Metro Train Prediction App - CLI Command Reference

Author: David Morrison

Project Repo: <https://github.com/DavMorr/wmata-app>

[Overview 1](#_Toc1272938598)

[Command Summary 1](#_Toc2035630445)

[Metro Sync Command 1](#_Toc1289641578)

[Command Signature 1](#_Toc724576563)

[Purpose 1](#_Toc671931090)

[Command Options 1](#_Toc1385278460)

[Basic Usage 2](#_Toc1208636573)

[Standard Sync 2](#_Toc743669753)

[Validation Check 2](#_Toc749004791)

[Command Output 2](#_Toc2034318447)

[Successful Sync 2](#_Toc499246822)

[Validation Success 2](#_Toc1493532154)

[Validation Failure with Sync 2](#_Toc1851323741)

[Error Scenarios 3](#_Toc1076098944)

[API Connection Failure 3](#_Toc264068550)

[Partial Sync Failure 3](#_Toc1512904830)

[Return Codes 3](#_Toc786985991)

[Command Usage Examples 3](#_Toc768484393)

[Development Workflow 4](#_Toc582132455)

[Initial Setup 4](#_Toc599482656)

[Daily Development 4](#_Toc2059091546)

[Debugging Data Issues 4](#_Toc890893237)

[Production Deployment 4](#_Toc1139697805)

[Initial Production Setup 4](#_Toc852992466)

[Production Maintenance 5](#_Toc1090592703)

[Container/Docker Usage 5](#_Toc1747363290)

[With Laravel Sail 5](#_Toc2039286659)

[Direct Docker Execution 5](#_Toc1110830500)

[Error Handling 5](#_Toc2079131269)

[Common Error Types 5](#_Toc1704307659)

[1. WMATA API Errors 5](#_Toc364226590)

[2. Database Errors 6](#_Toc1964534404)

[3. Configuration Errors 7](#_Toc1441256147)

[Error Recovery Strategies 7](#_Toc488156564)

[Graceful Degradation 7](#_Toc531469527)

[Manual Data Validation 7](#_Toc972923778)

[Automation and Scheduling 8](#_Toc735989051)

[Laravel Task Scheduler 8](#_Toc514530042)

[Basic Scheduling 8](#_Toc1993263449)

[Advanced Scheduling with Logging 8](#_Toc202009769)

[Cron Job Setup 9](#_Toc815096374)

[Direct Cron Entry 9](#_Toc972721080)

[Docker/Sail Cron Setup 9](#_Toc1169861918)

[Monitoring Automation 9](#_Toc1078920365)

[Health Check Script 9](#_Toc352851954)

[Monitoring and Logging 10](#_Toc1116674954)

[Log File Locations 10](#_Toc342613063)

[Default Laravel Logs 10](#_Toc1160959501)

[Log Monitoring Commands 10](#_Toc1859607294)

[Performance Monitoring 10](#_Toc1610199511)

[Command Timing 11](#_Toc1597307815)

[Database Impact Monitoring 11](#_Toc1017488215)

[Success Metrics 11](#_Toc1698652562)

[Key Performance Indicators 11](#_Toc395368535)

[Health Check Queries 11](#_Toc185142034)

[Troubleshooting 12](#_Toc255726881)

[Debug Mode 12](#_Toc912158802)

[Enable Detailed Logging 12](#_Toc541982804)

[Check Service Registration 12](#_Toc704384352)

[Data Validation 12](#_Toc711400991)

[Manual Data Integrity Checks 13](#_Toc203140106)

[Cache Debugging 13](#_Toc1870024929)

[Performance Issues 13](#_Toc106237984)

[Slow Sync Diagnosis 13](#_Toc415724263)

[Memory Usage Monitoring 14](#_Toc1219985668)

[Recovery Procedures 14](#_Toc1500069778)

[Complete Data Reset 14](#_Toc1712099464)

[Selective Data Refresh 14](#_Toc825053627)

## Overview

The Metro Train Prediction App provides a specialized CLI command for managing Metro data synchronization with the WMATA API. This command handles the complex process of fetching, transforming, and storing Metro system data including lines, stations, addresses, and geographic path information.

### Command Summary

|  |  |  |
| --- | --- | --- |
| **Command** | **Purpose** | **Frequency** |
| metro:sync | Synchronize all Metro data from WMATA API | Daily or as needed |
| metro:sync --validate | Validate cache integrity without sync | Hourly monitoring |

## Metro Sync Command

### Command Signature

bash

sail artisan metro:sync {--validate : Validate cache integrity first}

### Purpose

The metro:sync command performs a complete synchronization of Metro system data from the WMATA API, including:

1. **Metro Lines** - All available metro lines (Red, Blue, Green, Orange, Silver, Yellow)
2. **Station Data** - Complete station information including coordinates and line assignments
3. **Station Addresses** - Physical addresses for each station
4. **Station Paths** - Geographic ordering and distance calculations for proper route display

### Command Options

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Usage** |
| --validate | Check cache integrity before performing sync | metro:sync --validate |

### Basic Usage

#### **Standard Sync**

bash

*# Basic synchronization (most common usage)*

sail artisan metro:sync

#### **Validation Check**

bash

*# Check cache integrity without syncing*

sail artisan metro:sync --validate

### Command Output

#### Successful Sync

Starting Metro data synchronization...

+---------------------+-------+

| Type | Count |

+---------------------+-------+

| Lines synced | 6 |

| Stations synced | 95 |

| Path entries synced | 95 |

+---------------------+-------+

Metro data sync completed successfully!

Stations will now display in proper sequence order

#### **Validation Success**

bash

sail artisan metro:sync --validate

*# Output:*

Checking cache integrity...

Cache is valid

#### **Validation Failure with Sync**

bash

sail artisan metro:sync --validate

*# Output:*

Checking cache integrity...

Cache validation failed, proceeding with sync...

Starting Metro data synchronization...

*# ... continues with full sync*

### Error Scenarios

#### **API Connection Failure**

Starting Metro data synchronization...

+---------------------+-------+

| Type | Count |

+---------------------+-------+

| Lines synced | 0 |

| Stations synced | 0 |

| Path entries synced | 0 |

+---------------------+-------+

Errors encountered:

• API request failed with status: 500

Sync failed: API request failed with status: 500

#### **Partial Sync Failure**

Starting Metro data synchronization...

+---------------------+-------+

| Type | Count |

+---------------------+-------+

| Lines synced | 6 |

| Stations synced | 95 |

| Path entries synced | 0 |

+---------------------+-------+

Errors encountered:

• Failed to sync path for line RD: API request failed with status: 401

• Failed to sync path for line BL: Line BL not found

### Return Codes

|  |  |  |
| --- | --- | --- |
| **Code** | **Status** | **Description** |
| 0 | SUCCESS | Command completed without errors |
| 1 | FAILURE | Command failed due to errors |

## Command Usage Examples

### Development Workflow

#### **Initial Setup**

bash

*# After fresh installation or database reset*

cd laravel-app

sail artisan migrate

sail artisan metro:sync

#### **Daily Development**

bash

*# Check if sync is needed*

sail artisan metro:sync --validate

*# If validation fails, sync data*

sail artisan metro:sync

#### **Debugging Data Issues**

bash

*# Clear cache and resync*

sail artisan cache:clear

sail artisan metro:sync

*# Check database after sync*

sail artisan tinker

App\Models\Line::count(); // Should be 6

App\Models\Station::count(); // Should be ~95

App\Models\StationPath::count(); // Should be ~95

### Production Deployment

#### **Initial Production Setup**

bash

*# Production deployment*

php artisan migrate --force

php artisan metro:sync

*# Verify data*

php artisan metro:sync --validate

#### **Production Maintenance**

bash

*# Weekly data refresh*

php artisan metro:sync

*# Log output for monitoring*

php artisan metro:sync >> /var/log/metro-sync.log 2>&1

### Container/Docker Usage

#### **With Laravel Sail**

bash

*# Standard Sail usage*

sail artisan metro:sync

*# Run in background*

sail artisan metro:sync &

*# Capture output*

sail artisan metro:sync | tee sync-output.log

#### **Direct Docker Execution**

bash

*# If running without Sail*

docker exec laravel-app php artisan metro:sync

## Error Handling

### Common Error Types

#### **1. WMATA API Errors**

**Error**: API request failed with status: 401

bash

*# Check API key configuration*

grep WMATA\_API\_KEY .env

*# Verify API key at* [*https://developer.wmata.com*](https://developer.wmata.com)

*# Test API key directly*

curl -H "api\_key: YOUR\_KEY" <https://api.wmata.com/Rail.svc/json/jLines>

**Error**: API request failed with status: 429

bash

*# Rate limit exceeded - wait and retry*

*# Check current rate limit*

sail artisan tinker

Cache::get('wmata\_api\_rate\_limit');

*# Clear rate limit if needed*

Cache::forget('wmata\_api\_rate\_limit');

**Error**: API request failed with status: 500

bash

*# WMATA API is down - retry later*

*# Check WMATA status at* [*https://developer.wmata.com*](https://developer.wmata.com)

#### **2. Database Errors**

**Error**: SQLSTATE[42S02]: Base table or view not found

bash

*# Run migrations first*

sail artisan migrate

*# Then sync data*

sail artisan metro:sync

**Error**: SQLSTATE[23000]: Integrity constraint violation

bash

*# Foreign key constraint issue - reset database*

sail artisan migrate:fresh

sail artisan metro:sync

#### **3. Configuration Errors**

**Error**: No application encryption key has been specified

bash

*# Check .env file has APP\_KEY*

grep APP\_KEY .env

*# Generate key if missing (only if necessary)*

sail artisan key:generate

**Error**: Class 'App\Services\WmataApiService' not found

bash

*# Check service provider registration*

cat bootstrap/providers.php

*# Ensure WmataServiceProvider is listed*

*# Clear config cache*

sail artisan config:clear

### Error Recovery Strategies

#### **Graceful Degradation**

bash

*# If sync fails, check what data is available*

sail artisan tinker

App\Models\Line::count();

App\Models\Station::count();

*# App may work with cached data even if sync fails*

#### **Manual Data Validation**

bash

*# Validate specific data integrity*

sail artisan tinker

*# Check for lines without stations*

App\Models\Line::whereDoesntHave('stationPaths')->get();

*# Check for stations without addresses*

App\Models\Station::whereDoesntHave('address')->count();

*# Check for orphaned paths*

App\Models\StationPath::whereDoesntHave('station')->count();

## Automation and Scheduling

### Laravel Task Scheduler

#### **Basic Scheduling**

php

*// app/Console/Kernel.php*

protected function schedule(Schedule $schedule): void

{

*// Daily sync at 3 AM*

$schedule->command('metro:sync')

->dailyAt('03:00')

[->emailOutputOnFailure('admin@example.com](mailto:->emailOutputOnFailure('admin@example.com)');

*// Hourly validation check*

$schedule->command('metro:sync --validate')

->hourly()

->skip(function () {

*// Skip if recent sync was successful*

return Cache::get('metro\_last\_sync\_success', false);

});

}

#### **Advanced Scheduling with Logging**

php

*// app/Console/Kernel.php*

protected function schedule(Schedule $schedule): void

{

$schedule->command('metro:sync')

->dailyAt('03:00')

->appendOutputTo(storage\_path('logs/metro-sync.log'))

[->emailOutputOnFailure('admin@example.com](mailto:->emailOutputOnFailure('admin@example.com)')

->before(function () {

Log::info('Metro sync starting');

})

->after(function () {

Log::info('Metro sync completed');

Cache::put('metro\_last\_sync\_success', true, 3600);

})

->onFailure(function () {

Log::error('Metro sync failed');

Cache::put('metro\_last\_sync\_success', false, 3600);

});

}

### Cron Job Setup

#### **Direct Cron Entry**

bash

*# Add to crontab (crontab -e)*

*# Daily sync at 3 AM*

0 3 \* \* \* cd /path/to/laravel-app && php artisan metro:sync >> /var/log/metro-sync.log 2>&1

*# Hourly validation*

0 \* \* \* \* cd /path/to/laravel-app && php artisan metro:sync --validate >> /var/log/metro-validation.log 2>&1

#### **Docker/Sail Cron Setup**

bash

*# For Sail environment*

0 3 \* \* \* cd /path/to/project && ./vendor/bin/sail artisan metro:sync >> /var/log/metro-sync.log 2>&1

### Monitoring Automation

#### **Health Check Script**

bash

#!/bin/bash

*# metro-health-check.sh*

*# Check if recent sync was successful*

LAST\_SYNC=$(docker exec laravel-app php artisan metro:sync --validate 2>&1)

if [[ $LAST\_SYNC == \*"Cache is valid"\* ]]; then

echo "Metro sync health: OK"

exit 0

else

echo "Metro sync health: FAILED"

echo "$LAST\_SYNC"

exit 1

fi

## Monitoring and Logging

### Log File Locations

#### **Default Laravel Logs**

bash

*# Application logs*

storage/logs/laravel.log

*# Custom sync logs (if configured)*

storage/logs/metro-sync.log

storage/logs/metro-validation.log

#### **Log Monitoring Commands**

bash

*# Real-time log monitoring*

tail -f storage/logs/laravel.log

*# Search for Metro-specific logs*

grep "WMATA" storage/logs/laravel.log

grep "Metro sync" storage/logs/laravel.log

*# Check for errors*

grep "ERROR" storage/logs/laravel.log | grep -i metro

### **Performance Monitoring**

#### **Command Timing**

bash

*# Time command execution*

time sail artisan metro:sync

*# Example output:*

*# real 0m45.123s*

*# user 0m2.456s*

*# sys 0m0.789s*

#### **Database Impact Monitoring**

bash

*# Before sync*

sail artisan tinker

DB::table('lines')->count();

DB::table('stations')->count();

DB::table('station\_paths')->count();

*# Run sync with timing*

time sail artisan metro:sync

*# After sync - verify counts*

### Success Metrics

#### **Key Performance Indicators**

* **Sync Duration**: Should complete within 60 seconds
* **Data Counts**: Lines (6), Stations (~95), Paths (~95)
* **Error Rate**: < 5% failure rate over time
* **Cache Hit Rate**: > 90% for prediction requests

#### **Health Check Queries**

sql

*-- Verify data completeness*

SELECT

(SELECT COUNT(\*) FROM lines) as line\_count,

(SELECT COUNT(\*) FROM stations) as station\_count,

(SELECT COUNT(\*) FROM station\_paths) as path\_count,

(SELECT COUNT(\*) FROM station\_addresses) as address\_count;

*-- Check for recent updates*

SELECT

MAX(updated\_at) as last\_line\_update,

(SELECT MAX(updated\_at) FROM stations) as last\_station\_update,

(SELECT MAX(updated\_at) FROM station\_paths) as last\_path\_update

FROM lines;

## Troubleshooting

### Debug Mode

#### **Enable Detailed Logging**

bash

*# Temporarily enable debug mode*

echo "APP\_DEBUG=true" >> .env

*# Run sync with verbose output*

sail artisan metro:sync -v

*# Disable debug mode when done*

sed -i 's/APP\_DEBUG=true/APP\_DEBUG=false/' .env

#### **Check Service Registration**

bash

*# Verify services are registered*

sail artisan route:list --path=api

*# Check service container bindings*

sail artisan tinker

app()->bound(App\Services\WmataApiService::class);

app()->bound(App\Services\MetroDataService::class);

### Data Validation

#### **Manual Data Integrity Checks**

bash

*# Check for missing data*

sail artisan tinker

*# Lines should have start/end stations*

App\Models\Line::whereDoesntHave('startStation')->get();

App\Models\Line::whereDoesntHave('endStation')->get();

*# Stations should have addresses*

App\Models\Station::whereDoesntHave('address')->count();

*# Paths should be sequential*

App\Models\StationPath::where('line\_code', 'RD')->orderBy('seq\_num')->pluck('seq\_num')->toArray();

// Should be [1, 2, 3, 4, 5, ...]

#### **Cache Debugging**

bash

*# Check cache contents*

sail artisan tinker

Cache::get('wmata.lines');

Cache::get('metro.lines.frontend');

Cache::get('wmata.predictions.A01');

*# Clear specific cache keys*

Cache::forget('wmata.lines');

Cache::flush(); // Clear all cache

### Performance Issues

#### **Slow Sync Diagnosis**

bash

*# Check network connectivity to WMATA*

curl -w "[@curl-format.txt](mailto:@curl-format.txt)" -o /dev/null -s <https://api.wmata.com/Rail.svc/json/jLines>

*# Monitor database during sync*

*# In separate terminal:*

watch -n 1 "sail artisan tinker --execute='echo App\Models\Station::count();'"

#### **Memory Usage Monitoring**

bash

*# Check memory usage during sync*

*# Monitor with top/htop while running:*

sail artisan metro:sync

*# Check PHP memory limits*

sail artisan tinker

ini\_get('memory\_limit');

### Recovery Procedures

#### **Complete Data Reset**

bash

*# Nuclear option - complete reset*

sail artisan migrate:fresh

sail artisan metro:sync

#### **Selective Data Refresh**

bash

*# Clear only Metro-related cache*

sail artisan tinker

Cache::forget('wmata.lines');

Cache::forget('wmata.stations.all');

Cache::forget('metro.lines.frontend');

*# Resync without database reset*

sail artisan metro:sync

This CLI command reference provides comprehensive documentation for managing the Metro Train Prediction App's data synchronization and maintenance operations.