# Wall-to-Wall Python library

Contents

[Wall-to-Wall Python library 1](#_Toc121822028)

[Introduction 1](#_Toc121822029)

[User-facing tools 2](#_Toc121822030)

[Commands 2](#_Toc121822031)

[Configuration format 3](#_Toc121822032)

[Requirements 3](#_Toc121822033)

[Builder configuration 3](#_Toc121822034)

[Fully-specified configuration 3](#_Toc121822035)

[Examples 6](#_Toc121822036)

[CASFRI - YT 6](#_Toc121822037)

[Standalone Template 7](#_Toc121822038)

## Introduction

The wall-to-wall tool helps simplify and automate the workflow of GCBM simulations for a wall-to-wall Canadian framework. It is less general-purpose than the typical "Standalone Template" workflow found in the training materials and is not meant to replace those steps for more complicated projects, or projects with data that requires the full set of transformation and filtering options found in the tiler library.

The tool performs these steps for a GCBM project

1. tiles pre-rollback layers
2. creates pre-rollback input database
3. runs rollback (if rollback is configured)
4. tiles post-rollback layers (if rollback is configured)
5. creates post-rollback input database (if rollback is configured)
6. configures GCBM to run

## User-facing tools

Users interact with the wall-to-wall library through a Python command-line application. The application accepts a JSON configuration file containing all the necessary information for tiling the spatial layers, creating the input database, performing a spatial inventory rollback, configuring and running GCBM, and queueing a GCBM run on the cluster.

### Commands

walltowall build <builder config file path> [output config file path]

* Use the builder configuration contained in the config file to fill in and configure the rest of the project; creates the fully-specified configuration in a separate file in the same directory as the builder config file unless an output config file path is specified.

walltowall prepare <fully-specified config file path> [output root path]

* Using the fully-specified project configuration in the config file, tile the spatial layers, generate the input database, run the spatial rollback, and configure the GCBM run. Project files are generated in various subdirectories off the specified output root path, or the directory containing the config file by default.

walltowall merge <fully-specified config file path> <prepared project root 1> <prepared project root 2> [prepared project root n …] --output\_path <output root path>

* Using the fully-specified project configuration from the first argument to get the GCBM config file template path and disturbance order, merge two or more projects together in descending priority order into a single project in the specified output path. Projects must have been prepared by the walltowall scripts and have the same spatial extent and resolution.

walltowall run (local|cluster) <prepared project root> [--config\_path <fully-specified config file path>]

* Run the specified project either locally or on the cluster, using the optionally specified config file for the project title and overridden executable or distributed run client paths.

## Configuration format

The configuration format for projects in the wall-to-wall library is designed to be more streamlined and user-friendly than writing a tiler script and running the Recliner2GCBM GUI, as well as allowing known standardized collections of input data (i.e. CASFRI) to be configured automatically through Python code in the library instead of through explicit configuration by the user.

There are two parts to the configuration: the optional "builder" configuration used with the "walltowall build" command that acts as a shorthand format for automatically configuring a project based on standardized sets of input data, and the fully-specified configuration read by the "walltowall prepare" command that contains all of the specific details needed to prepare and run the simulation.

### Requirements

The configuration file starts with the project name, followed by the optional builder configuration, and then the fully-specified configuration which can either be initially blank and generated by the builder using "walltowall build", or written manually if not using a builder.

{

"project\_name": <project name>,

[optional builder configuration]

(fully-specified configuration)

}

### Builder configuration

The builder configuration is an optional shortcut for creating projects using input data that conforms to the requirements of a builder module, usually allowing a much shorter configuration to generate the more detailed fully-specified configuration. It consists of the type of builder to use, followed by any configuration items recognized by the selected builder, and optionally any fully-specified configuration items that should take precedence over configuration generated by the builder.

"builder": {

"type": <builder type name>,

(any configuration specific to the selected builder)

[any main configuration items that take priority over builder-generated configuration]

}

Example:

{

"builder": {

"type": "casfri",

"casfri\_data": "../00\_preprocessing/casfri\_data/processed/YT03",

"other\_data": "../00\_preprocessing",

"yield\_table": "../00\_preprocessing/aspatial/yields/afforestation\_national/YT\_yield\_table.csv",

"dm\_xls": "../00\_preprocessing/aspatial/archive\_index/casfri\_dms.xlsx",

"aidb": "../00\_preprocessing/aspatial/archive\_index/casfri\_archive\_index.mdb",

"age\_distribution": "../../00\_preprocessing/aspatial/BGI\_age\_class\_dist.xlsx"

"rollback": {

"age\_distribution": "rollback/age\_distribution.json",

"inventory\_year": 2022

}

}

}

### Fully-specified configuration

The fully-specified configuration contains all of the details required to prepare and run a GCBM simulation. Because the objectives for this tool are to increase user-friendliness and efficiency, many of the configuration items are optional, and the tool will attempt to detect or use defaults for omitted configuration.

#### Layer definition

There are several places in the configuration that accept either a layer path for simple layers that comply with all the default settings, or a layer definition where the details of the layer can be specified more explicitly. Layer definitions can take these settings:

"layer": <path to layer>

[optional] "attribute": <attribute to read; default: search by layer name, or use first attribute found>

[optional] "attribute": {

<attribute to read>: <attribute value to filter for; only matching polygons are rasterized>

}

[optional] "lookup\_table": <path to csv file>

The lookup table replaces original pixel or attribute values with new ones. For rasters, the first column is the pixel value, and subsequent columns are the attributes. For vectors, the columns are paired up; the first column in each pair is named after the attribute to replace and contains the original values, and the second column’s name doesn’t matter and contains the values to substitute.

If a substitution does not appear in the lookup table, the original value is used.

If no lookup table is specified, the wall-to-wall tool searches first for <layer filename>.csv in the same directory as this configuration file, and then in the layer directory.

#### Required configuration items

"aidb": <path to ArchiveIndex database (AIDB)>,

"yield\_table": <path to yield table>,

"yield\_interval": <yield table age/volume interval, in years>,

##### Classifiers

Classifiers are what link the yield table to the spatial landscape. They are configured with at least a layer definition for the spatial component, and by default the walltowall script will attempt to match the spatial layer to the yield table, or the corresponding yield column and even a separate file containing all possible classifier values can be configured explicitly.

At least one classifier must be included in the "classifiers" section, with the configuration format:

"classifiers": {

<classifier name>: {

<layer definition items for spatial layer linked to classifier>,

[optional] "values\_path": <file with classifier values; default: use yield table>,

[optional] "values\_col": <column name/# in values\_path; default: use only column in single-column

file, or classifier name or spatial attribute, or column where values intersect with

spatial values>,

[optional] "yield\_col": <column name/# in yield table; default: use values\_col if values\_path is

the same as yield table path, otherwise search by classifier name, or search for column where

values are a subset of values from values\_path>

}

}

#### Optional configuration items

"resolution": <pixel resolution in degrees lat/lon>,

##### Bounding box

The bounding box defines the simulation area; nodata pixels in the bounding box are propagated to all other spatial layers. If the bounding box is not configured, the "initial\_age" layer is used.

"bounding\_box": <path to layer or full layer definition>

##### Layers

This section holds all the "miscellaneous" spatial layers that are not the bounding box, classifiers, or disturbances.

"layers": {

<layer name>: <path to layer or full layer definition>

}

##### Disturbances

This section holds any disturbance layers to be included with the simulation. The keys are the filenames or file patterns to search for, followed by the disturbance details. If the disturbance layer is a shapefile, the wall-to-wall tool will automatically split it into tiled layers by year.

"disturbances": {

<layer path or glob file pattern>: {

[optional] "year": <specific year, or name of attribute containing year, or "filename">

If specific year is given or the special keyword "filename" is used, use that exact year or try to parse it from the filename, respectively. If year is not specified, wall-to-wall will first search for an attribute named "year", then for the first column where all the values are 4-digit integers, and finally will check if the disturbance year can be parsed from the filename.

[optional] "disturbance\_type": <specific disturbance type, or name of attribute

containing disturbance type>

If disturbance type is not specified, wall-to-wall will search for the first attribute where all the values appear in tblDisturbanceTypeDefault in the AIDB.

[optional] "age\_after": <specific age after, or name of attribute containing reset age>

If age\_after is not specified, wall-to-wall will search for an attribute named "age\_after" or fall back to no transition rule if not found. If age\_after or regen\_delay are present, this causes a transition rule to be attached directly to the disturbance layer. regen\_delay can be omitted even if age\_after is present.

[optional] "regen\_delay": <specific regen delay, or name of attribute containing regen delay;

default: 0>

If regen\_delay is not present, wall-to-wall will search for an attribute named "regen\_delay" or fall back to no transition rule if not found and age\_after is not present. If age\_after or regen\_delay are present, this causes a transition rule to be attached directly to the disturbance layer. age\_after must be configured if regen\_delay is present, as there is no sensible default.

[optional] <other Layer definition items>

}

}

##### Rollback

If present, this section causes a spatial inventory rollback to be run.

"rollback": {

"age\_distribution": <path to age distribution JSON or Excel file>,

"inventory\_year": <path to inventory vintage layer or full layer definition, or global

inventory vintage year>,

[optional] "rollback\_year": <year to roll back to; default: 1990>,

[optional] "prioritize\_disturbances": <true/false, default: false>,

[optional] "single\_draw": <true/false, default: false>,

[optional] "establishment\_disturbance\_type": <default establishment disturbance type name>,

[optional] "disturbance\_order": <path to disturbance order text file>

}

## Examples

### CASFRI - YT

This configuration uses the CASFRI builder in the wall-to-wall tool to set up the Yukon simulation – for this example, assume that it's stored in a file called yt\_casfri.json:

{

"project\_name": "casfri\_yt",

"builder": {

"type": "casfri",

"casfri\_data": "../00\_preprocessing/casfri\_data/processed/YT03",

"other\_data": "../00\_preprocessing/other\_data",

"yield\_table": "../00\_preprocessing/yields/afforestation\_national/YT\_yield\_table.csv",

"dm\_xls": "../00\_preprocessing/archive\_index/casfri\_dms.xlsx",

"aidb": "../00\_preprocessing/archive\_index/casfri\_archive\_index.mdb",

"rollback": {

"age\_distribution": "rollback/age\_distribution.json",

"inventory\_year": 2022

}

}

}

After running walltowall build yt\_casfri.json, the builder fills in the rest of the project details based on some assumptions about the way CASFRI projects are structured:

{

"project\_name": "casfri\_yt",

"builder": {

"type": "casfri",

"casfri\_data": "../00\_preprocessing/casfri\_data/processed/YT03",

"other\_data": "../00\_preprocessing/other\_data",

"yield\_table": "../00\_preprocessing/yields/afforestation\_national/YT\_yield\_table.csv",

"dm\_xls": "../00\_preprocessing/archive\_index/casfri\_dms.xlsx",

"aidb": "../00\_preprocessing/archive\_index/casfri\_archive\_index.mdb",

"rollback": {

"age\_distribution": "rollback/age\_distribution.json",

"inventory\_year": 2022

}

},

"resolution": 0.001,

"aidb": "../00\_preprocessing/archive\_index/casfri\_archive\_index.mdb",

"yield\_table": "../00\_preprocessing/yields/afforestation\_national/YT\_yield\_table.csv",

"yield\_interval": 10,

"classifiers": {

"RU": {

"layer": "../00\_preprocessing/pspu/pspus\_2016.shp",

"attribute": "Reconcilia"

},

"LeadingSpecies": {

"layer": "../00\_preprocessing/casfri\_data/processed/YT03/layer\_1/leading\_species.tiff",

"values\_col": "casfri\_species\_name"

}

},

"layers": {

"initial\_age": "../00\_preprocessing/casfri\_data/processed/YT03/layer\_1/age\_2022.tiff",

"mean\_annual\_temperature": "../00\_preprocessing/other\_data/NAmerica\_MAT\_1971\_2000.tif",

"admin\_boundary": {

"layer": "../00\_preprocessing/other\_data/pspu/pspus\_2016.shp",

"attribute": "ProvinceNa"

},

"eco\_boundary": {

"layer": "../00\_preprocessing/other\_data/pspu/pspus\_2016.shp",

"attribute": "EcoBound\_1"

}

},

"disturbances": {

"../00\_preprocessing/casfri\_data/processed/YT03/layer\_1/disturbances\_\*.tiff": {}

},

"rollback": {

"age\_distribution": "rollback/age\_distribution.json",

"inventory\_year": 2022

}

}

After the fully-specified project configuration is generated, the project can be prepared by running:

walltowall prepare yt\_casfri.json

This tiles the spatial layers, generates the input database, runs the spatial inventory rollback, generates the rollback input database, and finally configures the GCBM simulation for running.

Finally, the simulation can be run either locally or on the cluster (with wall-to-wall correctly configured and tunnels already connected):

walltowall run local . --config\_path yt\_casfri.json

walltowall run cluster . --config\_path yt\_casfri.json

### Standalone Template

This assumes the same directory structure as the standalone template training project, with the wall-to-wall configuration file located in the root directory, and the input files in the usual locations.

There is no builder shortcut for this type of custom project, so we skip the builder step and go straight to hand-writing the fully-specified configuration, relying on the defaults to keep everything concise:

{

"project\_name": "standalone\_template",

"start\_year": 2010,

"end\_year": 2020,

"resolution": 0.00025,

"aidb": "input\_database/ArchiveIndex\_Beta\_Install.mdb",

"yield\_table": "input\_database/yield.csv",

"yield\_interval": 10,

"classifiers": {

"Classifier1": {

"layer": "layers/raw/inventory/inventory.shp"

},

"Classifier2": {

"layer": "layers/raw/inventory/inventory.shp"

}

},

"layers": {

"initial\_age": {

"layer": "layers/raw/inventory/inventory.shp",

"attribute": "AGE\_2010"

},

"mean\_annual\_temperature": {

"layer": "layers/raw/inventory/inventory.shp",

"attribute": "AnnualTemp"

}

},

"disturbances": {

"layers/raw/disturbances/disturbances.shp": {

"age\_after": 0,

"regen\_delay": 0

}

}

}

After the fully-specified project configuration is written, the project can be prepared by running:

walltowall prepare yt\_casfri.json

This tiles the spatial layers, generates the input database, runs the spatial inventory rollback, generates the rollback input database, and finally configures the GCBM simulation for running.

Finally, the simulation can be run either locally or on the cluster (with wall-to-wall correctly configured and tunnels already connected):

walltowall run local . --config\_path yt\_casfri.json

walltowall run cluster . --config\_path yt\_casfri.json