

# Generic Carbon Budget Model (GCBM) Technical Guide: JSON Workflow (“gcbmwalltowall”)

Natural Resources Canada  
Canadian Forest Service  
January 2025

## Table of contents

1. Prerequisites.....	3
2. Overview.....	3
3. Standalone Template .....	4
3.1 Defining the project in JSON configuration format .....	4
3.2 Prepare the simulation .....	8
3.3 Run the GCBM.....	9
3.4 Run postprocessing tools.....	9
4. Sample Data Exercise .....	10
4.1 Prepare the input data files .....	10
4.2 Edit the project configuration.....	10
4.3 Run the simulation .....	14

## 1. Prerequisites

This new Generic Carbon Budget Model (GCBM) JavaScript Object Notation (JSON) configuration workflow (“gcbmwalltowall”) is intended to replace the pre-existing manual project configuration workflow.

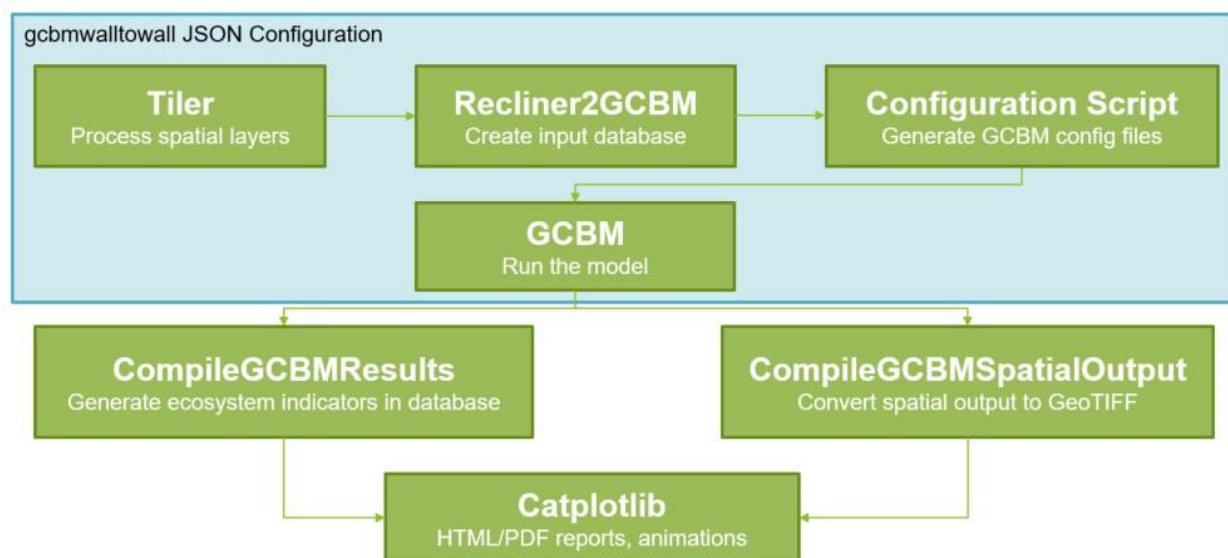
## 2. Overview

This new “gcbmwalltowall” Python package consolidates many of the individual preprocessing tools from the previous manual workflow into a single application driven by a .json configuration file.

This new workflow supports the same functionality as the previous manual workflow and can be used to assemble the same kinds of simulations, while also allowing for more advanced usage, such as:

- Composing simulations from fragments of the JSON configuration
- Layering on additional custom scripting, if required (for example, to add a set of disturbance layers to a base project created from the JSON configuration)

The “gcbmwalltowall” application consolidates the steps included in the blue box in the manual workflow in Figure 1.



**Figure 1. A flow diagram of the steps included in the manual workflow configuration of a Generic Carbon Budget Model (GCBM) project. Steps included in the blue box have been consolidated in the new JSON workflow configuration.**

### 3. Standalone Template

This section describes how to configure the “Standalone Template” training project and all the original input data using the JSON workflow instead of the previous manual workflow. There are no additional set-up steps required beyond the installation guide bundled with the training files. The procedure includes:

1. Defining the project in the “gcbmwalltowall” JSON configuration format
2. Running the “walltowall prepare” command to process spatial data, generate the input database, and prepare the GCBM run configuration
3. Running the GCBM using “walltowall run local”
4. Using postprocessing tools to generate the final simulation output, including:
  - a. Ecosystem indicators in database format
  - b. Raw spatial output to final layers

#### 3.1 Defining the project in JSON configuration format

Starting with a fresh copy of the Standalone Template project, create a subdirectory off the project root (i.e., where the “run\_all.bat” file is located) called “config” to hold the project configuration file.

The configuration file contains paths to other files. These paths can be either absolute or relative; when specified as relative paths, they are relative to the location of the configuration file.

##### **Create the base project configuration file**

Create a file called “walltowall\_config.json” in the new “config” directory, and enter the following basic project details:

```
{
  "project_name": "standalone_template",
  "start_year": 2010,
  "end_year": 2020,
  "resolution": 0.00025
}
```

The “project\_name” refers to the user-assigned project name. The “start\_year” and “end\_year” are the years when the simulation will start and end, respectively. The “resolution” represents the length of one side of a square pixel in EPSG:4326/WGS84 degrees latitude/longitude. All other layers in a GCBM simulation are reprojected and resampled to the selected resolution.

## Add the AIDB path


Identify the path to the Archive Index Database (AIDB). The AIDB contains all the (default Canadian) non-spatial modelling parameters (disturbance types and matrices, species and forest types, climate data, reporting jurisdictions, biomass, decay, and volume-to-biomass conversion parameters, etc.). For more information about the AIDB, consult the following document available on Natural Resources Canada's Open Science and Technology Repository: [The Carbon Budget Model of the Canadian Forest Sector \(CBM-CFS3\): Archive Index Database table and parameter descriptions](#). The path to the AIDB must be specified in the "aidb" configuration component below.

```
{
  "project_name": "standalone_template",
  "start_year": 2010,
  "end_year": 2020,
  "resolution": 0.00025,
  "aidb": "../input_database/ArchiveIndex_Beta_Install.mdb"
}
```

## Add the yield table information

The yield table can be in .xls, .xlsx, or .csv format and must contain one or more classifier columns, one AIDB species column (values entered should match species names in the AIDB), and a contiguous series of columns containing the yield curve age/volume pairs in a common increment (e.g., 1-year, 5-year, 10-year). Figure 2 displays an example of a partial yield table including classifier columns, the AIDBSP column (i.e., AIDB leading species), and gross merchantable volume yields in 10-year increments.

Classifier1	Classifier2	AIDBSP	0	10	20	n
BF	7	Poplar/Aspen	0	0	1.2	...
BP	5	Balsam fir	0	0	0	...
...	...	...	...	...	...	...



**Figure 2. An example of a yield table for a Generic Carbon Budget Model (GCBM) project displaying 2 classifier columns, the AIDBSP column, and 4 columns of gross merchantable volume for three age increments.**

The path to the yield table is specified for the "yield\_table" component, and the number of years between volume increment values is specified for the "yield\_interval" in the configuration below.

```
{
  "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
  }

```

## Add classifiers

Configure as many classifiers as needed to uniquely identify the rows in the yield table in the configuration below. Each classifier must be linked to a spatial layer to set its initial value in the simulation. Within the “classifiers” section, the top-level keys are the classifier names as they appear in the overall project and the results database tables. The project classifier names do not need to match the yield table column names or spatial layer attribute names—the “gcbmwalltowall” application will attempt to match the project classifiers to the corresponding yield table columns and spatial layer attributes based on the values that are found in one or more places.

```

{
  "project_name": "standalone_template",
  ...
  "classifiers": {
    "Classifier1": {
      "layer": "../layers/raw/inventory/inventory.shp",
      "attribute": "Classifier1"
    },
    "Classifier2": {
      "layer": "../layers/raw/inventory/inventory.shp",
      "attribute": "Classifier2"
    }
  }
}

```

In the configuration above, the explicit connection has been made between the project classifier names and their slightly abbreviated attribute names in the spatial layers (e.g., Classifier1 linked to Classifier1) using the optional “attribute” configuration item, and the link to the yield table is discovered automatically.

## Add other non-classifier layers

The GCBM accepts other non-classifier, non-disturbance layers in projects. In the configuration below, the “initial\_age” layer is required and sets the age of the forested pixels at the start of the simulation (this determines their position on their associated yield curves). An optional “mean\_annual\_temperature” layer can also be included in a simulation, and this will influence dead organic matter decay

rates. If no such spatial layer is provided, the default values will be based on the ecoboundary selected by the user for the project.

```
{
  "project_name": "standalone_template",
  ...
  "layers": {
    "initial_age": {
      "layer": "../layers/raw/inventory/inventory.shp",
      "attribute": "AGE_2010"
    },
    "mean_annual_temperature": {
      "layer": "../layers/raw/inventory/inventory.shp",
      "attribute": "AnnualTemp"
    }
  }
}
```

### Add disturbance layers

The user must configure the disturbance layers for their project. The sample dataset includes wildland fire and two different severities of mountain pine beetle. Disturbance layers are contained in the “disturbances” section of the configuration below, where each top-level key is a file name or file pattern, with any optional configuration items specified. The “gcbmwalltowall” application will attempt to detect the year of disturbance and disturbance type from layer attribute values or file naming conventions unless otherwise specified. Disturbance layers must contain—in either the attribute table or in part of the file name—a lookup (i.e., the year of disturbance in YYYY format) and the disturbance type, which must match a valid disturbance type name in the AIDB.

```
{
  "project_name": "standalone_template",
  ...
  "disturbances": {
    "../layers/raw/disturbances/disturbances.shp": {
      "age_after": 0,
      "regen_delay": 0
    }
  }
}
```

## 3.2 Prepare the simulation

Once the project has been configured, the next step is to prepare it for simulation using the “walltowall” command-line tool. This tool processes all the spatial and non-spatial input data into GCBM-format and configures the simulation to be run.

### Run walltowall prepare

The “prepare” step is included in the “run\_all.bat” file, but can also be run manually on its own with the following syntax:

```
walltowall prepare <path to config file> <path to project root>
```

**Note:** The project root should always be the directory containing the “run\_all.bat” file, for example, from the following command prompt where the current working directory is the same location as the “run\_all.bat” file:

```
walltowall prepare config\walltowall_config.json
```

### Examine the prepared project

The outputs of the “walltowall prepare” command are a set of GCBM-formatted spatial layers, the GCBM input database, and the model configuration files.

### Spatial layers

The “layers\tilled” directory contains the fully processed spatial input where each layer has been cropped, reprojected, and resampled according to the project configuration. By default, the “initial\_age” layer is used as the bounding box, unless explicitly configured by the user.

A GCBM-formatted layer consists of:

- The final cropped, reprojected, and rasterized layer padded to the nearest whole degree latitude and longitude
- A .json metadata file containing “layer\_data” (the data type of the layer), “nodata” (the nodata value), and “attributes” (an optional attribute table mapping pixel values to one or more attributes)

### GCBM input database

The GCBM input database (gcbm\_input.db), found in the “input\_database” folder, contains the yield curves, disturbance matrices, and all other non-spatial modelling parameters in SQLite format.

### Model configuration files



The model configuration is stored in the “gcbm\_project” directory and is spread across multiple .json files. These are generated from a set of templates bundled with the “gcbmwalltowall” application and control the behavior of the GCBM, including the start and end year, the paths to the various input data files created by the “prepare” step, and the spatial outputs that the model creates.

### 3.3 Run the GCBM

After the “prepare” step, the model can be run using the “walltowall run local” command, which is included in the “run\_all.bat” script, but can also be run manually from the command line. Bundled with the training files in the tools\GCBM directory, the path to the GCBM executable (moja.cli.exe) can be explicitly configured in the “gcbmwalltowall” project configuration file or globally configured in the “settings.json” file the <python root>\Tools\gcbmwalltowall\ directory.

#### Logs

The logs for most steps can be found in the “logs” folder, including the GCBM run log (moja\_debug.log). The “gcbmwalltowall” log (walltowall.log) is stored in the project root (i.e., the same location as the “run\_all.bat” file).

#### Raw GCBM output

The unprocessed GCBM output, both spatial and non-spatial, is stored in the gcbm\_project\output directory. After running GCBM, a pair of scripts must be run to convert the raw output into a more user-friendly format.

### 3.4 Run postprocessing tools

Users must run the GCBM post-processing tools, which involves both the database output and spatial output.

#### Database output

Raw GCBM tabular output is transformed into a flatter set of tables in a separate SQLite database by a script in the tools\CompileGCBMResults directory. The tables produced have a “v\_” prefix, and most contain an “indicator” column that groups pools or fluxes into named ecosystem indicators (e.g., NPP, NBP, Aboveground Biomass). The script is included in the “run\_all.bat” file, but can also be run on its own using:

```
tools\CompileGCBMResults\CompileGCBMResults.bat
```

The script generates the final results database called “compiled\_gcbm\_output.db”, found in the “processed\_output” folder.

## Spatial output

Raw GCBM spatial output is merged into single layers per indicator and year of simulation by a script in the `tools\CompileGCBMSpatialOutput` directory. The script is included in the “run\_all.bat” file, but can also be run separately using:

```
tools\CompileGCBMSpatialOutput\create_tiffs.bat
```

The resulting layers are stored in the `processed_output\spatial` directory.

## 4. Sample Data Exercise

This section illustrates how to adapt the Sample Data Exercise from the manual workflow to the JSON-based workflow. The Sample Data Exercise demonstrates how to use the same Standalone Template project structure to simulate a different set of data.

### 4.1 Prepare the input data files

The Sample Data Exercise uses the JSON workflow version of the Standalone Template project as a base, with a different set of spatial layers and yield curves.

To proceed, do the following:

- 1) Make a copy of the `3a_Standalone_Template_JSON_Workflow` directory and rename it to `3b_Sample_Data_Exercise_JSON_Workflow` (all other steps assume that this copy is the project root).
- 2) Delete the contents of the “raw” folder in the “layers” folder.
- 3) Copy the contents of the “Spatial” folder in the “2\_Sample\_Data” folder from the main training files directory into the “raw” folder in the “layers” folder.
- 4) Delete the “yield.csv” file in the “input\_database” folder.
- 5) Copy the “yield\_curves.csv” file in the `2_Sample_Data\Aspatial\` directory into the “input\_database” folder.

### 4.2 Edit the project configuration

In the “walltowall\_config.json” file in the “config” folder, make the necessary changes to the configuration to use the new input data.

## Update the basic project details

As indicated below, change the project name to “Sample data exercise”, remove the “start year” so that the default value of 1990 is used, change the “end year” to 2020, and set the “resolution” to 0.01. In essence, change the configuration from

```
{  
  "project_name": "standalone_template",  
  "start_year": 2010,  
  "end_year": 2020,  
  "resolution": 0.00025,
```

to

```
{  
  "project_name": "Sample data exercise",  
  "end_year": 2000,  
  "resolution": 0.01,
```

## Update yield table

Change the yield table file name in the configuration from

```
"yield_table": "../input_database/yield.csv",  
"yield_interval": 10,
```

to

```
"yield_table": "../input_database/yield_curves.csv",  
"yield_interval": 10,
```

## Update the classifiers

The new yield table only uses a single classifier, “LdSpp”, linked to an attribute in the Sample Data Exercise inventory layer. Remove the original two classifiers and replace them with the new classifier. This means changing the related configuration from

```
"classifiers": {
  "Classifier1": {
    "layer": "../layers/raw/inventory/inventory.shp",
    "attribute": "Classifier1"
  },
  "Classifier2": {
    "layer": "../layers/raw/inventory/inventory.shp",
    "attribute": "Classifier2"
  }
},
```

to

```
"classifiers": {
  "LdSpp": {
    "layer": "../layers/raw/inventory/inv_sample.shp"
  }
},
```

The application will try to match the classifier name (the top-level keys in the “classifiers” section) with an attribute of the same name in the spatial layer, and a column of the same name in the yield table. If this name-matching fails, it will try to match to the yield table by comparing the values from the spatial layer attribute with the column values in the yield table. These can be configured explicitly with the optional “attribute”, “values\_col”, and “yield\_col” configuration items.

### **Update the age layer**

The user will need to update the configuration for the age layer from

```
"initial_age": {
  "layer": "../layers/raw/inventory/inventory.shp",
  "attribute": "AGE_2010"
```

to

```
"initial_age": {
  "layer": "../layers/raw/inventory/inv_sample.shp",
  "attribute": "Age"
```

### **Configure the mean annual temperature**

The user will need to update the configuration for mean annual temperature from

```
"mean_annual_temperature": { "layer": "../layers/raw/inventory/inventory.shp",
  "attribute": "AnnualTemp"
}
```

to

```
"mean_annual_temperature": "../layers/raw/environment/namerica_mat_1971_2000.tif"
```

### Add wildland fire disturbance events

The Sample Data Exercise includes a set of raster-based fire disturbance layers in the layers\raw\Disturbances\Fire directory. Each year of disturbance is contained in a separate file that identifies the year as part of the file name. For this simulation, the GCBM's stand-replacing "Wildfire" disturbance type will be used, and "gcbmwalltowall" will detect that the year is part of the file name and extract it. The user will need to update the disturbance-related configuration from

```
"disturbances": {  
  "../layers/raw/disturbances/disturbances.shp": {  
    "age_after": 0,  
    "regen_delay": 0  
  }  
}
```

to

```
"disturbances": {  
  "../layers/raw/disturbances/fire/*.tif": {  
    "disturbance_type": "Wildfire"  
  }  
}
```

### Add insect disturbance events

There is a set of insect disturbance layers in the layers\raw\Disturbances\Insect directory. These layers are in vector format, and the disturbance type is mountain pine beetle in three different severity classes. The "SEVERITY" attribute contains the severity class (i.e., 1 [low], 2 [medium], or 3 [high]), and the "YEAR\_" attribute represents the year of disturbance and is also part of the file name.

Because no attribute in the shapefile matches a disturbance type name that the GCBM recognizes, a lookup table must be provided to map each severity class to a disturbance type. The sample data includes one that has already been prepared.

Copy the "mpb\_severity.csv" file in the Sample\_Data\Aspatial\ directory into the 3b\_Sample\_Data\_Exercise\_JSON\_Workflow\config directory. **Note:** The contents of the lookup table map an attribute that exists in the shapefile (i.e., "SEVERITY")

to new values in an adjacent column (i.e., `disturbance_type`); the name of the column with the substitutions can have any name that is unique within the file.

Finally, add the entry to the “disturbances” section of the configuration as follows

```
"./layers/raw/disturbances/insect/*.shp": {  
  "lookup_table": "mpb_severity.csv",  
  "disturbance_type": "SEVERITY",  
  "year": "filename"  
}
```

### 4.3 Run the simulation

The user must execute the “`run_all.bat`” file to run the simulation and postprocess the output into the final set of results in the “`processed_output`” directory.