PlexAnimeTools v2.0.0

A comprehensive PowerShell module for organizing anime, TV shows, cartoons, and movies for Plex Media Server. Automatically fetches metadata from MyAnimeList (Jikan API) and TMDb, renames files with proper episode titles, creates season folders, and downloads artwork.

Features

- **Automatic Content Detection** - Identifies anime, TV shows, cartoons, and movies
- **API Integration** - Fetches metadata from Jikan (MyAnimeList) and TMDb
- **Dual MAL Support** - Uses both Jikan (unofficial) and MAL Official APIs
- **Episode Title Fetching** - Renames files with accurate episode titles
- **Plex-Compatible Naming** - S01E01 format with customizable templates
- **Season Folders** - Automatically creates proper folder structure
- **Artwork Download** - Fetches and saves poster images
- **Multiple Config Profiles** - Default, Plex-strict, and Fansub-chaos presets
- **WhatIf Preview Window** - Visual preview of all changes with approval system
- **WhatIf Support** - Preview changes before applying them
- **GUI & CLI Modes** - Use graphical interface or command line
- **Comprehensive Logging** - Detailed logs with full transcript capture
- **Pipeline Support** - Process multiple folders efficiently
- **Interactive Launcher** - Easy-to-use menu system

Installation

Method 1: Manual Installation

1. Download or clone the module to one of your PowerShell module paths:

```powershell
$env:PSModulePath -split ';'
```

2. Common locations:

- C:\Users\<YourName>\Documents\PowerShell\Modules\PlexAnimeTools
- C:\Program Files\PowerShell\Modules\PlexAnimeTools

3. Verify installation:

```powershell
Import-Module PlexAnimeTools
Get-Command -Module PlexAnimeTools
```

Method 2: Import from Current Directory

```powershell
Import-Module .\PlexAnimeTools.psd1
```

Module Structure

```
PlexAnimeTools/
├── PlexAnimeTools.psd1          # Module manifest
├── PlexAnimeTools.psm1          # Module loader
├── Start-PlexAnimeTools.ps1     # Launcher script
├── README.md                    # Master Documentation
├── README.pdf                   # PDF Documentation
├── Config/
│   ├── default.json             # Default configuration
│   ├── plex-strict.json         # Strict Plex naming
│   └── fansub-chaos.json        # Preserve fansub tags
├── Public/
│   ├── Invoke-AnimeOrganize.ps1 # Main organization function
│   ├── Get-AnimeInfo.ps1        # Anime info retrieval
│   ├── Test-PlexScan.ps1        # Plex compatibility testing
│   └── Start-PlexGUI.ps1        # GUI launcher
├── Private/
│   ├── Detection.ps1            # Content type detection
│   ├── Jikan.ps1                # Jikan API (MAL unofficial)
│   ├── MALOfficial.ps1          # MAL Official API
│   ├── TMDb.ps1                 # TMDb API functions
│   ├── Naming.ps1               # File naming utilities
│   ├── Plex.ps1                 # Plex-specific functions
│   ├── Threading.ps1            # Logging & utilities
```

```
│   └── Preview.ps1            # WhatIf preview window
└── Logs/                      # Auto-created log directory
    ├── PlexAnimeTools_*.log   # Main application logs
    ├── Transcript_*.log       # Full console transcripts
    └── PlexScan_Errors_*.log  # Error reports
```

Documentation

- **README.md -** Complete markdown documentation (this file)
- **README.pdf -** Printable PDF version of documentation

To update the PDF after making changes to README.md, use any of these methods:

- VS Code with "Markdown PDF" extension
- Pandoc: `pandoc README.md -o README.pdf`
- Online converter: https://www.markdowntopdf.com/
- Browser: Open in browser and Print to PDF

Quick Start

Launch with Start Script (Easiest)

```powershell
# Navigate to PlexAnimeTools folder
cd C:\Path\To\PlexAnimeTools

# Run the launcher
.\Start-PlexAnimeTools.ps1
```

The launcher provides an interactive menu with options for:

- GUI mode
- CLI mode
- Quick start guide
- Plex library testing
- Module information

- README viewer

GUI Mode (Recommended for Beginners)

```powershell
Start-PlexGUI
```

1. Select source folder (where your media files are)
2. Select destination folder (your Plex library location)
3. Choose configuration profile
4. Check "Preview Only (WhatIf)" to test first
5. Click "Process"

CLI Mode

```powershell
# Preview mode with visual window (safe - no changes made)
Invoke-AnimeOrganize -Path "D:\Downloads\Anime" -OutputPath "D:\Plex\Anime" -WhatIf

# Process single folder
Invoke-AnimeOrganize -Path "D:\Downloads\Attack on Titan" -OutputPath "D:\Plex\Anime"

# Process multiple folders via pipeline
Get-ChildItem "D:\Downloads\Anime" -Directory |
    Invoke-AnimeOrganize -OutputPath "D:\Plex\Anime"

# Use strict Plex naming
Invoke-AnimeOrganize -Path "D:\Downloads" -OutputPath "D:\Plex" -ConfigProfile plex-strict
```

WhatIf Preview System

When you run with `-WhatIf`, a visual preview window opens showing all planned changes:

Preview Window Features:

- **Summary Statistics** - Total shows, files to rename, folders to create
- **Color-Coded Table:**

- 🔵 Light Blue = Folders being created
- 🟢 Light Green = New file creation
- 🟡 Light Yellow = File moves/renames

- **Detailed View** - Shows current filename → new filename for every file
- **Export to CSV** - Save preview for offline review
- **Two-Step Approval:**
  1. Review all changes in the preview window
  2. Click **Proceed** to execute, or **Cancel** to abort

Example:

```powershell
# Run in preview mode
Invoke-AnimeOrganize -Path "I:\Anime\Pokemon" -OutputPath "I:\Anime" -WhatIf

# Preview window opens showing:
# - Total Shows: 1
# - Files to Rename/Move: 731
# - Folders to Create: 2
#
# Table shows:
# Type    Show     Action  Source                    Destination
# Folder  Pokemon  Create  I:\Anime\Pokemon          I:\Anime\Pokemon\Season 01
# File    Pokemon  Move    pokemon_001.mkv           Pokemon - S01E01 - Pokemon I Choose You.mkv
# File    Pokemon  Move    pokemon_002.mkv           Pokemon - S01E02 - Pokemon Emergency.mkv
# ...

# Click "Proceed" button to execute all changes
# Click "Cancel" button to abort with no changes
```

The preview system ensures you never accidentally rename or move files incorrectly!

Configuration Profiles

default.json

- Balanced settings for most users
- Removes common fansub tags

- Uses English titles when available
- Standard Plex naming conventions

plex-strict.json

- Maximum Plex compatibility
- Strips ALL tags and quality markers
- English titles only
- Minimal file extensions (mkv, mp4 only)
- Stops on first error

fansub-chaos.json

- Preserves release group tags
- Keeps quality information in filenames
- Romaji titles preferred
- Backs up original filenames
- Processes all extensions

API Configuration

TMDb API Key (Required for TV Shows/Movies)

1. Get a free API key: https://www.themoviedb.org/settings/api
2. Open your config file (e.g., `Config/default.json`)
3. Replace `"YOUR_TMDB_API_KEY_HERE"` with your actual key:

```json
"TMDbAPIKey": "abc123yourkeyhere456"
```

Jikan API (MyAnimeList - Unofficial)

- No API key required
- Free public API
- Automatically rate-limited to respect API guidelines
- Used by default for all anime lookups

MAL Official API (Optional - Enhanced Features)

The module can also use MyAnimeList's official API for enhanced features:

1. **Create a MAL API Application** (free):
   - Go to: https://myanimelist.net/apiconfig
   - Click "Create ID"
   - App Name: `PlexAnimeTools` (or any name)
   - App Type: `web`
   - App Redirect URL: `http://localhost`
   - Fill in other required fields
   - Click "Submit"
2. **Copy your Client ID** and add it to your config:

```
json

  "MALClientId": "your_client_id_here"
```

3. **Test the connection:**

```
powershell

  Test-MALOfficialAPI
```

4. **First time authentication:**
   - When you first use MAL Official API, it will open your browser
   - Log in to MyAnimeList and authorize the app
   - Copy the redirect URL from your browser
   - Paste it into PowerShell when prompted
   - Token is saved and automatically refreshed

Benefits of Official MAL API:

- More detailed anime information
- Personal list integration (your anime list)
- Real-time updates

- Higher rate limits
- Official support

Both APIs work together:

- Jikan API (unofficial) - Used by default, no setup needed
- MAL Official API (optional) - Enhanced features when configured

---

Available Functions

Invoke-AnimeOrganize

Main function to organize media files.

Parameters:

- `-Path` - Source folder(s) containing media files
- `-OutputPath` - Destination Plex library location
- `-ConfigProfile` - Configuration to use (default, plex-strict, fansub-chaos)
- `-ForceType` - Override content detection (Auto, Anime, TV Series, Cartoon, Movie)
- `-WhatIf` - Preview mode - shows what would happen without making changes

Examples:

```powershell
# Preview changes with visual window
Invoke-AnimeOrganize -Path "D:\Downloads\Anime" -OutputPath "D:\Plex\Anime" -WhatIf

# Process with strict naming
Invoke-AnimeOrganize -Path "D:\Downloads" -OutputPath "D:\Plex" -ConfigProfile plex-strict

# Force as anime (skip auto-detection)
Invoke-AnimeOrganize -Path "D:\Downloads\Show" -OutputPath "D:\Plex" -ForceType Anime

# Pipeline processing
Get-ChildItem "D:\Downloads" -Directory | Invoke-AnimeOrganize -OutputPath "D:\Plex"
```

WhatIf Preview: When using `-WhatIf`, a graphical window opens showing:

- Summary of total shows, files, and folders

- Color-coded table of all planned changes
- Current filename → New filename for every file
- Option to export preview to CSV
- **Proceed** button to execute changes
- **Cancel** button to abort

The preview ensures you see exactly what will happen before any files are touched!

Get-AnimeInfo

Retrieves anime information from MyAnimeList.

Parameters:

- `-Title` - Anime title to search for
- `-MalId` - MyAnimeList ID for direct lookup
- `-IncludeEpisodes` - Include full episode list

Examples:

```powershell
# Search by title
Get-AnimeInfo -Title "Attack on Titan"

# Get details with episodes
Get-AnimeInfo -MalId 16498 -IncludeEpisodes

# Export episode list
Get-AnimeInfo -Title "One Piece" -IncludeEpisodes |
    Select-Object -ExpandProperty EpisodeList |
    Export-Csv "episodes.csv"
```

Test-PlexScan

Validates Plex compatibility of organized media.

Parameters:

- `-Path` - Path to organized media folder

- `-Detailed` - Show detailed validation results

Examples:

```powershell
# Test single show
Test-PlexScan -Path "D:\Plex\Anime\Attack on Titan"

# Test with details
Test-PlexScan -Path "D:\Plex\Anime\Naruto" -Detailed

# Test all shows in library
Get-ChildItem "D:\Plex\Anime" -Directory | Test-PlexScan
```

Validation Checks:

- Poster artwork exists (poster.jpg)
- Season folders properly named (Season 01, Season 02, etc.)
- Files follow S##E## naming convention
- Scores 0-100% compatibility

Interactive Options: After testing, you can choose:

- `1` - Continue (return to menu/prompt)
- `2` - Log detailed error report

Start-PlexGUI

Launches the graphical user interface.

Example:

```powershell
Start-PlexGUI
```

Features:

- Folder browser for source/destination
- Configuration profile selector

- Preview mode checkbox (WhatIf)
- Recursive subfolder processing option
- Real-time output log
- Progress tracking
- Error logging to Logs directory

Show-LatestLog

Opens the most recent log file.

Parameters:

- `-Type` - Log type to open (Main, Transcript, Error)

Examples:

```powershell
# View main application log
Show-LatestLog -Type Main

# View full console transcript
Show-LatestLog -Type Transcript

# View error report
Show-LatestLog -Type Error
```

Get-LatestLog

Gets path to the most recent log file.

Parameters:

- `-Type` - Log type to retrieve (Main, Transcript, Error, All)

Examples:

```powershell
```

```
# Get path to main log
$logPath = Get-LatestLog -Type Main

# Get all log files
$allLogs = Get-LatestLog -Type All

# Read last 50 lines of latest log
Get-Content (Get-LatestLog -Type Main) -Tail 50
```

Clear-OldLogs

Removes log files older than specified days.

Parameters:

- `-Days` - Number of days to keep (default: 30)

Example:

```
powershell

# Clean up logs older than 30 days
Clear-OldLogs -Days 30

# Keep only last 7 days
Clear-OldLogs -Days 7
```

Naming Conventions

TV Series Episodes

```
Show Title - S01E01 - Episode Title.mkv
Show Title - S01E02 - Episode Title.mkv
```

Movies

```
Movie Title (Movie).mkv
```

Specials

```
Show Title - S00E01 - Special Title.mkv
```

Folder Structure

TV Series:

```
Plex Library/
└── Show Title/
    ├── poster.jpg
    ├── Season 01/
    │   ├── Show Title - S01E01 - Episode Title.mkv
    │   └── Show Title - S01E02 - Episode Title.mkv
    └── Season 02/
        └── Show Title - S02E01 - Episode Title.mkv
```

Movies:

```
Plex Library/
└── Movies/
    └── Movie Title/
        ├── poster.jpg
        └── Movie Title (Movie).mkv
```

Supported File Types

Default Configuration:

- .mkv
- .mp4
- .avi
- .m4v
- .mov
- .wmv
- .flv
- .webm
- .ts

Additional formats can be added in configuration files.

Content Type Detection

The module automatically detects content types based on:

Anime Detection

- Keywords: anime, season, episode, ova, ona, special, BD, BluRay
- Release groups: SubsPlease, HorribleSubs, Erai-raws, Commie, etc.
- Japanese characters in folder names

Movie Detection

- Single video file in folder
- Keywords: movie, film
- Year in parentheses: (2024)

TV Series/Cartoon Detection

- Multiple episodes in folder
- Season numbering
- Standard TV show patterns

Logging

All operations are logged to the `Logs` directory within the module:

```
PlexAnimeTools/Logs/
├── PlexAnimeTools_YYYYMMDD_HHMMSS.log    # Main application log
├── Transcript_YYYYMMDD_HHMMSS.log        # Complete console output
└── PlexScan_Errors_YYYYMMDD_HHMMSS.log   # Test error reports
```

Log Types:

- **Main Log -** Application events, API calls, file operations

- **Transcript** - Complete PowerShell session capture (ALL console output)
- **Error Reports** - Detailed test results with issues

View Logs:

```powershell
# Open latest main log in Notepad
Show-LatestLog -Type Main

# Open latest transcript
Show-LatestLog -Type Transcript

# Open latest error report
Show-LatestLog -Type Error

# Get path to latest log
$logPath = Get-LatestLog -Type Main
Get-Content $logPath -Tail 50

# View all logs
Get-LatestLog -Type All
```

Log Levels:

- [INFO] - General information
- [SUCCESS] - Successful operations
- [WARNING] - Non-critical issues
- [ERROR] - Errors that occurred
- [DEBUG] - Detailed debugging info

Automatic Cleanup:

```powershell
# Remove logs older than 30 days
Clear-OldLogs -Days 30
```

The module automatically:

- Creates the Logs directory on first run
- Timestamps all log files

- Captures full console output via transcript
- Logs all errors with stack traces
- Provides easy access to latest logs

Troubleshooting

"No results found" Error

- Check folder name is recognizable (remove excessive tags)
- Try manual search with `Get-AnimeInfo -Title "Show Name"`
- Verify internet connection for API access

TMDb API Not Working

- Ensure API key is configured in config file
- Get free key from https://www.themoviedb.org/settings/api
- Check rate limits aren't exceeded (300ms between requests)

Files Not Moving

- Check destination path exists and is writable
- Verify source files aren't in use
- Use `-WhatIf` to preview without changes
- Check log files in `Logs` directory for detailed errors

Episode Titles Not Showing

- Verify anime is in MyAnimeList database
- Check episode count matches available episodes
- Some anime may have limited episode data

Character Encoding Issues

- Module uses UTF-8 encoding
- Log files saved with UTF-8
- Filenames sanitized for Windows compatibility

Finding Error Details

All errors are logged to the `Logs` directory:

```powershell
# View latest main log
Show-LatestLog -Type Main

# View complete console output (including all errors)
Show-LatestLog -Type Transcript

# Check specific error in log
$log = Get-LatestLog -Type Main
Select-String -Path $log -Pattern "ERROR" -Context 2, 5

# List all recent logs
Get-ChildItem (Join-Path $PSScriptRoot "Logs") |
    Sort-Object LastWriteTime -Descending |
    Select-Object -First 10 Name, LastWriteTime, Length
```

Best Practices

1. **Always test with -WhatIf first**

```powershell
Invoke-AnimeOrganize -Path "..." -OutputPath "..." -WhatIf
```

This opens a visual preview window where you can:

- Review every single change before it happens
- Export the preview to CSV for record-keeping
- Click Proceed only when you're confident
- Cancel if anything looks wrong

2. **Use the launcher script for easy access**

```powershell
```

```
.\Start-PlexAnimeTools.ps1
```

Provides interactive menu with all features

3. **Organize before moving to Plex**
   - Process to a staging folder first
   - Verify results with Test-PlexScan
   - Review the WhatIf preview carefully
   - Then move to Plex library
4. **Use appropriate config profile**
   - `default` - Most users
   - `plex-strict` - Maximum compatibility
   - `fansub-chaos` - Preserve original tags
5. **Backup original files**
   - Keep backups before processing
   - Or enable `BackupOriginalNames` in config
   - Use WhatIf preview to verify changes
6. **Monitor log files**
   - Review logs after processing: `Show-LatestLog -Type Main`
   - Check for errors or warnings
   - Logs saved to `Logs` directory automatically
   - Transcript captures ALL console output
7. **Force content type when needed**

```powershell
# Force as anime instead of auto-detect
Invoke-AnimeOrganize -Path "..." -OutputPath "..." -ForceType Anime -WhatIf
```

8. **Regular log cleanup**

```powershell
# Remove logs older than 30 days
Clear-OldLogs -Days 30
```

9. **Export previews for documentation**
    - Use the "Export to CSV" button in the preview window
    - Keep records of what was changed
    - Useful for troubleshooting or reverting changes

Examples & Use Cases

Organize Downloaded Anime

```powershell
# Preview first
Invoke-AnimeOrganize -Path "D:\Downloads\[SubsPlease] Attack on Titan" `
    -OutputPath "D:\Plex\Anime" -WhatIf

# Process after verification
Invoke-AnimeOrganize -Path "D:\Downloads\[SubsPlease] Attack on Titan" `
    -OutputPath "D:\Plex\Anime"
```

Batch Process Multiple Shows

```powershell
# Get all anime folders and process with preview
Get-ChildItem "D:\Downloads\Anime" -Directory |
    Where-Object { $_.Name -notmatch 'processed' } |
    Invoke-AnimeOrganize -OutputPath "D:\Plex\Anime" -WhatIf

# Review the preview window, then click Proceed to execute all
```

Clean Up Existing Library

```powershell
# Test compatibility of existing shows
Get-ChildItem "D:\Plex\Anime" -Directory | Test-PlexScan -Detailed

# Shows with issues
Test-PlexScan -Path "D:\Plex\Anime" | Where-Object { $_.Score -lt 70 }
```

Export Anime Information

```powershell
# Get comprehensive anime info
$info = Get-AnimeInfo -Title "Demon Slayer" -IncludeEpisodes

# Export episode list
$info.EpisodeList | Export-Csv "demon_slayer_episodes.csv" -NoTypeInformation

# View statistics
$info | Select-Object Title, Episodes, Score, Status, Genres
```

Process with Custom Settings

```powershell
# Use fansub-chaos profile to preserve tags with preview
Invoke-AnimeOrganize -Path "D:\Downloads\Anime" `
    -OutputPath "D:\Staging" `
    -ConfigProfile fansub-chaos `
    -WhatIf

# Review in preview window, export to CSV if needed, then proceed
```

Review Changes Before Execution

```powershell
# Run with WhatIf to see preview
Invoke-AnimeOrganize -Path "I:\Anime\Pokemon" -OutputPath "I:\Anime" -WhatIf

# Preview window shows:
# - 731 files will be renamed
# - Current names vs new Plex-compatible names
# - Color-coded by action type
# - Export button to save preview as CSV
# - Proceed button when ready
# - Cancel if anything looks wrong
```

Customization

Creating Custom Config Profiles

1. Copy existing profile from `Config/` folder
2. Modify settings as needed
3. Save with new name (e.g., `my-custom.json`)
4. Use with `-ConfigProfile my-custom`

Key Settings:

```json
{
  "NamingFormat": {
    "Episode": "{Title} - S{Season:D2}E{Episode:D2} - {EpisodeTitle}",
    "Movie": "{Title} (Movie)",
    "Special": "{Title} - S00E{Episode:D2} - {EpisodeTitle}"
  },
  "SearchSettings": {
    "RemoveTags": ["[SubsPlease]", "[Erai-raws]"],
    "RemoveQuality": ["1080p", "720p"],
    "PreserveGroupTags": false
  },
  "Features": {
    "DownloadArtwork": true,
    "RenameFiles": true,
    "BackupOriginalNames": false
  }
}
```

Requirements

- **PowerShell 5.1 or higher**
- **Windows** (Uses Windows Forms for GUI)
- **.NET Framework** (Included with Windows)
- **Internet connection** (For API access)
- **TMDb API Key** (For TV shows/movies - free)

Known Limitations

- Windows only (GUI requires Windows Forms)
- Jikan API rate limited (500ms between requests)
- TMDb API rate limited (300ms between requests)
- No native support for macOS/Linux (CLI may work with modifications)
- Episode data availability depends on MAL/TMDb databases

FAQ

Q: Do I need API keys? A: TMDb API key required for TV shows/movies. Jikan (anime) is free without key.

Q: Will this delete my original files? A: By default, files are moved (not copied). Use `-WhatIf` to preview. Enable `BackupOriginalNames` in config for safety.

Q: Can I undo changes? A: No built-in undo. Always test with `-WhatIf` first and keep backups.

Q: What if episode titles are wrong? A: Check the anime on MyAnimeList. Data comes directly from their database.

Q: Can I process files in-place? A: Not recommended. Module is designed to organize into clean Plex library structure.

Q: Does this work with other media servers? A: Yes, naming conventions work with Jellyfin, Emby, and other servers following Plex naming.

Contributing

This is an open-source community project. Contributions welcome!

How to Contribute:

- Report bugs in issue tracker
- Suggest features or improvements
- Submit pull requests
- Share your custom config profiles
- Help improve documentation

License

Released under MIT License. See LICENSE file for details.

Credits

- **APIs Used:**
  - Jikan API (MyAnimeList unofficial API)
  - TMDb API (The Movie Database)
- **Community:**
  - Plex community for naming guidelines
  - Anime community for fansub standards

Version History

v2.0.0 (Current)

- Complete rewrite as PowerShell module
- Added GUI interface
- Multiple configuration profiles
- TMDb integration for TV/movies
- WhatIf support
- Pipeline support
- Comprehensive logging
- Episode title fetching
- Artwork download
- Plex compatibility testing

Support

- Check logs on desktop for errors
- Use `-WhatIf` to preview changes

- Test with small batches first
- Review this README for examples
- Check configuration files for settings

Happy organizing! 📺🎬🎀