Mockup Data Generation System - Complete Commands Reference

# Table of Contents

1. Overview  
2. Command Line Interface  
3. Batch File Interface  
4. Available Commands  
5. Command Options  
6. Usage Examples  
7. Output Formats  
8. File Generation  
9. Troubleshooting

# 1. Overview

The Mockup Data Generation System provides comprehensive mock data generation capabilities for various scenarios including positive, negative, and exclusion cases. The system supports multiple interfaces and output formats to meet different requirements.

# 2. Command Line Interface

## Primary Generator: consolidated\_generator.py

The main command-line tool that consolidates all functionality:

python src/mockgen/consolidated\_generator.py [OPTIONS]

## Alternative CLI: mockgen.cli

A specialized CLI module for probability scenarios:

python -m src.mockgen.cli [OPTIONS]

# 3. Batch File Interface

## Interactive Menu: consolidated\_generator.bat

Run the interactive menu system:

scripts/consolidated\_generator.bat

## Direct Execution: run\_consolidated\_generator.bat

Execute directly from root directory:

run\_consolidated\_generator.bat

# 4. Available Commands

## Scenario Generation Commands

|  |  |  |
| --- | --- | --- |
| Command | Description | Requirements |
| --positive | Generate positive scenarios | Model specification recommended |
| --negative | Generate negative scenarios | Model specification recommended |
| --exclusion | Generate exclusion scenarios | Model specification recommended |
| --all | Generate all scenario types | Requires --wgs flag |
| --combined | Generate combined output file | Optional --wgs flag |
| --list | List available models and scenarios | None |
| --report | Generate project report | None |

# 5. Command Options

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Description | Default | Constraints |
| --model MODEL | Specify model name | All models | String |
| --count COUNT | Number of records to generate | 1 | 1-100 |
| --split | Generate separate files for each record | False | Boolean flag |
| --wgs | Use WGS format for output | False | Boolean flag |
| --config CONFIG | Path to config file | user\_input.json | File path |
| --output-dir OUTPUT\_DIR | Output directory | generated\_outputs | Directory path |

# 6. Usage Examples

## Basic Scenario Generation

### Generate Positive Scenarios

# Generate 1 positive scenario for Model\_1

python src/mockgen/consolidated\_generator.py --positive --model Model\_1

# Generate 5 positive scenarios with WGS format

python src/mockgen/consolidated\_generator.py --positive --model Model\_1 --count 5 --wgs

# Generate positive scenarios for all models

python src/mockgen/consolidated\_generator.py --positive --count 3

### Generate Negative Scenarios

# Generate 1 negative scenario for Model\_1

python src/mockgen/consolidated\_generator.py --negative --model Model\_1

# Generate 10 negative scenarios with WGS format and split files

python src/mockgen/consolidated\_generator.py --negative --model Model\_1 --count 10 --wgs --split

# Generate negative scenarios for all models

python src/mockgen/consolidated\_generator.py --negative --count 2

### Generate Exclusion Scenarios

# Generate 1 exclusion scenario for Model\_1

python src/mockgen/consolidated\_generator.py --exclusion --model Model\_1

# Generate 3 exclusion scenarios with WGS format

python src/mockgen/consolidated\_generator.py --exclusion --model Model\_1 --count 3 --wgs

# Generate exclusion scenarios for all models

python src/mockgen/consolidated\_generator.py --exclusion --count 1

## Advanced Generation

### Generate All Scenarios

# Generate all scenario types for all models (requires WGS format)

python src/mockgen/consolidated\_generator.py --all --wgs

# Generate all scenarios for specific model with 5 records

python src/mockgen/consolidated\_generator.py --all --wgs --model Model\_1 --count 5

# Generate all scenarios with split files

python src/mockgen/consolidated\_generator.py --all --wgs --count 3 --split

### Generate Combined Output

# Generate combined output file with standard format

python src/mockgen/consolidated\_generator.py --combined

# Generate combined output file with WGS format

python src/mockgen/consolidated\_generator.py --combined --wgs

## Utility Commands

### List Available Models

# List all available models and their scenario support

python src/mockgen/consolidated\_generator.py --list

### Generate Project Report

# Generate comprehensive project report

python src/mockgen/consolidated\_generator.py --report

## CLI Module Examples

### Using mockgen.cli

# Generate positive scenarios (WGS format required)

python -m src.mockgen.cli --probability --positive --model Model\_1 --wgs

# Generate multiple negative scenarios

python -m src.mockgen.cli --probability --negative --model Model\_1 --count 5 --wgs

# Generate all scenario types

python -m src.mockgen.cli --probability --all --model Model\_1 --wgs

# List available models

python -m src.mockgen.cli --list

# 7. Output Formats

## Standard Format

• Default output format

• Simplified structure

• Suitable for general use cases

## WGS Format

• Complete template structure

• Matches exact reference template

• Required for --all command

• Includes all template fields in specific order

# 8. File Generation

## Single File Output

• Default behavior

• All records in one JSON file

• Timestamped filename

## Split File Output

• Use --split flag

• Separate file for each record

• Individual timestamped filenames

• Useful for large datasets

## File Naming Convention

{model\_name}\_{scenario\_type}\_{timestamp}.json

{model\_name}\_{scenario\_type}\_{timestamp}\_{record\_number}Z.json (when split)

## Output Directory

• Default: generated\_outputs/

• Automatically created if not exists

• Configurable via --output-dir option

# Configuration

## Configuration File

• Default: user\_input.json

• Contains model definitions and data templates

• Configurable via --config option

## Configuration Structure

{

"Model\_1\_positive": { ... },

"Model\_1\_negative": { ... },

"Model\_1\_exclusion": { ... },

"Model\_2\_positive": { ... }

}

# Batch File Menu Options

## Interactive Menu Choices

• 1. Generate All Scenarios (All Models): Prompts for WGS format, record count, and split option

• 2. Generate Positive Scenarios: Prompts for model name, WGS format, record count, and split option

• 3. Generate Negative Scenarios: Prompts for model name, WGS format, record count, and split option

• 4. Generate Exclusion Scenarios: Prompts for model name, WGS format, record count, and split option

• 5. Generate WGS Format Scenarios: Prompts for scenario type, model name, record count, and split option

• 6. Generate Combined Output: Prompts for WGS format option

• 7. Generate Project Report: No additional prompts required

• 8. List Available Models: No additional prompts required

• 9. Help: Displays system information and usage

• 0. Exit: Closes the application

# Command Combinations

## Valid Combinations

• --positive/--negative/--exclusion + --model + --count + --wgs + --split

• --all + --wgs + --count + --split + --model (optional)

• --combined + --wgs (optional)

• --list (standalone)

• --report (standalone)

## Invalid Combinations

• --all without --wgs (will generate error)

• Multiple scenario types in same command

• --split without --count > 1

# Error Handling

## Common Errors

• Missing WGS flag for --all: Use --wgs with --all command

• Invalid model name: Check available models with --list

• Count out of range: Use values between 1-100

• Missing configuration: Ensure user\_input.json exists

## Error Messages

• Clear error descriptions

• Usage suggestions

• Help text display

# Performance Considerations

## Record Count Limits

• Maximum: 100 records per command

• Recommended: 1-50 for optimal performance

• Large counts may increase processing time

## File Size Considerations

• WGS format produces larger files

• Split files increase total disk usage

• Consider available disk space for large datasets

# Best Practices

## Command Usage

1. Start with small record counts for testing

2. Use --list to verify available models

3. Test with single records before batch generation

4. Use appropriate output format for your use case

## File Management

1. Use descriptive model names

2. Implement regular cleanup of generated files

3. Monitor output directory size

4. Use timestamps for file organization

# 9. Troubleshooting

## Common Issues

• "No such file or directory": Run from project root directory

• Configuration errors: Verify user\_input.json format

• Permission errors: Check write permissions for output directory

• Memory issues: Reduce record count for large datasets

## Debug Commands

# Check available models

python src/mockgen/consolidated\_generator.py --list

# Verify configuration

python src/mockgen/consolidated\_generator.py --report

# Test with minimal options

python src/mockgen/consolidated\_generator.py --positive --model Model\_1

# Support and Documentation

## Additional Resources

• README.md: Project overview and quick start

• MASTER\_REFERENCE.md: Detailed technical documentation

• Source code: Full implementation details

• Batch files: Interactive execution options

## Getting Help

# Display help for main generator

python src/mockgen/consolidated\_generator.py --help

# Display help for CLI module

python -m src.mockgen.cli --help

# Use interactive batch menu

scripts/consolidated\_generator.bat

---

*This document provides a comprehensive reference for all available commands and options in the Mockup Data Generation System. For the most up-to-date information, refer to the source code and help commands.*