AXIOM v2.0 - Quick Reference Guide

🚀 One-Line Commands

```batch
# Quick start (Windows)
setup_and_run.bat

# Quick start (Linux/macOS)
python AXIOM.py

# Resume interrupted session
setup_and_run.bat --resume

# Fast mode (experienced users)
setup_and_run.bat --workers 10 --delay 0.5

# Check system health
setup_and_run.bat --diagnostic
```

📊 Common SQL Queries

Get Statistics

```sql
```

```sql
-- Total skins
SELECT COUNT(*) FROM skins;

-- By status
SELECT download_status, COUNT(*)
FROM skins
GROUP BY download_status;

-- By category
SELECT category, COUNT(*)
FROM skins
GROUP BY category
ORDER BY COUNT(*) DESC;

-- Failed downloads
SELECT title, category, download_url
FROM skins
WHERE download_status IN ('download_failed', 'extraction_failed');

-- Most popular authors
SELECT author, COUNT(*) as skin_count
FROM skins
WHERE author != 'Unknown Author'
GROUP BY author
ORDER BY skin_count DESC
LIMIT 10;

-- Recently added
SELECT title, author, scraped_at
FROM skins
ORDER BY scraped_at DESC
LIMIT 20;

-- Skins with screenshots
SELECT title, screenshots
FROM skins
WHERE screenshots != '[]' AND screenshots != '';

-- Large files
SELECT title, file_size, local_path
FROM skins
```

```
WHERE file_size LIKE '%MB'
ORDER BY CAST(SUBSTR(file_size, 1, INSTR(file_size, ' ')-1) AS REAL) DESC;
```

🐍 Python Quick Scripts

Export Specific Category

python

```python
from AXIOM import SkinDatabase
import json

db = SkinDatabase('scraped_data/skins.db')

# Get all skins from a category
with db.get_connection() as conn:
    cursor = conn.execute(
        "SELECT * FROM skins WHERE category = ?",
        ("Suites",)
    )
    skins = [dict(row) for row in cursor.fetchall()]

# Export to JSON
with open('suites_only.json', 'w') as f:
    json.dump(skins, f, indent=2)

print(f"Exported {len(skins)} suites")
```

Find Skins by Author

python

```
from AXIOM import SkinDatabase

db = SkinDatabase('scraped_data/skins.db')

author_name = "YourFavoriteAuthor"

with db.get_connection() as conn:
    cursor = conn.execute(
        "SELECT title, category, download_url FROM skins WHERE author LIKE ?",
        (f"%{author_name}%",)
    )

    print(f"Skins by {author_name}:")
    for row in cursor.fetchall():
        print(f"  - {row['title']} ({row['category']})")
```

Retry Failed Downloads

```python
from AXIOM import SkinDatabase

db = SkinDatabase('scraped_data/skins.db')

# Reset failed downloads to pending
with db.get_connection() as conn:
    cursor = conn.execute(
        "UPDATE skins SET download_status = 'pending' WHERE download_status = 'download_failed'"
    )
    count = cursor.rowcount
    conn.commit()

print(f"Reset {count} failed downloads to pending")
print("Run scraper with --resume to retry")
```

Generate Custom Report

```python
```

```
from AXIOM import SkinDatabase
from datetime import datetime

db = SkinDatabase('scraped_data/skins.db')

with open('custom_report.txt', 'w', encoding='utf-8') as f:
    f.write(f"AXIOM Scraping Report\n")
    f.write(f"Generated: {datetime.now()}\n")
    f.write("=" * 70 + "\n\n")

    # Overall stats
    stats = db.get_statistics()
    f.write(f"Total Skins: {stats['total_skins']}\n\n")

    f.write("Status Breakdown:\n")
    for status, count in stats.get('by_status', {}).items():
        percentage = (count / stats['total_skins'] * 100) if stats['total_skins'] > 0 else 0
        f.write(f"  {status:20} {count:5} ({percentage:.1f}%)\n")

    f.write("\n" + "=" * 70 + "\n\n")

    f.write("Category Breakdown:\n")
    for category, count in stats.get('by_category', {}).items():
        f.write(f"  {category:30} {count:5}\n")

print("Report generated: custom_report.txt")
```

🔧 Troubleshooting Commands

Check Database Integrity

```bash
# SQLite integrity check
sqlite3 scraped_data/skins.db "PRAGMA integrity_check;"
```

Vacuum Database (Reclaim Space)

```bash
```

```
# Compact database after many deletes
sqlite3 scraped_data/skins.db "VACUUM;"
```

Count Actual Downloaded Files

```bash
# Windows
dir /s /b scraped_data\downloads\*.rmskin | find /c ".rmskin"
dir /s /b scraped_data\downloads\*.zip | find /c ".zip"

# Linux/macOS
find scraped_data/downloads -type f -name "*.rmskin" | wc -l
find scraped_data/downloads -type f -name "*.zip" | wc -l
```

Check Extracted Directories

```bash
# Windows
dir /s /a:d scraped_data\extracted_skins | find /c "<DIR>"

# Linux/macOS
find scraped_data/extracted_skins -type d | wc -l
```

📁 File Structure Reference

```
project/
├── AXIOM.py                    # Main scraper (v2.0)
├── axiom_tests.py              # Test suite
├── requirements.txt            # Dependencies
├── setup_and_run.bat           # Windows launcher
├── rainmeterui_categories.json     # Configuration
│
├── scraped_data/               # Output directory
│   ├── skins.db                # SQLite database ⭐
│   ├── complete_collection.json    # Full JSON export
│   ├── complete_collection.csv     # CSV export
│   ├── scraping_summary.txt        # Summary report
│   ├── axiom_scraper.log       # Detailed log
│   ├── downloads/              # Downloaded archives
│   │   └── [Category]/
│   │       └── [Skin]_[file].zip
│   └── extracted_skins/          # Extracted contents
│       └── [Category]/
│           └── [Skin]/
│               ├── Skins/
│               └── @Resources/
│
├── logs/                         # Setup logs
│   └── axiom_TIMESTAMP.log
│
└── venv/                         # Virtual environment
    └── ...
```

⚙️ Performance Tuning Matrix

Scenario	Workers	Delay	Batch	Notes
First Run	5	1.0	100	Safe defaults
Fast Network	10	0.5	200	High-speed home/office
Slow Network	3	1.5	50	Mobile/limited bandwidth
Server-Friendly	3	2.0	50	Respectful to source
Maximum Speed	15	0.3	300	Risk of rate limiting
Overnight Run	5	1.0	150	Balanced for stability

🚨 Error Code Reference

Exit Code	Meaning	Solution
0	Success	Scraping completed
1	Python not found	Install Python 3.8+
2	Dependencies failed	Run pip install manually
3	Config file error	Check JSON syntax
4	Network error	Check internet connection
5	Disk space error	Free up disk space
130	User interrupted	CTRL+C pressed - use --resume

💡 Pro Tips

Tip 1: Incremental Backups

batch

```batch
# Backup database periodically
copy scraped_data\skins.db scraped_data\skins_backup_%date%.db
```

Tip 2: Monitor Progress

batch

```batch
# In another terminal, watch database size
# Windows
dir scraped_data\skins.db

# Linux/macOS
watch -n 5 'ls -lh scraped_data/skins.db'
```

Tip 3: Estimate Time Remaining

sql

```sql
-- Check progress
SELECT
    download_status,
    COUNT(*) as count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM skins), 2) as percentage
FROM skins
GROUP BY download_status;
```

Tip 4: Schedule Scraping

batch

```batch
# Windows Task Scheduler
schtasks /create /tn "AXIOM Scraper" /tr "C:\path\to\setup_and_run.bat --resume" /sc daily /st 02:00

# Linux cron
0 2 * * * cd /path/to/axiom && ./run.sh --resume >> cron.log 2>&1
```

Tip 5: Export for Analysis

python

```python
# Export to pandas DataFrame for analysis
import sqlite3
import pandas as pd

conn = sqlite3.connect('scraped_data/skins.db')
df = pd.read_sql_query("SELECT * FROM skins", conn)

# Analysis examples
print(df.groupby('category')['title'].count())
print(df['download_status'].value_counts())
print(df[df['author'] != 'Unknown Author']['author'].value_counts().head(10))

conn.close()
```

🔄 Migration Script (v1.0 → v2.0)

Full Migration Script

python

```python
#!/usr/bin/env python3
"""
Migrate AXIOM v1.0 (JSON) to v2.0 (SQLite)
"""

import json
from pathlib import Path
from AXIOM import SkinDatabase, SkinMetadata
from datetime import datetime

def migrate_v1_to_v2(old_json_file, new_db_file):
    """Migrate old JSON format to new SQLite database"""

    print("=" * 70)
    print("AXIOM v1.0 → v2.0 MIGRATION")
    print("=" * 70)
    print()

    # Load old data
    print(f"Loading old data from: {old_json_file}")
    with open(old_json_file, 'r', encoding='utf-8') as f:
        old_data = json.load(f)

    old_skins = old_data.get('skins', [])
    print(f"Found {len(old_skins)} skins in old format")
    print()

    # Create new database
    print(f"Creating new database: {new_db_file}")
    db = SkinDatabase(new_db_file)

    # Migrate each skin
    migrated = 0
    failed = 0

    for i, skin_data in enumerate(old_skins, 1):
        try:
            # Convert tags and screenshots from list to JSON string if needed
            if 'tags' in skin_data and isinstance(skin_data['tags'], list):
                skin_data['tags'] = json.dumps(skin_data['tags'])
            if 'screenshots' in skin_data and isinstance(skin_data['screenshots'], list):
                skin_data['screenshots'] = json.dumps(skin_data['screenshots'])
            # Create SkinMetadata object
            skin = SkinMetadata(**skin_data)

            # Save to database
            if db.save_skin(skin):
                migrated += 1
            else:
                failed += 1
                print(f"  Failed to save: {skin.title}")

            # Progress indicator
            if i % 100 == 0:
                print(f"  Progress: {i}/{len(old_skins)} ({i/len(old_skins)*100:.1f}%)")

        except Exception as e:
            failed += 1
            print(f"  Error migrating skin {i}: {e}")

    print()
    print("=" * 70)
    print("MIGRATION COMPLETE")
    print("=" * 70)
    print(f"Successfully migrated: {migrated}")
    print(f"Failed: {failed}")
    print(f"Total: {len(old_skins)}")
    print()

    # Verify
    stats = db.get_statistics()
    print(f"Database now contains: {stats['total_skins']} skins")
    print()

    # Export verification
    verification_file = Path(new_db_file).parent / "migration_verification.json"
    print(f"Exporting verification file: {verification_file}")
    db.export_to_json(verification_file)

    print()
    print("Migration complete! You can now use the new database with AXIOM v2.0")
    print(f"Old file preserved at: {old_json_file}")
    print(f"New database at: {new_db_file}")


if __name__ == "__main__":
    import sys

    if len(sys.argv) != 3:
        print("Usage: python migrate_v1_to_v2.py <old_json_file> <new_db_file>")
        print()
        print("Example:")
        print("  python migrate_v1_to_v2.py scraped_data/complete_skin_collection.json scraped_data/skins.db")
        sys.exit(1)

    old_json = sys.argv[1]
    new_db = sys.argv[2]

    if not Path(old_json).exists():
        print(f"Error: Old JSON file not found: {old_json}")
        sys.exit(1)

    if Path(new_db).exists():
        response = input(f"Database {new_db} already exists. Overwrite? (y/n): ")
        if response.lower() != 'y':
            print("Migration cancelled")
            sys.exit(0)

    migrate_v1_to_v2(old_json, new_db)
```

📊 Database Schema

sql

```
-- Complete schema for reference
CREATE TABLE skins (
    url TEXT PRIMARY KEY,                  -- Unique identifier
    title TEXT NOT NULL,                   -- Skin name
    category TEXT NOT NULL,                -- Category name
    category_url TEXT,                     -- Category page URL
    page_number INTEGER DEFAULT 1,         -- Page number where found
    author TEXT,                           -- Author name
    description TEXT,                      -- Full description
    download_url TEXT,                     -- Direct download link
    download_filename TEXT,                -- Original filename
    file_size TEXT,                        -- Human-readable size
    downloads_count TEXT,                  -- Download count from site
    rating TEXT,                           -- Rating/stars
    tags TEXT,                             -- JSON array of tags
    screenshots TEXT,                      -- JSON array of image URLs
    created_date TEXT,                     -- Creation date
    updated_date TEXT,                     -- Last update date
    version TEXT,                          -- Version number
    compatibility TEXT,                    -- Rainmeter version
    scraped_at TEXT,                       -- When scraped (ISO format)
    download_status TEXT DEFAULT 'pending', -- Status: pending/downloaded/extracted/failed
    local_path TEXT,                       -- Path to downloaded file
    extracted_path TEXT,                   -- Path to extracted folder
    file_hash TEXT                         -- SHA-256 hash
);

-- Indexes for performance
CREATE INDEX idx_category ON skins(category);
CREATE INDEX idx_status ON skins(download_status);
CREATE INDEX idx_download_url ON skins(download_url);
```

🎯 Common Workflows

Workflow 1: Fresh Complete Scrape

```
batch
```

```
1. setup_and_run.bat --diagnostic          # Check system
2. setup_and_run.bat                       # Start scraping
3. Wait for completion (check logs)
4. Verify: dir /s scraped_data\downloads
```

Workflow 2: Resume After Interruption

batch

```
1. setup_and_run.bat --resume              # Resume scraping
2. Check progress: sqlite3 scraped_data\skins.db "SELECT download_status, COUNT(*) FROM skins GROUP BY downloa
```

Workflow 3: Update Collection

batch

```
# Scrapes only new skins (existing ones skipped automatically)
1. setup_and_run.bat                       # Discovers new skins
2. setup_and_run.bat --resume              # Downloads new ones only
```

Workflow 4: Export Subset

python

```python
# Export only extracted skins
from AXIOM import SkinDatabase
db = SkinDatabase('scraped_data/skins.db')

with db.get_connection() as conn:
    cursor = conn.execute("""
        SELECT title, category, extracted_path
        FROM skins
        WHERE download_status = 'extracted'
    """)

    with open('extracted_skins_list.txt', 'w', encoding='utf-8') as f:
        for row in cursor:
            f.write(f"{row['category']}/{row['title']}\n")
```

Workflow 5: Batch Processing

python

```
# Process skins in batches
from AXIOM import SkinDatabase
import time

db = SkinDatabase('scraped_data/skins.db')

batch_size = 50
processed = 0

while True:
    pending = db.get_pending_downloads(limit=batch_size)
    if not pending:
        break

    print(f"Processing batch of {len(pending)} skins...")

    # Your processing logic here
    for skin in pending:
        # Do something with skin
        pass

    processed += len(pending)
    print(f"Processed {processed} skins total")
    time.sleep(5)  # Brief pause between batches
```

🔍 Advanced Queries

Find Duplicate Titles

sql

```sql
SELECT title, COUNT(*) as count
FROM skins
GROUP BY title
HAVING count > 1
ORDER BY count DESC;
```

Find Skins Without Downloads

sql

```
SELECT title, category, url
FROM skins
WHERE download_url = '' OR download_url IS NULL;
```

Calculate Storage Usage

```sql
-- Requires file size parsing
SELECT
    category,
    COUNT(*) as skin_count,
    SUM(CAST(SUBSTR(file_size, 1, INSTR(file_size, ' ')-1) AS REAL)) as total_mb
FROM skins
WHERE file_size LIKE '%MB'
GROUP BY category
ORDER BY total_mb DESC;
```

Find Recently Updated Skins

```sql
SELECT title, author, updated_date
FROM skins
WHERE updated_date != ''
ORDER BY updated_date DESC
LIMIT 20;
```

Most Popular Tags

```sql
-- Requires JSON parsing (SQLite 3.38+)
SELECT
    json_each.value as tag,
    COUNT(*) as frequency
FROM skins, json_each(skins.tags)
WHERE tags != '[]'
GROUP BY tag
ORDER BY frequency DESC
LIMIT 20;
```

📦 Backup & Restore

Quick Backup

```batch
# Windows
copy scraped_data\skins.db backups\skins_%date:~-4,4%%date:~-7,2%%date:~-10,2%.db

# Linux/macOS
cp scraped_data/skins.db backups/skins_$(date +%Y%m%d).db
```

Full Backup (Everything)

```batch
# Windows
xcopy /E /I /Y scraped_data scraped_data_backup_%date:~-4,4%%date:~-7,2%%date:~-10,2%

# Linux/macOS
tar -czf axiom_backup_$(date +%Y%m%d).tar.gz scraped_data/
```

Restore Database

```batch
# Simply replace the database file
copy backups\skins_20241201.db scraped_data\skins.db
```

🎓 Learning Resources

SQLite Documentation

- SQLite Official Docs
- SQLite Tutorial

Python Async Programming

- asyncio Documentation
- aiohttp Documentation

Web Scraping Best Practices

- Respect robots.txt
- Use appropriate delays
- Handle errors gracefully
- Cache responses when possible

🏆 Achievement Unlocks

Track your progress:

- [ ] **First Run** - Complete initial scrape
- [ ] **Century Club** - Download 100+ skins
- [ ] **Millennium** - Download 1,000+ skins
- [ ] **Complete Collection** - Download all available skins
- [ ] **Resume Master** - Successfully resume after interruption
- [ ] **Query Expert** - Write custom SQL query
- [ ] **Export Pro** - Export data in 3+ formats
- [ ] **Automation King** - Schedule automated scraping

🆘 Emergency Commands

Kill Stuck Process

```batch
# Windows
taskkill /F /IM python.exe

# Linux/macOS
pkill -9 python
```

Recover Corrupted Database

```batch
# Export to SQL and reimport
sqlite3 skins.db .dump > backup.sql
sqlite3 new_skins.db < backup.sql
```

Reset Specific Category

```sql
-- Reset all skins in a category to pending
UPDATE skins
SET download_status = 'pending',
    local_path = '',
    extracted_path = '',
    file_hash = ''
WHERE category = 'Suites';
```

Clear Failed Downloads

```sql
DELETE FROM skins
WHERE download_status IN ('download_failed', 'extraction_failed');
```

📞 Quick Support Checklist

Before asking for help:

1. ✅ Ran `--diagnostic`
2. ✅ Checked `axiom_scraper.log`
3. ✅ Verified Python version (3.8+)
4. ✅ Confirmed disk space (5+ GB free)
5. ✅ Tested internet connectivity
6. ✅ Validated config JSON syntax
7. ✅ Read error messages in logs
8. ✅ Tried `--resume` for interruptions

Quick Reference v2.0
Last Updated: 2024

🔗 Related Files

- Complete Documentation
- Main Scraper
- Test Suite
- Requirements
- Batch Launcher

Keep this reference handy for quick command lookup!