildcard pattern in field at the top of the page. To see only those parameters related to the **BADFOOD** rule set, for example, enter “dam.BADFOOD.\*” in the “Wildcard” field.

#### Parameter Help

The on-line help includes documentation on each model parameter, arranged hierarchically by parameter name. There are many ways to access this information:

* Select **Help/Model Parameters** from the menu
* Select “Model Parameters” or one of its children from the “?” tab in the Detail Browser’s sidebar.
* Click on a model parameter name when browsing model parameters in the Detail Browser.

#### Editing Model Parameters

Before editing a model parameter, make sure you know what it is for and what the range of acceptable values is. Then, there are two ways to edit it. The usual way is to browse the model parameters in the Detail Browser; each parameter name has associated with it a pencil icon, . Click on the icon, and enter the new value in the resulting dialog box.

Some parameters can only be edited when the scenario is unlocked; if the scenario is locked, the pencil will be replaced by a padlock icon, . Note, however, that it is almost never a good idea to edit model parameters when the scenario is locked, except when running tests or troubleshooting a problem, because the parameter changes will be lost when the scenario is unlocked.

It is also possible to edit model parameters from the Athena command line or via an executive script using the **parm set** command. See the on-line help for more information.

#### Model Parameter Files

It is sometimes desirable to copy model parameter settings from one scenario to another. For example, an analyst might calibrate Athena for a particular region or use, and wish to use these settings for a number of scenarios. Consequently, Athena allows the user to import and export the model parameter database.

To export the model parameter database, select **File/Parameters/Export…** from the menu; you will be prompted to enter the name of a “.parmdb” file. To import the saved parameters into another scenario, open the scenario and select **File/Parameters/Import…** from the menu and select the “.parmdb” file.

### Rebasing the Scenario

Once time has been advanced for a scenario, that is, once it is in Simulation mode, it is no longer possible to change many of the model inputs. For example, one cannot add or delete groups or actors. However, given an existing scenario with time advanced one can save a new scenario whose inputs are equivalent to those of the original scenario at the current simulation time. This is called *rebasing the scenario*. For example, one might:

* Create scenario A
* Run time forward for three years (156 weeks), resulting in many changes to various attitude curves.
* Rebase the scenario, creating scenario B.

Scenario B is back in the Scenario Preparation state, and can be edited as desired. If scenario B is locked without any edits, the model values at time 0 will be approximately the same as those in Scenario A at 156 weeks.

# Part III: Athena Cookbook

## Recipes

This section contains procedures for accomplishing particular tasks or effects within an Athena scenario.

### Disabling the Economic Model

Athena contains a powerful Economic model based on a six-sector CGE; with power comes responsibility, however, and at times it may be desirable to disable the Economic model, as shown here:

1. Unlock the scenario, if it is locked.
2. In the Detail Browser, go to the sidebar’s **Objects** tab and click on the “econ.\*” entry under “Model Parameters”.
3. The resulting page will list the econ.disable parameter at the top.
4. Click on the pencil icon to the right of the parameter.
5. Enter “yes” in the “Value:” field of the resulting dialog, and press “Send&Close”.

The Economic model is now disabled. Set the parameter back to “no” to enable it again.

### Comparing Satisfaction Changes to the Real World

In order to gain a better understanding of what it means to have a persistent or transient effect of some percentage, here are some real world examples as provided by TRISA, a primary user of Athena. The following three examples are based on a week’s worth of change.

* 1%-2% change: barely noticeable to the population. Examples include regular road maintenance, local programs to improve education or perhaps a vaccination campaign.
* 3% change: fairly noticeable to the population. Examples include closing down a major access road, an ongoing political scandal, implementation of a rural electrification program.
* 5% change: very much effects the population. Examples are the shutdown of a major interstate, loss of food, water or basic services (eg. power) for an extended period of time, major government corruption.

Changes that are greater than these should be considered onetime events since they will have a large impact and may even affect the entire playbox.

* 7.5%-10% change: events whose effects may fade in a week or two. Examples include an election or a popular band coming to town.
* 15%-20% change: events whose effects last 1 to 3 months. Examples are a localized flood, a failure of government to act in a crisis and an extended food shortage.
* 30% change (or perhaps greater): events whose effects last 3 to 6 months. Examples consist of things on the order of a major disaster (eg. hurricane Katrina), destruction of well-known religious or cultural site, and a perceived complete failure of the government.

# Part IV: Reference

## Athena Objects

This section gives a complete description of the data attributes associated with the major Athena objects: neighborhoods, actors, and groups. This information is also available in the on-line help; however, the emphasis there is on helping the user to enter valid data. The emphasis here is on how the attributes are used by Athena.

Each attribute has two names: the human-readable name used in dialog boxes, browsers, and reports, and (where appropriate) the name used in SQLite database tables.

### Tactics, Conditions, Injects, and Payloads

Tactics, conditions, CURSE injects, and IOM payloads each come in a number of types. Consequently, these kinds of objects are given their own sections: tactic types are described in Section 17, condition types in Section 18, CURSE Injects in Section 19, and payloads in Section 20.

### Neighborhoods

A neighborhood is a polygonal area on the map in which simulation events occur. Civilian groups live in neighborhoods; force groups operate in neighborhoods; actors attempt to control neighborhoods.

The following table shows the attributes associated directly with each neighborhood. In addition to these, each neighborhood has a proximity to every other neighborhood; and of course, many, many things are explicitly related to particular neighborhoods.

| Attribute | Description |
| --- | --- |
| Neighborhood ID  (n) | The neighborhood ID, a unique string used to identify this neighborhood. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., group IDs). |
| Long Name  (longname) | A longer, human-readable name for use in reports and other output; it may also be used to document the precise meaning of the neighborhood ID, which might otherwise be cryptic. |
| Local Neighborhood?  (local) | A flag that indicates whether or not this neighborhood is part of the local economy described in Section 4.1.3. Local neighborhoods contribute consumers, workers, and production capacity to the local economy; non-local neighborhoods do not. |
| Urbanization  (urbanization) | The urbanization of the neighborhood: one of **Urban**, **Suburban**, **Rural**, or **Isolated**. Urbanization affects the attrition model (Section 6.3) and the ENI Services model (Section 4.10). |
| Controller  (controller) | The ID of the actor in control of this neighborhood prior to the start of the simulation (if any). As the simulation runs, the actor in control of the neighborhood may change. |
| Prod. Cap. Factor  (pcf) | A factor that determines the neighborhood's share of GOODS production capacity at simulation start. By default, neighborhoods are allocated production capacity in proportion to their work force. A neighborhood with a PCF of 2.0 will receive twice as much production capacity as that. (Section 9.2.) |
| Polygon  (polygon) | The neighborhood’s border, expressed as a list of map coordinates. |
| Reference Point  (refpoint) | The neighborhood’s reference point—a point that is known to be within the neighborhood. The software uses this point to detect improperly overlapping or nested neighborhoods, and also as a safe place to position icons relative to the polygon. |
| Stacking Order  (stacking\_order) | Overlapping neighborhood polygons stack like elements in a drawing program. When a point on the map is contained is two or more neighborhood polygons, it is presumed to be in the neighborhood at the top of the stacking order. |
| Obscured By  (obscured\_by) | It’s an error for a neighborhood higher in the stacking order to cover a lower neighborhood’s reference point. If this should happen to a neighborhood during scenario development, this attribute will be set to the ID of the offending neighborhood. |

### Actors

Actors are the significant decision makers in the playbox. Actors have belief systems, own force and organization groups, and own and execute strategies, which consist of goals, tactics, and conditions; see Sections 17 and 18 for information on the different tactic and condition types.

The following data attributes are directly associated with each actor.

| Attribute | Description |
| --- | --- |
| Actor ID  (a) | The actor ID, a unique string used to identify this actor. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., group IDs). |
| Long Name  (longname) | A longer, human-readable name for use in reports and other output; it may also be used to document the precise meaning of the actor ID, which might otherwise be cryptic. |
| Funding Type  (atype) | The actor's funding type, either "INCOME", e.g., income from the local economy, or "BUDGET", e.g., income from some foreign source outside of the local economy.  The significant difference is that a "BUDGET" actor's money does not enter the local economy unless it is spent. |
| Supports  (supports) | This attribute identifies the actor that usually receives this actor’s political support (Section 5.4); it may be an actor’s ID, or **SELF** if the actor usually supports himself, or **NONE** if the actor doesn’t usually make use of his political support.  Political support is computed weekly, neighborhood by neighborhood, and by default actors will support each other as determined by this attribute. However, it is also possible for the actor to override this setting in particular neighborhoods using the **SUPPORT** tactic (Section 17.21); thus, he can support himself in one neighborhood, a different actor in another neighborhood, and no one at all in a third. |
| Auto-maintain Infrastructure?  (auto\_maintain) | If **Yes**, any GOODS production infrastructure owned by this actor will be maintained automatically; if **No**, the actor will need to use the **MAINTAIN** tactic to expend resources to maintain his infrastructure. (See Sections 17.17 and 9.2.) |
| Cash Reserve  (cash\_reserve) | The actor’s cash reserve as of the start of the simulation, in dollars; see Section 5.2.1. |
| Cash-On-Hand  (cash\_on\_hand) | An actor’s cash-on-hand is the amount of money the actor has available to fund tactics. The actor receives income into his cash-on-hand each week, and he may also move money between his cash-on-hand and his cash reserve. Cash-on-hand that is unspent during one week rolls over to the next.  This attribute represents the actor’s cash-on-hand at the start of the simulation: the actor’s income, carry-over from the previous week, and any funds received from other actors prior to simulation start. |
| Income, GOODS Sector, $/week  (income\_goods) | **INCOME actors only.** The actor's income from the **goods** sector of the economy, in dollars per week. This is a tax-like income: genuine taxes, tolls, bribes, baksheesh, fees, and the like, paid to the actor during the production of goods.  If the Economics model is enabled (see Section 15.1), then this is the actor's income from **goods** as of time 0; income in subsequent weeks will vary with the size of the **goods** sector.  This is one component of the actor's regular weekly income. The actor's income flows into his cash-on-hand, where it may be used to fund tactics. |
| Income, BLACK Profits, shares  (income\_black\_nr) | **INCOME actors only.** The actor's income from the **black** sector of the economy, in shares of the profit. An actor receives one or more shares of the profit if the actor is in some sense an owner of the means of production and distribution. Actors receive the profits in proportion to their number of shares.  If the Economics model is enabled (see Section 15.1), then this is the actor's income from **black** as of time 0; income in subsequent weeks will depend on the actual profits in future weeks.  This is one component of the actor's regular weekly income. The actor's income flows into his cash-on-hand, where it may be used to fund tactics. |
| Income, BLACK Tax, $/week  (income\_black\_tax) | **INCOME actors only.** The actor's income from the **black** sector of the economy, in dollars per week. This is a tax-like income: genuine taxes, tolls, bribes, baksheesh, fees, and the like, paid to the actor during the production and sale of black market goods.  If the Economics model is enabled (see Section 15.1), then this is the actor's income from **black** tax-like payments as of time 0; income in subsequent weeks will vary with the size of the **black** sector.  This is one component of the actor's regular weekly income. The actor's income flows into his cash-on-hand, where it may be used to fund tactics. |
| Income, POP Sector, $/week  (income\_pop) | **INCOME actors only.** The actor's income from the **pop** sector of the economy, in dollars per week. This is a tax-like income: genuine taxes, tolls, bribes, baksheesh, fees, and the like, paid by consumers to the actor.  If the Economics model is enabled (see Section 15.1), then this is the actor's income from **pop** as of time 0; income in subsequent weeks will vary with the size of the **pop** sector.  This is one component of the actor's regular weekly income. The actor's income flows into his cash-on-hand, where it may be used to fund tactics. |
| Income, Graft on FA, $/week  (income\_graft) | **INCOME actors only.** The actor's income from graft on foreign aid to the region, in dollars per week.  If the Economics model is enabled (see Section 15.1), then this is the actor's income from graft as of time 0; income in subsequent weeks will depend on the amount of foreign aid flowing into the region.  This is one component of the actor's regular weekly income. The actor's income flows into his cash-on-hand, where it may be used to fund tactics. |
| Income, WORLD Sector, $/week  (income\_world) | **INCOME actors only.** The actor's income from outside the playbox in dollars per week. This includes foreign aid to local actors from actors outside the playbox who are not modeled explicitly in the scenario, as well as money used by foreign actors to fund their operations in the playbox.  If the Economics model is enabled (see Section 15.1), then this is the actor's income from **world** as of time 0; income in subsequent weeks will vary with the amount of money coming into the playbox.  This is one component of the actor's regular weekly income. The actor's income flows into his cash-on-hand, where it may be used to fund tactics. |
| Weekly Budget, $/week  (budget) | **BUDGET actors only.** The actor's weekly budget from his home country or other foreign source. It flows into his cash-on-hand, where it may be used to fund tactics. Note that unspent cash-on-hand is not carried over to the next week; it simply vanishes. |

### Civilian Groups

Civilian groups represent the population of the playbox, broken down by neighborhood, ethnicity, clan membership, political views, or any other criteria the analyst chooses. The identity of a civilian group is determined by its belief system, which determines its affinity with actors and other groups.

The following attributes are directly associated with civilian groups.

| Attribute | Description |
| --- | --- |
| Group ID  (g) | The group ID, a unique string used to identify this group. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., actor IDs). |
| Long Name  (longname) | A longer, human-readable name for use in reports and other output; it may also be used to document the precise meaning of the group ID, which might otherwise be cryptic. |
| Neighborhood  (n) | This attribute is the ID of the group’s neighborhood of residence. |
| Demeanor  (demeanor) | The demeanor of the group, i.e., its propensity for violence. The group may be aggressive, average, or apathetic.  Demeanor affects the group’s ability to project force in the Security model (Section ). It has had other uses in the past (i.e., in JNEM) and will likely be used in other models in the future. |
| Base Population  (basepop) | The population of this group in its neighborhood of residence just prior to the start of the simulation. This population can be reduced by attrition and by the flow of members to other groups.  Population size is one of the more important inputs, as it plays a role in many different models. See the *Athena Analyst’s Guide* for details. |
| Population Change Rate  (pop\_cr) | The population change in percent per year that the group will have. The rate can be positive, negative or zero. |
| Subsistence Agriculture Flag (sa\_flag) | This flag is true for groups that earn their living by doing subsistence agriculture rather than participating in the cash economy as either consumers or workers. |
| Color  (color) | Athena represents group personnel on the map using icons. This is the color of this group’s icons. It can be entered as a hexadecimal RGB string, e.g., red is “#FFOOOO”; however, it will usually be chosen using the GUI. The color has no effect on the model results. |
| Labor Force %  (lfp) | The percentage of the group that belongs to the labor force. Note that this must be zero for subsistence agriculture groups. |
| Housing  (housing) | The status of the group's housing: are they AT\_HOME, DISPLACED from their homes but mingling with the neighborhood population, or isolated from the neighborhood population IN\_CAMP. Note that subsistence agriculture groups must be AT\_HOME. |
| Unit Shape  (shape) | The shape of the group’s units: friend, enemy, or neutral, as defined in MIL-STD-2525a. The shape has no effect on the model results. |
| Unemployment Per Capita %  (upc) | In rebased scenarios (Section 14.20) this is the unemployment per capita for this group prior to the start of the simulation. (Section 8.2.5.2) |

### Force Groups

A force group represents a body of personnel that exists primarily to apply armed force at the behest of an actor, though the actor can use them for other purposes as well. The following attributes are directly associated with force groups.

| Attribute | Description |
| --- | --- |
| Group ID  (g) | The group ID, a unique string used to identify this group. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., actor IDs). |
| Long Name  (longname) | A longer, human-readable name for use in reports and other output; it may also be used to document the precise meaning of the group ID, which might otherwise be cryptic. |
| Owning Actor  (a) | The ID of the actor that owns this group. This attribute may initially be left blank, but must be filled in before the scenario is locked. |
| Force Type  (forcetype) | The kind of force this is: Regular Military, Paramilitary, Police, Irregular Military, and Organized Crime. The force type affects the group’s ability to project force in the Security model (Section ).  Regular Military includes traditional uniformed armies. Paramilitary includes national police forces trained in military tactics. Irregular Military includes militias and terrorist groups; many of these will be non-uniformed. |
| Training Level  (training) | The group's training level, which ranges from Not Trained to Proficient. The training level affects the group's discipline and skill. |
| Base Personnel  (base\_personnel) | The number of group personnel in the playbox on scenario lock. These personnel must be deployed, or they will be lost. |
| Demeanor  (demeanor) | The demeanor of the group, i.e., its propensity for violence. The group may be aggressive, average, or apathetic. This also affects the group’s ability to project force in the Security model (Section ). |
| Cost $/person/week  (cost) | The cost to the owning actor of maintaining one member of the group in the playbox for one week, in dollars. The cost is incurred during deployment each week, hence this is sometimes called the deployment cost.  It is reasonable to set this attribute to zero if the number of troops available does not depend on the actor’s financial assets in the playbox. A U.S. commander, for example, is given some number of troops; his ability to deploy them is not (to a first approximation, at least) limited by his own budget. |
| Local Group?  (local) | A flag indicating whether or not the civilians perceive the group as being local to the playbox. |
| Color  (color) | Athena represents group personnel on the map using icons. This is the color of this group’s icons. It can be entered as a hexadecimal RGB string, e.g., red is “#FFOOOO”; however, it will usually be chosen using the GUI. The color has no effect on the model results. |
| Unit Shape  (shape) | The shape of the group’s units: friend, enemy, or neutral, as defined in MIL-STD-2525a. The shape has no effect on the model results. |

### Organization Groups

An organization group represents a body of personnel that exists primarily for non-military reasons. Organization groups are owned by actors. The following attributes are directly associated with organization groups.

| Attribute | Description |
| --- | --- |
| Group ID  (g) | The group ID, a unique string used to identify this group. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., actor IDs). |
| Long Name  (longname) | A longer, human-readable name for use in reports and other output; it may also be used to document the precise meaning of the group ID, which might otherwise be cryptic. |
| Owning Actor  (a) | The ID of the actor that owns this group. This attribute may initially be left blank, but must be filled in before the scenario is locked. |
| Organization Type  (orgtype) | The kind of organization this is: Non-Governmental Organizations (NGOs), like the Red Cross or Doctors Without Borders; Inter-Governmental Organizations (IGOs) like UNESCO; and contractors, like Haliburton. The primary difference between the three types is that contractors can project a small amount of force in the Security model (Section 4.6), and consequently have reduced security requirements when performing activities in neighborhoods. |
| Base Personnel  (base\_personnel) | The number of group personnel in the playbox on scenario lock. These personnel must be deployed, or they will be lost. |
| Demeanor  (demeanor) | The demeanor of the group, i.e., its propensity for violence. The group may be aggressive, average, or apathetic. This also affects the group’s ability to project force in the Security model (Section 4.6). |
| Cost $/person/week  (cost) | The cost to the owning actor of maintaining one member of the group in the playbox for one week, in dollars. The cost is incurred during deployment each week, hence this is sometimes called the deployment cost.  It is reasonable to set this attribute to zero if the number of personnel available does not depend on the actor’s financial assets in the playbox. Foreign NGOs, for example, might belong to a “pseudo-actor”, a dummy actor whose strategy determines what the NGO does. In this case, the primary asset is the people on the ground, and the monetary cost can be ignored. |
| Color  (color) | Athena represents group personnel on the map using icons. This is the color of this group’s icons. It can be entered as a hexadecimal RGB string, e.g., red is “#FFOOOO”; however, it will usually be chosen using the GUI. The color has no effect on the model results. |
| Unit Shape  (shape) | The shape of the group’s units: friend, enemy, or neutral, as defined in MIL-STD-2525a. The shape has no effect on the model results. |

### Communications Asset Packages

A communications asset package (CAP) is the collection of hardware, skilled labor, and so forth that comprise a communications asset, i.e., a television station, radio station, or web site. Athena looks only at CAPs usable for the broadcast to the civilian population; see Section 9.1. CAPs are owned by actors, and have the following attributes.

| Attribute | Description |
| --- | --- |
| CAP  (k) | The CAP ID, a unique string used to identify this CAP. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., actor IDs). |
| Long Name  (longname) | A longer, human-readable name for use in reports and other output; it may also be used to document the precise meaning of the CAP ID, which might otherwise be cryptic. |
| Owning Actor  (a) | The ID of the actor that owns this CAP. This attribute may initially be left blank, but must be filled in before the scenario is locked. |
| Capacity  (capacity) | The CAP's capacity, a fraction from 0.0 to 1.0 indicating the CAP's level of repair. The CAP's coverage declines as the capacity decreases. |
| Cost $/message/week  (cost) | The cost of sending a message via this CAP, in dollars per message per week. |
| Neighborhoods | The CAP covers all or part of one or more neighborhoods. Neighborhood coverage is a fraction from 0.0 to 1.0 for each neighborhood, indicating the fraction of the neighborhood reached by the CAP's broadcasts. |
| Civ. Groups | Just because a CAP reaches a neighborhood doesn't mean that every group in the neighborhood pays attention. Therefore, the CAP has a penetration fraction for every civilian group, 0.0 to 1.0. |

### Information Operations Messages

An Information Operations Message (IOM) is a message (i.e., a television spot) sent by an actor via a CAP to influence civilian attitudes; see Section 10.2. IOMs have the following attributes.

| Attribute | Description |
| --- | --- |
| IOM  (iom\_id) | The IOM ID, a unique string used to identify this IOM. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., actor IDs). |
| Description  (longname) | An IOM is a stand-in for a real human-language message to be broadcast to the civilian population. This field should contain a brief statement of the real message as an aid to the analyst. |
| Semantic Hook  (hook\_id) | The message appeals to the belief systems of the targeted civilians by means of its *semantic hook*; see Sections 10.2.2 and 8.3. |

In addition, every IOM has one or more payloads that determine the effect it is intended to have on the civilians. See Section 20 for information about the specific payload types.

### Semantic Hooks

An Information Operations Message (IOM) appeals to the beliefs of the civilian population by means of a *semantic hook*; see Section 10.2.2. Each semantic hook has the following attributes.

| Attribute | Description |
| --- | --- |
| Hook ID  (hook\_id) | The hook ID, a unique string used to identify this hook. It should be short, and may consist of capital letters and digits. It must be distinct from all other entity IDs used in the scenario (e.g., actor IDs). |
| Description  (longname) | A brief human-readable statement of the beliefs to which the hook appeals. |

Associated with each hook is a position on one or more belief system topics (see Section 8.3). The following attributes are associated with each of the hook's topics.

| Attribute | Description |
| --- | --- |
| Hook ID  (hook\_id) | The ID of the hook to which this topic belongs. |
| Topic ID  (topic\_id) | The ID of a belief system topic. |
| Position  (position) | A position on the specific topic: **Passionately For**, **Strongly For**, **Weakly For**, **Ambivalent**, **Weakly Against**, **Strongly Against**, **Passionately Against**. |

The hook will appeal to a civilian group in proportion to the group's affinity with the hook's positions on the hook's topics.

### GOODS Production Infrastructure

GOODS Production Infrastructure determines the total production output capacity of the goods sector. The unit of GOODS production infrastructure is the “plant”. Rather than specify a specific number of plants that are owned by an actor in particular neighborhoods, Athena allows the user to specify a number of shares of the total production capacity along with an initial repair level by actor and neighborhood; see Section 9.2. Shares of GOODS production infrastructure have the following attributes.

| Attribute | Description |
| --- | --- |
| Agent ID  (a) | The agent ID. It may be any actor in Athena or 'SYSTEM'. |
| Neighborhood  (n) | A neighborhood ID. It may be any local neighborhood defined. |
| Number of Shares  (num) | The number of shares that should be owned by the agent in the neighborhood. If the production capacity of the neighborhood is zero, then no plants will be allocated to the agent in this neighborhood. |
| Average Repair Level  (rho) | This is the average repair level of the plants allocated to the agent in this neighborhood when the scenario is locked. It is a number between 0.0 and 1.0 |

### Complex User-defined Role-based Situations and Events (CURSEs)

A Complex User-defined Role-based Situation and Event (CURSE) provides a method for the analyst to model situations and events that are not already explicitly modeled in Athena. See Section 8.8.

CURSEs are sent using the **CURSE** tactic (Section 17.6).

Each curse is associated with one or more *injects*, which represent actual attitude changes.

See Section 19 for a list of the inject types.

| Attribute | Description |
| --- | --- |
| CURSE ID  (curse\_id) | The CURSE ID. It may contain letters and digits, and must begin with a letter. It should usually be short, between 2 and 10 characters in length, e.g., FLOOD. |
| Description  (longname) | This field should contain a brief statement of the real situation or event being modeled as an aid to the analyst. For example, **Widespread Flooding.** |
| Cause  (cause) | The "cause" for any inject associated with this CURSE. |
| Here Factor  (s) | Indirect effects of injects for this CURSE in the same neighborhood as the CURSE are multiplied by this factor. If it is 0.0, there will be no such indirect effects. |
| Near Factor  (p) | Indirect effects of injects for this CURSE in NEAR neighborhoods are multiplied by this factor. If it is 0.0, there will be no such indirect effects. |
| Far Factor  (q) | Indirect effects of injects for this CURSE in FAR neighborhoods are multiplied by this factor. If it is 0.0, there will be no such indirect effects. |

### Abstract Situations

Abstract situations, or *absits*, represent problems in neighborhoods that actors can ignore or resolve. The abstract situation model was brought into Athena from JNEM and is not yet completely integrated with the actor/strategy model. See the *Athena Rules* document for a list of all absit types.

#### Operations on Abstract Situations

There are two significant operations on abstract situations:

* They can be created.
* They can be resolved.

So long as an abstract situation exists, it will depress satisfaction by a certain amount, as indicated in the *Athena Rules* document.

Creating an abstract situation implies that a situation of the given type has just begin. This is generally a bad thing, and will usually cause a larger decline in satisfaction during the first week of its existence. This is called the *inception penalty*. Satisfaction will remain lower so long as the situation continues, up until the situation is *resolved*, i.e., the problem is fixed. At this point the transient satisfaction effects will cease, and there might be a *resolution benefit* which makes certain satisfaction levels go up. Again, see the *Athena Rules* document.

In JNEM, these situations were only resolved manually, when the exercise controllers deemed that the training audience had put forth sufficient effort.

In Athena, abstract situations may be created explicitly by the analyst, or when an actor’s **EXECUTIVE** tactic sends the **ABSIT:CREATE** order. Resolution may be scheduled when the situation is created, or may occur when an actor’s **EXECUTIVE** tactic sends the **ABSIT:RESOLVE** order.

#### Attributes of Abstract Situations

Abstract situations have the following attributes, which must be set when the situation is created:

| Attribute | Description |
| --- | --- |
| Type  (stype) | The type of abstract situation, as listed in the *Athena Rules Document*. |
| Coverage  (coverage) | The fraction of the neighborhood affected by the situation, as a real number from 0.0 to 1.0. |
| Inception  (inception) | A Boolean flag; **true** if there should be an inception penalty when the situation is created, and **false** otherwise. Use **false** for situations that are on-going at time 0. |
| Resolved By  (resolver) | The name of the group that will automatically resolve the situation, if automatic resolution is enabled. This information is used only to determine whether the resolver was a local or foreign group, which affects the Autonomy (AUT) effects in the rule sets. It could also be used to drive relationship or cooperation changes.  The resolver can also be specified if the situation is resolved explicitly. |
| Duration  (rduration) | The duration of the situation in days. If positive, the situation will be automatically resolved at the end of the interval, and the resolution will be attributed to the “Resolved By” group. If 0, no auto-resolution will take place. |

### Horizontal Relationships

Athena defines a horizontal relationship for every pair of groups; this figure governs much of Athena's behavior. See Section 8.5 for more about horizontal relationships.

| Attribute | Description |
| --- | --- |
| Group *f*  (f) | A group |
| Group *g*  (g) | Another group |
| Current  (hrel) | Group *f’*s relationship with group *g*, a number ranging from -1.0 to 1.0. It is frequently useful to use the following scale: **Supports**, **Likes**, **Is Indifferent To**, **Dislikes**, **Opposes**. |
| Baseline  (base) | The baseline relationship. The current level is the baseline level plus all transient effects. The baseline will change slowly over time due to these same effects. |
| Natural  (nat) | The natural level for this curve. In the absence of other drivers, the baseline level will regress back to the natural level over time. |

### Vertical Relationships

Athena defines a vertical relationship between every group and actor; this figure ultimately determines the support each actor receives. See Section 5.3 for more about vertical relationships.

| Attribute | Description |
| --- | --- |
| Group  (g) | A group |
| Actor  (a) | An actor |
| Current  (hrel) | The group's relationship with the actor, a number ranging from -1.0 to 1.0. It is frequently useful to use the following scale: **Supports**, **Likes**, **Is Indifferent To**, **Dislikes**, **Opposes**. |
| Baseline  (base) | The baseline relationship. The current level is the baseline level plus all transient effects. The baseline will change slowly over time due to these same effects. |
| Natural  (nat) | The natural level for this curve. In the absence of other drivers, the baseline level will regress back to the natural level over time. |

### Satisfaction Levels

Athena defines satisfaction levels for every civilian group and each of the four concerns: Autonomy (AUT), Culture (CUL), Quality of Life (QOL) and Safety (SFT). Each curve has the following attributes, to be set as part of scenario preparation.

The current satisfaction level and the saliency are the two most important attributes as the simulation runs. The initial baseline satisfaction needs to be set for each curve during scenario preparation.

See Section 8.6 for more about satisfaction levels.

| Attribute | Description |
| --- | --- |
| Group  (g) | A civilian group. |
| Concern  (c) | One of the four concerns. |
| Saliency | The saliency is the importance of this concern to this group: the greater the saliency relative to the other concerns, the greater the effect this concern has on the group’s overall mood. Use the following scale: **Crucial**, **Very Important**, **Important**, **Less Important**, **Unimportant**, **Negligible**.  Thus, if personal safety and comfort are less important to the group than honor, the **CUL** saliency might be **Crucial** while the **QOL** and **SFT** saliences are **Less Important**. |
| Current  (hrel) | The group’s current satisfaction with this concern, a number from -100.0 to 100.0 It is frequently useful to use the following scale: **Very Satisfied**, **Satisfied**, **Ambivalent**, **Dissatisfied**, **Very Dissatisfied**. |
| Baseline  (base) | The baseline satisfaction level. The current level is the baseline level plus all transient effects. The baseline will change slowly over time due to these same effects. |
| Natural  (nat) | The natural satisfaction level for this curve, if any. In the absence of other drivers, the baseline level will regress back to the natural level over time. |

### Cooperation Levels

Athena defines a cooperation level for every civilian group with every force group. Remember that cooperation is the likelihood that members of the civilian group will give intel to members of the force group when asked; a high cooperation level does not imply by itself any particular degree of assistance beyond the simple giving of information.

See Section 10.1 for more about cooperation levels.

| Attribute | Description |
| --- | --- |
| Civilian Group  (f) | A civilian group |
| Force Group  (g) | A force group |
| Current  (hrel) | The civilian group’s current cooperation with the force group, a number ranging from 0 to 100. It is frequently useful to use the following scale: **Always Cooperative**, **Very Cooperative**, **Cooperative**, **Marginally Cooperative**, **Uncooperative**, **Very Uncooperative**, **Never Cooperative**. |
| Baseline  (base) | The baseline cooperation level. The current level is the baseline level plus all transient effects. The baseline will change slowly over time due to these same effects. |
| Natural  (nat) | The natural cooperation level for this curve. In the absence of other drivers, the baseline level will regress back to the natural level over time. |

## Tactic Types

In Athena, actors use tactics to implement their decisions (Section 5.2.3). This section lists all of the available tactic types, with information about their parameters and how to make use of them.

There are a number of standard parameters that all or most tactic types have in common. They are presented here, since their meanings are the same in all cases.

| Parameter | Description |
| --- | --- |
| Tactic ID  (id) | An integer number that uniquely identifies this tactic. |
| Agent  (agent) | The name of the agent (i.e., an actor or the **SYSTEM** agent) in whose strategy the tactic appears. |
| Tactic Type  (typename) | The type of the tactic, e.g., **DEPLOY** or **ASSIGN**. |
| State  (state) | Every tactic has a state, one of **normal**, **disabled**, or **invalid**. The analyst can toggle a tactic's state between **normal** and **disabled** using the On/Off button in the **Strategy Browser**, and tactics are **invalid** if they fail a sanity check. Only tactics in the **normal** state are considered for execution. |

### ABSIT

The **ABSIT** tactic allows an actor or the SYSTEM agent to create an abstract situation in a neighborhood, and specify its coverage fraction, duration, and resolving group. Note that only one abstract situation of a given type can appear in a neighborhood at one time; if multiple ABSIT tactics attempt to create situations of the same type in the same neighborhood at overlapping times, only the first will succeed.

The created situation will remain until its duration expires, or until it is manually resolved. The situation will appear as an icon on Athena’s **Physical/Map** tab.

Abstract situations are described in Section 4.9; the effects of having the troops perform the activity are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Neighborhood  (n) | The ID of the neighborhood in which the situation will be created. |
| Type  (stype) | The situation type. See the *Athena Rules Document* for the complete list of situation types. |
| Coverage  (coverage) | The neighborhood coverage fraction, as a positive real number less than or equal to 1.0. The coverage represents the fraction of the neighborhood's population that will be affected by this situation. |
| Resolver  (resolver) | The name of the force group that the neighborhood population will consider responsible for resolving this situation, or **NONE** if none. This parameter applies only if the situation resolves automatically. |
| Duration  (duration) | A duration, in integer weeks. If positive, the situation will automatically resolve after the specified number of weeks; if 0, no auto-resolution will take place. |

### ACCIDENT

The **ACCIDENT** tactic allows an actor or the SYSTEM agent to create an "ACCIDENT" abstract event in a neighborhood, specifying its coverage fraction. The abstract event will occur in the neighborhood each week that the tactic executes.

Abstract events are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Neighborhood  (n) | The ID of the neighborhood in which the event will occur. |
| Coverage  (coverage) | The neighborhood coverage fraction, as a positive real number less than or equal to 1.0. The coverage represents the fraction of the neighborhood's population that will be affected by this situation. A fraction of 0.5 yields the nominal effect. |

### ASSIGN

The **ASSIGN** tactic allows an actor to assign his own force or organization group personnel to perform a particular activity in a particular neighborhood. The personnel must already be deployed to the neighborhood, and must not have been assigned to do any other activity. In addition, the actor must have sufficient cash-on-hand to pay for it. The cost in dollars per person per week of assigning troops to do an activity is defined for each activity type by parameters in the Model Parameter database. Look under activity.FRC.\*.cost and activity.ORG.\*.cost for the cost parameters. By default, all activity costs are zero.

The assignment lasts for the following week, up until the next time strategies are executed. The assigned personnel will appear as units on Athena’s **Physical/Map** tab.

If there are insufficient personnel, or insufficient funds, the tactic will not execute.

If the actor executes multiple **ASSIGN** tactics during the same week for the same neighborhood, group, and activity, the results are cumulative.

Activity assignment is described in Section 4.4; the effects of having the troops perform the activity are described in the *Athena Rules* document. See below for the list of possible activities.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Group  (g) | The ID of the force or organization group to be assigned an activity. |
| Neighborhood  (n) | The ID of the neighborhood in which the activity will take place. |
| Activity  (activity) | The name of the activity to be performed. Note that organization groups can perform only a few of the defined activities. |
| Personnel Mode  (pmode) | The user can choose the number of personnel to deploy in several different ways, as determined by the *pmode*, which takes one of the following values:  **ALL:** Athena will assign as many of the group's unassigned personnel as the actor can afford. If no unassigned personnel are available, the tactic succeeds trivially. If personnel are available but there's not enough cash to assign any, the tactic fails due to insufficient cash.  **SOME**: Athena will assign the specified number of *personnel*. The tactic fails if there are insufficient troops available to assign, or if the actor cannot afford to assign the requested number.  **UPTO**: Athena will assign as many troops as the actor can afford, from at least the *min* number of personnel up to the given *max* number. The tactic fails if there are fewer than *min* personnel available for assignment, or if the actor cannot afford at least *min* personnel.  **ALLBUT**: Athena will assign all but the specified number of *personnel*, leaving *personnel* available for additional assignments. The tactic operates on a best-efforts basis; if there are only *personnel* or fewer troops available, the tactic will succeed trivially with an empty assignment.  **PERCENT**: Athena will assign a given *percent* of the group's unassigned personnel, rounding up (so that it always assigns at least one person). If no personnel are available, succeeds trivially. The tactic fails if the actor cannot afford to assign the requested percentage. |
| Personnel  (personnel) | When the *pmode* is **SOME**, the number of unassigned personnel to be assigned.  When the *pmode* is **ALLBUT**, the number of unassigned personnel to retain; any excess will be assigned.  This parameter is ignored for other modes. |
| Min Personnel  (min) | When the *pmode* is **UPTO**, the minimum number of unassigned personnel to be assigned.  This parameter is ignored for other modes. |
| Max Personnel  (max) | When the *pmode* is **UPTO**, the maximum number of unassigned personnel to be assigned.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *pmode* is **PERCENT**, the percentage of unassigned troops to deploy, expressed as a real number, i.e., a value of 3.5 will assign 3.5% of the unassigned troops.  This parameter is ignored for other modes. |

#### Activities

The activities that can be assigned to force and organization group personnel are as follows; see the *Athena Analyst’s Guide* and the *Athena Rules Document* for more information.

| Activity | FRC | ORG | Description |
| --- | --- | --- | --- |
| CHKPOINT | • |  | Operating a checkpoint |
| CONSTRUCT | • | • | Doing or facilitating construction |
| COERCION | • |  | Coercing cooperation through threats of violence |
| CRIME | • |  | Engaging in criminal activities against enemies of the group |
| CURFEW | • |  | Enforcing a curfew |
| EDU | • | • | Providing schools |
| EMPLOY | • | • | Employing local civilians |
| GUARD | • |  | Guarding sites |
| INDUSTRY | • | • | Aiding local industry |
| INFRA | • | • | Improving local infrastructure |
| LAWENF | • |  | Enforcing the law/keeping the peace |
| MEDICAL | • | • | Providing healthcare |
| PATROL | • |  | Patrolling the neighborhood |
| PSYOP | • |  | Doing psychological operations |
| RELIEF | • | • | Providing humanitarian relief |

### ATTRIT

This tactic causes magic attrition to population of a neighborhood or to the personnel of a specific group in a neighborhood. The effects of the attrition are just as if they had occurred in combat. The personnel are killed and removed from the simulation. If civilians are killed, the **CIVCAS** rule set assesses the satisfaction changes; and if force groups were involved, the cooperation changes as well.

This tactic has no cost, and can be executed only by the SYSTEM agent.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Attrition Mode  (mode) | The attrition mode, which has one of the following values:  **NBHOOD**: Cause attrition to all units in the neighborhood (including the civilians).  **GROUP**: Cause attrition to units belonging to a specific group in the neighborhood. |
| Neighborhood  (n) | The ID of the neighborhood in which the attrition will take place. |
| Group  (f) | When the *mode* is GROUP, the ID of the group to receive the attrition. |
| Responsible Group  (g1) | A force group that is responsible for causing the attrition. This parameter is available only if civilians are being attrited. |
| Responsible Group 2  (g2) | A second force group that is responsible for causing the attrition. This parameter is available only if civilians are being attrited. |

### BROADCAST

The **BROADCAST** tactic allows an actor to broadcast an Information Operations Message (IOM) via a Communications Asset Package (CAP). The message may be attributed to a particular actor, or may be anonymous. The message will have a particular preparation cost, in addition to the cost of transmission by the chosen CAP.

An actor always has access to CAPs that he owns himself; and can be granted access to other CAPs if their owners execute the relevant **GRANT** tactic. If the actor chooses a CAP owned by another actor, and if access to the CAP is not granted during that week, then the broadcast will be cancelled and the cost refunded.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| CAP  (cap) | The ID of the CAP via which the IOM will be broadcast. |
| Attributed Source  (a) | The attributed source: the name of an actor, or **SELF** if the message is to be attributed to the tactic owner, or **NONE** if the message is to be anonymous. |
| IOM  (iom) | The ID of the IOM to be broadcast. |
| Prep. Cost  (cost) | The preparation cost, in dollars per week, as distinct from the CAP's broadcast cost. |

### BUILD

The **BUILD** tactic allows an actor to construct new goods sector production infrastructure thereby increasing the output capacity of that sector. There are two ways to allocate resources to construction jobs; see the "Construction Mode" parameter, below.

As long as this tactic executes, every effort is made to complete existing construction jobs in priority order with the most complete jobs getting resources first. If there are any resources left over after existing plants have had the maximum amount allocated, then new plants are started. The maximum amount of construction that can take place on a plant each week is controlled by a model parameter.

Once new infrastructure is completed, it adds to the production capacity of the goods sector and it must be maintained by using the MAINTAIN tactic. Nominally, it will take multiple weeks of construction before new infrastructure contributes to the production capacity of the goods sector.

If an actor cannot complete the construction of new infrastructure due to monetary constraints, then that infrastructure will remain in an unfinished state until more resources are allocated to it.

| Parameter | Description |
| --- | --- |
| Nbhood  (n) | The neighborhood in which infrastructure is to be built. |
| Construction Mode  (mode) | The construction mode. This determines how resources should be allocated to build new infrastructure. In both cases, the amount of construction done is on a best efforts basis and is constrained by the amount of cash-on-hand.  **CASH**: Construction cost is constrained by some amount of money and cannot exceed remaining cash-on-hand.  **EFFORT**: Construction cost is constrained by a number of work weeks allocated to build new infrastructure. The cost cannot exceed remaining cash-on-hand. |
| Amount  (amount) | If construction mode is CASH, the maximum amount of money to use when allocating work. This is constrained by remaining cash-on-hand. |
| Number  (num) | If construction mode is EFFORT, the maximum number of work-weeks per week to allocate to the job. The effort is converted to an amount of money which is constrained by remaining cash-on-hand. |

### CURSE

The **CURSE** tactic allows the SYSTEM agent to trigger a CURSE (Section 8.8), resulting in magic attitude changes. The **CURSE** tactic's parameters map actors and groups to the roles defined in the curse.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| CURSE ID  (curse) | The ID of the CURSE to send. |
| Role Mapping  (roles) | The mapping of particular groups and actors to the CURSE's roles. The structure of this value is complex; it should only be created using the GUI. |

### DAMAGE

The **DAMAGE** tactic sets the average repair level for goods production infrastructure plants owned by an actor in a neighborhood. It's possible to use this tactic not only to decrease average repair levels, but increase them as well.

This tactic consumes no resources, and consequently never fails. If the actor specified does not own any infrastructure in the neighborhood specified the tactic will execute trivially without changing anything. This tactic can only be executed by the SYSTEM agent.

Damage infrastructure tactics have the following attributes:

| Parameter | Description |
| --- | --- |
| Actor  (a) | The ID of an actor that owns infrastructure. |
| Nbhood  (n) | The neighborhood for which the average repair will be set. |
| Repair Level  (percent) | The new average repair level. |

### DEMO

The **DEMO** tactic allows an actor or the SYSTEM agent to create an "DEMO" (non-violent demonstration) abstract event in a neighborhood, specifying its coverage fraction and the identity of the demonstrating civilian group. The abstract event will occur in the neighborhood each week that the tactic executes.

Abstract events are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Neighborhood  (n) | The ID of the neighborhood in which the event will occur. |
| Demonstrating Group  (g) | The ID of the civilian group that is demonstrating in the neighborhood. It need not be a group that resides in the neighborhood. |
| Coverage  (coverage) | The neighborhood coverage fraction, as a positive real number less than or equal to 1.0. The coverage represents the fraction of the neighborhood's population that will be affected by this situation. A fraction of 0.5 yields the nominal effect. |

### DEMOB

The **DEMOB** tactic allows the actor to explicitly remove his force or organization group personnel from the playbox. For example, **DEMOB** could be used to model a withdrawal of U.S. troops from a particular region, when the appropriate withdrawal conditions were met. Only personnel who have not yet been deployed for the week can be demobilized.

Note that any troops remaining undeployed at the end of strategy execution will be demobilized automatically.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Group  (g) | The ID of a force or organization group belonging to the tactic’s owning actor. |
| Mode  (mode) | The demobilization mode, which has one of the following values:  **ALL**: Demobilize all undeployed personnel belonging to group *g*.  **SOME:** Demobilize the specified number of undeployed *personnel*, if available; fails if there are insufficient personnel.  **PERCENT:** Demobilizes the specified percentage of the undeployed personnel.  **EXCESS:** Demobilizes all undeployed personnel in excess of the specified number of *personnel*. |
| Personnel  (personnel) | When the mode is **SOME**, the number of undeployed personnel to be demobilized.  When the mode is **EXCESS**, the number of undeployed personnel to retain; any excess will be demobilized.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the mode is **PERCENT**, the percentage of undeployed troops to demobilized, expressed as a real number, i.e., a value of 3.5 will demobilize 3.5% of the undeployed troops.  This parameter is ignored for other modes. |

### DEPLOY

The **DEPLOY** tactic allows the actor to deploy his force and organization group personnel into particular neighborhoods in the playbox. The personnel must already be present in the playbox, either remaining from the previous week or newly mobilized using the **MOBILIZE** tactic. The number of personnel to deploy is determined by the personnel mode (*pmode*); see the table below.

**Deployment Cost:** Each force and organization group definition includes a deployment cost in dollars per person per week. This cost is incurred by the **DEPLOY** tactic; if the actor has insufficient funds, the troops will not be deployed (and will likely be demobilized).

**Multiple Deployments:** If multiple **DEPLOY** tactics are executed for the same group and neighborhood then the results are cumulative.

**Redeployment and Existing Deployments:** The neighborhood and personnel modes control new deployments only. If the tactic executed the previous week, then by default the remaining personnel will remain in place when the tactic executes this week. If the redeploy flag is set (*redeploy*), then every deployment is a new deployment.

**Undeployed Troops:** Troops remaining undeployed at the end of strategy execution are automatically demobilized, as though the **DEMOB** tactic was used. Thus, it is usually wise to have a catchall **DEPLOY** tactic with personnel mode **ALL**.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Group  (g) | The ID of a force or organization group belonging to the tactic’s owning actor. |
| Personnel Mode  (pmode) | The personnel mode, which has one of the following values.  **ALL**: Athena will deploy as many of the group's undeployed personnel as the actor can afford. If no undeployed personnel are available, the tactic succeeds trivially. If personnel are available but there's not enough cash to deploy any, the tactic fails due to insufficient cash.  **SOME**: Athena will deploy the specified number of personnel. The tactic fails if there are insufficient troops available to deploy, or if the actor cannot afford to deploy the requested number.  **UPTO**: Athena will deploy as many troops as the actor can afford, from at least the min number of personnel up to the given max number. The tactic fails if there are fewer than min personnel available for deployment, or if the actor cannot afford at least min personnel.  **ALLBUT**: Athena will deploy all but the specified number of personnel, leaving personnel available for additional deployments. The tactic operates on a best-efforts basis with respect to personnel; if there are only personnel or fewer troops available, the tactic will succeed trivially with an empty deployment.  **PERCENT**: Athena will deploy a given percent of the group's undeployed personnel, rounding up (so that it always deploys at least one person). If no personnel are available, succeeds trivially. Fails if the actor cannot afford to deploy the requested percentage. |
| Personnel  (personnel) | When the *pmode* is **SOME**, the number of undeployed personnel to be deployed.  When the *pmode* is **ALLBUT**, the number of undeployed personnel to retain; any excess will be deployed.  This parameter is ignored for other modes. |
| Min Personnel  (min) | When the *pmode* is **UPTO**, the minimum number of undeployed personnel to be deployed.  This parameter is ignored for other modes. |
| Max Personnel  (max) | When the *pmode* is **UPTO**, the maximum number of undeployed personnel to be deployed.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the pmode is **PERCENT**, the percentage of undeployed troops to deploy, expressed as a real number, i.e., a value of 3.5 will deploy 3.5% of the undeployed troops.  This parameter is ignored for other modes. |
| In Neighborhoods  (nlist) | The list of neighborhoods into which the personnel are to be deployed, as determined by rule using the NBHOODS gofer (Section 21.5). |
| Neighborhood Mode  (nmode) | The neighborhood mode, which has one of the following values.  **BY\_POP**: The chosen troops are deployed into the neighborhoods in proportion with the neighborhood population. If neighborhood A has twice the population of neighborhood B, it will receive twice the troops. This is the default.  **EQUAL**: Every neighborhood will get the same number of troops, give or take one. This was the only choice in Athena 5.1 and prior. |
| Redeploy?  (redeploy) | If false (the default), then any existing deployment will be retained when this tactic executes in subsequent weeks. If true, then the tactic will do a new deployment every week. |

### DEPOSIT

Each actor has two pots of money: his cash-on-hand and his cash reserve. Money in the first is available to be spent on tactics; money in the second is explicitly being saved for later. By managing his cash reserve using the **DEPOSIT** and **WITHDRAW** tactics, the actor can build up a war chest for a major offensive.

The **DEPOSIT** tactic transfers a sum of money from the actor’s cash-on-hand into his cash-reserve, according to the *mode*; see below.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Mode  (mode) | The deposit amount mode, which has one of the following values.  **ALL**: Deposit all remaining cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **EXACT**: Deposit the exact amount of money from cash-on-hand. The tactic will fail to execute if there are insufficient funds in the actor’s cash-on-hand.  **UPTO**: Deposit any amount of money up to the given amount from the actor’s cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **PERCENT**: Deposit the specified percent of cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **EXCESS**: Deposit any cash-on-hand that is in excess of the specified amount. The tactic will still execute successful if there is not enough cash, but no cash will be deposited. |
| Amount  (amount) | When the *mode* is **EXACT**, the exact amount of money to be deposited.  When the *mode* is **UPTO**, the maximum amount of money that should be deposited.  When the *mode* is **EXCESS**, the amount of money beyond which any excess should be deposited.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *mode* is **PERCENT**, the percentage of cash-on-hand that should be deposited.  This parameter is ignored for other modes |

### EXECUTIVE

The **EXECUTIVE** tactic allows any actor, or the **SYSTEM** agent, to execute an executive command during scenario execution. (See Section 12.8 for information about executive commands.) **EXECUTIVE** tactics can write messages to the log, pause the simulation, and send many different orders via the **send** command. It is also possible to group executive commands into a script, and then call the script using the **call** command. For example, a script might model some kind of catastrophe by creating a magic attitude driver, and then creating satisfaction or cooperation inputs against it, essentially writing a DAM rule on the fly.

**Note:** Using an **EXECUTIVE** tactic is really a kind of programming; be sure to test your executive commands before you assume that they are working. The Scripts Editor’s validation checker can be a big help here (Section 13.16). Any errors will appear in the Significant Events Log when the simulation pauses; see the **Time/Significant Events** tab.

The tactic has the following parameters:

| Parameter | Description |
| --- | --- |
| Command  (command) | The executive command to execute, often a **call** command, or a user-defined command defined in the **Scripts** tab. |

### EXPLOSION

The **EXPLOSION** tactic allows an actor or the SYSTEM agent to create an "EXPLOSION" abstract event in a neighborhood, specifying its coverage fraction. The abstract event will occur in the neighborhood each week that the tactic executes.

Abstract events are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Neighborhood  (n) | The ID of the neighborhood in which the event will occur. |
| Coverage  (coverage) | The neighborhood coverage fraction, as a positive real number less than or equal to 1.0. The coverage represents the fraction of the neighborhood's population that will be affected by this situation. A fraction of 0.5 yields the nominal effect. |

### FLOW

The **FLOW** tactic allows the SYSTEM agent to cause population to flow from one civilian group to another. The transfer is permanent (unless reversed using a subsequent **FLOW** tactic).

This tactic never executes on lock, and never fails due to insufficient resources.

The tactic has the following parameters:

| Parameter | Description |
| --- | --- |
| Source Group  (f) | The ID of the civilian group from which the population will flow. |
| Destination Group  (g) | The ID of the civilian group to which the population will flow. |
| Mode  (mode) | The *mode*, which has one of the following values.  **ALL**: Flow all remaining people from group f to group g.  **RATE**: Flow people from *f* to *g* at the given rate, expressed as a *percent* per year. If the percentage is 5, then over the course of a year this tactic would flow about 5% of the population.  **UPTO**: Flow available personnel up to a maximum of *personnel* people from *f* to *g*, but no more.  **ALLBUT**: Flow all but *personnel* of the available personnel from *f* to *g*. |
| Personnel  (personnel) | When the *mode* is **UPTO**, the maximum number of personnel to be transferred. If there are insufficient personnel available, then all remaining personnel are transferred.  When the *mode* is **ALLBUT**, the number of personnel to leave in the source group.  This parameter is ignored for other modes. |
| Rate  (percent) | When the mode is **RATE**, the rate at which personnel are to be transferred, expressed as a percentage of the group per year, i.e., 5%/year.  The parameter is ignored for other modes. |

### FUND

The **FUND** tactic allows an actor to give money to other actors. The funding actor may specify any amount of money, which will come out of the actor’s cash-on-hand. It will be added to the recipient’s cash-on-hand at the beginning of the subsequent strategy execution.[[22]](#footnote-22) Multiple **FUND** tactics with the same recipient during the same week are cumulative.

Each execution of the **FUND** tactic is a one-time event; but the tactic can be re-executed week after week to set up a continuing stream of money.

The tactic has the following parameters:

| Parameter | Description |
| --- | --- |
| Actor  (a) | The ID of the actor to receive the funds. |
| Mode  (mode) | The funding amount mode, which has one of the following values.  **ALL**: Fund with all remaining cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **EXACT**: Fund with the exact amount of money from cash-on-hand. The tactic will fail to execute if there are insufficient funds in the actor’s cash-on-hand.  **UPTO**: Fund with any amount of money up to the given amount from the actor’s cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **PERCENT**: Fund with the specified percent of cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **EXCESS**: Fund with any cash-on-hand that is in excess of the specified amount. The tactic will still execute successful if there is not enough cash, but no cash will be deposited. |
| Amount  (amount) | When the *mode* is **EXACT**, the exact amount of money to be funded.  When the *mode* is **UPTO**, the maximum amount of money that should be funded.  When the *mode* is **EXCESS**, the amount of money beyond which any excess should be funded.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *mode* is **PERCENT**, the percentage of cash-on-hand that should be funded.  This parameter is ignored for other modes |

### FUNDENI

The **FUNDENI** tactic allows an actor to fund Essential Non-Infrastructure (ENI) services for particular groups living in local neighborhoods (Section 4.1.3). The actor may specify any amount of money, which will come out of the actor’s cash-on-hand. The tactic will not execute if the funding actor has insufficient funds.

**Effect on the Civilians:** Civilian groups require a minimum amount of ENI services, and may expect a much higher amount. When a group's requirements and expectations are not met, the group's mood will fall, and the group's vertical relationships with the actors may also fall. If the group's requirements and expectations are exceeded, the group's mood and vertical relationships may rise.

**Multiple Tactics:** If multiple **FUNDENI** tactics are executed for a group, by the same or multiple actors, the effects are cumulative.

**Saturation:** Any given group can use only so much in the way of ENI services; money spent in excess of this saturation level is wasted. When multiple actors are funding ENI services for a group, and saturation is reached, the actor in control in the neighborhood is the first to get credit.

**Continuous Funding:** Each execution of the **FUNDENI** tactic provides funding for the following week. To provide continuous funding over time, the tactic must be executed each week.

The tactic has the following parameters:

| Parameter | Description |
| --- | --- |
| Groups  (glist) | The IDs of the civilian groups to receive the funds, specified by rule using the CIVGROUPS gofer (Section 21.2). The tactic will attempt to fund all groups returned by the selected gofer value. However, the tactic will never fund the following kinds of groups:   * Groups in non-local neighborhoods; these do not require or expect ENI services. * "Empty" groups, i.e., groups whose population is zero; there is no one to serve. * Groups in neighborhoods that lack a minimal level of support for the funding actor. |
| Mode  (mode) | The funding amount mode, which has one of the following values. In all cases, the amount of money is capped by default to the amount of money it would take to fund the saturation level of service (assuming the actor that owns this tactic is the only one funding ENI). It's possible to cap at a lower percentage of the SLOS; see the *los* parameter.  **ALL**: Fund with all remaining cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **EXACT**: Fund with the exact amount of money from cash-on-hand. The tactic will fail to execute if there are insufficient funds in the actor’s cash-on-hand.  **UPTO**: Fund with any amount of money up to the given amount from the actor’s cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **PERCENT**: Fund with the specified percent of cash-on-hand. The tactic will still execute successfully if cash-on-hand is zero, but no cash will be deposited.  **EXCESS**: Fund with any cash-on-hand that is in excess of the specified amount. The tactic will still execute successful if there is not enough cash, but no cash will be deposited. |
| Amount  (amount) | When the *mode* is **EXACT**, the exact amount of money to be funded.  When the *mode* is **UPTO**, the maximum amount of money that should be funded.  When the *mode* is **EXCESS**, the amount of money beyond which any excess should be funded.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *mode* is **PERCENT**, the percentage of cash-on-hand that should be funded.  This parameter is ignored for other modes |
| LOS  (los) | This parameter is used to cap the amount an actor spends on ENI as a percentage of the Saturation Level of Service (SLOS) If this parameter is set to 50, Athena will compute the amount of money it would take to fund ENI for the groups in "Groups" to meet the 50 percent of the saturation level of service and ensure that the owning actor does not spend any more than that. |

### GRANT

The **GRANT** tactic grants one or actors access to one or more of the tactic owner's Communications Asset Packages (CAPs). The actors granted access can then broadcast Information Operations Messages (IOMs) via any of the listed CAPs.

Access is granted for a single week; the same tactic must be executed in each week for which access is to be granted.

The tactic has the following parameters:

| Parameter | Description |
| --- | --- |
| CAP List  (klist) | A list of one or more CAPs belonging to the tactic owner. |
| Actor List  (alist) | A list of one or more actors, specified by rule using the **ACTORS** gofer (Section 21.1). |

### MAINTAIN

The **MAINTAIN** tactic maintains repair levels of the manufacturing infrastructure owned by an actor in the neighborhoods specified. An actor may specify any amount of money as a cap on repair spending. The total cost of repairs is computed and then compared to the cap. If the cost is less than the cap, the maximum amount of repair that could take place does. If the cap is less, as much repair than can be done within that cap is allocated to the actor’s infrastructure in proportion to the total cost of repairs across the neighborhoods specified in the tactic.

Multiple **MAINTAIN** tactics on the same infrastructure in the same week is only cumulative up to one week’s worth of maintenance.

| Parameter | Description |
| --- | --- |
| Nbhoods  (nlist) | The list of neighborhoods in which infrastructure is to be maintained, as determined by rule using the NBHOODS gofer (Section 21.5). |
| Funding Mode  (fmode) | The funding mode, which has one of the following values. In all cases, the actual amount of money spent is the lesser of:   * The cost to do one weeks worth of repairs. * The amount specified in the funding mode. * The amount of cash-on-hand.   **ALL**: Spend as much cash on hand as it takes to maintain the infrastructure.  **EXACT**: Spend up to some exact amount of cash on hand to maintain the infrastructure.  **PERCENT**: Spend up to a percentage of cash on hand to maintain the infrastructure. |
| Amount  (amount) | If the "Funding Mode" is EXACT the amount of money that any spending for repairs should be capped. Setting this amount to a large number will cause all repairs to take place, assuming that the actor has enough cash-on-hand to do them. |
| Fund Percentage  (percent) | If the "Funding Mode" is PERCENT the percentage of cash-on-hand to use as a maximum amount of funds for repairs. |
| Maintenance Mode  (rmode) | The maintenance mode has one of the following values. In all cases, the cost of repairs is capped by default to the amount of money it would take to bring up the repair level by one weeks worth of repairs.  **FULL**: Try to maintain a repair level of 100% of capacity. The tactic will still execute if cash-on-hand is less than the cost to maintain this level of repair, just the repairs will fall short.  **UPTO**: Try to maintain a repair level at some amount less than full capacity. The tactic will still execute if there is not enough cash to maintain this level, just the repairs will fall short. |
| Capacity Percentage  (level) | If the "Maintenance Mode" is UPTO this parameter is used to cap the maximum level of repair to some percentage of full capacity. Setting this to 100% is equivalent to specifying a "Maintenance Mode" of FULL. |

### MOBILIZE

Each force and organization group has an initial quantity of personnel in the playbox, based on the scenario inputs; the **MOBILIZE** tactic allows the actor who owns a force or organization group to bring additional personnel into the playbox, e.g., to implement a surge or to reflect recruitment from the local population.[[23]](#footnote-23) Once mobilized, the troops can be **DEPLOY**ed as usual.

**Tactic Ordering:** Only **DEPLOY** tactics executing *after* the **MOBILIZE** tactic but during the same week can make use of the newly mobilized personnel. Consequently, this tactic should appear before the relevant **DEPLOY** tactics in the same block, or in a higher priority block.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Group  (g) | The ID of a force or organization group belonging to the tactic’s owning actor. |
| Group  (mode) | The mobilization mode, which has one of the following values.  **ADD**: Mobilizes a specific number of new *personnel* for group *g*.  **PERCENT**: Increases the number of mobilized troops by a specific percentage, *percent*. If no troops are currently mobilized, none will be added.  **UPTO**: Increases the number of mobilized troops up to some maximum number of *personnel*. If the current number of mobilized troops exceeds that number, none will be added.  **ENSURE**: Increases the number of mobilized troops so as to ensure that least the specified number of *personnel* is available for deployment. If the current number of undeployed troops exceeds that number, none will be added. |
| Personnel  (personnel) | When the *mode* is **ADD**, the number of personnel to mobilize.  When the *mode* is **UPTO**, mobilizes troops up to the specified maximum number of personnel.  When the *mode* is **ENSURE**, ensures that at least personnel troops are available for deployment, mobilizing troops if necessary.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *mode* is **PERCENT**, increases the number of mobilized troops by the given percentage, *percent*, i.e., a value of 3.5 will mobilize 3.5% more troops.  This parameter is ignored for other modes |

### RIOT

The **RIOT** tactic allows an actor or the SYSTEM agent to create a "RIOT" abstract event in a neighborhood, specifying its coverage fraction. The abstract event will occur in the neighborhood each week that the tactic executes.

Abstract events are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Neighborhood  (n) | The ID of the neighborhood in which the event will occur. |
| Coverage  (coverage) | The neighborhood coverage fraction, as a positive real number less than or equal to 1.0. The coverage represents the fraction of the neighborhood's population that will be affected by this situation. A fraction of 0.5 yields the nominal effect. |

### SIGEVENT

The **SIGEVENT** tactic logs a message to the significant events log. The message will log level of

1 (highest priority) and a component field of "tactic".

Abstract events are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Message  (msg) | The message text to log when the tactic executes. |

### SPEND

The **SPEND** tactic allows the actor to spend his cash-on-hand on things not explicitly modeled in Athena. The tactic specifies the amount of money to be spent and the Economic sectors on which it is to be spent.

Money is allocated to sectors by means of shares. If the **goods** sector gets 3 shares while the **pop** sector gets 1 share, for example, then **goods** will receive 75% of the money and **pop** will receive 25%. Shares can be expressed as percentages if that seems clearer.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Mode  (mode) | The spending mode, which has one of the following values.  **ALL**: Spend all remaining cash-on-hand. The tactic will still execute if cash-on-hand is zero, but no cash will be spent.  **EXACT**: Spend the exact *amount* of money from cash-on-hand. The tactic will fail to execute if there are insufficient funds in cash-on-hand (except on scenario lock).  **UPTO**: Spend any amount of money up to the given *amount* from available cash-on-hand. The tactic will still execute if cash-on-hand is zero, but no cash will be spent.  **PERCENT**: Spend the specified percentage of cash-on-hand. The tactic will still execute if cash-on-hand is zero, but no cash will be spent.  **EXCESS**: Spend any cash-on-hand that is in excess of the specified *amount*. The tactic will still execute if there is not enough cash, but no cash will be spent. |
| Amount  (amount) | When the *mode* is **EXACT**, the exact amount of money to be spent.  When the *mode* is **UPTO**, the maximum amount of money that should be spent.  When the *mode* is **EXCESS**, the amount of money over which any excess should be spent.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *mode* is **PERCENT**, the percentage of cash-on-hand that should be spent.  This parameter is ignored for other modes. |
| Goods  (goods) | The share of the money to spent on the **goods** sector. |
| Black  (black) | The share of the money to spent on the **black** sector. |
| Pop  (pop) | The share of the money to spent on the **pop** sector. |
| Region  (region) | The share of the money to spent on the **region** sector. |
| World  (world) | The share of the money to spent on the **world** sector. |

### STANCE

The **STANCE** tactic allows an actor to tell his force groups to adopt a particular stance toward other groups. The stance is a relationship value; the force groups are to act as though they had the designated relationship, with consequent effects on security. How well they succeed depends on their training, and on their actual relationships toward the designated groups. See Section 4.6. The stance applies for that week, and must be repeated each week if it is to persist.

An actor can execute multiple **STANCE** tactics during a single week. If two such tactics affect the same relationship, the higher priority tactic determines the outcome.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Force Group  (f) | The force group whose stance is being set. |
| Mode  (mode) | This parameter determines how the designated civilian groups are identified: as an explicit list of groups (**BY\_GROUP**) or as all civilians living in a particular list of neighborhoods (**BY\_NBHOOD**). |
| Groups  (glist) | When the mode is **GROUP**, this parameter is the list of civilian groups for which the force group's stance is being set, specified by rule using the **GROUPS** gofer (Section 21.4). |
| Neighborhoods  (nlist) | When the mode **NBHOOD**, this parameter is the list of neighborhoods for which the force group's stance is being set, specified by rule using the **NBHOODS** gofer (Section 21.5). |
| Designated Relationship  (drel) | The relationship that the force group is ordered to adopt toward the designated civilians, using the usual scale of -1.0 to 1.0. |

### SUPPORT

The **SUPPORT** tactic allows the actor to give his political support to various actors on a neighborhood-by-neighborhood basis (Section 5.4). The actor’s default allocation of support is part of the actor’s definition; he may, by default, support himself, some other actor, or no actor, in all neighborhoods. This tactic allows him to tailor his support neighborhood by neighborhood. For example, an actor may support himself in one neighborhood, a political ally in a nearby neighborhood, and no one at all in any other neighborhood. By so doing, he indicates that he does not intend to be a political player in these other neighborhoods.

Support is given only for the following week, at which time it must be explicitly renewed if the support is to continue.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Supported Actor  (a) | The ID of the actor that will receive the actor’s political support, or **SELF**, or **NONE**. |
| In Neighborhoods  (nlist) | The IDs of the neighborhoods in which the actor’s political support is given, specified by rule using the **NBHOODS** gofer (Section 21.5). |

### VIOLENCE

The **VIOLENCE** tactic allows an actor or the SYSTEM agent to create a "VIOLENCE" abstract event in a neighborhood, specifying its coverage fraction. The abstract event will occur in the neighborhood each week that the tactic executes.

Abstract events are described in the *Athena Rules* document.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Neighborhood  (n) | The ID of the neighborhood in which the event will occur. |
| Coverage  (coverage) | The neighborhood coverage fraction, as a positive real number less than or equal to 1.0. The coverage represents the fraction of the neighborhood's population that will be affected by this situation. A fraction of 0.5 yields the nominal effect. |

### WITHDRAW

Each actor has two pots of money: his cash-on-hand and his cash reserve. Money in the first is available to be spent on tactics; money in the second is explicitly being saved for later. By managing his cash reserve using the **DEPOSIT** and **WITHDRAW** tactics, the actor can build up a war chest for a major offensive.

The **WITHDRAW** tactic transfers a sum of money from the actor's cash reserve into his cash-on-hand, so that it can be spent on tactics. The tactic always succeeds; and there are no constraints on the level of the cash-reserve, i.e., the actor can go into debt by withdrawing money he doesn't have. Note that banking is not modeled explicitly in Athena; if the actor's

cash-reserve is negative one has to assume that he floated a loan somewhere.

Cash withdrawn from the reserve can be spent by subsequent tactics during the same week. Unspent cash-on-hand rolls over to the next week, as usual.

The tactic parameters are as follows:

| Parameter | Description |
| --- | --- |
| Mode  (mode) | The withdrawal amount mode, which has one of the following values.  **ALL**: Withdraw all remaining cash reserve. The tactic will still execute if the cash reserve is zero or negative, but no cash will be withdrawn.  **EXACT**: Withdraw the exact *amount* of money from cash reserve. The tactic will fail to execute if there is not at least this amount of money in cash reserve.  **UPTO**: Withdraw any amount of money up to the given *amount* from cash reserve. The tactic will still execute if the reserve is zero or negative, but no cash will be withdrawn.  **PERCENT**: Withdraw the specified percent of the cash reserve. The tactic will still execute if the reserve is zero or negative, but no cash will be withdrawn.  **BORROW**: Withdraw any *amount* from cash reserve, allowing the balance to go negative if need be. This is a workaround for borrowing money in the absence of a model of a banking system. The tactic will always execute successfully. |
| Amount  (amount) | When the *mode* is **EXACT**, the exact amount of money to be withdrawn.  When the *mode* is **UPTO**, the maximum amount of money that should be withdrawn.  When the *mode* is **BORROW**, the amount of money to be withdrawn, allowing the reserve to go negative, if necessary.  This parameter is ignored for other modes. |
| Percentage  (percent) | When the *mode* is **PERCENT**, the percentage of cash-on-hand that should be deposited.  This parameter is ignored for other modes. |

## Condition Types

Conditions are Boolean predicates about the state of the simulation that are used to control whether and when the tactics in actor’s strategies are executed (Section 5.2). This section lists all of the conditions, explains their precise meanings, and documents their parameters.

**Note:** Athena 6 has many fewer condition types than Athena 5, but no functionality has been lost. The **AFTER**, **AT**, **BEFORE**, and **DURING** conditions have been replaced by the block time constraints (Section 5.2.7); and the rest have been merged into the single **COMPARE** condition.

Every condition has the following parameters:

| Parameter | Description |
| --- | --- |
| Condition ID  (id) | An integer number that uniquely identifies this condition. |
| Condition Type  (typename) | The type of the condition, e.g., **EXPR** or **COMPARE**. |
| Agent  (agent) | The name of the agent (i.e., an actor or the **SYSTEM** agent) in whose strategy the condition appears. |
| State  (state) | The condition’s state, one of **normal**, **disabled**, or **invalid**. A condition is **invalid** if it fails a sanity check, and can be **disabled** by the user. Otherwise, the condition will be **normal**. |

### COMPARE

This condition compares two numbers, which can be simulation outputs, simulation inputs, or just plain numbers; for the full range of values that can be used, see the NUMBER “gofer” in the on-line help. For example, the **COMPARE** condition might be met when:

* The mood of group G1 is greater than the mood of neighborhood N1
* Actor A1’s cash-reserve is less than $100,000.

The condition has the following parameters:

| Parameter | Description |
| --- | --- |
| X Value  (x) | The X value in the comparison, chosen by rule using the **NUMBER** gofer (Section 21.6). |
| Is  (comp) | The kind of comparison to do:   * EQ: equal to * AE: approximately equal to * GE: greater than or equal to * GT: greater than * LE: less than or equal to * LT: less than |
| Y Value  (y) | The Y value in the comparison, chosen by rule using the **NUMBER** gofer (Section 21.6). |

### CONTROL

The **CONTROL** condition checks an actor’s control of one or more neighborhoods (Section 5.4) according to its parameters..

The condition has the following parameters:

| Parameter | Description |
| --- | --- |
| Actor  (a) | The name of an actor whose control is being queried. |
| Sense  (sense) | One of the following values: are we checking whether actor *a* does or doesn’t control the neighborhood(s)?   * **DOES** * **DOESNT** |
| Any/All  (anyall) | One of the following values: are we checking whether he controls any or all of the neighborhoods?   * **ANY** * **ALL** |
| Neighborhoods  (nlist) | A list of one or more neighborhoods, chosen by rule using the **NBHOODS** gofer (Section 21.5). |

Thus, using this condition the user can ask these kinds of questions:

* Does actor A control all of the neighborhoods N1, N2, and N3?
* Does actor A control any of the neighborhoods N1, N2, and N3?
* Does actor A *not* control all of the neighborhoods N1, N2, and N3 (but he might control some of them)
* Does A *not* control any of the neighborhoods N1, N2, and N3?

### EXPR

The **EXPR** condition is met when a particular Boolean expression is true. The expression has TCL expression syntax, and is evaluated by the Athena executive (Section 12.8). The executive provides a large number of functions that can be used in **EXPR** expressions; see “Executive Functions” in the on-line help for the complete list. Also, note that all of the numbers available to the **COMPARE** condition are also available as executive functions.

For example, the expression

nbmood(’N1’) < 10 && mood(’G1’) < -15.0

will be true when the security of group G1 is less than 10 in neighborhood N1, and the mood of G1 is less than -15.0.

The condition parameters are as follows:

| Parameter | Description |
| --- | --- |
| Expression  (expression) | A Boolean expression with TCL syntax to be evaluated by the Athena executive. |

## CURSE Inject Types

In order for a CURSE (Section 8.8) to affect attitudes, they must have one or more injects associated with them. Injects consist of a mode, type, roles and magnitude of effect. Injects can also toggle between being normal or disabled depending on the desires of the analyst. See Section 8.8.2 for a discussion of roles.

The following table lists the attribute that all inject types have in common.

| Attribute | Description |
| --- | --- |
| CURSE ID  (curse\_id) | The CURSE ID, the identifier of the curse to which this inject is attached. |
| Inject Number  (inject\_num) | The inject number, which identifies the inject relative to its curse. |
| Inject  (id) | The ID of this inject, which consists of the ID of the curse to which it is attached, and its inject number with respect to the CURSE, as a list {curse\_id inject\_number}. |
| Mode  (inject\_mode) | The mode, **persistent** or **transient**. Transient injects affect the current level of the curve during the next time step; persistent injects affect the curve's baseline, and hence usually persistent during subsequent weeks. |
| Inject Type  (inject\_type) | The inject's type, one of: **COOP**, **HREL**, **SAT** or **VREL**. |
| State  (state) | The inject's state, one of **normal**, **disabled**, or **invalid**. Both disabled and invalid injects are ignored when the curse is sent; the difference is that disabled injects are explicitly disabled by the user, while invalid injects are disabled due to failing a sanity check. |

### COOP Inject

In addition to the common inject attributes, cooperation injects contain the following attributes.

| Attribute | Description |
| --- | --- |
| Of Civ Group Role  (f) | The name of a role representing a list of one or more civilian groups. |
| With Force Group Role  (g) | The name of a role representing a list of one or more for force groups. |
| Magnitude  (mag) | The nominal magnitude of the effect of the inject, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

### HREL Inject

In addition to the common inject attributes, horizontal relationship injects contain the following attributes.

| Attribute | Description |
| --- | --- |
| Of Group Role  (f) | The name of a role representing a list of one or more groups of any group type. |
| With Group Role  (g) | The name of a role representing a list of one or more groups of any group type. |
| Magnitude  (mag) | The nominal magnitude of the effect of the inject, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

### SAT Inject

In addition to the common inject attributes, satisfaction injects contain the following attributes.

| Attribute | Description |
| --- | --- |
| Civ. Group Role  (f) | The name of a role representing a list of one or more civilian groups. |
| With  (c) | The name of a concern: AUT, CUL, QOL or SFT |
| Magnitude  (mag) | The nominal magnitude of the effect of the inject, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

### VREL Inject

In addition to the common inject attributes, satisfaction injects contain the following attributes.

| Attribute | Description |
| --- | --- |
| Group Role  (g) | The name of a role representing a list or one or more groups. |
| With Actor Role  (a) | The name of a role representing a list of one or more actors. |
| Magnitude  (mag) | The nominal magnitude of the effect of the inject, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

## Payload Types

An Information Operations Message (IOM) contains one or more payloads that affect civilian attitudes (see Sections 10.2 and 8.4). There are several different kinds of payload; what they all have in common is that they affect the attitudes of the "covered civilian groups", i.e., those groups in the coverage area of the CAP used to broadcast the IOM and with whom the IOM resonates. This section describes the individual payload types.

Every payload has the following parameters in common.

| Parameter | Description |
| --- | --- |
| Message ID  (iom\_id) | The ID of the IOM to which the payload is attached. |
| Payload Number  (payload\_num) | The payload number, which identifies the payload relative to its IOM. |
| State  (state) | Every payload has a state, one of **normal**, **disabled**, or **invalid**. The analyst can toggle a tactic's state between **normal** and **disabled** using the On/Off button in the **IOM Browser**, and payloads are **invalid** if they fail a sanity check. Only payloads in the **normal** state affect civilian attitudes. |

### COOP Payload

The **COOP** payload affects civilian group *f’*s cooperation with some specific force group *g*. It has the following parameters.

| Parameter | Description |
| --- | --- |
| Group  (g) | The ID of a force group. |
| Magnitude  (mag) | The nominal magnitude of the effect of the payload, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

### HREL Payload

The **HREL** payload affects civilian group *f’*s horizontal relationship with some specific group *g*, of any group type. It has the following parameters.

| Parameter | Description |
| --- | --- |
| Group  (g) | The ID of a group. |
| Magnitude  (mag) | The nominal magnitude of the effect of the payload, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

### SAT Payload

The **SAT** payload affects civilian group *f’*s satisfaction with some specific concern *c*. It has the following parameters.

| Parameter | Description |
| --- | --- |
| Concern  (c) | The ID of a concern: **AUT**, **CUL**, **QOL**, or **SFT**. |
| Magnitude  (mag) | The nominal magnitude of the effect of the payload, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

### VREL Payload

The **VREL** payload affects group *f’*s vertical relationship with some specific actor *a*. It has the following parameters.

|  |  |
| --- | --- |
| Actor  (a) | The ID of an actor. |
| Magnitude  (mag) | The nominal magnitude of the effect of the payload, expressed as a real number, or as one of the Athena magnitude symbols from the *Athena Rules* document, e.g., **XL+**. |

## Gofers

A gofer is a simulation object that retrieves a particular kind of data by rule: for example, the list of neighborhoods controlled by actor A. Gofers are used by many of Athena’s tactics and conditions, and make them much more flexible. This section defines the kind of gofers used by Athena; for complete details, see the on-line help.

### ACTORS Gofer

This gofer retrieves a list of actors that meet some rule, e.g., all actors with influence in neighborhood N1. The user may also pick the desired actors from a list.

### CIVGROUPS Gofer

This gofer retrieves a list of civilian groups that meet some rule, e.g., all groups resident in N1, or all groups whose mood is “bad”. The user may also pick the desired groups from a list.

### FRCGROUPS Gofer

The gofer retrieves a list of force groups that meet some rule, e.g., all force groups owned by actor A, or all force groups deployed to neighborhood N1. The user may also pick the desired groups from a list.

### GROUPS Gofer

The gofer retrieves a list of groups that meet some rule, e.g., all groups owned by actor A, or all The user may also pick the desired groups from a list.

### NBHOODS Gofer

This gofer retrieves a list of neighborhoods from a list, e.g., all neighborhoods controlled by actor A, or all neighborhoods in which group G is deployed. The user may also pick the desired neighborhoods from a list.

### NUMBER Gofer

The gofer retrieves a single number, which may be a simulation input, a simulation output, or an explicit number chosen by the user. This gofer is especially useful in the **COMPARE** condition.

## Simulation Orders

Inputs to the Athena scenario and models are presented to Athena as *orders*.

Orders may be sent interactively by a variety of methods; such orders can usually be undone. The complete history of orders sent interactively for the current scenario is shown on the **Time/Orders**tab. This tab is hidden by default, but can be revealed by selecting **View/Order History** from the menu bar.

Orders can also be sent by the **EXECUTIVE** tactic.

See the on-line help for complete documentation of all Athena orders. Select **Help/Orders** from the menu, or “Orders” or its children from the “?” tab in the Detail Browser’s sidebar.

## Glossary

This section contains a glossary of terms. When a glossary term is included in another term’s definition, it is printed in *italics*.

absit

An *abstract situation*.

abstract situation

An *abstract situation*, or *absit*, is an on-going *situation* within a *neighborhood*; it is an *attitude* *driver*, and affects all of the people living in the neighborhood. It may also affect surrounding neighborhoods. For a complete list of situation types, and the effect of each, see the *Athena Rules* document.

activity

An *activity* is an action that a *group* can perform in a *neighborhood*, as initiated by a *tactic*. Activities are staffed by personnel deployed to neighborhoods during *strategy* execution. Personnel assigned to an activity appear as a *unit* on the **Physical/Map** tab; the unit is said to perform the activity. There are different sets of activities for the different group types. For the full list of activities, and the meaning and effect of each, see the *Athena Rules* document.

actor

A significant decision maker in the *playbox*. See Sections 5.1 and 16.3.

affinity

The affinity of one *group* or *actor* for another is a measure of the depth of feeling or support of the one for the other. Affinity is a number from +1.0 to −1.0; it is computed by comparing the *belief system*s of the two entities, and is the basis for group/group and group/actor *relationships*. Affinity is often described using the following scale:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Long Name** | **Value** | **Bounds** |
| SUPPORT | Supports | 0.8 | 0.7 < *value* <= 1.0 |
| LIKE | Likes | 0.4 | 0.2 < *value* <= 0.7 |
| INDIFF | Is Indifferent To | 0.0 | -0.2 < *value* <= 0.2 |
| DISLIKE | Dislikes | -0.4 | -0.7 < *value* <= -0.2 |
| OPPOSE | Opposes | -0.8 | -1.0 < *value* <= -0.7 |

agent

An agent is a simulation object that can own a strategy; that is, it is an object that can execute tactics subject to certain conditions to achieve its goals.

Political actors are the primary kind of agent in Athena; actors have financial and personnel assets that they use to achieve their political goals. In addition, there is also the SYSTEM agent, which uses a strategy to drive other parts of the simulation, e.g., to displace civilian groups or to make events happen at particular times.

attitude

An *attitude* of a particular *group* toward something. Attitudes include:

* Group *satisfaction* with various *concern*s
* The *cooperation* of *civilian group*s with *force group*s
* The *horizontal relationship*s between all pairs of groups
* The *vertical relationship*s between groups and *actor*s.

Attitudes change over time in response to attitude *driver*s: *event*s and *situation*s in the simulated world that affect attitudes.

Autonomy (AUT)

One of the four *concerns*. See Section 8.5.1.

belief system

Each *civilian group* and each *actor* has a belief system, which is a set of beliefs with regard to a number of *topic*s. Each belief consists of a *position* for or against the topic, along with an *emphasis* on agreement or disagreement for that position.

Belief systems are used to compute *affinities*, which in turn are the basis for group/group and group/actor relationships.

bgerror

When Athena encounters an unexpected run-time error, it does not usually halt execution. Instead, the error and a complete stack trace are written to the *debugging log*, and the user is notified. This is called a *background error*, or *bgerror* for short. See Section 12.7 for details.

block

A component of a *strategy*, containing *conditions* and *tactics*. See Section 5.2.4.

calibration

Athena's models can be calibrated by tweaking Athena's *Model Parameters*. The model descriptions in the *Athena Analyst's Guide* and *Mars Analyst's Guide* call out the names of the relevant model parameters.

CAP

A *communications asset package*.

capacity

1. (Economics) Each *sector* in the Economics model has a maximum size, or *capacity*. The size of the **pop** sector is limited by the size of the labor force, for example.

Usually when we speak of capacity, however, we mean the capacity of the **goods** sector: the ability of the economy to produce goods and services. This capacity is set when the scenario is locked and can then be increased or decreased *neighborhood* by neighborhood by building and maintain **goods** production infrastructure (Section 9.2).

2. (Information Operations) Each *Communications Asset Package* (CAP) has a *capacity*, a measure of the CAP's level of repair. When operating at full capacity, the CAP reaches its full audience.

cause

Every *attitude* *driver* has an associated *cause*. All *inputs* to *URAM* for the driver will be tagged with this cause; for example, the **DISEASE** and **EPIDEMIC** *abstract situation*s both have the cause **SICKNESS**.

The purpose of causes is to prevent multiple inputs of a similar type from piling on in an unreasonable way. For example, suppose that there is a serious epidemic in *neighborhood* A which then spreads to nearby neighborhood B. While the epidemic remained in A, the folks in B were somewhat unhappy about it (because it might spread, and because they care about some or all of the people in A). Once the epidemic spreads to B, though, the people in B have their own problems; it's unlikely to expect them to feel all that bad about the situation in neighborhood A.

Next, consider neighborhood C, which is near to both A and B. The spread of the epidemic to B might heighten their alarm somewhat, but it shouldn't double their alarm.

In these cases, where there are multiple inputs to URAM for a given *satisfaction* curve (say, group G's *SFT* in neighborhood C), the smaller inputs get swamped by the larger rather than adding to them.

The cause associated with each kind of driver can be found in the *Athena Rules* document.

civilian group

A *group* that represents civilian population in a *neighborhood*. See Sections 8.1.1 and 16.4.

CLI

Athena has a *command-line interface* (CLI), where *executive commands* and scripts can be executed. The CLI is hidden by default; to make it visible, select **View/Command Line** from the menu bar.

command

A TCL *command* used to script Athena through the Athena *executive*. See Section 12.8.

communications asset package (CAP)

A newspaper, television station, telephone system, internet service provider, website, or other bundle of assets that is used to communication information from one person to another (or, in the case of electronic warfare CAPs, to prevent information from spreading). Athena 6 implements only broadcast CAPs, which are used by *actors* to broadcast *information operations messages* (IOMs) to civilian population. CAPs are owned by *actors*. See Section 9.1.

concern

Athena tracks the *satisfaction* of *civilian group*s with respect to a number of *concerns*, also known as *soft factors*. See Section 8.6.1.

condition

A Boolean condition regarding the current *state* of the simulation which can be met or unmet. Conditions are used to control the execution of *blocks*. See Section 5.2.2.

consumer

A *consumer* is a member of the civilian population who consumes goods produced by the *local economy*. Consumers are thus distinguished from members of the *subsistence agriculture population*, who live off of their own land and do not participate in the local economy. Some consumers are also *workers*.

control

*Actors* with sufficient *support* in a *neighborhood* gain *influence* in that neighborhood; and given sufficient influence, may take *control* of the neighborhood. See Section 5.4.

cooperation

A cooperation level is an *attitude* that represents the likelihood that a member of a *civilian group* will give information to a member of a particular *force group*. Cooperation levels are represented using the following scale:

| **Name** | **Long Name** | **Value** | **Bounds** |
| --- | --- | --- | --- |
| AC | Always Cooperative | 100.0 | 99.9 < *value* <= 100.0 |
| VC | Very Cooperative | 90.0 | 80.0 < *value* <= 99.9 |
| C | Cooperative | 70.0 | 60.0 < *value* <= 80.0 |
| MC | Marginally Cooperative | 50.0 | 40.0 < *value* <= 60.0 |
| U | Uncooperative | 30.0 | 20.0 < *value* <= 40.0 |
| VU | Very Uncooperative | 10.0 | 1.0 < *value* <= 20.0 |
| NC | Never Cooperative | 0.0 | 0.0 < *value* <= 1.0 |

Athena creates the required cooperation curves automatically as groups are created. The cooperation between groups is viewed and modified on the **Attitudes/Cooperation**tab. Cooperation can also be changed by attitude*driver*s that trigger Athena's *DAM Rules*.

coverage

Coverage is a measure of the fraction of a *neighborhood* that is affected by something—a group *activity*, an *abstract situation*, etc.—expressed as a number between 0.0 and 1.0. It is a common input to the *DAM Rules*.

Culture (CUL)

One of the four *concerns*. See Section 8.6.1.

CURSE

A Complex User-defined Role-based Situation or Event. CURSEs are an *IOM*-like scheme for doing magic attitude inputs.

DAM Rules

Athena assesses the effect of attitude *drivers* using a collection of rule sets that belong to Athena's Driver Assessment Model (DAM); the rule sets are collectively known as the *DAM Rules*. The full set of rules and their effects are defined in the *Athena Rules* document.

debugging log

As Athena runs, it writes a detailed log of its behavior, primarily intended for use in debugging the application and its models; it is displayed on the **Time/Log** tab. It is distinct from the *Significant Events Log*, which logs significant simulation events. See Section 13.14.3.

demobilize

To *demobilize* *force group* or *organization group* personnel is to remove them from the playbox. See *personnel*.

demographic situation

A *demographic situation*, or *demsit*, is an on-going *situation* involving the residents of a neighborhood as detected by the Demographics model. It is an attitude *driver*. See the *Athena Rules* document for details.

demsit

A *demographic situation*.

deployment

*Actors* own *force group*s and *organization group*s, and can make use of their *mobilized* *personnel*. To do useful work, such personnel must be *deployed* into neighborhoods. A particular group's *deployment* is simply the number of personnel allocated to each neighborhood.

In particular, deployed personnel can be assigned to do *activities*, and can be *demobilized*, thus removing them from the playbox.

driver

**1.** Any *event* or *situation* occurring in the *playbox* that causes a change in one or more *attitude*s. Examples of drivers include casualties to *civilian groups*, the presence and *activities* of *force groups* in neighborhoods, *abstract situation*s, *demographic situations*, *IOMs*, *magic attitude drivers*, and *CURSEs*.

**2.** *A magic attitude driver*.

**3.** A *URAM* driver ID, used by URAM to track individual drivers. This ID is useful when examining the URAM data in the *RDB*.

Driver Assessment Model

Athena's *Driver Assessment Model* (DAM) is the collection of rule sets that assess the effects of attitude *driver*s. These rule sets are referred to as the *DAM Rules*.

emphasis

In a *civilian group* or *actor*'s *belief* *system*, a belief about a *topic* is represented as a *position* and an *emphasis* on agreement or disagreement with that position. The emphasis is a number from 0.0 to 1.0. When computing *affinity*, 0.0 indicates that disagreement counts entirely and agreement not at all, and 1.0 indicates that agreement counts entirely and disagreement not at all. In general, the lower the emphasis, the lower the resulting affinities.

**Note:** the extreme values of 0.0 and 1.0 represent extreme pathological cases and should not be used.

Emphasis should be defined using the following symbolic constants:

|  |  |  |
| --- | --- | --- |
| **Name** | **Long Name** | **Value** |
| ASTRONG | Agreement--Strong | 0.9 |
| AWEAK | Agreement | 0.7 |
| NEITHER | Neither | 0.5 |
| DWEAK | Disagreement | 0.35 |
| DSTRONG | Disagreement--Strong | 0.25 |
| DEXTREME | Disagreement--Extreme | 0.15 |

entity

A simulation *entity* is simply some object in the simulation that has identity: an *actor*, a *group*, a *neighborhood*, etc. In the context of the *belief system* model, *actor*s and *civilian group*s are sometimes referred to as "belief system entities".

event

**1.** An attitude *driver* that occurs at some particular point in time, as opposed to a *situation*.

**2.** A significant event, as included in the *Significant Events Log*.

**3.** Any occurrence in the simulation.

executive

The Athena *executive* is the object that processes *commands* entered at the Athena *CLI* or in Athena *executive scripts*. In addition, it is responsible for executing **EXECUTIVE** *tactic*s and for evaluating the **EXPR** *condition*’s Boolean expressions.

executive script

A script of *commands* to be processed by the Athena *executive*, e.g., via an **EXECUTIVE** *tactic*.

force group

A *group* that exists in order to project force on behalf of an *actor*, e.g., the U.S. Army. See Sections 6.1 and 16.5.

group

A related collection of people in the *playbox*. There are three kinds of *group*: *civilian groups*, *force groups*, and *organization groups*. See Section 8.1.

horizontal relationship

The *relationship* of one *group* with another, expressed as a number from −1.0 to +1.0. Horizontal relationships need not be symmetric—that is, group F might like group G more or less than group G likes group F. Relationships are often described using the following scale:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Long Name** | **Value** | **Bounds** |
| SUPPORT | Supports | 0.8 | 0.7 < *value* <= 1.0 |
| LIKE | Likes | 0.4 | 0.2 < *value* <= 0.7 |
| INDIFF | Is Indifferent To | 0.0 | −0.2 < *value* <= 0.2 |
| DISLIKE | Dislikes | −0.4 | −0.7 < *value* <= -0.2 |
| OPPOSE | Opposes | −0.8 | −1.0 < *value* <= -0.7 |

influence

*Influence* is a measure of an *actor*'s influence in a *neighborhood*, based on the *support* for the actor by the residents of the neighborhood relative to other actors. Mathematically, influence is a fraction of neighborhood *control*; the influence of all actors in the neighborhood sums to one. In general, an actor takes control of a neighborhood by having an influence greater than 0.5.

inject

A CURSE modifies attitudes by “injecting” attitude changes; each such change is called an “inject”. See Section 8.8.

information operations message (IOM)

A message sent by an *actor* via a *communications asset package* to affect the *attitudes* of *civilian groups*. An IOM includes a *semantic hook*, used to pull in the civilians, and one or more *payloads*, which actually change civilian attitudes. See Section 10.2.

input

**1.** Any value given to Athena as an input.

**2.** More specifically, an *attitude* input given to *URAM* by one of the *DAM Rules*.

JNEM

The Joint Non-kinetic Effects Model(JNEM) is a predecessor of JNEM used to add civilian responses to federations of battlefield simulations. Some of the models in Athena, notably *URAM* and the *DAM Rules*, have their roots in JNEM.

Julian week number

Athena maps simulation times to calendar dates using Julian week numbers. In this scheme, each year has exactly 52 weeks, numbered 1 to 52. Week 1 begins on January 1st; week 52 ends on December 31st.

For data entry, Julian weeks are represented by a week string, "*yyyy*W*ww*", where *yyyy* is the 4-digit year and *ww* is the week number. Thus, the first week of 2014 is represented as **2014W01** and the last week as **2014W52**.

Of course, 52 seven-day weeks adds up to a 364-day year. Therefore, the final week of the year always has eight days, and in leap years the week containing February 28th also has eight days. Note that Athena doesn't care about this, as it doesn't deal with durations smaller than one week.

In real world use, Julian week numbers have been replaced by the ISO8601 week numbering scheme. In this scheme, every week begins on a Monday, and all of the weeks have seven days. On the other hand, the first day of the first week of the year will usually be in December of the previous year. As Athena doesn't need this level of detail, we stick with a simple, 52-week Julian numbering scheme.

labor force

The *labor force* consists of the *workers* in the *local economy*.

local economy

Athena's Economics model concerns the economy of a region: a single country, part of that country, or perhaps several small countries taken together. Athena's *neighborhood*s can be "local", i.e., part of the economic region of interest, or non-local, outside the region. The *local economy* is then the economy of this local region of interest, which might be a subset of the *playbox*.

lock

To *lock* the *scenario* is to leave the Scenario Preparation *state*, initialize the simulation, and make it possible for simulation time to advance. See also *unlock*.

**log level**

Every entry written to the *debugging log* has an associated severity, as follows:

* **fatal**: Used when the program is about to halt. Displayed with a red background.
* **error**: Used only for *bgerror*s. Displayed with an orange background.
* **warning**: Used when a potential problem is noticed. Displayed with a yellow background.
* **normal**: Normal informational message.
* **detail**: Detailed informational message.
* **debug**: Debugging message, low-level details. Note that **debug** messages are only written to the log when enabled by the developer.

The Debugging Log Browser’s *log level*  determines which messages are actually displayed. By default the log level is **normal**, meaning that the **Log** tab messages with severity **normal** and above.

magic attitude driver

A *magic attitude driver* (MAD) is a "magic" *driver* defined by the user; it affects *attitudes* in any way the user determines.

map

**1.** A visualization of the *playbox*, including *neighborhoods*, *group* personnel represented as *units*, and other data, as displayed on the **Physical/Map** tab.

**2.** An image file used as a background on the **Physical/Map** tab as an aid to visualization.

map reference string

A map reference string is a six-character string that represents a point on the *scenario*'s map. See Section 13.7.1.

mobilized personnel

A *force group* or *organization group*'s *mobilized* personnel are the group's personnel actually present in the playbox. See *personnel*.

model parameter

An *input* to Athena that is used to calibrate or otherwise control Athena’s models. *Model parameters* are stored in the model parameter database, which is part of the *scenario*. Select **Help/Model Parameters** from the Athena menu to browse a description of the existing model parameters; select **Model Parameters** or one of its subheadings from the **Detail Browser**’s Objects tree to see default and current parameter values, and to change parameter values. Also, see Section 14.19.

mood

**1.** A *civilian group*'s *satisfaction* averaged across its *concern*s, and taking *saliency* into account. In other words, how satisfied or dissatisfied the group is with things in general.

**2.** A *neighborhood*'s mood, i.e., the average mood across all civilian groups in the neighborhood.

neighborhood

A region of the *playbox* in which *civilian groups* live and other *groups* are *deployed* and conduct *activities*. See Section 4.1.1.

order

Inputs to the Athena *scenario* and models are presented to Athena as *orders*.

Orders may be sent interactively by a variety of methods; such orders can usually be undone. The complete history of orders sent interactively for the current scenario is shown on the **Orders**tab.

Finally, orders can be sent by the **EXECUTIVE** *tactic*.

organization group

A *group* representing an organization that is active in the *playbox*, e.g., for humanitarian relief. See Section 8.1.2.

parameter

**1.** A *model parameter*.

**2.** An input to a *condition*, *order*, or *tactic*.

payload

The part of an *information operations message* that affects the *attitudes* of the civilian population. See Section 10.2.3.

personnel

In Athena, *force group*s and *organization group*s exist to provide manpower to *actor*s. The people who make up a group are called the group's *personnel*.

plant

A unit of **goods** production infrastructure that can built, damaged, and maintained. See Section 9.2.

playbox

The *playbox* is the region of interest to the analyst in the current *scenario*, i.e., the region for which *neighborhood*s are defined.

position

In a *civilian group* or *actor*'s *belief system*, the entity’s belief about a *topic* is represented as a *position* and an *emphasis* on agreement or disagreement with that position. The position is a number from 1.0 to −1.0 that represents not only whether the entity is for or against the topic, but also the strength of that position: the greater the absolute magnitude of the number, the more likely the entity is to take political action in support of its belief. Note that political action does not necessarily imply violent action.

**Note:** the extreme values of −1.0 and 1.0 represent extreme pathological cases and should not be used.

Positions can be defined using the following symbolic constants:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Long Name** | **Value** | **Bounds** |
| P+ | Passionately For | 0.9 | 0.8 < *value* <= 1.0 |
| S+ | Strongly For | 0.6 | 0.45 < *value* <= 0.8 |
| W+ | Weakly For | 0.3 | 0.05 < *value* <= 0.45 |
| A | Ambivalent | 0.0 | -0.05 < *value* <= 0.05 |
| W- | Weakly Against | −0.3 | −0.45 < *value* <= −0.05 |
| S- | Strongly Against | −0.6 | −0.8 < *value* <= −0.45 |
| P- | Passionately Against | −0.9 | −1.0 < *value* <= −0.8 |

proximity

The *proximity* of one *neighborhood* with another is expressed as one of the following four values, **HERE**, **NEAR**, **FAR**, and **REMOTE**, where **HERE** represents the proximity of a neighborhood to itself.

The *attitudes* of a *civilian group* can be affected by attitude *driver*s that occur **HERE** in the same neighborhood; to a lesser degree by drivers occurring in a **NEAR** neighborhood, even less by drivers in a **FAR** neighborhood, and not at all by drivers in a **REMOTE** neighborhood.

Proximity is assigned by the analyst, and can take into account physical distance, physical obstacles, psychological distance, and so forth. Further, it need not be symmetric: neighborhood A might regard neighborhood B as **NEAR** while B regards A as **FAR**, or even as **REMOTE**.

Quality of Life (QOL)

One of the four *concerns*. See Section 8.6.1.

**relationship**

The *attitude* a *group* has toward another group (the *horizontal relationship*) or toward an actor (the *vertical* *relationship*).

role

A *CURSE inject* is defined in terms of “roles”: place holders for lists of groups and actors. The **CURSE** tactic maps these roles to actual lists of groups and actors. See Section 8.8.

rule

A small model that assesses the *state* of the simulation and gives SME-based input to another model. The term is especially used of the *DAM Rules*; however, many other models also contain rules.

rule set

A collection of one or more *rules*. In some cases, as in the *DAM Rules*, there really are individual rules in a rule set; in other cases, a rule set consists of a look-up table which is assumed to virtually represent a family of individual rules.

Run-time Database (RDB)

Most of Athena’s working data is stored in the *Run-time Database* (RDB). It can be queried directly while Athena is running by using the **rdb** and **select** *commands*.

Safety (SFT)

One of the four *concerns*. See Section 8.6.1.

**saliency**

The *saliency* of a *concern* to a *group* is the importance of that concern to the group. Some groups will value *Quality of Life* more than *Autonomy*; for others, *Culture* will trump *Safety*. This comes into play when computing a group or *neighborhood*'s *mood*, and when comparing *satisfaction* levels across groups.

Saliency is represented using the following scale:

|  |  |  |
| --- | --- | --- |
| **Name** | **Long Name** | **Value** |
| CR | Crucial | 1.000 |
| VI | Very Important | 0.850 |
| I | Important | 0.700 |
| LI | Less Important | 0.550 |
| UN | Unimportant | 0.400 |
| NG | Negligible | 0.000 |

**satisfaction**

A *satisfaction* level is an *attitude* that represents a *civilian group*'s satisfaction with respect to some *concern*. Strong positive or negative satisfaction will encourage a group to take action. Satisfaction levels are represented using the following scale:

|  |  |  |
| --- | --- | --- |
| **Name** | **Long Name** | **Value** |
| VS | Very Satisfied | 80.0 |
| S | Satisfied | 40.0 |
| A | Ambivalent | 0.0 |
| D | Dissatisfied | -40.0 |
| VD | Very Dissatisfied | -80.0 |

scenario

An Athena *scenario* is the complete collection input data required to run the simulation.

**scenario file**

Athena is a document-centric application, rather like a word processor or spreadsheet application. An Athena document is called a *scenario file*. The scenario file contains the *scenario* data, the *map* (if any), reports, *snapshot*s, and in general everything to do with the scenario. Athena scenario files have an ".adb" file type.

**sector**

Athena's Economics model divides the economy into six sectors, as described in Section 7. The three sectors are **goods**, **black**, **pop**, **actors**, **region**, and **world**.

**security**

A *group*'s *security* is computed by the Security model. It is a number that indicates how safe members of the group are in a *neighborhood*, as opposed to how satisfied they are with their safety. Security is represented by the following scale:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Long Name** | **Value** | **Bounds** |
| H | High | 60 | 25 < *value* <= 100 |
| M | Medium | 15 | 5 < *value* <= 25 |
| L | Low | -10 | -25 < *value* <= 5 |
| N | None | -60 | -100 < *value* <= -25 |

A group's security is affected its size and by its *horizontal relationships* with other groups present in the neighborhood.

semantic hook

The part of an *information operations message* that appeals to a civilian group's *belief system* in order to sell them the IOM's *payloads*.

service situation

An attitude *driver* that involves the provision of some service to the civilian population. At present, the only service situation is the Essential Non-Infrastructure (ENI) services situation; see Section 4.10.

Significant Events Log

As the Athena simulation runs, it writes significant simulation *events* to the Significant Events Log. This log can be browsed on the **Time/Significant Events** tab, and in the **Detail Browser**; see Sections 13.14.1 and 13.5. The Significant Events Log is distinct from the *debugging log*.

situation

An attitude *driver* that takes place over time, e.g., an epidemic or the presence of a force group in the neighborhood, as opposed to a one-time *event*.

soft factors

The four *concerns*.

stance

A *force group*'s *stance* toward another *group* is a designated *horizontal relationship* value established by the force group's owning *actor* that indicates how the force group is supposed to behave toward the other group.

For example, a group given peace-keeping duties in a *neighborhood* should be given a strongly positive stance toward the groups that reside in the neighborhood, indicating that the group is to treat the residents well. This will increase the *security* of the residents.

How well a force group succeeds in behaving according to its designated stance depends on the group's training level and on the difference between the stance and the group's actual relationship with the designated groups. If the force group is poorly trained, and has a hatred for the designated groups, the increase in security will be much less than if the force group is highly trained or better disposed toward the residents.

SQLite

An SQL database engine that stores its data in a single disk file. Athena uses SQLite to store its *run-time database* and its *scenario files*. SQLite is easily available and widely used in industry; users may find it convenient to use it to extract Athena outputs from scenario files.

**state**

**1.** The complete state of the simulation at a given time: all data associated with the simulation. This is what is saved in the *scenario file*.

**2.** The simulation mode in which Athena is currently operating. The Athena simulation can be in one of the following states at any given time:

* Prep
* Running
* Paused

See Section 12.2.

strategy

An *actor*'s *strategy* is the collection of *tactic*s he will use to achieve his *goal*s, along with the set of *condition*s under which he will use them. The actor's strategy is browsed and edited on the **Strategy**tab.

subsistence agriculture population

That part of the population that lives by subsistence agriculture, whether farming or herding, and hence does not participate in the cash economy. Athena assumes that there is a hard line between the subsistence farmers and the *consumer*s, although there is certainly some overlap in reality.

support

An *actor*'s *support* in a *neighborhood* is the degree to which the people in the neighborhood (including *force group* and *organization group* personnel) will do his bidding. Support is based on the *vertical relationship* between the groups and the actor, on each group's *security* in the neighborhood, and on the number of people in each group. Support is used to compute *influence*.

SYSTEM agent

An *actor*-like *entity* that has a *strategy* and uses it to drive parts of the simulation that are not owned by actors.

tactic

An action that can be taken by an *actor* as part of his *strategy* to achieve his *goals*. Tactics are contained in *blocks*. See Sections 5.2.3 and 17.

tick

Athena advances time *week* by week; the processing done for each week as the simulation runs is referred to as happening at or on the "tick". In earlier versions of Athena, certain things happened every so many ticks; this interval was called a *tock*. In Athena 6, everything happens on the tick.

time specification string

A time specification string identifies a particular simulation time. It has the form "*baseTime* +/- *offset*", where the offset is some number of *weeks* and the base time has one of the following forms:

* **LOCK**: The time at which the scenario was locked. Note that this will not be t=0 if the start tick has been changed.
* **RUN**: The time at which the most recent run began. If the Run button was pressed while paused at t=10, then RUN will be t=11, the first time tick of the run.
* **NOW**: The current simulation time.
* ***weeks***: The time as some number of weeks since **T0**.
* ***weekString***: A specific calendar week, specified as "*yyyy*W*ww*", where *yyyy* is the four-digit year and *ww* is the *Julian week number*, 1 to 52. Thus, the weeks of 2014 range from **2014W01** to **2014W52*.***

Either the base time or the offset may be omitted. If the base time is omitted, it defaults to **NOW**; if the offset is omitted, it is 0. For example,

* **NOW**: The current simulation time.
* **NOW+10**: 10 weeks from now.
* **2014W26+5**: 5 weeks after the middle of 2014.
* **-5**: Five weeks ago
* **+5**: Five weeks from now
* **5**: Week five.

**tock**

See *tick*.

topic

A *civilian group* or *actor*'s *belief system* consists of beliefs about a number of *topics*. A topic is something that unites or divides people, and which can drive people in the playbox to political action: e.g., Shia Islam or the presence of the US forces in the playbox.

There are an unlimited number of topics in any given region of the world. Topics are of interest from Athena's point of view if feelings for or against the topic are strong enough to motivate political action, and if there is strong disagreement about the topic among the groups and actors in the playbox.

Topics must be stated in absolute terms, so that they mean the same thing to all actors and groups. For example, "I control the country" is relative to the actor or group, and so is not a topic. "Actor A controls the country", on the other hand, is an absolute statement and so is a topic.

Topics are also referenced in *semantic hooks*.

**unit**

A *unit* is a collection of personnel belonging to some *group* that can be positioned in the *playbox* and displayed on the **Map** tab.

Units come into existence automatically to represent the *neighborhood* population; to represent *force group* and *organization group* personnel deployed to neighborhoods by the **DEPLOY** *tactic*; to represent personnel assigned to particular *activities* by the **ASSIGN**tactic. Thus, every unit belongs to some group.

Each unit is tied to a particular neighborhood and can be moved about on the map within that neighborhood for the purposes of visualization; however, a unit's specific location within a neighborhood has no effect on the simulation. Units cannot be moved from one neighborhood to another.

In conventional war games, units correspond to a Table of Organization and Equipment (TOE); thus, a unit might be Company A of such-and-such a battalion. In Athena, a unit is simply an allocation of personnel to some purpose over a period of time, e.g., an allocation of members of a force group to the activity of law enforcement.

unlock

To *unlock* the scenario is to terminate a simulation run and re-enter the Scenario Preparation *state*, thus making it possible to edit the scenario inputs. Unlocking the scenario erases all of the current simulation results; if they should be retained, then save a copy of the *scenario file* prior to unlocking the scenario.

URAM

The Unified Regional Attitude Model (*URAM*) is that part of Athena that tracks changes to *attitudes* (e.g., satisfaction levels). It is given *inputs* by the *DAM Rules*, which fire in response to attitude *drivers* as described in the *Athena Rules* document.

urbanization

The urbanization level of the neighborhood: **Isolated**, **Rural**, **Suburban**, or **Urban**. See Section 4.1.1

vertical relationship

The *relationship* between a *civilian group* and an *actor*, expressed as a number from −1.0 to +1.0. Vertical relationships change over time, based on circumstances. Relationships are often described using the following scale:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Long Name** | **Value** | **Bounds** |
| SUPPORT | Supports | 0.8 | 0.7 < *value* <= 1.0 |
| LIKE | Likes | 0.4 | 0.2 < *value* <= 0.7 |
| INDIFF | Is Indifferent To | 0.0 | −0.2 < *value* <= 0.2 |
| DISLIKE | Dislikes | −0.4 | −0.7 < *value* <= −0.2 |
| OPPOSE | Opposes | −0.8 | −1.0 < *value* <= −0.7 |

**volatility**

The *volatility* of each *neighborhood* is a number from 0 to 100 that indicates how likely it is that a random person in the neighborhood will get caught up in violence unrelated to his presence; as such, it is a component of a *group*’s *security* in the neighborhood.

week

The basic unit of time in Athena is one week. See *tick*.

**worker**

A member of the *labor force*, as determined by the Demographics model. All *workers* are *consumer*s.

## Acronyms

AAG *Athena Analyst’s Guide*

AE Approximately equal to

ALOS Actual Level of Service

AUG *Athena User’s Guide*

AUT Autonomy

CAP Communications Asset Package

CGE Computable General Equilibrium

CLI Command Line Interface

CPI Consumer Price Index

CUL Culture

CURSE Complex User-defined Role-based Situation or Event

DAM Driver Assessment Model

ELOS Expected Level of Service

ENI Essential Non-Infrastructure services

EQ Equal to

GDP Gross Domestic Product

GE Greater than or equal to

GT Greater than

GUI Graphical User Interface

HTML Hypertext Markup Language

HUMINT Human Intelligence

ID Identifier

IGO Inter-Governmental Organizations

IOM Information Operations Message

JNEM Joint Non-kinetic Effects Model

LE Less than or equal to

LT Less than

LOS Level of Service

MAD Magic Attitude Driver

MAG *Mars Analyst’s Guide*

MAM Mars Affinity Model

MIL-STD Military Standard

NGO Non-Governmental Organizations

PCF Production Capacity Factor

QOL Quality of Life

RDB Run-time Database

RGB Red-Green-Blue

RLOS Required Level of Service

S&RO Stability and Recovery Operations

SFT Safety

SLOS Saturation Level of Service

SQL Standard Query Language

SWAT Special Weapons And Tactics

TBD To Be Determined

TCL Tool Command Language

TOE Table of Organization and Equipment

TRADOC U.S. Army Training and Doctrine Command

TRISA TRADOC G-2 Intelligence Support Activity

UNESCO United Nations Educational, Scientific, and Cultural Organization

URAM Unified Regional Attitude Model

URL Universal Resource Locator

## Scenario Checklist

When creating a scenario, follow the steps outlined in Section 14. These steps are summarized here for convenience. They need not be completed in this precise order, but every step should be considered.

1. Define each actor
   1. What financial resources does the actor have?
   2. Does his income come from the local economy, or does he have a budget from outside the local economy?
   3. What personnel assets does the actor have?
   4. How does the actor use his political support?
2. Select a map (if desired)
3. Define each neighborhood
   1. Omit irrelevant neighborhoods
   2. Scale neighborhoods according to your needs
   3. Each neighborhood is local or non-local
   4. Production Capacity Factor (PCF)
   5. Assign proximities based on the social/psychological distance between the neighborhoods
4. Define each civilian group
   1. Neighborhood of residence
   2. Population
   3. Housing (e.g., at home or displaced)
   4. Consider belief systems when breaking the population into groups
   5. Population change rates
   6. Are any empty placeholder groups needed (e.g., in camps) to receive population later?
5. Define each force group
   1. Owning actor
   2. Uniformed or not
   3. Deployment costs
   4. Base number of personnel in the playbox
6. Define each organization group (if any)
   1. Owning actor
   2. Deployment costs
   3. Base number of personnel in the playbox
7. Define the belief systems of the actors and civilian groups. This is likely to be an iterative process.
   1. Topics
   2. Beliefs
   3. Sanity check
   4. Tweak results as needed
8. Define satisfaction levels and trends for each civilian group
9. Define cooperation levels and trends for each civilian group and force group
10. Create initial abstract situations (if any)
11. Specify ownership shares, by neighborhood and actor, for **goods** production infrastructure.
12. Calibrate or disable the Economics model
13. Prepare to conduct Information Operations campaigns (if any)
    1. Create CAPs
    2. Create IOMs
       1. Define semantic hooks, including any needed belief system topics.
       2. Attach IOM payloads.
14. Prepare to do magic inputs by creating the relevant CURSEs.
15. Define the actors’ strategies
    1. Group related tactics into blocks
    2. Put the blocks in priority order
    3. Add conditions to blocks to support contingencies.
    4. Set the **On Lock** flag for those blocks used to establish the initial conditions.
16. Define the SYSTEM agent's strategy
    1. CURSEs, to represent significant one-of-a-kind attitude drivers
    2. Movement of civilian population around the playbox
17. Make any required changes to the model parameter database

1. Political, Military, Economics, Social, Infrastructure, Information, plus Physical Environment and Time. [↑](#footnote-ref-1)
2. Rebasing a scenario is the process of saving a new scenario equivalent to the state of the simulation at some advanced time. [↑](#footnote-ref-2)
3. Any particular point on the ground is located in one and only one neighborhood. If a city polygon is on top of a province polygon, the points belonging to the city do not belong to the province so far as Athena is concerned. [↑](#footnote-ref-3)
4. A Z-curve is a piece-wise linear approximation to an S-curve. See the *Mars Analyst’s Guide* for details. [↑](#footnote-ref-4)
5. In the context of this section, both mere presence and assigned activities count as activities. [↑](#footnote-ref-5)
6. Services which do require infrastructure are equally important, and will be modeled in a future version of Athena. [↑](#footnote-ref-6)
7. That is, the infrastructure required to produce the goods and services in the Economic model's **goods** sector. See Section 9.2. [↑](#footnote-ref-7)
8. The analyst prioritizes the blocks when the strategy is created. [↑](#footnote-ref-8)
9. The term "Supports" meaning a high vertical relationship is distinct from the kind of active political support defined in Section 5.4. [↑](#footnote-ref-9)
10. We use a Z-curve to convert the base security level, -100 to +100, into a multiplier ranging from 0.0 to (usually) 1.0. [↑](#footnote-ref-10)
11. Recruitment can be simulated by moving civilians out of groups using the FLOW tactic and mobilizing new troops for force groups using the MOBILIZE tactic. Desertion can be handled similarly. [↑](#footnote-ref-11)
12. Except, of course, in non-local neighborhoods. The number of consumers and laborers in those neighborhoods is ignored. [↑](#footnote-ref-12)
13. In Athena 6, horizontal relationships can be affected by information operations messages (IOMs) and by magic attitude drivers and CURSEs. [↑](#footnote-ref-13)
14. Athena, and JNEM before it, has always used the term “concern”; our sponsors prefer the term “soft factor”. [↑](#footnote-ref-14)
15. Magic Attitude Drivers are now deprecated, having been replaced with CURSEs, and will not be discussed further in this document. [↑](#footnote-ref-15)
16. Future versions of Athena might include the ability to build, damage, and repair communications infrastructure, with consequent effects on civilian and non-civilian communications. [↑](#footnote-ref-16)
17. This is controlled by a parmdb(5) parameter: plant.buildtime [↑](#footnote-ref-17)
18. See sqlite.org for more information. [↑](#footnote-ref-18)
19. See www.tcl.tk for more information. [↑](#footnote-ref-19)
20. Note that CURSEs do the same job without scripting. [↑](#footnote-ref-20)
21. See the values of *p* and *q* for each rule set in the *Athena Rules Document* to see how effects are reduced by decreasing proximity for that rule set. [↑](#footnote-ref-21)
22. The actors are presumed to execute their strategies in parallel; but the time the money is available, the recipient has already executed their strategy for the week. Hence, the money becomes available the following week. [↑](#footnote-ref-22)
23. In the future, we expect to have explicit models of recruitment and desertion. [↑](#footnote-ref-23)