LYTRO

ILLUM

Lytro Power Tools Beta User Manual

Documentation Version 1.0 – August 18th, 2015

Introduction to Lytro Power Tools

What are the Lytro Power Tools?

Lytro Power Tools (LPT) are a set of Python camera tools, image control tools, web tools, recipe tools and support for Android Debug Bridge (ADB) command line instructions to write new camera applications. For programmers, developers and researchers, or anyone who is interested in experimenting with Light Field imagery using their Lytro camera, the LPT provides full access for complete control. Program additional functionality, create and execute scripts to automate processes in the camera, customize upload processes and content management, and gain deeper control at every step from raw LFR to Living Picture to animation.

Requirements - System, Hardware, Expertise

Recommended system hardware

PC

64-bit Windows® 7 or 64-bit Windows® 8
8 GB Ram or more
Intel® Core™ i5 quad-core processor or better
DirectX® 11
Graphics:
AMD Radeon™ 7970 or better
NVIDIA® GeForce® GTX 760 or better
Python® 2.7.10

Mac

Mac OS X® 10.9 or better
8 GB Ram or more
Intel Core i5 processor or better
Graphics:
NVIDIA GeForce GTX 760 or better
AMD Radeon 7970 or better
Python® 2.7.10

Expertise with command line tools

The Lytro Power Tools' (LPT) Camera Tool uses Python and Android Debug Bridge (ADB) to provide Root level access to your Lytro camera's Android operating system. If you are not comfortable with command line tools, Python or programming, you should not use the Lytro Power Tools resources.

Cable

USB 3.0 Cable (included with ILLUM)

Supported Light Field Cameras

Lytro ILLUM

Online manual for Lytro ILLUM:
http://manuals.lytro.com/illum/

- Camera Tool
- Camera Controls
- LFP Tool
- Web Tool
- Recipe Tool

Lytro First Generation

Online knowledge base for Lytro 1st Gen Camera:
https://support.lytro.com/hc/en-us/sections/200163680-First-Generation-Lytro-Camera

- LFP Tool
- Web Tool
- Recipe Tool

Software Resources

Lytro Resources

- cameratool
- cameracontrols
- lfptool
- webtool
- recipetool
- Lytro Desktop App https://lytro.com/desktop/

Third Party Resources

- Android™ Debug Bridge (ADB)

Warranty Notice

This documentation describes specific ways to obtain and use Root level access to your Lytro camera's Android operating system. Any other access and operations not described in this document are considered by Lytro to be unauthorized modifications to the Lytro camera. Unauthorized modifications can permanently damage and/or render the Lytro camera inoperable and are not covered under your camera's warranty.

The authorized modifications and operations are:

- Develop and execute Light Field Processing Tool Python scripts as listed within the Light Field Processing Tool Commands section below
- Develop and execute Camera Tool Python scripts as listed within the Camera Tool section below, modifying the Commands and Arguments listed within the Camera Tool Help Menu.
- Develop and execute Camera Control Python scripts as listed within the Camera Control section below, modifying the Commands and Arguments listed within the Camera Control Help Menu.
- Develop and execute Web Tool Python scripts as listed in the Web Tool section below.
- Develop and execute Recipe Tool Scripts as listed in the Recipe Tool section below

Glossary

List of terms and acronyms used in the document

4D Coordinates:X,Y,U,V data for working with a External Standardized Light Field (ESLF)

ADB: Android Debug Bridge: command line tool that lets you communicate with an emulator instance or connected Android-powered device.

Camera Tool Camera Tool is a Python interface which enables various capture features of the camera to be scripted and controlled.

Camera Control Tool is a Python interface which enables running specifc camera features.

CCM: Color Correction Matrix

Depth Map Image: an image map which graphically indicates the depth of a captured scene

ESLF: External Standardized Light Field

Extended Depth of Field (EDOF): EDOF describes an image where everything is in focus. These are post-processed from .LFR captures.

FNC: Frame-space Normalized Coordinate, normalized image coordinates where [0.0, 0.0] is top left and [1.0, 1.0] is bottom right.

FW: Firmware for the camera

INT: Integer

Lambda Values: Lambda values are a measurement of dioptric distance. The Lambda Value corresponding to a particular object distance depends on the focus and zoom settings of the lens at the time the picture was taken. For a given Light Field Picture, objects closer to the lens have lower negative values of lambda, while objects which are further away have more positive Lambda Values.

LFE: Light Field Engine, the post-capture light field image processing system.

LFP File: Light Field Picture: A file (.LFP) that contains a series of pictures with light and color, depth information and metadata which were post processed from one raw .LFR capture.

LFP Tool: The Light Field Processing Tool can process raw light field pictures (.LFR) files, including

- adjust parameters of the image
- generate images EDOF images
- generate a depth map
- generate images focused at different distance planes
- create perspective shift images
- create Recipes for post-processing controls
- work with metadata in an LFR file

LFP Schema: Light Field Picture format underlying data framework

LFR - Raw Light Field Picture: A Light Field Raw file (.LFR) that contains raw light field data or a raw.lfp file that contains raw light field data along with recipe or acceleration data.

Light Field: A Light Field is all of the light rays traveling in every direction throughout a volume of space in a given area. Light field photography captures light, color, directional information and depth information from all light rays interacting with the objects in a scene.

LPT - Lytro Power Tools (LPT) are a set of Python camera and image control tools, web tools and support for Android Debug Bridge (ADB) command line instructions. The LPT allows users to program additional functionality, create and execute scripts to automate processes in the camera, customize upload processes and gain deeper access to the LFR.

Perspective Shift: Perspective Shift allows changing the point of view in a picture after the picture has already been taken. This will reveal different angles of view for the foreground, middle, and background subjects.

Recipe: A Recipe is a set of parameters used to make image adjustments to a given LFR during image processing.

Recipe Tool is a set of Python commands which provide the ability to create recipe files.

RNC: Reorient-space Normalized Coordinate, coordinates used during living picture playback. Because this coordinate system is only used during Living Picture playback, parameters using this system (Zoom, Pan) should be excluded for generating single image outputs with LFP Tool/Recipe Tool.

Warp / Warp Stack: A Warp LFP with a stack of images is generated from the LFR and is the appropriate input to the Lytro desktop and web players.

Web Tool is a set of Python commands which provide the ability to upload and manage your LFP pictures and albums on pictures.lytro.com.

Components list

Lytro Resources

- Camera Tool
- Camera Controls
- LFP Tool
- Recipe Tool
- Web Tool
- Lytro Desktop App: https://lytro.com/desktop/

Lytro Desktop App is a free software image editor from Lytro that provides a powerful set of tools to help you create amazing Living Pictures with custom animations in 2D or cinematic 3D. Adjust aperture, shift focal planes, tilt, rotate and adjust image properties.

Third Party Resources

- Android Debug Bridge: http://developer.android.com/tools/help/ADB.html
- Android Debug Bridge (ADB) is a powerful third party command line tool that lets you communicate with an emulator instance or your connected ILLUM camera. It is a client-server program that includes three components:
- Client - runs on your development machine. You can invoke a client from a shell by issuing an ADB command. Other Android tools such as the ADT plugin and DDMS also create ADB clients.
- Server - runs as a background process on your development machine. The server manages communication between the client and the ADB daemon running on an emulator or device.
- Daemon - runs as a background process on each emulator or device instance.

Quick Start Guide

Note: Python and the 3rd Party Python libraries listed below are required for the Lytro Power Tools installer to run.

Python & 3rd Party Python Libraries

Required:
Python 2.7.10: To install Lytro Power Tools, Python version 2.7.10 is required..

- **OS X:** Note that OS X Yosemite (version 10.10) ships with Python version 2.7.6. Upgrading can be accomplished in several ways, but it is recommended that the installer is either downloaded and ran from python.org or it is installed using a package manager, such as brew.

- **Windows:** It is recommended that the 2.7.10 installer is downloaded and executed from python.org
  Install Notes: During installation, **it is essential** that the *Add python.exe to Path* option be enabled on the *Customize Python 2.7.10* installer screen. This adds Python install directories to the user or system path environment variable and allows Python (as well as Lytro Power Tools) to be executed from any directory in the system. Also, if *Install for all users* is selected, it is required that the current user account log out of and back into Windows before the system environment path is updated.

NumPy, SciPy: Both of these libraries are used extensively for various mathematical functions within LFP Tool and Recipe Tool. The Lytro Power Tools setup script will fail to install if these modules are not found or cannot be automatically installed. Due to the nature of these libraries' installations, automatic installation during Lytro Power Tools setup is not always possible.

- **OS X:** If the recommended Python version (2.7.10) is installed, the setup script should not have an issue installing these libraries. If the setup script fails to install, it is recommended that these two libraries are installed with Python Package Index (pip), or downloading these libraries' installers from their respective websites and installing manually.
- **Windows:** The easiest method of installation is to install the pre-built Windows installers linked to from SciPy's *Installing the SciPy Stack* page (direct link to the pre-built installers). The most recent cp27 versions (i.e., cpython 2.7) are required (as of writing this document: *numpy=1.9.2, scipy=0.16*), for 32-bit or 64-bit Python, whichever is applicable. Once downloaded, use the Python Package Installer (pip) to install the download *whl* files (e.g. **`pip install path\to\numpy-1.9.2+mkl-cp27-none-win32.whl`**).

The following packages are automatically installed during Lytro Power Tools installation:

- jsonschema: validates LFP meta information and recipe files against Lytro's LFP Schema
- dictdiffer: shows differences between Python dictionary objects
- pytweening: animation easing functions for Recipe Tool

Recommended:
matplotlib: While not used as extensively as NumPy or SciPy, matplotlib is used for plotting animation data in Recipe Tool. This is a convenience feature and not a necessity to work with Lytro Power Tools.
OS X/Windows: It is recommended that Python Package Index (pip) is used to install matplotlib

Lytro Power Tools Installation

1. Launch the Lytro Power Tools installation executable
2. Accept the installer's license agreement and extract its contents to a desired location
3. From a terminal, change directories to the extracted folder and run the setup.py installer

```
$ cd path/to/ lytro-power-tools-1.0.0b0
$ python setup.py install
```

Note: To uninstall Lytro Power Tools, use pip to remove the lytro-power-tools package:

```
$ pip uninstall lytro-power-tools
```

Lytro User Forum

Have any questions, feature requsts or comments? Looking to collaborate or share your LPT project? Join the conversation at: http://forums.lytro.com

Enabling ADB on Mac

You will need to install some internal components to enable ADB to interact with the camera.

1) Go here to get the ADB installer. https://code.google.com/p/adb-fastboot-install/downloads/list

2) Run the Mac installer script:

```
$ ./ADB-Install-Mac.sh
```

3) Send these two commands via Terminal.

```
$ mkdir ~/.android
```

```
$ echo 0x24CF >> ~/.android/adb_usb.ini
```

If you are having problems connecting camera with ADB then run this on Terminal and disconnect, reconnect camera.

```
$ adb kill-server
```

Enabling ADB on Windows

Refer to the detailed ***Lytro Power Tools / ADB Installation for Windows*** document here:
https://support.lytro.com/hc/en-us/articles/204797410

Camera

To control your Lytro Light Field camera, Lytro Power Tools includes two components, which permit you to script and automate camera and capture functions. The Camera Tool uses Python to control capture. Camera Tool Commands can be utilized to write new camera applications. Camera Control allows you to directly control camera features and settings when tethered via USB cable.

Camera features that can be controlled

White Balance
Set and get current White Balance setting

Exposure Mode
Set and get current Exposure Mode setting

Exposure Compensation
Set and get current Exposure Compensation setting

ISO
Set and get current ISO setting

Shutter Speed
Set and get current Shutter Speed setting

Focus Step
Set and get current Focus Step (Distance) setting

Zoom Step
Set and get current Zoom Step (Focal Length) setting

Optical Offset
Set and get current Optical Offset (see ILLUM User Manual for details) setting

Focus Lock
Set and get current Focus Lock state

Zoom Lock
Set and get current Zoom Lock state

AE Lock
Set and get current Auto Exposure Lock state

Depth Feedback
Set and get current Depth Feedback state

Focus Mode
Set and get current Focus Mode (auto or manual) setting

Shutter Mode
Set and get current Shutter Mode (single or continuous) setting

Self Timer
Set and get current Self Timer (0, 2, 10 seconds) setting

Sleep Timer
Set and get current Sleep Timer value

Exposure Bracketing
Set and get current Exposure Bracketing mode and value

Focus Bracketing
Set and get current Exposure Bracketing mode and value

Metering Mode
Set and get current Metering Mode (average or evaluative) setting

Flash Sync
Set and get current Flash Sync (back or front curtain) setting

Flash AF Assist
Set and get current Flash Auto Focus Assist setting

Flash Exposure Compensation
Set and get current Flash Exposure Compensation setting

Send Touch Event
Send Touch Event to actuate a screen touch by specifying screen's X and Y coordinates

Send Swipe Event
Send Swipe Event to actuate a screen swipe by specifying up, down, left or right

Physical Controls
Send button press event to actuate shutter, depth assist, power, AF, AEL, hyper focal or FN button push.

Capture Image
Send a capture command to record an image or sequence of images

Warranty Notice

This documentation describes specific ways to obtain and use Root level access to your Lytro camera's Android operating system. Any other access and operations not described in this document are considered by Lytro to be unauthorized modifications to the Lytro camera. Unauthorized modifications can permanently damage and/or render the Lytro camera inoperable and are not covered under your camera's warranty.

Authorized Camera Tool Modifications and Operations

Authorized Camera Tool modifications and operations are:

- ***Develop and execute Camera Tool Python scripts as listed within the Camera Tool Commands and Arguments section below.***
- ***Develop and execute Camera Control Python scripts as listed within the Camera Control section below, modifying the Commands and Arguments listed within the Camera Control Help Menu.***

About Camera Tool

Camera Tool contains pre-programmed capture sequence applications. The complete list of scripts that can be created are listed in the Camera Tool help menu, which can be accessed by entering **`-h`** in the python terminal window.

Individual Command level help details along with available Arguments can be accessed by entering the Help argument data (-h) into the python terminal window with the following convention: **`Command name -h`** For example, the Help argument for Focus Sweep would be: **`cameratool focus-sweep -h`**

Enabling ADB to use Camera Tool

1) The ILLUM Firmware should be 2.0 or newer
2) To Enable ADB in the ILLUM Hold down 'AEL' + half-press Lytro button on boot.
3) Plug camera in via the USB 3 cable
4) Check if ADB is running:
   Send command "ADB shell" should enter "shell@android".
   If this does not work re-try step 2 or restart ADB by sending command "ADB kill-server"

Transferring and running scripts

Camera Tool generates a python script and JSON file. The python file contains the function instructions and values. The JSON file provides data and parameters to be executed by the script. Camera Tool verifies current ADB and camera state, sd-card capacity and capture parameters and both the python and JSON files are transferred automatically to the camera when it is connected to the computer via USB. Your run.py file can be viewed on the camera's SD card.

The camera must be turned on and plugged in via USB to the computer you are running Camera Tool on. Once the script and JSON files are copied onto the camera's SD card, the ILLUM will automatically generate a control interface to start the script.

Camera Tool Scripts

Captures

This script takes a series of captures. Users specify number of captures and sleep interval.

Usage: **`captures [parameter]`**

`--pictures`

- Total number of pictures to take.
- i.e "**`captures --pictures 50`**".
- If no argument is specified it defaults to 10.

`--interval`

- Time interval in between each capture.
- i.e "**`captures --interval 15`**".
- If no argument is specified it defaults minimum interval time. This number may vary between 2 to 10 seconds depending on SD Card write speed.

Continuous Captures

This script takes multiple series of burst captures. Users specify the burst size, the time interval and the number of iterations.

Note: This tool may not perform properly at shutter speeds slower than 1/25

Usage: **`cont-captures [parameter]`**

`--size`

- Number of consecutive pictures to take for each series of burst captures
- i.e "**`cont-captures --size 5`**". Possible sizes are 1 - 8.
- If no argument is specified it defaults to 3.

`--repeat`

- Repeat series n times
- i.e "**`cont-captures --repeat 5`**".
- If no argument is specified it defaults to 1.

`--interval`

- Time interval in between each burst capture series (in secs).
- i.e "**`cont-captures --interval 60`**".
- If no argument is specified it defaults minimum interval time. This number varies depending on burst capture size and SD Card write speed

Exposure Bracketing

This script takes multiple series of exposure bracketed shots. Users specify bracket series size, exposure stop increments, time intervals, and the number of iterations.

Usage: **`exp-bracketing [parameter]`**

`--size`

- Number of consecutive pictures to take for each exposure bracketed series.
- i.e "**`exp-bracketing --size 5`**". Defaults to 3.

`--offset`

- Exposure stop increments
- i.e "**`exp-bracketing --offset 1`**". Possible increments are: .3, .6, 1, 2. Defaults to .3.

`--repeat`

- Repeat exposure bracketed cycle n times
- i.e "**`exp-bracketing --repeat 5`**". Defaults to 1.

`--interval`

- Time interval in between exposure-bracketed series.
- i.e "**`exp-bracketing --interval 60`**". If no argument is specified it defaults minimum interval time. This number varies depending on burst capture size and SD Card write speed.

Focus Bracketing

This script takes multiple series of focus bracketed shots. Users specify burst size, focus stop increments, time intervals, and the number of iterations.

Usage: **`focus-bracketing [parameter]`**

`--size`

- Number of consecutive pictures to take for each focus bracketed series
- i.e "**`focus-bracketing --size 5`**". Defaults to 3.

`--offset`

- Focus stop increments
- i.e "**`focus-bracketing --offset 1`**". Possible increments are: 1 - 10. Defaults to 1.

`--repeat`

- Repeat focus bracketed cycle N times
- i.e "**`focus-bracketing --repeat 5`**". Defaults to 1

`--interval`

- Time interval in between focus-bracketed series.
- i.e "**`focus-bracketing --interval 60`**". If no argument is specified it defaults minimum interval time. This number varies depending on burst capture size and SD Card write speed.

Focus Sweep

This script takes a number of captures evenly distributed across the specified focus range. (Applicable focus range size depends on camera's current zoom position).

Usage: **`focus-sweep [parameter]`**

`--focus-range`

- Focus stop increments.
- i.e "**`focus-sweep --focus-range 300 1000`**". Available focus range depends on focal length (zoom).

`--pictures`

- Number of pictures to take.
- i.e "**`focus-sweep --pictures 5`**". Defaults to 10.

`--interval`

- Time interval in between each capture.
- i.e "**`focus-sweep --interval 15`**".
- If no argument is specified it defaults minimum interval time. This number may vary between 2 to 10 seconds depending on SD Card write speed.

Zoom Sweep

This script takes a number of captures evenly distributed across the specified zoom range.

Usage: **`zoom-sweep [parameter]`**

`--zoom-range`

- Zoom position.
- i.e "**`zoom-sweep --zoom-range 20 1000`**". Zoom ranges from 1 to 1522

`--pictures`

- Number of pictures to take.
- i.e "**`focus-sweep --pictures 5`**". Defaults to 10.

`--interval`

- Time interval in between each capture.
- i.e "**`zoom-sweep --interval 15`**".
- If no argument is specified it defaults minimum interval time. This number may vary between 2 to 10 seconds depending on SD Card write speed.

Calibration Data
Pulls calibration data from camera to the local file system. Refer to Help menu for detail.
Usage: **`pull-cal-data`**

Download
Downloads images and stores them in local file system. Refer to Help menu for detail.
Usage: **`download-images`**

Delete
Deletes ALL pictures from the camera
Usage: **`delete-all`**

About Camera Control Tool

Camera Control Tool allows you to directly control camera features and settings through the command line when the ILLUM is tethered via USB cable. These Commands can be utilized to control the camera or retrieve the current values. The complete list of camera features that can be controlled are listed in the Camera Control help menu, which can be accessed by entering **`-h`** in the python terminal window. For additional output detail, use the Verbose argument **`-v.`**

Individual Command level help details along with available arguments can be accessed by entering the Help argument data (-h) into the python terminal window with the following convention: **`Command name -h`** For example, the Help argument for Exposure Mode would be: **`cameracontrol exposureMode -h`**

Camera Control Tool Commands and Arguments

White Balance

Choose and retrieve current White Balance setting
Usage: **`whiteBalance`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
WhiteBalance choices are: **`['auto', 'tungsten', 'fluorescent', 'flash', 'daylight', 'cloudy', 'shade', 'custom']`**
`-g, --get`
Get current WhiteBalance Mode

Exposure Mode

Choose and retrieve current Exposure Mode setting
Usage: **`exposureMode`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
ExposureMode choices are: **`['program', 'iso', 'shutter', 'manual']`**
`-g, --get`
Get current exposureMode state

Exposure Compensation

Choose and retrieve current Exposure Compensation setting
Usage: **`exposureMode`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Exposure Compensation: **`[-2 to +2]`**
`-g, --get`
Get current exposureMode state

ISO

Choose and retrieve current ISO setting
Usage: **`iso`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
ISO ranges 80-3200
`-g, --get`
Get current iso value

Shutter Speed

Choose and retrieve current Shutter Speed setting
Usage: **`shutterSpeed`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Shutter speed ranges 1/4000 - 32, can be entered as fraction or decimal
`-g, --get`
Get current shutter speed value

Focus Step

Choose and retrieve current Focus Step (Distance) setting
Usage: **`focusStep`**
Optional arguments:

`-h, --help`
Show this help message and exit

-s, --set
Set focus step **[1 - 1522]** (range depends on camera focal length position)
-g, --get
Get current focus step

Zoom Step

Choose and retrieve current Zoom Step (Focal Length) setting
Usage: **zoomStep**
Optional arguments:
-h, --help
Show this help message and exit
-s, --set
Set zoomSteps range **[1 - 1522]**
-g, --get
Get current zoom step

Optical Offset

Choose and retrieve current Optical Offset (see ILLUM User Manual for details) setting
Usage: **opticalOffset**
Optional arguments:
-h, --help
Show this help message and exit
-s, --set
Set opticalOffset range **[-10 to +10]**
-g, --get
Get current optical offset value
-r, --restore
Restore to default, value of -4

Focus Lock

Choose and retrieve current Focus Lock state
Usage: **`focusLock`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Disable or enable use 0 or 1
`-g, --get`
Get current focusLock state
`-r, --restore`
Restore to default, value of 0

Zoom Lock

Choose and retrieve current Zoom Lock state
Usage: **`zoomLock`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Disable or enable use 0 or 1
`-g, --get`
Get current zoomLock state
`-r, --restore`
Restore to default, value of 0

AE Lock

Choose and retrieve current Auto Exposure Lock state
Usage: **`aeLock`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Disable or enable use 0 or 1
`-g, --get`
Get current auto exposure lock state
`-r, --restore`
Restore to default, value of 0

Depth Assist

Choose and retrieve current Depth Assist state
Usage: **`depthAssist`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Disable or enable use 0 or 1
`-g, --get`
Get current Depth Assist state
`-r, --restore`
Restore to default, value of 0

Focus Mode

Choose and retrieve current Focus Mode (auto or manual) setting
Usage: **`focusMode`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Choices are **`['auto' or 'manual']`**
`-g, --get`
Get current Focus Mode state
`-r, --restore`
Restore value to 'auto'

Shutter Mode

Choose and retrieve current Shutter Mode (single or continuous) setting
Usage: **`shutterMode`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Choices are **`['single' or 'continuous']`**
`-g, --get`
Get current Shutter Mode state
`-r, --restore`
Restore to default, value of 'single'

Self Timer

Choose and retrieve current Self Timer (0, 2, 10 seconds) setting
Usage: **`selfTimer`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Set capture timer in secs **`[0, 2, 10]`** (Pass '0' to disable)
`-g, --get`
Get current Self-Timer state
`-r, --restore`
Restore to default, value of disabled'

Sleep Timer

Choose and retrieve current Sleep Timer value
Usage: **`sleepTimer`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Sleep Timer in secs. Set to '0' to disable.
`-g, --get`
Get current Sleep Timer state
`-r, --restore`
Restore to default, value of '120' secs

Exposure Bracketing

Choose and retrieve current Exposure Bracketing mode and value
Usage: **`expBracketing`**
Optional arguments:

`-h, --help`
Show this help message and exit
`-s, --set`
Disable or enable Exposure Bracketing. Choose 0 or 1
`-o, --offset`
Exposure stop size **`[1/3, 2/3, 1, 2]`**
`-c, --count`
Number of bracketed captures 3 or 5

-g, --get
Get current Exposure Bracketing state
-r, --restore
Restore to default, disabled, value of f ‘0’

Focus Bracketing
Choose and retrieve current Exposure Bracketing mode and value
Usage: **focusBracketing**
Optional arguments:
-h, --help
Show this help message and exit
-s, --set
Disable or enable Focus Bracketing. Choose 0 or 1
-o, --offset
Focus DS stop values **[1 - 10]**
-c, --count
Number of bracketed captures 3 or 5
-g, --get
Get current Focus Bracketing state
-r, --restore
Restore to default, disabled, value of f ‘0’

Metering Mode
Choose and retrieve current Metering Mode (average or evaluative) setting
Usage: **meteringMode**
Optional arguments:
-h, --help
Show this help message and exit
-s, --set
set Metering Mode to **[‘average’ or ‘evaluative’]**
-g, --get
Get current Metering Mode state
-r, --restore
Restore to default, value of ‘evaluative’

Flash Sync

Choose and retrieve current Flash Sync (back or front curtain) setting

Usage: **flashSync**

Optional arguments:

-h, --help
Show this help message and exit

-s, --set
Switch between front curtain or back curtain modes **['back' or 'front']**

-g, --get
Get current Flash Sync state

-r, --restore
Restore to default, value of 'front curtain'

Flash AF Assist

Choose and retrieve current Flash Auto Focus Assist setting

Usage: **flashAF**

Optional arguments:

-h, --help
Show this help message and exit

-s, --set
Disable or Enable Flash AF Assist mode **[0 or 1]**

-g, --get
Get current Flash AF Assist state

-r, --restore
Restore to default, disabled, value of f '0'

Flash Exposure Compensation

Choose and retrieve current Flash Exposure Compensation setting

Usage: **flashExpComp**

Optional arguments:

-h, --help
Show this help message and exit

-s, --set
Flash Exposure Compensation values **[-3 to +3]**

-g, --get
Get current Flash Exposure Compensation state

-r, --restore
Restore to default, value of '0'

Send Touch Event

Send Touch Event to actuate a screen touch by specifying screen's X and Y coordinates
Usage: **sendTouchEvent**
Optional arguments:

-h, --help
Show this help message and exit
-x, --xcoord
x coordinate ranges **[1 to 799]**
-y, --ycoord
y coordinate ranges **[1 to 479]**
-sl, --sleep
Sleep in secs after sending touch event
-re, --repeat
Number of iterations

Send Swipe Event

Send Swipe Event to actuate a screen swipe by specifying up, down, left or right
Usage: **sendSwipeEvent**
Optional arguments:

-h, --help
Show this help message and exit
-d, --direction
Set swipe direction, **[up, down left or right]**
-sl, --sleep
Sleep in secs after sending touch event
-re, --repeat
Number of iterations

Physical Controls

Send button press event to actuate shutter, half shutter, depth assist, power, AF, AEL, hyper focal or FN button push.

Usage: **`physicalControls`**

Optional arguments:

`-h, --help`
Show this help message and exit

`-pr, --press`
Send button press event for **`['shutter', 'depthAssist', 'power', 'AF', 'AEL', 'hyperFocal', 'Fn']`**

`-sl, --sleep`
Sleep in secs after sending button press

`-re, --repeat`
Number of iterations

Capture Image

Send a capture command to record an image or sequence of images

Usage: **`captureImage`**

Optional arguments:

`-h, --help`
Show this help message and exit

`-p, --pictures`
Number of pictures to take

`-as, --addSleep`
Additional Sleep on top of buffer clearance duration

Image Processing

To automate light field image processing and customization, Lytro Power Tools includes the Recipe Tool and Light Field Processing Tool. The Recipe Tool is used to create complex scripts (recipes) that can drive attributes of the raw light field image, replicate and control Lytro Desktop features, enabling you to go far beyond what is possible using only the Lytro Desktop and your ILLUM. The Light Field Processing Tool provides controls for processing and altering light field pictures. Recipe Tool provides control over the Adjustment Parameter Commands.

Image Adjustment Parameters

2D-Denoise Commands

ColorNoiseReduction
Remove or reduce image color information that is determined not to be a part of the original scene captured.
Usage: **`color-noise-reduction [-h]`**
Parameter: **`viewColorNoiseReduction`**
Type: **`integer`**
Min: **`0`**
Max: **`+100`**
Default: **`50`**

LuminanceNoiseReduction
Remove or reduce image brightness information that is determined not to be a part of the original scene captured.
Usage: **`luminance-noise-reduction [-h]`**
Parameter: **`viewLuminanceNoiseReduction`**
Type: **`integer`**
Min: **`0`**
Max: **`+100`**
Default: **`50`**

4D-2D Commands

Aperture
Change the aperture equivalent depth of field in the output .lfp, where 0.0 corresponds to f/16.0, 1.0 corresponds to f/2, and 2.0 corresponds to f/1.0.
Usage: **`aperture [-h]`**
Parameter: **`viewAperture`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`1.0`**

Focus
Adjust the focus in your picture
Usage: **`focus [-h]`**
Parameter: **`viewFocus`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

Focus-X
Adjust the focus in X coordinate values.
Usage: **`focus-x [-h]`**
Parameter: **`viewFocusX`**
Type: **`float`**
Depends: **`viewFocusY`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.5`**

Focus-Y
Adjust the focus in Y coordinate values.
Usage: **`focus-y [-h]`**
Parameter: **`viewFocusY`**
Type: **`float`**
Depends: **`viewFocusX`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.5`**

FocusSpread
Enable Focus Spread of image depth of field, specified as lambda added both near and far.
Usage: **`focus-spread [-h]`**
Parameter: **`viewFocusSpread`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

PerspectiveU
Change perspective in the coordinate U, corresponding to the horizontal axis. Large perspective changes can result in artifacts. Recommended range is -0.5 to 0.5, but you may try values in the range of -1.0 to 1.0.
Usage: **`perspective-u [-h]`**
Parameter: **`viewPerspectiveU`**
Type: **`float`**
Depends: **`viewPerspectiveV`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

PerspectiveV
Change perspective in the coordinate V, corresponding to the vertical axis. Large perspective changes can result in artifacts. Recommended range is -0.5 to 0.5, but you may try values in the range of -1.0 to 1.0.
Usage: **`perspective-v [-h]`**
Parameter: **`viewPerspectiveV`**
Type: **`float`**
Depends: **`viewPerspectiveU`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

Pivot
Sets the convergence point for standard pictures. Large values can result in artifacts.
Usage: **`pivot [-h]`**
Parameter: **`viewPivot`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

StereoBaseline
Set baseline for a stereo picture. Large values can result in artifacts.
Usage: **`stereo-baseline [-h]`**
Parameter: **`viewStereoBaseline`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

StereoPivot
Set the convergence point for stereo pictures. Large values can result in artifacts.
Usage: **`stereo-pivot [-h]`**
Parameter: **`viewStereoPivot`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

TiltX
Change the focal plane position in X.
Usage: **`tilt-x [-h]`**
Parameter: **`viewTiltX`**
Type: **`float`**
Depends: **`viewTiltY`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

TiltY
Change the focal plane position in Y.
Usage: **`tilt-y [-h]`**
Parameter: **`viewTiltY`**
Type: **`float`**
Depends: **`viewTiltY`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`0.0`**

Basic Tone Command

Blacks
Adjust only the very darkest shadows of the image without affecting mid tones.
Usage: **`blacks [-h]`**
Parameter: **`viewBlacks2`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0.0`**

Highlights
Adjust the upper range of mid tones and highlights.
Usage: **`highlights [-h]`**
Parameter: **`viewLuminanceNoiseReduction`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

Shadows
Adjust the lower range of mid tones and shadows.
Usage: **`shadows [-h]`**
Parameter: **`viewShadows2`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

Whites
Adjust only the very brightest highlights of the image without affecting mid tones.
Usage: **`whites [-h]`**
Parameter: **`viewWhites2`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

Color Correction Matrix Command

CCM
Color Correction Matrix
Parameter: **`viewCcm`**
Type: **`float`**
Min: **`n/a`**
Max: **`n/a`**
Default: **`null`**
Min items: **`9`**
Max items: **`9`**

Contrast Command

Contrast
Adjust in the positive to make dark areas darker and bright areas brighter. Adjust in the negative to make dark areas lighter and bright areas darker. Affects the entire tonal range of the image, but mostly shadows and highlights.
Usage: **`contrast [-h]`**
Parameter: **`viewContrast`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0.0`**

Crop Command

Crop
Crop by changing angle and adjusting the width and height of your output image.
Usage: **`crop [-h]`**
Parameter: **`viewCrop`**
Properties: **`angle, top, left, bottom, right`**
Type: **`float`**
Angle
- Default: **`0.0`**
- Min: **`-45.0`**
- Max: **`+45.0`**

Top, left
- Default: **`0.0`**
- Min: **`n/a`**
- Max: **`n/a`**

Bottom, right
- Default: **`1.0`**
- Min: **`n/a`**
- Max: **`n/a`**

Defringe Commands

Defringe
Enable Defringe to reduce Chromatic Aberration fringe artifacts.
Usage: **`defringe [-h]`**
Parameter: **`viewDefringe`**
Type: **`bool`**
Min: **`false`**
Max: **`true`**
Default: **`false`**

DefringeRadius
Set Defringe effect's pixel radius of fringe-reduction region.
Usage: **defringe-radius [-h]**
Parameter: **viewDefringeRadius**
Type:**float**
Min: **0**
Max: **10.0**
Default: **5.5**

DefringeThreshold
Set Defringe effect's threshold value. Value determines threshold at which fringe reduction begins: 100 substantially defeats defringing.
Usage: **defringe-threshold [-h]**
Parameter: **viewDefringeThreshold**
Type: **integer**
Min: **0**
Max: **+100**
Default: **50**

Exposure Command

Exposure
Adjust to simulate a larger or smaller amount of light from the original exposure. This expands or contracts the entire tonal range of the image.
Usage: **exposure [-h]**
Parameter: **viewExposure**
Type: **float**
Min: **-5.0**
Max: **+5.0**
Default: **0.0**

Reorient Command

Orientation
Rotate and orient the picture based on standard picture metadata values.
Usage: **`orientation [-h]`**
Parameter: **`viewOrientation`**
Type: **`integer`**
Choices: **`1, 2, 3, 4, 5, 6, 7, 8`**
Default: **`1`**

Saturation Commands

Saturation
Adjust the overall colorfulness of all image data in a global/linear fashion. Increasing the value raises the color saturation of all colors. Decreasing the value reduces the color saturation of all colors. A saturation of -100 is complete grayscale.
Usage: **`saturation [-h]`**
Parameter: **`viewSaturation`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

SaturationRed
Adjust overall saturation of Red channel.
Usage: **`saturation-red [-h]`**
Parameter: **`viewSaturationRed`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

SaturationGreen
Adjust overall saturation of Green channel.
Usage: **`saturation-green [-h]`**
Parameter: **`viewSaturationGreen`**

Type: **integer**
Min: **-100**
Max: **+100**
Default: **0**

SaturationBlue
Adjust overall saturation of Blue channel.
Usage: **saturation-blue [-h]**
Parameter: **viewSaturationBlue**
Type: **integer**
Min: **-100**
Max: **+100**
Default: **0**

SaturationCyan
Adjust overall saturation of Cyan channel.
Usage: **saturation-cyan [-h]**
Parameter: **viewSaturationCyan**
Type: **integer**
Min: **-100**
Max: **+100**
Default: **0**

SaturationMagenta
Adjust overall saturation of Magenta channel.
Usage: **saturation-magenta [-h]**
Parameter: **viewSaturationMagenta**
Type: **integer**
Min: **-100**
Max: **+100**
Default: **0**

SaturationYellow
Adjust overall saturation of Yellow channel.
Usage: **`saturation-yellow [-h]`**
Parameter: **`viewSaturationYellow`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

Vibrance
Adjust the colorfulness of the image in a weighted manner, biasing areas that are already strongly colored (non-neutral).
Usage: **`vibrance [-h]`**
Parameter: **`viewVibrance`**
Type: **`integer`**
Min: **`-100`**
Max: **`+100`**
Default: **`0`**

Sharpen Commands

SharpenDetail
Sets Sharpening Detail value
Usage: **`sharpen-detail [-h]`**
Parameter: **`viewSharpenDetail`**
Type: **`integer`**
Min: **`0`**
Max: **`+100`**
Default: **`0`**

SharpenEdgeMasking
Sets Sharpening Edge Masking Value
Usage: **`sharpen-edge-masking [-h]`**
Parameter: **`viewEdgeMasking`**
Type: **`integer`**
Min: **`0`**
Max: **`+100`**
Default: **`0`**

SharpenRadius
When increased, it affects a larger edge area, making the sharpening more pronounced and dramatic.
Usage: **`sharpen-radius [-h]`**
Parameter: **`viewStereoBaseline`**
Type: **`float`**
Min: **`+0.5`**
Max: **`+3.0`**
Default: **`1.0`**

Sharpness
Raising gives the appearance of a sharper image. Lowering gives the appearance of a less sharp, softer image. A value of zero removes all sharpening.
Usage: **`sharpness [-h]`**
Parameter: **`viewSharpness2`**
Type: **`integer`**
Min: **`0`**
Max: **`+150`**
Default: **`2`**

White Balance Commands

Temperature
Change the white temperature in Kelvin.
Usage: **`temperature [-h]`**
Parameter: **`viewTemperature`**
Type: **`integer`**
Min: **`+2000`**
Max: **`+50000`**
Default: **`null`**

Tint
Change the white temperature from green to magenta.
Usage: **`tint [-h]`**
Parameter: **`viewTint`**
Type: **`integer`**
Min: **`-150`**
Max: **`+150`**
Default: **`0`**

White Balance
Change the white balance to differing presets.
Usage: **white-balance [-h]**
Parameter: **viewWhiteBalance**
Choices: **as shot, auto, custom, daylight, cloudy, flash, fluorescent, shade, tungsten**
Default: **auto**

Window Commands

Note: These parameters are only available for living picture playback (i.e., Lytro Desktop or Web players). They have no affect on generating output for single images (e.g. lfptool batch).

PanX
Pan around the picture in X. Large pans can leave blank pixels.
Usage: **pan-x [-h]**
Parameter: **viewPanX**
Type: **float**
Depends: **viewPanY**
Min: **n/a**
Max: **n/a**
Default: **0.5**

PanY
Pan around the picture in Y. Large pans can leave blank pixels.
Usage: **pan-y [-h]**
Parameter: **viewPanY**
Type: **float**
Depends: **viewPanX**
Min: **n/a**
Max: **n/a**
Default: **0.5**

Zoom
Change zoom window in the picture. No limit. Small zoom can leave blank pixels.
Usage: **zoom [-h]**
Parameter: **viewZoom**
Type: **float**
Min: **n/a**
Max: **n/a**
Default: **1.0**

About LFP Tool

The Light Field Processing Tool provides the ability to process, import, export and metadata operations for LFP files generated by Lytro cameras.

- Process raw light field pictures (.LFR) files.
- Produce Extended Depth of Field (EDOF) images with everything in focus.
- Generate a depth map of objects in a scene.
- Generate images focused at different object distance planes in the scene.
- Generate EDOF and refocused images at different perspective shift values. Perspective Shift allows changing the point of view in a picture interactively.
- Query information about files.

The complete list of LFP Tool Commands and arguments are listed in the LFP Tool help menu, which can be accessed by using the **-h** argument with LFP Tool from a terminal window.

Individual Command level help details along with available arguments can be accessed with the following convention: **Command name -h** For example, the Help argument for Warp would be: **lfptool warp -h**

LFP Tool Commands and Arguments

Raw
Light Field Raw (LFR) files contain raw sensor data and metadata. contain image and depth information in addition to perspective views. This Raw data can be retrieved and processed for a variety of end uses including genertaing image and depth information and perspective views. The default locations for data can be found in Help: **raw -h**
Usage: **raw**
Optional arguments:

-h, --help
Show this help message and exit
-P PROCESSORS, --processors PROCESSORS
Processes to run concurrently [max: 8] (default: 1)

Input arguments:

-i PATH [PATH ...], --lfp-in PATH [PATH ...]
Execute operation on input LFP files or scanned directories

--range INT INT
Range of LFP files to parse (start/end)

--pattern FILE_PATTERN
--range pattern | * = wildcard, # = start of sequence (default: **IMG_#**)

--calibration-in PATH
Calibration data directory

--depth-in PATH
Input BMP or PNG depth map

--recipe-in PATH
Input view parameter JSON file

Action arguments:

--depth-out
Output depth map

--eslf-out
Output external standardized light field

--image-out
Output image file

--lfp2raw
Unpackage RAW and corresponding TXT from an LFP container

--lfp-out
Output warp LFP file

--lfr2xraw
Output XRAW LFR

--raw2lfp
Package RAW and corresponding TXT within an LFP container

--recipe-out
Output LFP view parameters

--unpack
Output non-packed LFP asset (default action)

Representation arguments:

`--depthrep {bmp,png,dat}`
Depth map representation

`--imagerep {jpeg,png,tiff,bmp,exr}`
Image representation

Output arguments:

`--dir-out PATH`
Output directory (default: output to the source LFP file's directory)

Perspective arguments:

`--focus FLOAT`
One element of the lambda list

`-u u [u ...], --perspective-u u [u ...]`
One element of the horizontal viewpoint list

`-v v [v ...], --perspective-v v [v ...]`
One element of the vertical viewpoint list

Window arguments:

`--height INT`
Output y resolution

`--orientation INT`
Orientation enumeration 1-8, matches EXIF definitions (default: 1)

`--width INT`
Output x resolution

Threads arguments:

`--threads INT`
Number of processing threads (per LFE instance)

Batch

Batch processing LFR files allows a number of files to be processed with the same instructions, which is extremely useful for projects with large volumes of captures.

Usage: **batch**

Optional arguments:

-h, --help
Show this help message and exit

-P PROCESSORS, --processors PROCESSORS
Processes to run concurrently [max: 8] (default: 1)

--per-lfp IMAGES
Apply batch process against each individual LFP

Input arguments:

-i PATH [PATH ...], --lfp-in PATH [PATH ...]
Execute operation on input LFP files or scanned directories

--range INT INT
Range of LFP files to parse (start/end)

--pattern FILE_PATTERN
--range pattern | * = wildcard, # = start of sequence (default: **IMG_#**)

--calibration-in PATH
Calibration data directory

--depth-in PATH
Input BMP or PNG depth map

--recipe-in PATH
Globally applied recipe (see: --per-lfp)

Output arguments:

--dir-out PATH
Output directory (default: output to the source LFP file's directory)

Window arguments:

-height INT
Output y resolution

-width INT
Output x resolution

Representation arguments:

--imagerep {jpeg,png,tiff,bmp,exr}
Image representation

Perspective arguments:

`-u u [u ...], --perspective-u u [u ...]`
One element of the horizontal viewpoint list

`-v v [v ...], --perspective-v v [v ...]`
One element of the vertical viewpoint list

Threads arguments:

`--threads INT`
Number of processing threads (per LFE instance)

Warp

The Warp command enables control for extraction and insertion of image data into warp LFP files. The default locations for data can be found in Help: **`warp -h`**

Usage: **`warp`**

Optional arguments:

`-h, --help`
Show this help message and exit

Input arguments:

`-i PATH [PATH ...], --lfp-in PATH [PATH ...]`
Execute operation on input LFP files or scanned directories

`--range INT INT`
Range of LFP files to parse (start/end)

`--pattern FILE_PATTERN`
`--range pattern | * = wildcard, # = start of sequence` (default: **`IMG_#`**)

Representation arguments:

`--depthrep {bmp,png,dat}`
Depth map representation

`--imagerep {jpeg,png,tiff,bmp,exr}`
Image representation

Output arguments:

`--dir-out PATH`
Output directory (default: output to the source LFP file's directory)

Window arguments:

--height INT
Output y resolution

--width INT
Output x resolution

Action arguments:

--lfp-out
Output warp LFP file

--recipe-out
Output LFP view parameters

--pack
Output packed LFP from unpacked LFP asset

--unpack
Output non-packed LFP asset (default action)

Threads arguments:

--threads INT
Number of processing threads (per LFE instance)

4D-Coord

X,Y,U,V data for working with a External Standardized Light Field (ESLF)
Usage: **4d-coord**
Optional arguments:

-h, --help
Show this help message and exit

-r ROW, --row ROW
ESLF row

-c COLUMN, --column COLUMN
ESLF column

Info

View, write, query, and validate metadata from an individual LFP file, a list of LFP files, or a directory containing LFP files.

Usage: **`info`**

Optional arguments:

`-h, --help`
Show this help message and exit

`-o, --json-out`
Write results as JSON [conflicts: -p/-v]

`-v, --validate`
Validate LFP schema [conflicts: -o/-p/-s]

`-p PROP [PROP ...], --property PROP [PROP ...]`
Query LFP metadata [conflicts: -o/-v]

`-x, --exact`
With **`--property`**, search for exact match

Input arguments:

`-i PATH [PATH ...], --lfp-in PATH [PATH ...]`
Execute operation on input LFP files or scanned directories

`--range INT INT`
Range of LFP files to parse (start/end)

`--pattern FILE_PATTERN`
`--range pattern | * = wildcard, # = start of sequence` (default: **`IMG_#`**)

About Recipe Tool

The Recipe Tool provides the ability to create Recipe scripts that can perform complex operations including Living Picture image processing, editing, and animation, merging of multiple scripts, getting info.

The Recipe Tool is broken into a set of primary commands that each have unique arguments, view parameter commands and animation parameter commands.

The complete list of Recipe Tool Commands and arguments are listed in the Recipe Tool help menu, which can be accessed using the **-h** argument with Recipe Tool from a terminal window.

Individual Command level help details along with available arguments can be accessed with the following convention: **Command name -h** For example, the Help argument for Merge would be: **recipetool merge -h**

Recipe Tool Primary Commands

Destroy

Deletes view parameter values and all animation time/value data for a given view parameter. Note that the individual view parameter commands (such as Saturation) include an option to delete specific animation keyframe times/values.

Usage: **destroy [-h] [-A] [--all]**

Optional arguments:

-h, --help
Show this help message and exit

-A, --animation
Use animation parameter in place of view parameter

--all
Destroy all view parameters

Input arguments:

-i, --recipe-in [<PATH> ...]
Execute operation on input recipe files or scanned directories

Info
Used to display view/animation parameter values.
Usage: **info [-h] [-A]**
Optional arguments:
-h, --help
Show this help message and exit

Input arguments:
-i, --recipe-in [<PATH> ...]
Execute operation on input recipe files or scanned directories

Info arguments:
-S, --show
Show view parameter value (default)
-A, --show-animation
Show animation data
-K, --keyframes [<index>]
Show animation data as keyframes
index: display only specified index
-P, --points [<index>/x/y]
Show animation data as x/y points
x : time data points
y : value data points
index : x/y points for specified index
none : all x/y data points

Merge
Combines source recipe view parameters into a single animation.
Usage: **merge [-h]**
Optional arguments:
-h, --help
Show this help message and exit
Output arguments:
-o, --recipe-out [<PATH> ...]
Output LFP view parameters
Input arguments:
-i, --merge-in [<PATH> ...]
Source recipe OR warp LFP file(s)
Merge arguments:
-s, --select
Merge only selected parameter arguments

```
-n, --steps       <STEPS>
      Steps      (default: 12)
-e, --ease        <ACCEL>
      Easing acceleration      (default: in_out)
      Choices: in, out, in_out
-f, --shape       <SHAPE>
      Easing shape      (default: quad)
      Choices: linear, bounce, cubic, expo, quart, sine, back, circ, elastic, quad,
quint
--t0 <SECOND>
      Start time in seconds
--t1 <SECOND>
      End time in seconds
```

Parameter arguments:
Comma separated overrides for individual parameters;
Missing sub-arguments will default to the global argument value;
Using desired sub-arguments, strings must match the following format:

```
--param option=value
```

e.g.: `--param t0=5,t1=15,shape=cubic,ease=out`

New

Generate new recipe file(s); multiple filenames are accepted

Usage: **`new [-h]`**

Optional arguments:

```
-h, --help
      Show this help message and exit
```

Output arguments:

```
-o, --recipe-out [<PATH> ...]
      Output LFP view parameters
```

Plot

Plot animation time/value (x/y) line (requires matplotlib); if no view parameter is specified, plot all

Usage: **`plot [-h] [-S]`**

Optional arguments:

```
-h, --help
      Show this help message and exit
-S, --save
      Save graph to disk (do not display)
```

Input arguments:
-i, --recipe-in [<PATH> ...]
Execute operation on input recipe files or scanned directories

Validate
Validate recipe file(s) against LFP schema
Usage: **validate [-h]**
Optional arguments:
-h, --help
Show this help message and exit
Input arguments:
-i, --recipe-in [<PATH> ...]
Execute operation on input recipe files or scanned directories
View
View parameter adjustments and values
Usage: **view**

Optional arguments:
-h, --help
Show this help message and exit
Input arguments:
-i, --recipe-in [<PATH> ...]
Execute operation on input recipe files or scanned directories
2D-denoise:
--color-noise-reduction {0..100}
Corresponds to crs:ColorNoiseReduction
--luminance-noise-reduction {0..100}
Corresponds to crs:LuminanceSmoothing
4D-to-2D:
--aperture FLOAT
Image depth of field, specified as normalized aperture diameter
--focus FLOAT
Adjust picture focus; depth of focal plane at FNC [0.5, 0.5
--focus-spread FLOAT
Spread image depth of field, as lambda added both near and far
--focus-x FLOAT
X coordinate in FNC of the location for which viewFocus is specified
--focus-y FLOAT
Y coordinate in FNC of the location for which viewFocus is specified

--perspective-u FLOAT
Center-of-perspective U coordinate
--perspective-v FLOAT
Center-of-perspective V coordinate
--pivot FLOAT
Distance at which objects are stationary under perspective change
--stereo-baseline FLOAT
Full length of stereo baseline
--stereo-pivot FLOAT
Stereo distance at which objects are stationary under perspective change
--tilt-x FLOAT
Signed change in lambda from left to right image edges, in FNC
--tilt-y FLOAT
Signed change in lambda from top to bottom image edges, in FNC

Basic Tone:

--blacks {-100..100}
Corresponds to crs:Blacks2012
--highlights {-100..100}
Corresponds to crs:Highlights2012
--shadows {-100..100}
Corresponds to crs:Shadows2012
--whites {-100..100}
Corresponds to crs:Whites2012

ccm:

--ccm FLOAT FLOAT FLOAT FLOAT FLOAT FLOAT FLOAT FLOAT FLOAT
row major; if not specified, a system-computed ccm is used

Contrast:

--contrast {-100..100}
Corresponds to crs:Contrast2012

Defringe:

--defringe
Enable reduction in fringe artifacts
--defringe-radius {0.0..10.0}
Pixel radius of fringe-reduction region
--defringe-threshold {0..100}
Threshold at which fringe reduction begins

Exposure:

--exposure {-5.0..5.0}
Corresponds to crs:Exposure2012

Reorient:

--orientation {1..8}
: Matches EXIF definitions

Saturate:

--saturation {-100..100}
: Corresponds to crs:Saturation

--saturation-blue {-100..100}
: Corresponds to crs:SaturationAdjustmentBlue

--saturation-cyan {-100..100}
: No direct crs correspondence; candidates: orange, aqua, purple

--saturation-green {-100..100}
: Corresponds to crs:SaturationAdjustmentGreen

--saturation-magenta {-100..100}
: Corresponds to crs:SaturationAdjustmentMagenta

--saturation-red {-100..100}
: Corresponds to crs:SaturationAdjustmentRed

--saturation-yellow {-100..100}
: Corresponds to crs:SaturationAdjustmentYellow

--vibrance {-100..100}
: Corresponds to crs:Vibrance

Sharpen:

--sharpen-detail {0..100}
: Corresponds to crs:SharpenDetail

--sharpen-edge-masking {0..100}
: Corresponds to crs:SharpenEdgeMasking

--sharpen-radius {0.5..3.0}
: Corresponds to crs:SharpenRadius

--sharpness {0..150}
: Corresponds to crs:Sharpness

White-balance:

--temperature {2000..50000}
: Corresponds to crs:Temp; estimated if not specified

--tint {-150..150}
: Corresponds to crs:Tint

--white-balance
: {as shot,auto,daylight,cloudy,shade,tungsten,fluorescent,flash,custom} corresponds to crs:WhiteBalance

Window:

--pan-x FLOAT
Position in RNC of the center of the window

--pan-y FLOAT
Position in RNC of the center of the window

--zoom FLOAT
Scale factor from RNC to WNC

Animation Parameter Commands

Auto animation arguments

Start time & value priority:
Time: inputted t0 time -> preceding keyframe's time -> t=0.0001
Value: inputted v0 value -> preceding keyframe's value ->
Initial value -> view parameter -> parameter default (1.0)

End time & value priority:
Time: inputted t1 time -> a duration of 10.0 seconds is applied
Value: inputted v1 value (required)

--t0 <TIME>
Start time

--v0 <VALUE>
Start value

--t1 <TIME>
End Time

--v1 <VALUE>
End Value

-e, --ease <ACCEL>
Easing acceleration (default: **in_out**)
Choices: **in, in_out, out**

-f, --shape <METHOD>
Easing shape (default: **quad**)
Choices: **linear, bounce, cubic, expo, quart, sine, back, circ, elastic, quad, quint**

-n, --steps <STEPS>
Easing animation steps (default: 12)

Manual animation arguments

-T, --times [<TIME> ...]
Times omitting t=0.0

-V, --values [<VALUE> ...]
Values omitting t=0.0

-I, --initial-value <VALUE>
Initial value at t=0.0

--dt0 [<(-T) OFFSET> ...] handle pair time offset 0 (positive float)

--dt1 [<(+T) OFFSET> ...] handle pair time offset 1 (negative float)

--dv0 [<(-V) OFFSET> ...] handle pair value offset 0 (min: n/a | max: n/a)

--dv1 [<(+V) OFFSET> ...] handle pair value offset 1 (min: n/a | max: n/a)

Animation keyframe arguments

-a, --adjust <KEYFRAME>
Adjust animation keyframe

-d, --destroy <KEYFRAME>
Destroy animation keyframe

Info arguments

-S, --show
Show parameter value

-A, --show-animation
Show animation data

-K, --keyframes [<index>]
Show animation data as keyframes
index : display only specified index

-P, --xy, --points [x/y/<index>]
Show animation data as x/y points
x : time data points
y : value data points
index : x/y points for specified index
none : all x/y data points

Scaling arguments

--scale-time <START> <END> scale animation time to new start/end

--scale-value <START> <END> scale animation value to new start/end

Managing Living Pictures and Albums

Light Field Pictures (LFP), commonly referred to as Living Pictures, can be hosted and shared on the pictures.lytro.com website (user account required). LFP images can be named, captioned, arranged in albums and presented with full interactivity permitting refocusing, perspective shift and virtual aperture control (Virtual Aperture only available on Lytro Mobile) via the Lytro player in a web browser or on iOS and Android mobile device apps. Web Tool can help automate and manage these functions on pictures.lytro.com.

Web Tool

Web Tool provides support for creating, updating, deleting, and retrieving data for pictures.lytro.com living pictures and albums. The complete list of Web Tool features that can be controlled are listed in the help menu, which can be accessed using the **-h** argument with LFP Tool from a terminal window.

Individual Command level help details along with available arguments can be accessed with the following convention: **Command name -h** For example, the Help argument for Upload would be: **webtool upload -h**

Web Tool Commands and Arguments

Upload

Upload an individual LFP file, a list of LFP files, or a directory containing LFP files to pictures.lytro.com; pictures will be uploaded to a new album unless an existing album ID is specified.
NOTE: if an existing album ID is specified, and other album arguments are specified (-d/--description | -n/--name | --public/--unlisted), the specified album will be updated to reflect these arguments
Usage: **upload [-u USERNAME] [-p PASSWORD] [-n NAME] [-d DESC] -i LFP_IN [LFP_IN ...] [-a ID] [--public | --unlisted] [--auto-caption | -c CAPTION | --caption-file CAPTIONS]**

Optional arguments:

-h, --help show this help message and exit

-i LFP_IN [LFP_IN ...], --lfp-in LFP_IN [LFP_IN ...]
execute operation on input LFP files or scanned directories

-a ID, --album-id ID
upload to specified album

Authentication:

-u USERNAME, --username USERNAME
Specify username; prompt if left blank

-p PASSWORD, --password PASSWORD
Specify password; prompt if left blank

Album:

-n NAME, --name NAME
Album name (140 chars. max, default: auto-generated)

-d DESC, --description DESC
Album description (3000 chars. max, default: auto-generated)

Album Privacy:

--public
Toggle album privacy status to public

--unlisted
Toggle album privacy status to unlisted (default option)

Picture caption:

--auto-caption
Auto-generates picture captions; e.g. '01/15 - test_output.lfp'

-c CAPTION, --caption CAPTION
Picture caption (140 char. max, default: None)

--caption-file CAPTIONS
Input CSV file containing captions; format: /absolute/path/to/picture.lfp,caption)

Album

Retrieve information for albums, update, or create albums on pictures.lytro.com
Usage: **album [-h] [-u USERNAME] [-p PASSWORD] [-n NAME] [-d DESC] [-i [ID]] [--public | --unlisted] (--get-all | -G | -U | -C | -D)**

Optional arguments:

-h, --help
Show this help message and exit

-i [ID], --album-id [ID]
Perform action on specified album ID; required for --update/-U | --get/-G actions

Authentication:

-u USERNAME, --username USERNAME
Specify username; prompt if left blank

-p PASSWORD, --password PASSWORD
Specify password; prompt if left blank

Album:

-n NAME, --name NAME
Album name (140 chars. max, default: auto-generated)

-d DESC, --description DESC
Album description (3000 chars. max, default: auto-generated)

Album privacy:

--public
Toggle album privacy status to public

--unlisted
Toggle album privacy status to unlisted (default option)

Album actions:

--get-all
Retrieve information for all available albums

-G, --get
Retrieve information for specified album and its pictures

-U, --update
Update information for specified album; at least one album argument is required

-C, --create
Create an empty album; album arguments are optional

-D, --delete
Delete specified album; all other arguments are ignored

Picture
Retrieve information for individual pictures or update pictures on pictures.lytro.com
Usage: **picture [-h] [-u USERNAME] [-p PASSWORD] [-i [ID]] [-c CAPTION] (--get-all | -G | -U | -D)**

Optional arguments:

-h, --help
Show this help message and exit

-i [ID], --picture-id [ID]
Perform action on specified picture ID; required for:
--update/-U | --get/-G |delete/-D actions

Authentication:

-u USERNAME, --username USERNAME
Specify username; prompt if left blank

-p PASSWORD, --password PASSWORD
Specify password; prompt if left blank

Caption:

-c CAPTION, --caption CAPTION
Picture caption (140 char. max, default: None)

Picture actions:

--get-all
Retrieve information for all available pictures

-G, --get
Retrieve picture information for specified picture

-U, --update
Update information for specified picture (requires --caption)

-D, --delete
Delete specified picture

Web Tool Upload

A processed LFP image can be uploaded to pictures.lytro.com (account required for all actions) using the following steps:

upload -c “Picture taken July 31, 2015” -u foobar -a 12345 raw/output.lfp

- Uploads ‘output.lfp’ to album ‘12345’ for user ‘foobar’ with the caption “Picture taken July 31, 2015”
- With all actions, username & password can be passed as arguments or if preferred, left blank. The web tool will prompt for either missing value when ran, e.g.:

```
upload -c “Picture taken July 31, 2014” -a 12345 raw/output.lfp
enter username for pictures.lytro.com: foobar
enter password for pictures.lytro.com: *********
```

Managing pictures, albums and caption

Web Tool functions include retrieving and updating album, picture and caption data with the following commands:

album --get-all

Retrieves information for all albums belonging to the user including: album name/description, privacy status, creation data, picture count

picture --get 567890

Retrieves information for picture **‘567890’** including: picture caption, privacy status, upload date

picture --update 567890 --caption “updated caption”

Updates picture caption for picture id “**567890**”

Lytro Power Tools Example Workflows

Workflow #1: Changing static parameters (non-animated) to your Recipe Tool file.

1. Find an existing recipe.json file or generate a new one by using the command:

   ```
   recipetool new -o path\to\recipe.json
   ```

2. Enter the command **`recipetool -h`** to look up the parameter you want to change.
3. Let's say we want to change contrast, we can look up the available commands for this by entering:

   ```
   recipetool contrast -h
   ```

4. We see that contrast is an integer type, labeled as "viewContrast" in the recipe.json file, with a range of -100 to +100, and a default of zero.
5. Using the commands referenced in help, we'll change the static (non-animated) value of contrast to -25 by the following command:

   ```
   recipetool -i path\to\recipe.json -v -25
   ```

6. Open your recipe file in a text editor to confirm that the viewContrast parameter value is now set to -25.

Workflow #2: Creating animations out of adjustable parameters.

1. Let's say we want to animate contrast instead of just changing it statically. Let's do some prep work to calculate what kind of animation we would like.
2. First let's count up how many individuals LFR files we have. Recipe Tool and LFP Tool can interpolate the correct animations for a given time, frame rate, and number of source raw files, but let's say we want to calculate it so that each raw file in our time lapse will consist of a single frame in the output animation.
3. For example, a time-lapse video of 212 raw files can output a single raw file per frame for 8.83 seconds if creating an animation at 24 frames per second. We'll use these numbers for the rest of this example.

4. Using the same help menu from workflow #1, we can view the animation parameters at the bottom of the output. We'll be using the auto-animation parameters in this example.
5. The auto-animation parameters are useful because they automatically calculate the correct handle pairs for an animation's given keyframes. Each keyframe contains a handle pair, a subset of time/value coordinates that help smooth and shape the transition of values across time for animated view parameters.
6. We'll be stating a start time, start value, end time, and end value in our animation, everything in between will be generated automatically by the auto-animate command.
7. The only other parameters we'll need other than the time-value pairs are the function, method, and steps parameters, which determine how the auto-animate functions builds the keyframes and handles in between our start and end pairs.
8. **`--ease`** is the type of the easing acceleration used to create the animation. This is a non-linear set of instructions for how a value transition should animate across a given time. "In" begins the transition slowly and accelerates it at the end. "Out" does the inverse, beginning quickly and slowing down at the end. "In-Out" uses In at the beginning and Out at the end, creating a transition that begins and ends slowly while transitioning more rapidly in the middle of the time field.
9. **`--shape`** determines the characteristic of the given acceleration. Starting with a set of non-linear transition instructions (start slow / end fast, etc…) a shape is added that gives further definition across the transition. Many of these choices are based off of simple mathematical functions (cubic, quad, quint, expo, etc…) while some are based off of recognizable physical phenomena (bounce, elastic, etc…). *Note*: the **`plot`** command is useful for observing the affects of the differing easing accelerations and shapes..
10. **`--steps`** is the number of time/value pairs that the auto-animate command will generate. This does not have to match the given number of raw files for animating across a given time and frame-rate, but since we know we want one frame per raw file for our animation output, we'll input the total number of our source time lapse raw files as the value for steps.
11. We're now ready to input our auto-animate command for contrast. For this example, we'll start at low contrast (-25) and use the expo shape with the "in" ease acceleration choice to gradually increase contrast in the beginning and then rapidly increase the contrast at the end of the animation to full contrast, +100. Here's what our command looks like:

```
recipetool contrast -i path\to\recipe.json --t0 0 --v0 -25 --t1 8.83 --v1 100
--shape expo --ease in --steps 212
```

12. When you open the resulting recipe.json file after running this command, you'll see a long list of keyframes, mapping each of the view parameter changes gradually moving across 212 steps in time.
13. Use the lfptool command to implement the animation on a batch of time lapse raws:

    lfptool batch -i path\to\lfrs --recipe-in path\to\recipe.json --dir-out path\to\ output-dir -P (number of processors)

14. Note: if you don't know the number of processors to use for multiprocessing, run:

    lfptool batch -h

    From the printout look for:
    -P PROCESSORS, --processors PROCESSORS processes to run concurrently [max: 4] (default: 1)"
    In the above example the max number of processors the machine can use is 4 for multiprocessing.
15. You now have a collection of still files that can be used to generate an animated movie in a 3rd party application.

Workflow #3: Using the **merge** command to generate animations, such as focus and perspective.

1. Some view parameters can be difficult to gauge the values on, especially when they list their range as being infinite float values. In this workflow we'll look at a practical way to animate focus and perspective parameters using Lytro Desktop and the Recipe Tool **merge** command.
2. Open a sample lfr raw file from your time lapse into Lytro Desktop.
3. Click to change the focus to the desired focus point you'd like your animation to start at.
4. Click and drag to also change the perspective shift to the desired perspective you wish to start at.
5. Export the file as an "Editable Living Picture". We'll return to this file later.
6. Now change the focus and the perspective to the positions you would like to end your animation at.
7. Export as an Editable Living Picture again. You may wish to rename your original export from the previous step, or export to a different file path, to avoid overwriting files.
8. Now that you have each editable living picture, use the Recipe Tool merge command to combine their recipe files into a new output recipe file. The merge command will generate an animation in the new output recipe file whenever it detects a difference in values for an animate-able view parameter.
9. Let's use our previous workflow example of 212 steps, so we can get a time/value points for each raw

file for 8.83 seconds of animation at 24 frames per second, with an expo shape and "in" choice for ease acceleration:

```
recipetool merge -i path\to\first.lfp path\to\second.lfp --t0 0 --t1 8.83 --shape
expo --ease in --steps 212 -o path\to\new\recipe.json
```

10. This method extracts the recipe files automatically from the editable living picture folder's lfp file. You now have a recipe file you can use to generate animation still frames.
11. Addendum: You can use the **`--select`** parameter if you want to only target specific view parameters to be merge into animations for a given pair of recipe.json files. For example, if there were other difference in parameters between the two editable living pictures and you only wanted to merge-animate focus and perspective, the command would be:

```
recipetool merge -i path\to\first.lfp path\to\second.lfp --t0 0 --t1 8.83
--shape expo --ease in --steps 212 -o path\to\new\recipe.json --select --focus
--persective-u --perspective-v
```

12. You can also set different animation parameters per selected parameter from the global ones used in the command, for example:

```
recipetool merge -i path\to\first.lfp path\to\your\second.lfp --t0 0 --t1 8.83
--shape expo --ease in --steps 212 -o path\to\new\recipe.json --select --focus
t0=5,t1=15,shape=cubic,ease=in_out --persective-u --perspective-v
```

Workflow #4: Generating stereo output pairs for compositing 3D animations.

1. Generating stereo pairs for 3D output animation requires a brief overview of two parameter concepts: pivot and baseline.

   Pivot the point at which objects that are perspective shifted, or viewed in stereo parallax, will move relative to. Think of it this way: hold a pencil in front of your face so that you're staring at the end of the pencil point, with the rest of the pencil behind it. Now grab the very end of the pencil, the eraser, and hold that point fixed while moving the pencil in a conical motion. Now grab the pencil from the middle and make the same motion.

When the pivot is on the eraser, everything in front of the eraser moves while the eraser stays fairly steady. When the pivot is in the middle, Objects in the front will move clock-wise, while objects in the back will move counter-clockwise in the opposite motion. Objects closer to the pivot move less, object further away move more.

This is how stereo parallax is determined. The value for the pivot is the given focus level in the image. A pivot of zero will tend to move as though the pivot were held in the middle of the image, especially with content at negative and positive focus levels (orange and blue in your depth view display).

Most images for 2D viewing with perspective shift have a negative pivot point. For 3D this generates a "fish tank" effect, where objects in the background have more parallax than objects in the foreground. A negative value indicates the "front" of the image, or even further than the front-most object.

Lytro Desktop default 3D pivot uses a zero or positive value for the pivot. This results in a "pop-out" effect, where objects in the foreground have more parallax than objects in the background.

Baseline is the maximum level of spacing in between objects. Instead of controlling this by a single value, as in the Lytro Desktop 3D StereoBaseline parameter, we'll simply be outputting pairs of image with two different perspective values.

2. For a given recipe.json file with animation instructions, set the pivot view parameter to that one you wish to use for the stereo animation (we'll use a value of 5 in this example):

   ```
   recipetool pivot -i path\to\recipe.json -v 5
   ```

3. Using lfptool, include two **-u** values and two **-v** values to output a given uv-coordinate perspective pair. If you'd like to just adjust the horizontal perspective for stereo vision, use zeros for v coordinates:

   ```
   lfptool batch -i path\to\lfrs --recipe-in path\to\recipe.json --dir-out path\to\
   output-dir -P (number of processors) -u -0.5 0.5 -v 0 0
   ```

4. Addendum: You can use the **lfptool warp --recipe-out** command to extract a recipe file from an editable living picture and get an idea for the range you may wish to set for the perspective. Starting -0.5 to +0.5 is a safe start.

Lytro Power Tools Configuration File

The Lytro Power Tools configuration file stores a number of user-defined common attributes. Here are details of what can be modified within this configuration file.

Location
Windows: **C:\Users\USERNAME\AppData\Local\Lytro\lytro-power-tools.cfg**
OS X: **/Users/USERNAME/Library/Application Support/Lytro/lytro-power-tools.cfg**

LFP Tool

Image & Depth Map Representation

Default image and depth map output representation types used during processing. Modifying these options sets the **--imagerep** and **--depthrep** argument defaults for the applicable command action. For example:

Setting **imagerep_raw_image_out** to jpeg is the equivalent to the following syntax:

```
$ lfptool raw --image-out --imagerep jpeg
```

Option	Choices
imagerep_raw_depth_out	*bmp, jpeg, png, tiff, exr*
imagerep_raw_image_out	*bmp, jpeg, png, tiff, exr*
imagerep_raw_eslf_out	*jpeg, png*
imagerep_raw_lfp_out	*jpeg, png, tiff*
imagerep_raw_unpack	*jpeg, png, tiff*
imagerep_warp_pack	*jpeg, png, tiff*
imagerep_warp_unpack	*jpeg, png, tiff*
depthrep_raw_depth_out	*bmp, png*
depthrep_raw_lfp_out	*bmp, png, dat*
depthrep_raw_unpack	*bmp, png, dat*
depthrep_warp_pack	*bmp, png, dat*
depthrep_warp_unpack	*bmp, png, dat*

Calibration In

Sets the absolute path to camera calibration data. Useful to change if the calibration data is not in a default location and if set, prohibits the need to call the argument every time a raw action is completed (if the calibration data is not in it's default location). Alternatively, set to False if using the XRAW LFP file format.

option: **calibration_in**
default: None (autodetect)

Verbose

Increase output verboseness; namely, increases output during LFE processing. Setting this argument is the equivalent to running a LFP Tool command with the **--verbose** argument flag; i.e.,

```
$ lfptool --verbose
```

option: **verbose**
default: False

Validate

Validate LFP metadata; set to False to skip validation. This should be safe to leave False if the LFPs in use are only processed through Lytro Power Tools or Desktop. If external modifications are being made to your LFP files, especially in regards to metadata, it would be safer to set this to True.

option: **`validate`**
default: False

Recipe Tool

Recipe Version

Specify recipe version; check the matrix below for compatibility with Lytro Desktop.

1: Lytro Desktop Version(s): 4.0.0 - 4.0.4
2: -NOT USED-
3: Lytro Desktop Version(s): 4.1.0 - 4.1.2
4: Lytro Desktop Version(s): 4.2.0 - ...

option: **`recipe_version`**
default: 4

Auto Animation

option: **`auto_ease`**
description: default animation easing acceleration
default: in_out
choices: in, out, in_out, out_in

option: **`auto_shape`**
description: default animation easing shape
default: in_out
choices: linear, quad, cubic, quart, quint, sine, expo, circ

option: **`auto_steps`**
description: default easing animation steps, more steps increases animation smoothness and provides a better shape but also increases CPU calculations and data point management
default: 12

option: **auto_duration**
description: used when **--t1** (end time) is not specified in an applicable recipetool command; the animation end time will instead equal this set value, plus **--t0** (start time); if **--t0** is also not specified, animation start time will equal in an internal millisecond start value (.0001) plus **auto_duration**
default: 10

Verbose

Increase output verboseness; namely, shows the difference before/after a Recipe Tool command is executed. Setting this argument is the equivalent to running a Recipe Tool command with the **--verbose** argument flag; i.e.,

```
$ recipetool --verbose
```

option: **verbose**
default: False

Web Tool

Authentication

There are two authentication methods to choose from:

1)

Hardcoded username & password - not very safe in multi-user environment. To enable, set both of these options:
username
password

2)

Alternatively, a safer method to allow seamless authentication is to store a user id and authentication token. However, since authentication tokens are dynamic, the authentication token may need to be periodically changed.

To get the user id and authentication token of an account, perform any webtool operation with the **--verbose/-v** argument flagged; the user id and authentication token will be stored in the POST Session call (the first call), located here:

```
Lytro POST Session: ['response']['data']['user']['id']
Lytro POST Session: ['response']['data']['authentication_token']
```

To enable, set both of these options:
user_id
auth_token

Verbose

Increase output verboseness; namely, increases output during HTTP API calls. Setting this argument is the equivalent to running a Web Tool command with the **--verbose** argument flag; i.e.,

```
$ webtool --verbose
```

option: **verbose**
default: False

Online Resources

- Lytro website — www.Lytro.com
- Training — training.Lytro.com
- Lytro Desktop — www.Lytro.com/downloads
- Lytro Support — support.Lytro.com

Legal