Contest Log Analytics

Version: 1.0.0-alpha.1
A post-mortem analysis engine for amateur radio contesting

Overview

Contest Log Analytics (CLA) is a sophisticated analysis tool for amateur radio contest logs. Think of it as ”game tape” for radiosport—just as professional sports teams review game footage to understand why plays succeeded or failed, CLA analyzes your contest logs to reveal the hidden story behind the final score.

Most logging software is designed for the heat of battle—getting QSOs into the log efficiently. **Contest Log Analytics is designed for the day after**, when you want to understand:

- Who owned the band at 0200Z?
- Which multipliers did you miss that your competitors found?
- Exactly how your operating strategy differed from the winner's approach?
- Where you gained or lost your advantage in the margins?

Key Features

Flexible Analysis

Analyze a single log to understand your performance metrics, or compare 2-3 logs for head-to-head competitive analysis. The system automatically adapts reports based on the number of logs provided.

Heuristic Analysis

Cabrillo logs don't record whether you were ”Running” (Calling CQ) or ”Search & Pouncing” (Hunting). CLA uses a sophisticated algorithm to infer your operating mode based on time and frequency deltas, creating insights that don't exist in the raw file.

Unique vs. Common Analysis

”Common” QSOs are the price of admission—everyone works them. Contests are won in the margins. CLA isolates ”Unique Run” and ”Unique S&P” QSOs to show exactly where you gained (or lost) your advantage relative to the competition.

Public Log Archive Integration

Forget manually downloading competitor logs. CLA connects directly to public contest archives (starting with CQ WW). Simply select the year and mode, search for competitors by callsign, and fetch data instantly for head-to-head analysis.

Dual Interface

- **Web Dashboard**: Containerized Django application with interactive visualizations (Plotly charts), QSO and Multiplier dashboards with drill-down analysis, progress tracking, and comprehensive report viewing
- **Command-Line Interface**: Full-featured CLI for batch processing, automation, and text report generation

Comprehensive Reporting

- **Interactive Animations**: Replay the contest hour-by-hour to visualize momentum shifts
- **Cumulative Difference Plots**: See the "zero line" of competition—where leads were built or lost
- **QSO Breakdown Charts**: Visualize Unique vs. Common QSOs by Run/S&P/Mixed mode
- **Multiplier Analysis**: Detailed missed multiplier reports with "Group Par" benchmarking
- **Multiplier Breakdown Dashboards**: Interactive HTML dashboards with drill-down analysis of missed multipliers by band and operating mode
- **Rate Sheets**: Hourly breakdown by band and mode
- **Score Summaries**: Comprehensive scoring with band-by-band details

---

Supported Contests

CLA supports major international contests including:

- **CQ World Wide DX Contest** (CW/SSB)
- **CQ WPX Contest** (CW/SSB)
- **ARRL DX Contest** (CW/SSB)
- **ARRL Sweepstakes**
- **ARRL IARU HF World Championship**
- **WAE Contest** (CW/SSB with QTC support)
- **WRTC** (with specialized propagation analysis)
- **CQ 160-Meter Contest**
- **North American QSO Party (NAQP)**
- **ARRL Field Day**

The analyzer utilizes external data files (CTY.DAT, Section files) to ensure accurate scoring and multiplier calculations.

Quick Start

Installation (Docker - Recommended)

The easiest way to run CLA is via Docker with the pre-configured web dashboard:

1. **Clone the repository**:

   ```
   git clone https://github.com/yourusername/Contest-Log-Analyzer.git
   cd Contest-Log-Analyzer
   ```

2. **Start the application**:

   ```
   docker-compose up --build
   ```

3. **Access the web interface**: Open your browser to `http://localhost:8000`

For detailed installation instructions including CLI setup, see the Installation Guide.

Basic Usage (CLI)

```
# Analyze a single log
python main_cli.py --report summary MyLog.log

# Compare two logs with all reports
python main_cli.py --report all MyLog.log CompetitorLog.log

# Generate specific report types
python main_cli.py --report plot MyLog.log CompetitorLog.log
python main_cli.py --report chart MyLog.log CompetitorLog.log

# WRTC scoring for IARU-HF logs
python main_cli.py --report all --wrtc 2026 MyLog.log

# Debug options
python main_cli.py --report all --debug-data MyLog.log
```

Note: The analyzer supports 1, 2, or 3 logs. Single-log analysis provides detailed performance metrics. Multi-log analysis enables head-to-head comparison and identifies unique vs. common QSOs.

Environment Variables: You must set two environment variables before running:

- `CONTEST_INPUT_DIR`: Root directory containing your 'Logs' and 'data' subdirectories
- `CONTEST_REPORTS_DIR`: Output directory for generated reports (must be local, not cloud-synced)

For complete usage instructions, see the User Guide.

Documentation

Comprehensive documentation is available in the `Docs/` directory:

- **Installation Guide**: Complete installation instructions for Docker and CLI environments
- **User Guide**: How to run the analyzer, interpret reports, and understand output
- **Contributing Guide**: Engineering standards, versioning strategy, and contribution guidelines
- **Programmer's Guide**: Technical documentation for developers extending CLA
- **Report Interpretation Guide**: Detailed explanation of report formats and what they reveal
- **Style Guide**: Standards for report formatting and visualization
- **Git Workflow Setup**: Git workflow conventions and setup instructions
- **Technical Algorithms**:
    - Run & S&P Classification Algorithm
    - Callsign Lookup Algorithm
    - WPX Prefix Lookup

Architecture Highlights

Data-Driven Design

Contest behavior is controlled by data, not code. Contest rules, multiplier definitions, and parsing logic are defined in JSON files, allowing new contests to be added without modifying Python scripts.

Plugin Architecture

Reports and contest-specific logic modules can be dropped into appropriate directories—the main engine discovers and integrates them automatically.

Data Abstraction Layer

The `contest_tools.data_aggregators` package provides pure Python primitive outputs (dictionaries, lists, scalars) ensuring complete decoupling between data processing and presentation layers.

Stateless Web Architecture

The Django web dashboard is stateless and containerized, using the file system for session persistence. It shares HTML templates with the CLI via a unified presentation layer.

Project Structure

```
Contest-Log-Analyzer/
  main_cli.py                # CLI entry point
  contest_tools/             # Core analysis engine
      contest_log.py         # ContestLog class
      log_manager.py         # Multi-log orchestration
      report_generator.py    # Report execution engine
      version.py             # Version management (git-based)
      contest_definitions/   # JSON contest rule files
      reports/               # Report generator modules
      data_aggregators/      # Data analysis layer
      styles/                # Matplotlib & Plotly styling
      utils/                 # Shared utilities
  web_app/                   # Django web application
      analyzer/              # Views, forms, templates
      config/                # Django settings
  Docs/                      # Documentation
  CONTEST_LOGS_REPORTS/      # Sample logs and reports
```

Contributing

CLA is under active development. To contribute:

1. Fork the repository
2. Create a feature branch
3. Follow the coding standards in the Contributing Guide and Programmer's Guide
4. Submit a pull request

See Git Workflow Setup for branching conventions and workflow details.

License

Mozilla Public License, Version 2.0

Author & Contact

Mark Bailey, KD4D
Email: kd4d@kd4d.org

Acknowledgments

- CTY.DAT country file maintained by Jim Reisert, AD1C
- Contest rules and specifications from ARRL, CQ Magazine, DARC, and other sponsoring organizations
- The amateur radio contesting community for feedback and testing

---

New to contesting? Check out the ARRL Contest Basics Guide.