**­­­­Linkage Priority Developer Documentation**

*Version 2.0—Updated October 2017*

John Gallo1 and Randal Greene2

1Conservation Biology Institute

2Feaver’s Lane

**Table of Contents**

[1 Introduction 2](#_Toc511918600)

[2 Coding Conventions 2](#_Toc511918601)

[3 LP Code Organization 2](#_Toc511918602)

[3.1 lp\_settings.py 2](#_Toc511918603)

[3.2 lp\_config.py 3](#_Toc511918604)

[3.3 lp\_main.py 3](#_Toc511918605)

[3.4 Summary Diagram of all the modules 3](#_Toc511918606)

[3.5 Linkage Mapper Arc10.tbx 3](#_Toc511918607)

[3.6 LmDlgContent.xsl 3](#_Toc511918608)

[4 Geoprocessing Summary 5](#_Toc511918609)

[5 LP Classes and Functions 6](#_Toc511918610)

[5.1 lp\_config.py 6](#_Toc511918611)

[5.2 lp\_main.py 7](#_Toc511918612)

[6 LM Architecture 10](#_Toc511918613)

[6.1 Configuration and Settings 10](#_Toc511918614)

[6.2 Logging 10](#_Toc511918615)

[6.3 Other Utilities 10](#_Toc511918616)

[7 Other Notes 11](#_Toc511918617)

[7.1 Code Editing and Debugging 11](#_Toc511918618)

[7.2 Source Control Using GitHub 11](#_Toc511918619)

[7.3 Becoming a Linkage Mapper Contributor 11](#_Toc511918620)

[7.4 Project Management Using Trello 11](#_Toc511918621)

[7.5 Future Enhancements 11](#_Toc511918622)

# Introduction

Linkage Priority (LP) is an ArcGIS 10.x geoprocessing tool within the Linkage Mapper (LM) toolkit. Developers should first understand the LM and LP functionality from the user’s perspective. Please review the following user-oriented introductory material:

* Linkage Mapper User Guide version 2.0, including the LM tutorial.
* Linkage Priority User Guide version 2.0, including the LP tutorial. Pay particular attention to the Geoprocessing Overview for high-level introduction to the processing logic that underlies LP.
* Linkage Mapper web site (<http://www.circuitscape.org/linkagemapper>) and related publications (<http://www.circuitscape.org/pubs>).

Developers should also be familiar with the ArcGIS geoprocessing framework on which LM and LP are built. Please refer to <http://desktop.arcgis.com/en/arcmap/10.4/analyze/main/geoprocessing-framework.htm> for an introduction to this framework.

The LM family of tools are built as Python script tools in a standard ArcGIS toolbox. Please refer to <http://desktop.arcgis.com/en/arcmap/10.4/analyze/creating-tools/a-quick-tour-of-creating-tools-in-python.htm> for an introduction to creating ArcGIS geoprocessing tools in Python.

# Coding Conventions

As a Python project, developers are encouraged to follow the PEP8 style guide (<https://www.python.org/dev/peps/pep-0008/>). Please note the following conventions:

* Class, method, function and variable names – lowercase with underscore separators (e.g. function\_name()).
* Constant “variable” names – all uppercase (e.g. CONSTANT).
* Maximum line length – 120 characters. The other LM tools use an 80-character line length, but 120 characters improves readability on the typical modern high-resolution display. We recommend it be adopted for future enhancements to all LM tools.

# LP Code Organization

Most of the LP functionality is defined in three Python modules (.py files). LP is called by ArcGIS from the Linkage Mapper toolbox, which is defined in two additional files.

ArcGIS is flexible with regards to the installation location of custom toolboxes like Linkage Mapper. However, Linkage Mapper expects all .py files to be in a subdirectory called “scripts”.

## lp\_settings.py

A module containing “constants” representing advanced LP settings/parameters that are not included in the tool dialog.

## lp\_config.py

The LP configuration module, which defines the lp\_config class and creates an instance of it called lp\_env. It also has a number of local helper functions, which are described in the LP Classes and Functions section below.

## lp\_main.py

The primary LP module, which includes:

* The execution starting point (if \_\_name\_\_ == "\_\_main\_\_ "), which calls the main() controlling function.
* The main() controlling function.
* Functions for the preparation steps.
* The run\_analysis() function and numerous called functions, which implement the primary analysis logic. The called functions are organized as per the Geoprocessing Overview in the LM user guide, and the Geoprocessing Summary above.
* Local helper functions.
* All functions are described in the LP Classes and Functions section below.

## Summary Diagram of all the modules

* On the following page there is a summary diagram of all the modules in the Linkage Mapper toolbox (including those unrelated to Linkage Priority).
* It gives a good overview of dependancies.
* It is a snapshot in time (2018-04-19).
* Thanks to Darren Kavanagh, using the Pyreverse packacge.

## Linkage Mapper Arc10.tbx

This file defines the Linkage Mapper ArcGIS toolbox. It is modified by right-clicking on it in ArcMap or ArcCatalog and selecting:

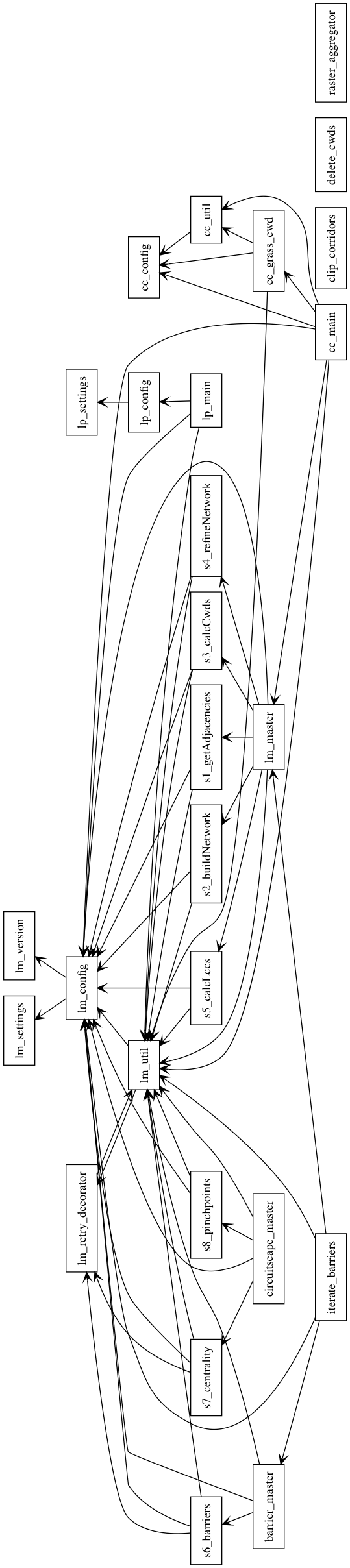
* Item Description, to view the tool metadata (including help text) and edit it.
* Properties, to define the tool’s:
  + Name
  + Label
  + Description
  + Optional underlying stylesheet
  + Underlying .py script file
  + Parameters (the type and order of these must match the order in which they are processed in the code)
  + Optional validation code to be run before the .py script is called
  + Optional compiled help file

In general, the toolbox should only be modified when there are new tools, new parameters for existing tools or updated help text.

## LmDlgContent.xsl

This file defines the stylesheet used by the Build Network and Map Linkages tool and the Linkage Priority tool. It facilitates group labeling and nesting of parameters in the tool dialog.

See note on previous page regarding this summary diagram.



# Geoprocessing Summary

The LP User Guide includes a geoprocessing overview. Additional details are provided here.

* Check that LM in the same Project Directory successfully finished Steps 3 and 5, and terminate if issues
* Make preliminary calculations for each corridor
  + Calculate Permeability
    - In project\_LCPs line feature class, calculate the attribute Raw\_Perm as LCP\_Length / CW\_Dist
    - In project\_LCPs line feature class, calculate the attribute Rel\_Perm as a normalization of all Raw\_Perm values
  + Calculate Relative Closeness
    - In project\_LCPs line feature class, calculate the attribute Rel\_Close as a normalization of all LCP\_Length values
  + Invert and normalize
    - For each intermediate corridor raster created by LM, normalize the raster values to create a corresponding intermediate inv\_norm raster
* Calculate Core Area Value (CAV) and its components for each core
  + Check weights and warn if issues
  + Add and calculate attributes in the input Core Area Feature Class
    - Mean resistance: mean\_res
    - Normalized resistance: norm\_res
    - Size: area
    - Normalized size: norm\_size
    - Perimeter: perimeter
    - Area/perimeter ratio: ap\_ratio
    - Normalized area/perimeter ratio: norm\_ratio
    - [Optional] Other core area value: ocav
    - [Optional] Normalized other core area value: nocav
    - [Optional] Expert core area value: ecav
    - [Optional] Normalized expert core area value: necav
    - [Optional] Current Flow Centrality value (from Centrality Mapper): CF\_Central
    - [Optional] Normalized CF\_Central value: ncfc
    - Core area value: cav
      * (norm\_res \* Resistance Weight) +

(norm\_size \* Size Weight) +\_

(norm\_ratio \* Area/Perimeter Weight) +

(necav \* Expert Core Area Value Weight) +

(ncfc \* Current Flow Centrality Weight) +

(nocav \* Other Core Area Value Weight)

* + - Normalized core area value: norm\_cav
* [Optional] Calculate climate envelope attributes for each core
  + Current climate envelope: clim\_env
    - Average of current climate envelope raster values within core
  + Normalized current climate envelope: nclim\_env
  + Future climate envelope: fut\_clim
    - Average of future climate envelope raster values within core
  + Normalized future climate envelope: nfut\_clim
* Complete calculations for each corridor
  + [Optional] Add and calculate attributes of the Core Pairs table
    - Normalize expert corridor importance value: neciv
  + Calculate corridor specific priority (CSP) raster for each corridor
    - Check weights and warn if issues
    - Calculate in-memory variables
      * Average the core area value of the two cores in the corridor: avg\_cav
      * [Optional] Difference the climate envelope of the two cores in the corridor: diff\_clim\_env
        + [Optional] If future climate envelope provided, use future climate envelope for the cooler core
    - Calculate CSP raster:
      * (Rel\_Close \* Closeness Weight) +

(Rel\_Perm \* Permeability Weight) +

(avg\_cav \* Core Area Value Weight) +

(neciv \* Expert Corridor Importance Value Weight) +

(diff\_clim\_env \* Climate Envelope Difference Weight)

* + - Apply Proportion CSP Values to Keep setting to reduce size of corridor and create CSP\_TOP raster
* Create overall result rasters
  + Combine CSP\_TOP rasters using Max and Mean to create overall Corridor Priority Value (project\_CPV) raster
    - Calculate sum, count and max of all CSP rasters
    - Calculate project\_CPV raster
      * (max \* MAXCSPWEIGHT) +

((sum / count) \* MEANCSPWEIGHT)

* + Clip project\_CPV to the MINCPV and normalize to create relative corridor importance (project\_RCI) raster
  + Clip project\_RCI to extent of truncated raster to create project\_linkage\_priority raster
  + Invert and normalize truncated raster to create project\_NORMTRUNC raster
  + Calculate overall project\_blended\_priority raster
    - (project\_NORMTRUNC \* Truncated Corridors Weight) +

(project\_linkage\_priority \* Linkage Priority Weight)

# LP Classes and Functions

## lp\_config.py

def nullfloat(innum):

"""Convert ESRI float or null to Python float."""

# returns float or None

def nullstring(arg\_string):

"""Convert ESRI nullstring to Python null."""

# returns string or None

def str2bool(pstr):

"""Convert ESRI boolean string to Python boolean type."""

# returns Boolean or None

class lp\_config(object):

"""Class container to hold Linkage Priority parameters and settings."""

# instantiated as lp\_env

def configure(self, arg):

"""Assign parameters and settings from passed arguments and advanced settings."""

## lp\_main.py

def delete\_datasets\_in\_workspace():

"""Delete all datasets in workspace."""

def delete\_dataset(dataset):

"""Delete one dataset."""

def delete\_arcpy\_temp\_datasets():

"""Delete datasets left behind by arcpy."""

def print\_runtime(stime):

"""Print process time when running from script."""

def main(argv=None):

"""Main function for Linkage Priority tool"""

# preparation steps

lp\_env.configure(argv)

check\_lp\_project\_dir()

check\_out\_sa\_license()

arc\_wksp\_setup()

config\_lm()

log\_setup()

# primary analysis

run\_analysis()

# includes exception handling and shutdown code

# explicitly or implicitly raised exceptions bubble up to here

def check\_lp\_project\_dir():

"""Checks to make sure path name is not too long. Long path names can cause problems with

ESRI grids."""

def check\_out\_sa\_license():

"""Check out the ArcGIS Spatial Analyst extension license."""

def arc\_wksp\_setup():

"""Setup ArcPy workspace."""

def config\_lm():

"""Configure Linkage Mapper."""

# get log file for last LM run

# read parameters section from file and turn into tuple for passing

# use LM config in addition to LP config by passing in parms from last LM run

def log\_setup():

"""Set up Linkage Mapper logging."""

# see LM Architecture below for additional details

def check\_add\_field(feature\_class, field\_name, data\_type):

"""Check if field exists, and if not then add."""

# return Boolean indicating whether the field previously existed

def normalize\_field(in\_table, in\_field, out\_field, normalization\_method,

invert=False):

"""Normalize values in in\_field into out\_field using score range or max score method, with

optional inversion."""

def normalize\_raster(in\_raster, normalization\_method, invert=False):

"""Normalize values in in\_raster using score range or max score method, with optional

inversion."""

# return normalized raster

def calc\_permeability(lcp\_lines):

"""Calculate raw and relative permeability for each Least Cost Path."""

def calc\_closeness(lcp\_lines):

"""Calculate relative closeness for each Least Cost Path."""

def inv\_norm():

"""Invert and normalize each corridor."""

# could be multiple nlc folders

def cav():

"""Calculate Core Area Value (CAV) and its components for each core."""

# check weights and warn if issues

# check/add fields

# calc mean resistance

# calc area, perimeter and ratio

# normalize CAV inputs

# resistance – invert

# size

# area/perimeter ratio

# optionally ecav

# optionally cfc

# optionally calc OCAV

# calc CAV

# normalize CAV with score range normalization

def clim\_env():

"""Calculate Climate Envelope(s) for each core."""

# calc score range normalization on current climate envelope

# calc aerial mean climate envelope for each core

# score range normalize resulting values

# optionally repeat for future climate envelope

def eciv():

"""Normalize Expert Corridor Importance Value (ECIV) for each corridor."""

def csp(sum\_rasters, count\_non\_null\_cells\_rasters, max\_rasters, lcp\_lines):

"""Calculate Corridor Specific Priority (CSP) for each corridor."""

# check weights

# could be multiple folders

# process each corridor raster in folder

# check for max 1

# get cores from raster name

# check for corresponding link

# get and avg CAVs for the core pair

# optionally get ECIV for the core pair

# optionally get and difference climate envelopes for the core pair to create CED

# optionally use future climate envelope for cooler core

# calc weighted sum CSP raster

# get max and min

# determine and apply threshold from Proportion Top CSP Values to Keep

# perform intermediate calculations on CSPs leading toward CPV

def cpv(sum\_rasters, count\_non\_null\_cells\_rasters, max\_rasters, cpv\_raster):

"""Combine CSPs using Max and Mean to create overall Corridor Priority Value (CPV)."""

def rci(cpv\_raster, rci\_raster):

"""Clip CPV to the MINCPV and renormalize to create relative corridor importance (RCI)."""

def linkage\_priority(rci\_raster, trunc\_raster, lp\_raster):

"""Clip RCI to extent of truncated raster (LP)."""

def norm\_trunc(trunc\_raster, norm\_trunc\_raster):

"""Invert and normalize truncated raster (NORMTRUNC)."""

def blended\_priority(norm\_trunc\_raster, lp\_raster, bp\_raster):

"""Calculate overall Blended Priority."""

def run\_analysis():

"""Run main Linkage Priority analysis."""

# check that LM finished with steps 3 and 5

# check/create gdb for scratch

# check/create gdb for intermediate

# set key dataset locations

# calc permeability

# calc relative closeness

# invert and normalize each corridor

# calc Core Area Value (CAV) and its components for each core

# normalize Expert Corridor Importance Value (ECIV)

# calc climate envelope

# calc Corridor Specific Priority (CSP)

# calc Corridor Priority Value (CPV)

# calc Relative Corridor Importance (RCI)

# calc Linkage Priority (LP)

# calc Blended Priority (BP)

# save a copy of Cores as the "Output for ModelBuilder Precondition"

# LM Architecture

As detailed above, LP modifies and builds on the standard LM outputs using the storage structure established by LM. As part of the LM family of tools, LP also takes advantage of some shared elements of the LM architecture.

## Configuration and Settings

For accessing the details of the successful LM run that LP builds on, LP instantiates and fills the lm\_config() class (defined in lm\_config.py) as lm\_env. This includes access to LM’s advanced settings defined in lm\_settings.py.

## Logging

LM provides logging to both the ArcGIS geoprocessing framework (for display and geoprocessing results history within ArcGIS) and to text files within the LM project structure (in the run\_history folder). LP uses the following functions logging functions from lm\_util.py:

def create\_log\_file(messageDir, toolName, inParameters):

"""creates and new text file for logging and remembers it for the current run"""

def write\_log(string):

"""write the string to the current log file"""

def close\_log\_file():

"""write the current time to the current log file and close the file"""

def gprint(string):

"""write the string to the current log file and pass to the ArcGIS geoprocessing framework as

a message or warning"""

def raise\_error(msg):

"""write the message to the current log file, pass it to the ArcGIS geoprocessing framework

as an error, and close the log file"""

## Other Utilities

LP also uses the following functions from lm\_util.py:

def build\_stats(raster):

"""Builds statistics and pyramids for output rasters"""

def delete\_data(dataset):

"""Delete the passed ArcGIS dataset"""

# Other Notes

## Code Editing and Debugging

The default Python editor is fairly rudimentary and, for many developers, not appropriate for larger projects like the LM family of tools. Code-aware text editors like notepad++ are an improvement, but do not offer interactive debugging. For interactive debugging, a Python-aware integrated development environment (IDE) is recommended. We have successfully used:

* PyCharm (<https://www.jetbrains.com/pycharm/>)
* Microsoft Visual Studio Community Edition (<https://www.visualstudio.com/vs/community/>)

Unfortunately, the current ArcGIS architecture does not support interactive debugging of Python geoprocessing tools when the tools are run from within ArcGIS. However, the code has been setup to support interactive debugging by starting the tool from within an IDE. Do this by:

* Opening lp\_main.py and going to the execution starting point (if \_\_name\_\_ == "\_\_main\_\_ ").
* Commenting out the call to main().
* Uncommenting out the subsequent block that builds an args list and passes it by calling main(args).
* Changing args to contain values relevant for the scenario to be debugged.
* Indicating to your IDE that lp\_main.py is the startup module.
* Starting the program in debug mode.
* Using breakpoints, watches and other interactive debugging techniques.

## Source Control Using GitHub

The LM family of tools are managed as an open source repository on GitHub. The repository URL is <https://github.com/linkagescape/linkage-mapper>.

## Becoming a Linkage Mapper Contributor

We encourage contributions to the LM project by ArcGIS/Python developers. This could include enhancements and fixes to existing tools, and development of new tools for the LM toolbox. We encourage new tools to follow the protocols in Linkage Priority and Climate Linkage Mapper, which are currently the two newest tools in the LM toolbox.

## Project Management Using Trello

Depends on outcome of discussion with Tim and Darren.

Does this belong here, or is it just for CBI staff/contractors?

## Future Enhancements

As with any active software project, there have been numerous suggestions for future enhancements. These include:

* Adding normalization method options (max score vs. score range) everywhere LP does normalization.
* Adding additional details to the LP log file, such as the parameters passed from ArcGIS (currently, these are accessible only from the ArcGIS geoprocessing results history).
* Improving tutorial with additional examples demonstrating LP’s optional parameters.

Please use the LM User Group (see the Support section of the LM User Guide) to register your suggestions.