# Lightroom and Photo Processing Automation Guide

## Overview
This guide covers automation possibilities for Adobe Lightroom and other high-end photo processing applications using command scripts and programming.

## Adobe Lightroom Automation Options

### 1. Lightroom Classic Scripting (Lua)
Lightroom Classic supports Lua scripting for automation:

What you can automate:
- Apply presets to multiple photos
- Batch export with specific settings
- Organize photos by metadata
- Apply adjustments programmatically
- Generate contact sheets
- Watermarking
- Rename files based on EXIF data

Example Lua Script:
```lua
-- Apply preset to selected photos
local LrApplication = import 'LrApplication'
local catalog = LrApplication.activeCatalog()
local photos = catalog:getTargetPhotos()

for _, photo in ipairs(photos) do
    photo:applyPreset("My Custom Preset")
end
```

### 2. Lightroom Mobile/Cloud API
For Lightroom Mobile and Cloud versions:

What you can do:
- Access photos via Adobe Creative SDK
- Apply edits remotely
- Sync collections
- Download/upload photos
- Manage metadata

Python Example:
```python
import requests

class LightroomAPI:
    def __init__(self, access_token):
        self.access_token = access_token

    def get_photos(self):
        # Get all photos from Lightroom Cloud
        headers = {'Authorization': f'Bearer {self.access_token}'}
        response = requests.get('https://lr.adobe.io/v2/catalog', headers=headers)
        return response.json()
```

## Alternative Photo Processing Applications

### 1. Darktable (Open Source)
Automation via:
- Command line tools
- Lua scripting
- Batch processing scripts

Capabilities:
- Raw file processing
- Batch operations
- Export workflows
- Metadata management

### 2. Capture One
Automation via:
- AppleScript (Mac)
- COM automation (Windows)
- Command line tools

What you can automate:
- Session management
- Batch processing
- Color grading
- Export settings

### 3. RawTherapee
Automation via:
- Command line interface
- Batch processing
- Profile application

## Python-Based Photo Processing

### Core Libraries
- **PIL/Pillow**: Basic image manipulation
- **OpenCV**: Advanced computer vision
- **RawPy**: Raw file processing
- **ExifRead**: Metadata extraction
- **ImageIO**: Multiple format support

### Example Batch Processing Script
```python
from PIL import Image, ImageEnhance
import os
from pathlib import Path

def process_photos(input_dir, output_dir):
    """Process all photos in a directory"""
    # Create output directory
    Path(output_dir).mkdir(exist_ok=True)

    # Process each photo
    for file_path in Path(input_dir).glob('*.jpg'):
        with Image.open(file_path) as img:
            # Apply enhancements
            enhancer = ImageEnhance.Contrast(img)
            enhanced = enhancer.enhance(1.2)

            # Save processed image
            output_path = Path(output_dir) / f"processed_{file_path.name}"
            enhanced.save(output_path, quality=95)
```

## Automation Use Cases

### 1. Wedding Photography Workflow
- Auto-import from memory cards
- Apply wedding presets
- Generate client galleries
- Watermark proofs
- Create slideshows

### 2. Real Estate Photography
- Batch HDR processing
- Apply real estate presets
- Resize for web/mobile
- Generate property galleries
- Upload to MLS systems

### 3. Portrait Photography
- Skin retouching automation
- Color grading presets
- Batch export for social media
- Client proofing workflows

### 4. Event Photography
- Auto-categorize by time/location
- Apply event-specific presets
- Generate contact sheets
- Batch upload to galleries

## Command Line Tools

### 1. ImageMagick
Powerful command-line image processing:

```bash
# Resize images
convert input.jpg -resize 1920x1080 output.jpg

# Apply effects
```

```
convert input.jpg -blur 0x1 -sharpen 0x1 output.jpg

# Batch processing
mogrify -resize 50% *.jpg
```

### 2. ExifTool
Metadata manipulation:
```bash
# Read EXIF data
exiftool photo.jpg

# Modify metadata
exiftool -Artist="Your Name" *.jpg

# Remove metadata
exiftool -all= photo.jpg
```

### 3. FFmpeg
Video and image processing:
```bash
# Convert image formats
ffmpeg -i input.png output.jpg

# Create timelapse from images
ffmpeg -framerate 30 -i img%03d.jpg output.mp4
```

## Workflow Automation Examples

### 1. Automated Import Workflow
```python
import shutil
from pathlib import Path
import datetime

def auto_import_photos(source_dir, destination_dir):
    """Automatically import and organize photos"""

    today = datetime.date.today()
    session_folder = Path(destination_dir) / f"Session_{today}"
    session_folder.mkdir(exist_ok=True)

    # Copy and rename files
    for i, file_path in enumerate(Path(source_dir).glob('*.jpg')):
        new_name = f"{today}_{i+1:03d}.jpg"
        shutil.copy2(file_path, session_folder / new_name)
```

### 2. Batch Preset Application
```python
def apply_preset_batch(photo_list, preset_name):
    """Apply preset to multiple photos"""
    for photo_path in photo_list:
        # This would integrate with Lightroom API
        # or use command-line tools
        apply_preset(photo_path, preset_name)
```

### 3. Automated Export
```python
def export_for_web(photo_path, output_dir):
    """Export photos optimized for web"""

    with Image.open(photo_path) as img:
        # Resize for web
        img.thumbnail((1920, 1080), Image.Resampling.LANCZOS)

        # Optimize quality
        img.save(output_dir / f"web_{photo_path.name}",
                quality=85, optimize=True)
```

## Getting Started

### 1. Choose Your Approach
- **Lightroom Classic**: Use Lua scripting
- **Lightroom Cloud**: Use Adobe Creative SDK
- **General Processing**: Use Python + PIL/OpenCV
- **Command Line**: Use ImageMagick/ExifTool

### 2. Start Simple
- Begin with basic batch operations
- Test on small photo sets
- Gradually add complexity

### 3. Common First Projects
- Batch resize photos
- Apply watermarks
- Organize by date/metadata
- Convert file formats
- Generate thumbnails

## Tools and Resources

### Software
- **Adobe Lightroom Classic**: Professional photo management
- **Darktable**: Open-source alternative
- **Capture One**: High-end raw processing
- **RawTherapee**: Free raw processor

### Programming Libraries
- **PIL/Pillow**: Python image processing
- **OpenCV**: Computer vision
- **RawPy**: Raw file handling
- **ImageIO**: Format support

### Command Line Tools
- **ImageMagick**: Image manipulation
- **ExifTool**: Metadata handling
- **FFmpeg**: Media processing

## Conclusion

Automation can significantly speed up photo processing workflows. Start with simple batch operations and gradually build more complex automation as needed. The key is to identify repetitive tasks and automate them systematically.

Choose the right tool for your needs:
- Lightroom users: Lua scripting or Adobe Creative SDK
- General users: Python + PIL/OpenCV
- Command-line fans: ImageMagick/ExifTool

Remember to always test automation on small batches before processing large photo collections!