
emod-api

Institute for Disease Modeling

May 20, 2022

CONTENTS

- 1 emod-api installation 3**
 - 1.1 Prerequisites 3
 - 1.2 Installation instructions 3
 - 1.3 Windows 4
- 2 Frequently asked questions 5**
- 3 API reference 7**
 - 3.1 emod_api package 7
 - 3.1.1 Subpackages 7
 - 3.1.2 Submodules 54
- 4 Glossary 57**
- Python Module Index 59**
- Index 61**

emod-api is the interface for Epidemiological MODELing software (EMOD) that users of idmtools interact with to create and modify EMOD simulations. Additional functionality for interacting with EMOD is provided in the [emodpy](#) package and [idmtools](#).

See [Welcome to idmtools](#) for a diagram showing how idmtools and each of the related packages are used in an end-to-end workflow using EMOD as the disease transmission model.

EMOD-API INSTALLATION

Follow the steps below to install emod-api.

1.1 Prerequisites

First, ensure the following prerequisites are met.

- Windows 10 Pro or Enterprise, Linux, or Mac
- Python 3.9 64-bit (<https://www.python.org/downloads/release>)
- A file that indicates the pip index-url:

- Windows
- Linux

In C:\Users\Username\pip\pip.ini, containing the following:

```
[global]  
index-url = https://packages.idmod.org/api/pypi/pypi-production/simple
```

In \$HOME/.config/pip/pip.conf, containing the following:

```
[global]  
index-url = https://packages.idmod.org/api/pypi/pypi-production/simple
```

1.2 Installation instructions

1. Open a command prompt and create a virtual environment in any directory you choose. The command below names the environment “v-emod-api”, but you may use any desired name:

```
python -m venv v-emod-api
```

2. Activate the virtual environment:

- Windows
- Linux

Enter the following:

```
v-emod-api\Scripts\activate
```

Enter the following:

```
source v-emod-api/bin/activate
```

3. Install emod-api packages:

```
pip install emod-api
```

If you are on Linux, also run:

```
pip install keyrings.alt
```

4. When you are finished, deactivate the virtual environment by entering the following at a command prompt:

```
deactivate
```

1.3 Windows

To properly install Shapely on Windows and/or if Snappy compression support is desired or needed, consider downloading and installing the latest python-snappy package for Windows from Christoph Gohlke's python package [website](#).

FREQUENTLY ASKED QUESTIONS

As you get started with emod-api, you may have questions. The most common questions are answered below. If you are using emodpy packages, see the FAQs from those packages for additional guidance.

I notice that I can import `emod_api.campaign` and use that as an object. I haven't seen that before. Sure.

Python modules are a lot like singletons. There's no need to add a static class inside that module in many cases.

Think of the module (which can have variables and methods) as a static class.

API REFERENCE

3.1 emod_api package

To generate a config.json from a param_overrides.json (or params-of-interest.json): python -m emod_api.config.from_overrides </path/to/po.json>

To generate a default config.json based on the schema for a given Eradication binary: python -m emod_api.config.from_schema -e </path/to/Eradication.[exe]> ...

To generate a schema.json: python -m emod_api.schema.get_schema </path/to/Eradication[.exe]>

For rest of emod-api documentation, please go to <https://github.com/InstituteforDiseaseModeling/emod-api>

3.1.1 Subpackages

emod_api.channelreports package

Submodules

emod_api.channelreports.channels module

Module for reading InsetChart.json channels.

```
class emod_api.channelreports.channels.Header(**kwargs)
    Bases: object
    property num_channels: int
    property dtk_version: str
    property time_stamp: str
    property report_type: str
    property report_version: str
    property step_size: int
        >= 1
    property start_time: int
        >= 0
    property num_time_steps: int
        >= 1
    as_dictionary() → dict
```

```
class emod_api.channelreports.channels.Channel(title, units, data)
    Bases: object
    property title: str
    property units: str
    property data
    as_dictionary() → dict

class emod_api.channelreports.channels.ChannelReport(filename: Optional[str] = None, **kwargs)
    Bases: object
    property header: emod_api.channelreports.channels.Header
    property dtk_version: str
    property time_stamp: str
    property report_type: str
    property report_version: str
        major.minor
    property step_size: int
        >= 1
    property start_time: int
        >= 0
    property num_time_steps: int
        > 0
    property num_channels: int
    property channel_names: list
    property channels: dict
        Channel objects keyed on channel name/title
    as_dataframe()
    write_file(filename: str, indent: int = 0, separators=(',', ':'))
        Write inset chart to specified text file.
```

emod_api.config package

Submodules

emod_api.config.default_from_schema module

emod_api.config.default_from_schema.write_default_from_schema(path_to_schema)
This module is deprecated. Please use default_from_schema_no_validation.

emod_api.config.default_from_schema_no_validation module

`emod_api.config.default_from_schema_no_validation.schema_to_config_subnode(schema_path_in,
subnode_list)`

This is the code from regular `schema_to_config`:

```
config = json.load(open("default_config.json"), object_hook=s2c.ReadOnlyDict) os.remove("default_config.json")
```

`emod_api.config.default_from_schema_no_validation.get_default_config_from_schema(path_to_schema,
schema_node=True,
as_rod=False,
out-put_filename=None)`

This returns a default config object as defined from reading a schema file.

Parameters `output_filename` (*str*) – if not None, the path to write the loaded config to

`emod_api.config.default_from_schema_no_validation.write_default_from_schema(path_to_schema,
out-put_filename='default_config.json',
schema_node=True)`

DEPRECATED: This function simply calls `get_default_config_from_schema` with specific arguments.

This function writes out a default config file as defined from reading a schema file. It's as good as the schema it's given. Note that this is designed to work with a schema from a disease-specific build, otherwise it may contain a lot of params from other disease types.

`emod_api.config.default_from_schema_no_validation.load_default_config_as_rod(config)`

Parameters `config` (*string/path*) – path to default or base config.json

Returns config (as `ReadOnlyDict`) with schema ready for schema-verified param sets.

`emod_api.config.default_from_schema_no_validation.get_config_from_default_and_params(config_path=None,
set_fn=None,
con-fig=None)`

Use this function to create a valid config.json file from a schema-derived base config, a callback that sets your parameters of interest

Parameters

- **config_path** (*string/path*) – Path to valid config.json
- **config** – read-only dict configuration object. Pass this XOR the `config_path`.
- **set_fn** (*function*) – Callback that sets params with implicit schema enforcement.

Returns read-only dict

Return type config

`emod_api.config.default_from_schema_no_validation.write_config_from_default_and_params(config_path,
set_fn,
con-fig_out_path)`

Use this function to create a valid config.json file from a schema-derived base config, a callback that sets your parameters of interest, and an output path.

Parameters

- **config_path** (*string/path*) – Path to valid config.json
- **set_fn** (*function*) – Callback that sets params with implicit schema enforcement.
- **config_out_path** – (string/path) Path to write new config.json

Returns Nothing

emod_api.config.dtk_post_process_adhocevents module

`emod_api.config.dtk_post_process_adhocevents.application(output_path)`

emod_api.config.dtk_pre_process_adhocevents module

`emod_api.config.dtk_pre_process_adhocevents.do_mapping_from_events(config, adhoc_events)`

Given a config file, a campaign file, and a list of `adhoc_events`, do the mappings. The `adhoc_event` list originally came from scraping an existing campaign file but now comes from `emod_api.campaign`.

`emod_api.config.dtk_pre_process_adhocevents.application(config)`

This is the public interface function to the submodule.

emod_api.config.dtk_pre_process_w5ml module

`emod_api.config.dtk_pre_process_w5ml.application(filename)`

emod_api.config.from_overrides module

`emod_api.config.from_overrides.flattenConfig(configjson_path, new_config_name='config.json')`

Historically called ‘flattening’ but really a function that takes a parameter override json config that includes a `Default_Config_Path` and produces a `config.json` from the two.

emod_api.config.from_poi_and_binary module

`emod_api.config.from_poi_and_binary.schema_to_config(schema_path_in)`

Purpose: Take a `schema.json` and return a “smart” config object that can be assigned to with schema-enforcement.

Use in conjunction with `to_file()`. Params: `schema_path_in` (str/path) Returns: config (smart dict)

`emod_api.config.from_poi_and_binary.set_schema(schema_path_in)`

`emod_api.config.from_poi_and_binary.make_config_from_poi_and_config_dict(start_config_dict,
poi_set_param_fn)`

Use this function to create a `config.json` from an existing param dict (defaults or base) and a function with your parameter overrides or parameters of interest.

`emod_api.config.from_poi_and_binary.make_config_from_poi_and_config_file(start_config_path,
poi_set_param_fn)`

Use this function to create a `config.json` from an existing config json file (defaults or base) and a function with your parameter overrides or parameters of interest.

`emod_api.config.from_poi_and_binary.make_config_from_poi_and_schema(schema_path,
poi_set_param_fn)`

Use this function to create a `config.json` from an existing schema json file and a function with your parameter overrides or parameters of interest.

`emod_api.config.from_poi_and_binary.make_config_from_poi(eradication_path, poi_set_param_fn)`

This function uses `emod_api` to produce a guaranteed working config starting with an Eradication binary and a parameters-of-interest python function. This is a usable and useful function.

Parameters

- **eradication_path** (*string*) – Fully-qualified path to Eradication binary that can be invoked to get a schema.
- **poi_set_param_fn** (*function*) – User-provided function/callback/hook that looks like:
- **set_params** (*def*) – `config.parameters.<param_name> = <schema valid param_value>`
 <repeat for each param> return config

Returns Hardcoded configuration filename written to pwd.

Return type “config.json” (string)

emod_api.config.from_schema module

argparse for command-line usage -s schema file -m model name -c config file

Sample code: `from emod_api.config import schema_to_config as s2c builder = s2c.SchemaConfigBuilder()
 builder.enumerate_params() builder.validate_dependent_params() builder.write_config_file()`

That will look for a local file called `schema.json` and produce a file called `config.json` that should work with an Eradication binary that produced the `schema.json`.

To build a default config for MALARIA_SIM, do: `builder = s2c.SchemaConfigBuilder(model="MALARIA_SIM")`

To generate a `schema.json` file from a binary, see help text for `emod_api.schema`.

```
class emod_api.config.from_schema.SchemaConfigBuilder(schema_name='schema.json',
                                                       model='GENERIC_SIM',
                                                       config_out='config.json', debug=False)
```

Bases: `object`

emod_api.config.schema_to_config module

```
class emod_api.config.schema_to_config.SchemaConfigBuilder(schema_name='schema.json',
                                                            model='GENERIC_SIM',
                                                            config_out='config.json', debug=False)
```

Bases: `emod_api.config.from_schema.SchemaConfigBuilder`

Deprecated in API v.1. Supported temporarily as pass-through functionality to `emod_api.config.from_schema`.

emod_api.demographics package

Submodules

emod_api.demographics.BaseInputFile module

```
class emod_api.demographics.BaseInputFile.BaseInputFile(idref)
```

Bases: `object`

```
    abstract generate_file(name)
```

generate_headers(*extra=None*)

emod_api.demographics.Demographics module

emod_api.demographics.Demographics.from_template_node(*lat=0, lon=0, pop=1000000, name='Erewhon', forced_id=1*)

Create a single-node Demographics instance from a few params.

emod_api.demographics.Demographics.from_file(*base_file*)

Create a Demographics instance from an existing demographics file.

emod_api.demographics.Demographics.get_node_ids_from_file(*demographics_file*)

Get a list of node ids from a demographics file.

emod_api.demographics.Demographics.get_node_pops_from_params(*tot_pop, num_nodes, frac_rural*)

Get a list of node populations from the params used to create a sparsely parameterized multi-node Demographics instance.

emod_api.demographics.Demographics.from_params(*tot_pop=1000000, num_nodes=100, frac_rural=0.3, id_ref='from_params', random_node_id=True*)

Create an EMOD-compatible Demographics object with the population and numbe of nodes specified.

Parameters

- **tot_pop** – The total population.
- **num_nodes** – Number of nodes. Can be defined as a two-dimensional grid of nodes [longitude, latitude]. The distance to the next neighbouring node is 1.
- **frac_rural** – Determines what fraction of the population gets put in the ‘rural’ nodes, which means all nodes besides node 1. Node 1 is the ‘urban’ node.
- **id_ref** – Facility name
- **random_node_id** – Create a random distanced grid with num_nodes nodes.

Returns Object of type Demographics

emod_api.demographics.Demographics.from_csv(*input_file, res=0.008333333333333333, id_ref='from_csv'*)

Create an EMOD-compatible Demographics instance from a csv population-by-node file.

emod_api.demographics.Demographics.from_pop_csv(*pop_filename_in, pop_filename_out='spatial_gridded_pop_dir', site='No_Site'*)

class emod_api.demographics.Demographics.Demographics(*nodes, idref='Gridded world grump2.5arcmin', base_file=None*)

Bases: [emod_api.demographics.BaseInputFile.BaseInputFile](#)

This class is a container of data necessary to produce a EMOD-valid demographics input file. It can be initialized from an existing valid demographics.joson type file or from an array of valid Nodes.

apply_overlay(*overlay_nodes: list*)

Parameters **overlay_nodes** – Overlay list of nodes over existing nodes in demographics

Returns

to_dict()

generate_file(name='demographics.json')

Write the contents of the instance to an EMOD-compatible (JSON) file.

property node_ids

Return the list of (geographic) node ids.

property nodes

property node_count

Return the number of (geographic) nodes.

get_node(nodeid)

Return the node identified by nodeid. Search either name or actual id :param nodeid: :return:

SetIndividualAttributesWithFertMort(CrudeBirthRate=0.04, CrudeMortRate=0.02)

AddIndividualPropertyAndHINT(Property: str, Values: List[str], InitialDistribution: Optional[List[float]] = None, TransmissionMatrix: Optional[List[List[float]]] = None, Transitions: Optional[List] = None)

Add Individual Properties, including an optional HINT configuration matrix.

Parameters

- **Property** – property (if property already exists an exception is raised).
- **Values** – property values.
- **InitialDistribution** – initial distribution.
- **TransmissionMatrix** – transmission matrix.

Returns N/A/

AddAgeDependentTransmission(Age_Bin_Edges_In_Years=[0, 1, 2, -1], TransmissionMatrix=[[1.0, 1.0, 1.0], [1.0, 1.0, 1.0], [1.0, 1.0, 1.0]])

Set up age-based HINT. Since ages are a first class property of an agent, Age_Bin is a special case of HINT. We don't specify a distribution, but we do specify the age bin edges, in units of years. So if Age_Bin_Edges_In_Years = [0, 10, 65, -1] it means you'll have 3 age buckets: 0-10, 10-65, & 65+. Always 'book-end' with 0 and -1.

Parameters

- **Age_Bin_Edges_In_Years** – array (or list) of floating point values, representing the age bucket boundaries.
- **TransmissionMatrix** – 2-D array of floating point values, representing epi connectedness of the age buckets.

SetDefaultIndividualAttributes()

NOTE: This is very Measles-ish. We might want to move into MeaslesDemographics

SetMinimalNodeAttributes()

SetBirthRate(birth_rate)

Set Default birth rate to birth_rate. Turn on Vital Dynamics and Births implicitly.

SetMortalityRate(mortality_rate, node_ids: Optional[List[int]] = None)

Set constant mortality rate to mort_rate. Turn on Enable_Natural_Mortality implicitly.

SetMortalityDistribution(distribution: Optional[emod_api.demographics.PropertiesAndAttributes.IndividualAttributes.MortalityDistribution] = None, node_ids: Optional[List[int]] = None)

Set a default mortality distribution for all nodes or per node. Turn on Enable_Natural_Mortality implicitly.

Parameters

- **distribution** – distribution
- **node_ids** – a list of node_ids

Returns None

SetMortalityOverTimeFromData(*data_csv*, *base_year*)

Set default mortality rates for all nodes or per node. Turn on mortality configs implicitly.

Parameters

- **data_csv** – Path to csv file with the mortality rates by calendar year and age bucket.
- **base_year** – The calendar year the sim is treating as the base.

Returns None

SetAgeDistribution(*distribution*:

emod_api.demographics.PropertiesAndAttributes.IndividualAttributes.AgeDistribution,
node_ids: *Optional[List[int]] = None*)

Set a default age distribution for all nodes or per node. Sets distribution type to COMPLEX implicitly.
:param distribution: age distribution :param node_ids: a list of node_ids

Returns None

SetDefaultNodeAttributes(*birth=True*)

Set the default NodeAttributes (Altitude, Airport, Region, Seaport), optionally including birth, which is most important actually.

SetDefaultIndividualProperties()

Initialize Individual Properties to empty.

SetDefaultProperties()

Set a bunch of defaults (age structure, initial susceptibility and initial prevalence) to sensible values.

SetDefaultPropertiesFertMort(*CrudeBirthRate=0.04*, *CrudeMortRate=0.02*)

Set a bunch of defaults (birth rates, death rates, age structure, initial susceptibility and initial prevalence) to sensible values.

SetDefaultFromTemplate(*template*, *setter_fn=None*)

Add to the default IndividualAttributes using the input template (raw json) and set corresponding config values per the setter_fn. The template should always be constructed by a function in DemographicsTemplates. Eventually this function will be hidden and only accessed via separate application-specific API functions such as the ones below.

SetNodeDefaultFromTemplate(*template*, *setter_fn*)

Add to the default NodeAttributes using the input template (raw json) and set corresponding config values per the setter_fn. The template should always be constructed by a function in DemographicsTemplates. Eventually this function will be hidden and only accessed via separate application-specific API functions such as the ones below.

SetEquilibriumAgeDistFromBirthAndMortRates(*CrudeBirthRate=0.04*, *CrudeMortRate=0.02*)

Set the initial ages of the population to a sensible equilibrium profile based on the specified input birth and death rates. Note this does not set the fertility and mortality rates.

SetInitialAgeExponential(*rate=0.0001068*, *description=""*)

Set the initial age of the population to an exponential distribution with a specified rate. :param rate: rate
:param description: description, why was this distribution chosen

SetInitialAgeLikeSubSaharanAfrica(*description=""*)

Set the initial age of the population to a overly simplified structure that sort of looks like sub-Saharan Africa. This uses the SetInitialAgeExponential. :param description: description, why was this age chosen?

SetOverdispersion(*new_overdispersion_value*, *nodes=[]*)

Set the overdispersion value for the specified nodes (all if empty).

SetConstantSusceptibility()

Set the initial susceptibility for each new individual to a constant value of 1.0.

SetInitPrevFromUniformDraw(*min_init_prev*, *max_init_prev*, *description=""*)

Set Initial Prevalence (one value per node) drawn from a uniform distribution. :param min_init_prev: minimal initial prevalence :param max_init_prevalence: maximal initial prevalence :param description: description, why were these parameters chosen?

SetConstantRisk(*risk=1*, *description=""*)

Set the initial risk for each new individual to the same value, defaults to full risk :param risk: risk :param description: description, why was this parameter chosen?

SetHeteroRiskUniformDist(*min_risk=0*, *max_risk=1*)

Set the initial risk for each new individual to a value drawn from a uniform distribution.

SetHeteroRiskLognormalDist(*mean=1.0*, *sigma=0*)

Set the initial risk for each new individual to a value drawn from a log-normal distribution.

SetHeteroRiskExponDist(*mean=1.0*)

Set the initial risk for each new individual to a value drawn from an exponential distribution.

SetFertilityOverTimeFromParams(*years_region1*, *years_region2*, *start_rate*, *inflection_rate*, *end_rate*)

Set fertility rates that vary over time based on a model with two linear regions. Note that fertility rates use GFR units: babies born per 1000 women of child-bearing age annually.

Parameters

- **years_region1** – The number of years covered by the first linear region. So if this represents 1850 to 1960, years_region1 would be 110.
- **years_region2** – The number of years covered by the second linear region. So if this represents 1960 to 2020, years_region2 would be 60.
- **start_rate** – The fertility rate at t=0.
- **inflection_rate** – The fertility rate in the year where the two linear regions meet.
- **end_rate** – The fertility rate at the end of the period covered by region1 + region2.

Returns rates array (Just in case user wants to do something with them like inspect or plot.)

infer_natural_mortality(*file_male*, *file_female*, *interval_fit=[1970, 1980]*, *which_point='mid'*,
predict_horizon=2050, *csv_out=False*, *n=0*,
results_scale_factor=0.0027397260273972603)

Calculate and set the expected natural mortality by age, sex, and year from data, predicting what it would have been without disease (usually HIV).

```
class emod_api.demographics.Demographics.DemographicsOverlay(nodes=None, meta_data:
    Optional[dict] = None,
    individual_attributes=None,
    node_attributes=None,
    mortality_distribution=None)
```

Bases: `object`

to_dict()

to_file(*file_name='demographics_overlay.json'*)

emod_api.demographics.DemographicsGenerator module**exception** emod_api.demographics.DemographicsGenerator.InvalidResolutionBases: `BaseException`

Custom Exception

class emod_api.demographics.DemographicsGenerator.DemographicsType(value)Bases: `enum.Enum`**STATIC** = 'static'

emod_api.demographics.DemographicsGenerator.arcsec_to_deg(arcsec: float) → float

Arc second to degrees :param arcsec: arcsecond as float

Returns arc second converted to degrees

emod_api.demographics.DemographicsGenerator.validate_res_in_arcsec(res_in_arcsec)

Validate that the resolution is valid :param res_in_arcsec: Resolution in arcsecond. Supported values can be found in VALID_RESOLUTIONS

Returns None.**Raise:** KeyError: If the resolution is invalid, a key error is raised**class** emod_api.demographics.DemographicsGenerator.DemographicsGenerator(nodes, concerns: *Optional[Union[emod_api.dtk_tools.demographics.DemographicsGeneratorConcern, List[emod_api.dtk_tools.demographics.DemographicsGeneratorConcern]]]*, res_in_arcsec='custom', node_id_from_lat_long=False)
Bases: `object`

Generates demographics file based on population input file. The population input file is csv with structure

node_label*, lat, lon, pop*

*-ed columns are optional

set_resolution(res_in_arcsec)

The canonical way to set arcsecond/degree resolutions on a DemographicsGenerator object. Verifies everything is set properly

Parameters res_in_arcsec – The requested resolution. e.g. 30, 250, 'custom'**Returns:** No return value.**generate_nodes**(defaults)

generate demographics file nodes

The process for generating nodes starts with looping through the loaded demographics nodes. For each node, we:

1. First determine the node's id. If the node has a forced id set, we use that. If we are using a custom resolution, we use the index (ie 1, 2, 3...). Lastly, we build the node id from the lat and lon id of the node

2. We then start to populate the node_attributes and individual attributes for the current node. The node_attributes will have data loaded from the initial nodes fed into DemographicsGenerator. The individual attributes start off as an empty dict.

3. We next determine the birthrate for the node. If the node attributes contains a Country element, we first lookup the birthrate from the World Pop data. We then build a MortalityDistribution configuration with

country specific configuration elements and add that to the individual attributes. If there is no Country element in the node attributes, we set the birth rate to the default_birth_rate. This value was set in initialization of the DemographicsGenerator to the birth rate of the specified country from the world pop data

4. We then calculate the per_node_birth_rate using get_per_node_birth_rate and then set the birth rate on the node attributes

5. We then calculate the equilibrium_age_distribution and use that to create the AgeDistribution in individual_attributes

6. We then add each new demographic node to a list to end returned at the end of the function

generate_metadata()

generate demographics file metadata

generate_demographics()

return all demographics file components in a single dictionary; a valid DTK demographics file when dumped as json

```
emod_api.demographics.DemographicsGenerator.from_dataframe(df, demographics_filename:
Optional[str] = None, concerns: Optional[Union[emod_api.dtk_tools.demographics.DemographicsNodeGeneratorConcern, List[emod_api.dtk_tools.demographics.DemographicsNodeGeneratorConcern]]] = None, res_in_arcsec='custom',
node_id_from_lat_long=True,
default_population: int = 1000,
load_other_columns_as_attributes=False,
include_columns: Optional[List[str]] = None, exclude_columns: Optional[List[str]] = None,
nodeid_column_name: Optional[str] = None, latitude_column_name: str = 'lat', longitude_column_name: str = 'lon', population_column_name: str = 'pop')
```

Generates a demographics file from a dataframe

Parameters

- **df** – pandas DataFrame containing demographics information. Must contain all the columns specified by latitude_column_name, longitude_column_name. The population_column_name is optional. If not found, we fall back to default_population
- **demographics_filename** – demographics file to save the demographics file too. This is optional
- **concerns** (*Optional[DemographicsNodeGeneratorConcern]*) – What DemographicsNodeGeneratorConcern should
- **DefaultWorldBankEquilibriumConcern** (*we apply. If not specified, we use the*) –
- **res_in_arcsec** – Resolution in Arcseconds
- **node_id_from_lat_long** – Determine if we should calculate the node id from the lat long. By default this is true unless you also set res_in_arcsec to CUSTOM_RESOLUTION. When not using lat/long for ids, the first fallback is to check the node for a forced id. If that is not found, we assign it an index as id
- **load_other_columns_as_attributes** – Load additional columns from a csv file as node attributes

- **include_columns** – A list of columns that should be added as node attributes from the csv file. To be used in conjunction with `load_other_columns_as_attributes`.
- **exclude_columns** – A list of columns that should be ignored as attributes when `load_other_columns_as_attributes` is enabled. This cannot be combined with `include_columns`
- **default_population** – Default population. Only used if `population_column_name` does not exist
- **nodeid_column_name** – Column name to load nodeid values from
- **latitude_column_name** – Column name to load latitude values from
- **longitude_column_name** – Column name to load longitude values from
- **population_column_name** – Column name to load population values from

Returns demographics file as a dictionary

```
emod_api.demographics.DemographicsGenerator.from_file(population_input_file: str,
                                                       demographics_filename: Optional[str] =
                                                       None, concerns: Op-
                                                       tional[Union[emod_api.dtk_tools.demographics.DemographicsC
                                                       List[emod_api.dtk_tools.demographics.DemographicsGenerator
                                                       = None, res_in_arcsec='custom',
                                                       node_id_from_lat_long=True,
                                                       default_population: int = 1000,
                                                       load_other_columns_as_attributes=False,
                                                       include_columns: Optional[List[str]] = None,
                                                       exclude_columns: Optional[List[str]] =
                                                       None, nodeid_column_name: Optional[str] =
                                                       None, latitude_column_name: str = 'lat',
                                                       longitude_column_name: str = 'lon',
                                                       population_column_name: str = 'pop')
```

Generates a demographics file from a CSV population

Parameters

- **population_input_file** – CSV population file. Must contain all the columns specified by `latitude_column_name`, `longitude_column_name`. The `population_column_name` is optional. If not found, we fall back to `default_population`
- **demographics_filename** – demographics file to save the demographics file too. This is optional
- **concerns** (*Optional[DemographicsNodeGeneratorConcern]*) – What DemographicsNodeGeneratorConcern should
- **DefaultWorldBankEquilibriumConcern** (*we apply. If not specified, we use the*) –
- **res_in_arcsec** – Resolution in Arcseconds
- **node_id_from_lat_long** – Determine if we should calculate the node id from the lat long. By default this is true unless you also set `res_in_arcsec` to `CUSTOM_RESOLUTION`. When not using lat/long for ids, the first fallback it to check the node for a forced id. If that is not found, we assign it an index as id
- **load_other_columns_as_attributes** – Load additional columns from a csv file as node attributes

- **include_columns** – A list of columns that should be added as node attributes from the csv file. To be used in conjunction with `load_other_columns_as_attributes`.
- **exclude_columns** – A list of columns that should be ignored as attributes when `load_other_columns_as_attributes` is enabled. This cannot be combined with `include_columns`
- **default_population** – Default population. Only used if `population_column_name` does not exist
- **nodeid_column_name** – Column name to load nodeid values from
- **latitude_column_name** – Column name to load latitude values from
- **longitude_column_name** – Column name to load longitude values from
- **population_column_name** – Column name to load population values from

Returns demographics file as a dictionary

emod_api.demographics.DemographicsInputDataParsers module

This file contains functions used to read, parse, and process input data files and convert the data into Nodes. Plus utility support function that are part of that process. There is no fixed fileformat for the incoming data. Any file format that is supported by a function here is a supported format. You can add to this.

```
emod_api.demographics.DemographicsInputDataParsers.node_ID_from_lat_long(lat, long,
                                                                           res=0.008333333333333333)

emod_api.demographics.DemographicsInputDataParsers.duplicate_nodeID_check(nodelist)

emod_api.demographics.DemographicsInputDataParsers.fill_nodes_legacy(node_info, DemoDf,
                                                                           res=0.008333333333333333)

emod_api.demographics.DemographicsInputDataParsers.ConstructNodesFromDataFrame(node_info,
                                                                           ex-
                                                                           tra_data_columns=[],
                                                                           res=0.008333333333333333)
```

emod_api.demographics.DemographicsTemplates module

```
emod_api.demographics.DemographicsTemplates.NoRisk()
    NoRisk puts everyone at 0 risk.

emod_api.demographics.DemographicsTemplates.FullRisk(demog, description="")
    FullRisk puts everyone at 100% risk.

emod_api.demographics.DemographicsTemplates.InitRiskUniform(demog, min_lim=0, max_lim=1,
                                                             description="")
    InitRiskUniform puts everyone at somewhere between 0% risk and 100% risk, drawn uniformly.
```

Parameters

- **min** (*float*) – Low end of uniform distribution. Must be ≥ 0 , < 1 .
- **max** (*float*) – High end of uniform distribution. Must be $\geq \text{min}$, ≤ 1 .
- **description** – Why were these values chosen?

Returns json object aka python dict that can be directly passed to `Demographics::SetDefaultFromTemplate`

Raises None –

`emod_api.demographics.DemographicsTemplates.InitRiskLogNormal(demog, mean=0.0, sigma=1.0)`
InitRiskLogNormal puts everyone at somewhere between 0% risk and 100% risk, drawn from LogNormal.

Parameters

- **mean** (*float*) – Mean of lognormal distribution.
- **sigma** (*float*) – Sigma of lognormal distribution.

Returns json object aka python dict that can be directly passed to `Demographics::SetDefaultFromTemplate`

Raises None –

`emod_api.demographics.DemographicsTemplates.InitRiskExponential(demog, mean=1.0)`
InitRiskExponential puts everyone at somewhere between 0% risk and 100% risk, drawn from Exponential.

Parameters **mean** (*float*) – Mean of exponential distribution.

Returns json object aka python dict that can be directly passed to `Demographics::SetDefaultFromTemplate`

Raises None –

`emod_api.demographics.DemographicsTemplates.NoInitialPrevalence(demog)`
NoInitialPrevalence disables initial prevalence; outbreak seeding must be done from an Outbreak intervention (or serialized population).

Parameters **demog** – `emod-api.demographics.Demographics` instance.

Returns None

Raises None –

`emod_api.demographics.DemographicsTemplates.InitPrevUniform(demog, low_prev, high_prev, description="")`

`emod_api.demographics.DemographicsTemplates.InitSusceptConstant(demog)`

`emod_api.demographics.DemographicsTemplates.EveryoneInitiallySusceptible(demog, setting=1.0)`

`emod_api.demographics.DemographicsTemplates.StepFunctionSusceptibility(demog, protected_setting=0.0, threshold_age=1825.0)`

`emod_api.demographics.DemographicsTemplates.SimpleSusceptibilityDistribution(demog, meanAgeAtInfection=2.5)`

`emod_api.demographics.DemographicsTemplates.DefaultSusceptibilityDistribution(demog)`

`emod_api.demographics.DemographicsTemplates.MortalityRateByAge(demog, age_bins, mort_rates)`
Set (non-disease) mortality rates by age bins. No checks are done on input arrays.

Parameters

- **age_bins** – list of age bins, with ages in years.
- **mort_rates** – list of mortality rates, where mortality rate is daily probability of dying..

Returns N/A.

`emod_api.demographics.DemographicsTemplates.MortalityStructureNigeriaDHS(demog)`

`emod_api.demographics.DemographicsTemplates.get_fert_dist_from_rates(rates)`

Write something...

`emod_api.demographics.DemographicsTemplates.get_fert_dist(path_to_csv)`

This function takes a fertility csv file (by year and age bin) and populates a DTK demographics.json file, and the corresponding config file to do individual pregnancies by age and year from data.

Parameters

- **demog** – `emod_api.demographics.Demographics` instance.
- **path_to_csv** – absolute path to csv input file. The file should have columns for 5-year age bins
- **"1950-1955"**. (labelled "15-19", etc. up to "45-49", and a column named "Years" with values like) –
- **anywhere**. (There can be extra columns and the columns can be) –

Returns (complex) dictionary. fertility distribution, ready to be added to demographics file.

`emod_api.demographics.DemographicsTemplates.InitAgeUniform(demog)`

`emod_api.demographics.DemographicsTemplates.AgeStructureUNWPP(demog)`

emod_api.demographics.Node module

```
class emod_api.demographics.Node.Node(lat, lon, pop, name: Optional[str] = None, area: Optional[float] =
    None, forced_id: Optional[int] = None, individual_attributes: Op-
    tional[emod_api.demographics.PropertiesAndAttributes.IndividualAttributes]
    = None, individual_properties: Op-
    tional[emod_api.demographics.PropertiesAndAttributes.IndividualProperties]
    = None, node_attributes: Op-
    tional[emod_api.demographics.PropertiesAndAttributes.NodeAttributes]
    = None, meta: Optional[dict] = None)
```

Bases: `emod_api.demographics.Updateable.Updateable`

default_population = 1000

res_in_degrees = 0.041666666666666664

to_dict() → dict

to_tuple()

property id

classmethod init_resolution_from_file(fn)

classmethod from_data(data: dict)

Function used to create the node object from data (most likely coming from a demographics file) :param data: :return:

property pop

property lon

property lat

property birth_rate

```
class emod_api.demographics.Node.OverlayNode(node_id, latitude=None, longitude=None,
                                              initial_population=None, **kwargs)
    Bases: emod_api.demographics.Node.Node
    Node that only requires an ID. Use to overlay a Node.
emod_api.demographics.Node.get_xpix_ypix(nodeid)
emod_api.demographics.Node.lat_lon_from_nodeid(nodeid, res_in_deg=0.041666666666666664)
emod_api.demographics.Node.xpix_ypix_from_lat_lon(lat, lon, res_in_deg=0.041666666666666664)
emod_api.demographics.Node.nodeid_from_lat_lon(lat, lon, res_in_deg=0.041666666666666664)
emod_api.demographics.Node.nodes_for_DTK(filename, nodes)
emod_api.demographics.Node.basicNode(lat: float = 0, lon: float = 0, pop: int = 1000000, name: str =
                                     'node_name', forced_id: int = 1)
```

emod_api.demographics.PreDefinedDistributions module

```
class emod_api.demographics.PreDefinedDistributions.ConstantDistribution(distribution)
    Bases: object
    Wrapping this class around a Distributions disables __setattr__ and makes the wrapped objects constant.
    to_dict()
        Calls the to_dict() method of the wrapped distribution.
    copy()
        Creates a deepcopy of the wrapped Distribution object.
```

emod_api.demographics.PropertiesAndAttributes module

```
class emod_api.demographics.PropertiesAndAttributes.IndividualProperty(initial_distribution:
                                                                    Optional[List[float]] =
                                                                    None, property=None,
                                                                    values:
                                                                    Optional[List[float]] =
                                                                    None, transitions:
                                                                    Optional[List[float]] =
                                                                    None,
                                                                    transmission_matrix:
                                                                    Optional[List[float]] =
                                                                    None)
    Bases: emod_api.demographics.Updateable.Updateable
    to_dict() → dict
class emod_api.demographics.PropertiesAndAttributes.IndividualProperties(individual_property:
                                                                    Op-
                                                                    tional[emod_api.demographics.PropertiesAndAttributes.IndividualProperty] =
                                                                    None)
    Bases: emod_api.demographics.Updateable.Updateable
    add(individual_property)
    add_parameter(key, value)
        Adds a user defined key-value pair to demographics. :param key: Key :param value: Value :return: None
```

`to_dict()` → dict

```
class emod_api.demographics.PropertiesAndAttributes.IndividualAttributes(age_distribution_flag=None,
                                                                           age_distribution1=None,
                                                                           age_distribution2=None,
                                                                           age_distribution=None,
                                                                           prevalence_distribution_flag=None,
                                                                           prevalence_distribution1=None,
                                                                           prevalence_distribution2=None,
                                                                           immunity_distribution_flag=None,
                                                                           immunity_distribution1=None,
                                                                           immunity_distribution2=None,
                                                                           risk_distribution_flag=None,
                                                                           risk_distribution1=None,
                                                                           risk_distribution2=None,
                                                                           migration_heterogeneity_distribution_flag=None,
                                                                           migration_heterogeneity_distribution1=None,
                                                                           migration_heterogeneity_distribution2=None,
                                                                           mortality_distribution=None,
                                                                           susceptibility_distribution=None)
```

Bases: `emod_api.demographics.Updateable.Updateable`

```
class SusceptibilityDistribution(distribution_values: Optional[List[float]] = None,
                                result_scale_factor=None, result_values=None)
```

Bases: `emod_api.demographics.Updateable.Updateable`

`to_dict()` → dict

```
class AgeDistribution(distribution_values=None, result_scale_factor=None, result_values=None)
```

Bases: `emod_api.demographics.Updateable.Updateable`

`to_dict()` → dict

`from_dict(age_distribution: dict)`

```
class MortalityDistribution(axis_names: Optional[List[str]] = None, axis_scale_factors:
                            Optional[List[float]] = None, axis_units=None,
                            num_distribution_axes=None, num_population_axes=None,
                            num_population_groups=None, population_groups=None,
                            result_scale_factor=None, result_units=None, result_values=None)
```

Bases: `emod_api.demographics.Updateable.Updateable`

`to_dict()` → dict

`from_dict(mortality_distribution: dict)`

`to_dict()` → dict

from_dict(*individual_attributes: dict*)

```
class emod_api.demographics.PropertiesAndAttributes.NodeAttributes(airport: Optional[int] =  
None, altitude=None, area:  
Optional[float] = None,  
birth_rate: Optional[float] =  
None, country=None,  
growth_rate: Optional[float]  
= None, name: Optional[str]  
= None, latitude:  
Optional[float] = None,  
longitude: Optional[float] =  
None, metadata:  
Optional[dict] = None,  
initial_population:  
Optional[int] = None,  
region: Optional[int] =  
None, seaport: Optional[int]  
= None,  
larval_habitat_multiplier:  
Optional[List[float]] =  
None, ini-  
tial_vectors_per_species=None,  
infectivity_multiplier:  
Optional[float] = None,  
extra_attributes:  
Optional[dict] = None)
```

Bases: *emod_api.demographics.Updateable.Updateable*

from_dict(*node_attributes: dict*)

to_dict() → *dict*

emod_api.demographics.Updateable module

class *emod_api.demographics.Updateable.Updateable*

Bases: *object*

(Base) class that provides *update()* method for each class that inherits from this class.

to_dict() → *dict*

update(*overlay_object*)

Updates an object with the values from *overlay_object*. :param *overlay_object*: Object that is used to update
self :return: None

add_parameter(*key, value*)

Adds a user defined key-value pair to demographics. :param *key*: Key :param *value*: Value :return: None

emod_api.demographics.demographics_utils module

`emod_api.demographics.demographics_utils.set_risk_mod(filename, distribution, par1, par2)`
Set the RiskDistributionFlag, RiskDistribution1 and RiskDistribution2 in a demographics file.

Parameters

- **filename** – The demographics file location
- **distribution** – The selected distribution (need to come from `distribution_types`)
- **par1** – Parameter 1 of the distribution
- **par2** – Parameter 2 of the distribution (may be unused depending on the selected distribution)

Returns Nothing

`emod_api.demographics.demographics_utils.set_immune_mod(filename, distribution, par1, par2)`
Set the ImmunityDistributionFlag, ImmunityDistribution1 and ImmunityDistribution2 in a demographics file.

Parameters

- **filename** – The demographics file location
- **distribution** – The selected distribution (need to come from `distribution_types`)
- **par1** – Parameter 1 of the distribution
- **par2** – Parameter 2 of the distribution (may be unused depending on the selected distribution)

Returns Nothing

`emod_api.demographics.demographics_utils.apply_to_defaults_or_nodes(demog, fn, *args)`
Apply the `fn` function either to the Defaults dictionary or to each of the nodes depending if the IndividualAttributes parameter is present in the Defaults or not.

Parameters

- **demog** – The demographic file represented as a dictionary
- **fn** – The function to apply the Defaults or individual nodes
- **args** – Argument list needed by `fn`

Returns Nothing

`emod_api.demographics.demographics_utils.set_demog_distributions(filename, distributions)`
Apply distributions to a given demographics file. The distributions needs to be formatted as a list of (name, distribution, par1, par2) with:

- **name**: Immunity, Risk, Age, Prevalence or MigrationHeterogeneity
- **distribution**: One distribution contained in `distribution_types`
- **par1, par2**: the values for the distribution parameters

```
# Set the PrevalenceDistribution to a uniform distribution with 0.1 and 0.2
# and the ImmunityDistributionFlag to a constant distribution with 1
demog = json.load(open("demographics.json", "r"))
distributions = list()
distributions.add(("Prevalence", "UNIFORM_DISTRIBUTION", 0.1, 0.2))
distributions.add(("Immunity", "CONSTANT_DISTRIBUTION", 1, 0))
set_demog_distribution(demog, distributions)
```

Parameters

- **filename** – the demographics file as json
- **distributions** – the different distributions to set contained in a list

Returns Nothing

`emod_api.demographics.demographics_utils.set_static_demographics(cb, use_existing=False)`

Create a static demographics based on the demographics file specified in the config file of the DTKConfigBuilder object passed to the function.

This function takes the current demographics file and adjust the birth rate/death rate to get a static population (the deaths are always compensated by new births).

Parameters

- **cb** – The config builder object
- **use_existing** – If True will only take the demographics file name and add the .static to it. If False will create a static demographics file based on the specified demographics file.

Returns Nothing

`emod_api.demographics.demographics_utils.set_growing_demographics(cb, use_existing=False)`

This function creates a growing population. It works the same way as the [set_static_demographics](#) but with a birth rate more important than the death rate which leads to a growing population.

Parameters

- **cb** – The DTKConfigBuilder object
- **use_existing** – If True will only take the demographics file name and add the .growing to it. If False will create a growing demographics file based on the specified demographics file.

Returns Nothing

emod_api.demographics.grid_construction module

- construct a grid from a bounding box
- label a collection of points by grid cells
- input: - points csv file with required columns lat,lon # see example input files (structures_households.csv)
- **output: - csv file of grid locations**
 - csv with grid cell id added for each point record

`emod_api.demographics.grid_construction.get_grid_cell_id(idx, idy)`

`emod_api.demographics.grid_construction.construct(x_min, y_min, x_max, y_max)`
Creating grid

`emod_api.demographics.grid_construction.get_bbox(data)`

`emod_api.demographics.grid_construction.lon_lat_2_point(lon, lat)`

`emod_api.demographics.grid_construction.point_2_grid_cell_id_lookup(point, grid_id_2_cell_id, origin)`

emod_api.interventions package

Submodules

emod_api.interventions.ccdl module

Proto-schema

WHEN :: WHERE :: WHO :: WHAT

WHEN: <Start_Time>-<End_Time> OR <Start_Time>(x<Repetitions>)/<Time_BetweenReps>

WHERE: AllPlaces OR Node_List

WHO: <Coverage%>/<IP>/<Min_Age>/<Max_Age>/<Sex>

WHAT: <Triggers>-><Intervention_Name1(Payload)>+<Intervention_Name2(Payload)>+...

emod_api.interventions.ccdl_viz module

Early draft of a very handy utility that takes a CCDL file (Concise Campaign Definition Language) and creates a graph(viz) visualization of it.

`emod_api.interventions.ccdl_viz.get_nickname_from_event(event_num, pieces)`

Allow nodes to get briefer and potentially more helpful nicknames. Default will probably remain a the nasty autogen above. Users can override this function with a callback of their own.

`emod_api.interventions.ccdl_viz.get_colour_from_event(tokens)`

Allow nodes to get a content-dependent colour. Default to just white. Users can override this function with a callback of their own. Have been using colour to capture IP categories.

`emod_api.interventions.ccdl_viz.get_shape_from_event(tokens)`

Allow nodes to get a content-dependent shape. Default to circle. Users can override this function with a callback of their own. Have been using shape to capture 'epoch' categories. Possible shapes include ellipse, circle, square, and diamond. Full list can be found at: <https://www.graphviz.org/doc/info/shapes.html>

`emod_api.interventions.ccdl_viz.set_beautifiers(name_cb=None, colour_cb=None, shape_cb=None)`

Override default no-op callbacks for setting nicknames, colours, and shapes of campaign nodes

`emod_api.interventions.ccdl_viz.viz(in_name='campaign.ccdl', out_name='camp.svg', display=True, whitelist=None)`

emod_api.interventions.common module

`emod_api.interventions.common.BroadcastEvent(camp, Event_Trigger: str = 'Births')`

Wrapper function to create and return a BroadcastEvent intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **Event_Trigger** – A valid trigger/event/signal.

Returns Schema-based smart dictionary representing a new BroadcastEvent intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.BroadcastEventToOtherNodes(camp, Event_Trigger,  
                                                         Node_Selection_Type='DISTANCE_ONLY',  
                                                         Max_Distance_To_Other_Nodes_Km=- 1,  
                                                         Include_My_Node=1)
```

Wrapper function to create and return a BroadcastEventToOtherNodes intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **Event_Trigger** – A valid trigger/event/signal.
- **Node_Selection_Type** – TBD.
- **Max_Distance_To_Other_Nodes_Km** – TBD.
- **Include_My_Node** – TBD.

Returns Schema-based smart dictionary representing a new BroadcastEvent intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.MultiInterventionDistributor(camp, Intervention_List)
```

Wrapper function to create and return a MultiInterventionDistributor intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **Intervention_List** – List of 1 or more valid intervention dictionaries to be
- **together.** (*distributed*) –

Returns Schema-based smart dictionary representing a new MultiInterventionDistributor intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.DelayedIntervention(camp, Configs, Delay_Dict=None)
```

Wrapper function to create and return a DelayedIntervention intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **Config** – Valid intervention config.
- **Delay_Dict** – Dictionary of 1 or 2 params that are the literal Delay_Distribution
- **E.g.,** (*parameters, but without the distribution, which is inferred.*) –
- **"Delay_Period_Exponential"** (*{}*) – 5 }

Returns Schema-based smart dictionary representing a new DelayedIntervention intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.HSB(camp, Event_Or_Config='Event', Config=None, Event='NoTrigger',  
                                  Tendency=1.0, Single_Use=True, Name='HSB')
```

Wrapper function to create and return a HealthSeekingBehaviour intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **Event_Or_Config** – “Event” or “Config”.

- **Config** – Complete, valid intervention configuration to be distributed.
- **Event** – Event/Trigger/Signal to be broadcast, alternative to an intervention.
- **Tendency** – Daily probability of ‘seeking care’ aka distributing payload intervention.
- **Single_Use** – One-and-done, or continuous?
- **Name** – Intervention Name. Useful if you want to provide uniqueness and not worry about
- **management.** (*duplicate intervention*) –

Returns Schema-based smart dictionary representing a new HSB intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.NLHTI(camp, Triggers, Interventions, Property_Restrictions=None,
Demographic_Coverage=1.0, Target_Age_Min=0,
Target_Age_Max=45625, Target_Gender='All',
Target_Residents_Only=False, Duration=- 1,
Blackout_Event_Trigger=None, Blackout_Period=None,
Blackout_On_First_Occurrence=None,
Disqualifying_Properties=None)
```

Wrapper function to create and return a NodeLevelHealthTriggeredIntervention intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **Triggers** – List of Triggers/Events/Signals
- **Interventions** – List of interventions to distribute when signal is heard.
- **Property_Restrictions** – Individual Properties that an agent must have to qualify for intervention.
- **Demographic_Coverage** – Percentage of individuals to receive intervention.
- **Target_Age_Min** – Minimum age (in years).
- **Target_Age_Max** – Maximum age (in years).
- **Target_Gender** – All, Male, or Female.
- **Target_Residents_Only** – Not used.
- **Duration** – How long this listen-and-distribute should last.
- **Blackout_Event_Trigger** – Not used.
- **Blackout_Period** – Not used.
- **Blackout_On_First_Occurrence** – Not used.
- **Disqualifying_Properties** – Not used.

Returns Schema-based smart dictionary representing a new NLHTI intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.PropertyValueChanger(camp, Target_Property_Key,  
                                                    Target_Property_Value, Daily_Probability=1.0,  
                                                    Maximum_Duration=1, Revert=- 1,  
                                                    Intervention_Name="",  
                                                    Event_Trigger_Distributed="",  
                                                    Event_Trigger_Expired="")
```

Wrapper function to create and return a PropertyValueChanger intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **IP.** (*Target_Property_Value*. The value part of the new key-value pair of the) –
- **IP.** –
- **key** (*New_Property_Value*.. Optional *IP*) – value part to be set, common to all interventions.
- **Target_Property_Value.** (*Daily_Probability*. The daily probability that an individual will move to the) –
- **Daily_Probability.** (*Maximum_Duration*. The maximum amount of time individuals have to move to a new group. This timing works in conjunction with) –
- **group.** (*Revert*. The number of days before an individual moves back to their original) –
- **policy.** (*Intervention_Name*. Optional *Intervention_Name*. Useful if managing a replacement) –
- **distributed.** (*Event_Trigger_Distributed*. Optional broadcast trigger to be published when PVC is) –
- **expired.** (*Event_Trigger_Expired*. Optional broadcast trigger to be published when PVC is) –

Returns Schema-based smart dictionary representing a new PropertyValueChanger intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.ScheduledCampaignEvent(camp, Start_Day: int, Node_Ids=None,  
                                                    Nodeset_Config=None, Number_Repetitions:  
                                                    int = 1, Timesteps_Between_Repetitions: int =  
                                                    - 1, Event_Name: str =  
                                                    'Scheduled_Campaign_Event',  
                                                    Property_Restrictions=None,  
                                                    Demographic_Coverage: float = 1.0,  
                                                    Target_Age_Min=0, Target_Age_Max=45625,  
                                                    Target_Gender: str = 'All',  
                                                    Target_Residents_Only: bool = False,  
                                                    Intervention_List=None)
```

Wrapper function to create and return a ScheduledCampaignEvent intervention. The alternative to a ScheduledCampaignEvent is a TriggeredCampaignEvent.

Parameters

- **camp** – emod_api.campaign object with schema_path set.

- **Start_Day** – When to start.
- **Event_Name** – Name for overall campaign event, of no functional meaning. Not in schema and not yet used.
- **Node_Ids** – Nodes to target with this intervention
- **Nodeset_Config** – Nodes to target with this intervention, return from `utils.do_nodes()`.
Deprecated since version 2.x: Use parameter `Node_Ids` instead
- **Property_Restrictions** – Individual Properties a person must have to receive the intervention(s).
- **Number_Repetitions** – N/A
- **Timesteps_Between_Repetitions** – N/A
- **Demographic_Coverage** – Percentage of individuals to receive intervention.
- **Target_Age_Min** – Minimum age (in years).
- **Target_Age_Max** – Maximum age (in years).
- **Target_Gender** – All, Male, or Female.
- **Intervention_List** – List of 1 or more valid intervention dictionaries to be
- **together.** (*distributed*) –

Returns Schema-based smart dictionary representing a new `ScheduledCampaignEvent` intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.TriggeredCampaignEvent(camp, Start_Day: int, Event_Name: str,
Triggers: List[str], Intervention_List:
List[dict], Node_Ids=None,
Nodeset_Config=None,
Node_Property_Restrictions=None,
Property_Restrictions=None,
Number_Repetitions: int = 1,
Timesteps_Between_Repetitions: int = - 1,
Demographic_Coverage: float = 1.0,
Target_Age_Min=0, Target_Age_Max=45625,
Target_Gender: str = 'All',
Target_Residents_Only=False, Duration=- 1,
Blackout_Event_Trigger: Optional[str] =
None, Blackout_Period=0,
Blackout_On_First_Occurrence=0,
Disqualifying_Properties=None, Delay=None)
```

Wrapper function to create and return a `TriggeredCampaignEvent` intervention. The alternative to a `TriggeredCampaignEvent` is a `ScheduledCampaignEvent`.

Parameters

- **camp** – `emod_api.campaign` object with `schema_path` set.
- **Start_Day** – When to start.
- **Event_Name** – Name for overall campaign event, of no functional meaning. Not in schema and not yet used.
- **Node_Ids** – Nodes to target with this intervention

- **Nodeset_Config** – Nodes to target with this intervention, return from `utils.do_nodes()`.
Deprecated since version 2.x: Use parameter `Node_Ids` instead
- **Triggers** – List of triggers/events/signals to listen to in order to trigger distribution.
- **Intervention_List** – List of 1 or more valid intervention dictionaries to be
- **together.** (*distributed*) –
- **Node_Property_Restrictions** – N/A.
- **Property_Restrictions** – Individual Properties a person must have to receive the intervention(s).
- **Demographic_Coverage** – Percentage of individuals to receive intervention.
- **Target_Age_Min** – Minimum age (in years).
- **Target_Age_Max** – Maximum age (in years).
- **Target_Gender** – All, Male, or Female.
- **Target_Residents_Only** – TBD.
- **Duration** – How long this listen-and-distribute should last.
- **Blackout_Event_Trigger** – Not used.
- **Blackout_Period** – Not used.
- **Blackout_On_First_Occurrence** – Not used.
- **Disqualifying_Properties** – Not used.
- **delay** – Optional delay between trigger and actual distribution.

Returns Schema-based smart dictionary representing a new `TriggeredCampaignEvent` intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.StandardDiagnostic(camp, Base_Sensitivity: float = 1.0,  
                                                Base_Specificity: float = 1.0, Days_To_Diagnosis:  
                                                float = 0.0, Event_Trigger_Distributed:  
                                                Optional[str] = None, Event_Trigger_Expired:  
                                                Optional[str] = None,  
                                                Positive_Diagnosis_Intervention=None,  
                                                Positive_Diagnosis_Event: str = 'PositiveResult',  
                                                Negative_Diagnosis_Intervention=None,  
                                                Negative_Diagnosis_Event: str = 'NegativeResult',  
                                                Treatment_Fraction: float = 1.0)
```

Wrapper function to create and return a `StandardDiagnostic` intervention.

Parameters

- **camp** – `emod_api.campaign` object with `schema_path` set.
- **Base_Sensitivity** – base sensitivity [0..1]
- **Base_Specificity** – base specificity [0..1]
- **Days_To_Diagnosis** – days to diagnosis
- **Event_Trigger_Distributed** – A trigger that is fired when intervention was distributed
- **Event_Trigger_Expired** – A trigger that is fired when intervention has expired

- **Positive_Diagnosis_Intervention** – Intervention that is distributed in case of a positive diagnosis. If set, no events may be configured.
- **Positive_Diagnosis_Event** – A trigger that is fired in case of a positive diagnosis
- **Negative_Diagnosis_Intervention** – Intervention that is distributed in case of a Negative diagnosis. If set, no events may be configured. Not used outside of Malaria-Ongoing yet.
- **Negative_Diagnosis_Event** – A trigger that is fired in case of a Negative diagnosis. Not used outside of Malaria-Ongoing yet.
- **Treatment_Fraction** – treatment fraction [0..1]

Returns Schema-based smart dictionary representing a new MultiInterventionDistributor intervention ready to be added to a campaign.

Return type *ReadOnlyDict*

```
emod_api.interventions.common.triggered_campaign_delay_event(camp, start_day, trigger, delay,  
                                                             intervention, ip_targeting=[],  
                                                             coverage=1.0)
```

Create and return a campaign event that responds to a trigger after a delay with an intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **start_day** – When to start.
- **delay** – Dictionary of 1 or 2 params that are the literal Delay_Distribution parameters,
- **"Delay_Period_Exponential"** (but without the distribution, which is inferred. E.g., {)-5 }.
- **trigger** – E.g., "NewInfection".
- **intervention** – List of 1 or more valid intervention dictionaries to be distributed together.
- **ip_targeting** – Optional Individual Properties required for someone to receive the intervention(s).

Returns Campaign event.

```
emod_api.interventions.common.triggered_campaign_event_with_optional_delay(camp, start_day,  
                                                                           triggers,  
                                                                           intervention,  
                                                                           delay=None,  
                                                                           duration=- 1,  
                                                                           ip_targeting=None,  
                                                                           coverage=1.0, tar-  
                                                                           get_age_min=0,  
                                                                           tar-  
                                                                           get_age_max=45625,  
                                                                           target_sex='All',  
                                                                           tar-  
                                                                           get_residents_only=False,  
                                                                           blackout=True,  
                                                                           check_at_trigger=False)
```

Create and return a campaign event that responds to a trigger after a delay with an intervention.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **start_day** – When to start.
- **triggers** – List of signals to listen for/trigger on. E.g., “NewInfection”.
- **intervention** – List of 1 or more valid intervention dictionaries to be distributed together.
- **delay** – Optional dictionary of 1 or 2 params that are the literal Delay_Distribution parameters,
- **"Delay_Period_Exponential"** (but without the distribution, which is inferred. E.g., { } – 5 }. If omitted,
- **immediate.**(intervention is) –
- **duration** – How long to listen.
- **ip_targeting** – Optional Individual Properties required for someone to receive the intervention(s).
- **coverage** – Fraction of target population to reach.
- **target_age_min** – Minimum age to target.
- **target_age_max** – Maximum age to target.
- **target_sex** – Optional target just “MALE” or “FEMALE” individuals.
- **target_residents_only** – Set to True to target only the individuals who started the simulation in this node and are still in the node.
- **blackout** – Set to True if you don’t want the triggered intervention to be distributed to the same person more than once a day.
- **check_at_trigger** – if triggered event is delayed, you have an option to check individual/node’s eligibility at the initial trigger or when the event is actually distributed after delay.

Returns Campaign event.

```
emod_api.interventions.common.change_individual_property_at_age(camp, new_ip_key,  
                                                                new_ip_value,  
                                                                change_age_in_days,  
                                                                revert_in_days, ip_targeting_key,  
                                                                ip_targeting_value,  
                                                                coverage=1.0)
```

Create and return a campaign event that changes a person’s Individual Properties once they turns a certain age. e.g., change_individual_property_at_age(cb, ‘ForestGoing’, ‘LovesForest’, coverage=0.6, change_age_in_days=15*365, revert=20*365)

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **new_ip_key** – The new IP key.
- **new_ip_value** – The new IP value.
- **change_age_in_days** – The age at which the individual transitions (in units of days).
- **revert_in_days** – How many days they remain with the new property.
- **ip_targeting_key** – The IP key a person must have to receive this.
- **ip_targeting_value** – The IP value a person must have to receive this.
- **coverage** – Optional fraction to limit this to a subset of the target population.

Returns Campaign event.

```
emod_api.interventions.common.change_individual_property_triggered(camp, triggers: list,
                                                                    new_ip_key: str,
                                                                    new_ip_value: str, start_day:
                                                                    int = 0, daily_prob: float =
                                                                    1, max_duration: int =
                                                                    9.3228e+35, revert_in_days:
                                                                    int = - 1, node_ids:
                                                                    Optional[list] = None,
                                                                    ip_restrictions:
                                                                    Optional[list] = None,
                                                                    coverage: float = 1.0,
                                                                    target_age_min: float = 0,
                                                                    target_age_max: float =
                                                                    45625, target_sex: str = 'All',
                                                                    target_residents_only: bool
                                                                    = False, delay=None,
                                                                    listening_duration: int = - 1,
                                                                    blackout: bool = True,
                                                                    check_at_trigger: bool =
                                                                    False)
```

Change Individual Properties when a certain trigger is observed.

Parameters

- **camp** – The instance containing the campaign builder and accumulator.
- **triggers** – A list of the events that will trigger the intervention.
- **new_ip_key** – The individual property key to assign to the individual. For example, InterventionStatus.
- **new_ip_value** – The individual property value to assign to the individual. For example, RecentDrug.
- **start_day** – The day on which to start distributing the intervention (**Start_Day** parameter).
- **node_ids** – The list of nodes to apply this intervention to. If not provided, defaults to all nodes.
- **daily_prob** – The daily probability that an individual's property value will be updated (**Daily_Probability** parameter).
- **max_duration** – The maximum amount of time individuals have to move to a new **daily_prob**; individuals not moved to the new value by the end of **max_duration** keep the same value.
- **revert_in_days** – The number of days before a node reverts to its original property value. Default of 0 means the new value is kept forever.
- **ip_restrictions** – The IndividualProperty key:value pairs to target.
- **coverage** – The proportion of the population that will receive the intervention (**Demographic_Coverage** parameter).
- **target_age_min** – Minimum age to target.
- **target_age_max** – Maximum age to target.

- **target_sex** – Optional target just “MALE” or “FEMALE” individuals.
- **target_residents_only** – Set to True to target only the individuals who started the simulation in this node and are still in the node.
- **delay** – The number of days the campaign is delayed after being triggered.
- **listening_duration** – The number of time steps that the triggered campaign will be active for. Default is -1, which is indefinitely.
- **blackout** (*advanced*) – Set to True if you don’t want the triggered intervention to be distributed to the same person more than once a day.
- **check_at_trigger** (*advanced*) – if triggered event is delayed, you have an option to check individual/node’s eligibility at the initial trigger or when the event is actually distributed after delay.
- **Returns** – N/A.

```
emod_api.interventions.common.change_individual_property_scheduled(camp, new_ip_key,  
                                                                    new_ip_value, start_day: int  
                                                                    = 0, number_repetitions: int  
                                                                    = 1,  
                                                                    timesteps_between_reps: int  
                                                                    = - 1, node_ids:  
                                                                    Optional[list] = None,  
                                                                    daily_prob: float = 1,  
                                                                    max_duration: int =  
                                                                    9.3228e+35, revert_in_days:  
                                                                    int = - 1, ip_restrictions:  
                                                                    Optional[list] = None,  
                                                                    coverage: float = 1.0,  
                                                                    target_age_min: float = 0,  
                                                                    target_age_max: float =  
                                                                    45625, target_sex: str = 'All',  
                                                                    target_residents_only: bool  
                                                                    = False)
```

Change Individual Properties at a given time.

Parameters

- **camp** – The instance containing the campaign builder and accumulator.
- **new_ip_key** – The individual property key to assign to the individual. For example, InterventionStatus.
- **new_ip_value** – The individual property value to assign to the individual. For example, RecentDrug.
- **start_day** – The day on which to start distributing the intervention (**Start_Day** parameter).
- **node_ids** – The list of nodes to apply this intervention to. If not provided, defaults to all nodes.
- **daily_prob** – The daily probability that an individual’s property value will be updated (**Daily_Probability** parameter).

- **max_duration** – The maximum amount of time individuals have to move to a new **daily_prob**; individuals not moved to the new value by the end of **max_duration** keep the same value.
- **revert_in_days** – The number of days before an individual reverts to its original property value. Default of -1 means the new value is kept forever.
- **ip_restrictions** – The IndividualProperty key:value pairs to target.
- **coverage** – The proportion of the population that will receive the intervention (**Demographic_Coverage** parameter).
- **target_age_min** – Minimum age to target.
- **target_age_max** – Maximum age to target.
- **target_sex** – Optional target just “MALE” or “FEMALE” individuals.
- **target_residents_only** – Set to True to target only the individuals who started the simulation in this node and are still in the node.
- **Returns** – N/A.

```
emod_api.interventions.common.change_individual_property(camp, target_property_name: str,
                                                         target_property_value: str, start_day: int
                                                         = 0, number_repetitions: int = 1,
                                                         timesteps_between_reps: int = - 1,
                                                         node_ids: Optional[list] = None,
                                                         daily_prob: float = 1, max_duration: int
                                                         = 9.3228e+35, revert: int = - 1, coverage:
                                                         float = 1, ip_restrictions: Optional[list] =
                                                         None, target_age_min: float = 0,
                                                         target_age_max: float = 45625,
                                                         target_sex: str = 'All',
                                                         target_residents_only: bool = False,
                                                         trigger_condition_list: Optional[list] =
                                                         None, triggered_campaign_delay: int = 0,
                                                         listening_duration: int = - 1,
                                                         blackout_flag: bool = True,
                                                         check_eligibility_at_trigger: bool =
                                                         False)
```

Add an intervention that changes the individual property value to another on a particular day OR after a triggering event using the **PropertyValueChanger** class. Deprecated. Prefer `change_individual_property_scheduled` or `change_individual_property_triggered` depending on the use case.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **target_property_name** – The individual property key to assign to the individual. For example, Risk.
- **target_property_value** – The individual property value to assign to the individual. For example, High.
- **start_day** – The day on which to start distributing the intervention.
- **number_repetitions** – Optional repeater value. Does not work with triggers.

- **timesteps_between_reps** – Gap between repetitions, optional. Does not work with triggers.
- **node_ids** – The list of nodes to apply this intervention to. Defaults to all.
- **daily_prob** – The daily probability that an individual's property value will be updated (**Daily_Probability** parameter).
- **max_duration** – The number of days to continue the intervention after **start_day**.
- **revert** – The number of days before an individual reverts to its original property value. Default of -1 means the new value is kept forever.
- **coverage** – The proportion of the population that will receive the intervention (**Demographic_Coverage** parameter).
- **ip_restrictions** – The IndividualProperty key:value pairs to target. Usually this will be the same key but different from the **target_property_xxx** entries.
- **target_residents_only** – Set to True to target only the individuals who started the simulation in this node and are still in the node.
- **target_age_min** – Optional minimum age, defaults to 0.
- **target_age_max** – Optional maximum age, defaults to inf.
- **target_sex** – Optional target sex, defaults to both.
- **triggered_campaign_delay** – The number of days the campaign is delayed after being triggered.
- **trigger_condition_list** – A list of the events that will trigger the intervention. If included, **start_day** is the day when monitoring for triggers begins.
- **listening_duration** – The number of time steps that the triggered campaign will be active for. Default is -1, which is indefinitely.
- **blackout_flag** – Set to True if you don't want the triggered intervention to be distributed to the same person more than once a day.
- **check_eligibility_at_trigger** – if triggered event is delayed, you have an option to check individual/node's eligibility at the initial trigger or when the event is actually distributed after delay.

Returns None

emod_api.interventions.import_pressure module

`emod_api.interventions.import_pressure.new_intervention(timestep, durs=[], dips=[], nods=[])`

`emod_api.interventions.import_pressure.new_intervention_as_file(timestep, filename=None)`

emod_api.interventions.migration module

`emod_api.interventions.migration.add_migration_event`(*camp*, *nodeto*, *start_day*: *int* = 0, *coverage*: *float* = 1, *repetitions*: *int* = 1, *tsteps_btwn*: *int* = 365, *duration_at_node*: *Optional[dict]* = None, *duration_before_leaving*: *Optional[dict]* = None, *target_age*: *Optional[dict]* = None, *nodes_from_ids*: *Optional[List[int]]* = None, *ind_property_restrictions*=None, *node_property_restrictions*=None, *triggered_campaign_delay*=0, *trigger_condition_list*=None, *listening_duration*=- 1)

Add a migration event to a campaign that moves individuals from one node to another.

Parameters

- **camp** – emod_api.campaign object with schema_path set.
- **nodeto** – The NodeID that the individuals will travel to.
- **start_day** – A day when intervention is distributed
- **coverage** – The proportion of the population covered by the intervention
- **repetitions** – The number of times to repeat the intervention
- **tsteps_btwn** – The number of time steps between repetitions.
- **duration_before_leaving** – Dictionary of parameters that define the distribution for duration before leaving node, including the distribution. Durations are in days. .. rubric:: Examples


```
{“Duration_Before_Leaving_Distribution”:”GAUSSIAN_DISTRIBUTION”,
“Duration_Before_Leaving_Gaussian_Mean”:          14,          “Duration_Before_Leaving_Gaussian_Std_Dev”          3}
{“Duration_Before_Leaving_Distribution”:”POISSON_DISTRIBUTION”,
“Duration_Before_Leaving_Poisson_Mean” 30}
```
- **duration_at_node** – Dictionary of parameters that define the distribution for duration at node, including the distribution Durations are in days. .. rubric:: Examples


```
{“Duration_At_Node_Distribution”:”GAUSSIAN_DISTRIBUTION”,          “Duration_At_Node_Gaussian_Mean”:          14,          “Duration_At_Node_Gaussian_Std_Dev”          3}
{“Duration_At_Node_Distribution”:”POISSON_DISTRIBUTION”,          “Duration_At_Node_Poisson_Mean” 30}
```
- **target_age** – The individuals to target with the intervention. To restrict by age, provide a dictionary of {‘agemin’: x, ‘agemax’: y}. Default is targeting everyone.
- **nodes_from_ids** – The list of node ids to apply this intervention to.
- **ind_property_restrictions** – The IndividualProperty key:value pairs that individuals must have to receive the intervention (**Property_Restrictions_Within_Node** parameter). In the format [{"BitingRisk": "High"}, {"IsCool": "Yes"}].
- **node_property_restrictions** – The NodeProperty key:value pairs that nodes must have to receive the intervention. In the format [{"Place": "RURAL"}, {"ByALake": "Yes"}].
- **triggered_campaign_delay** – After the trigger is received, the number of time steps until distribution starts. Eligibility of people or nodes for the campaign is evaluated on the start day, not the triggered day.

- **trigger_condition_list** – A list of the events that will trigger the intervention. If included, **start_days** is then used to distribute **NodeLevelHealthTriggeredIV**.
- **listening_duration** – The number of time steps that the distributed event will monitor for triggers. Default is -1, which is indefinitely.

Returns None

Example

```
from emod_api import campaign as camp
dan = {"Duration_At_Node_Distribution": "POISSON_DISTRIBUTION",
      "Duration_At_Node_Poisson_Mean": 30}
dbl = {"Duration_Before_Leaving_Distribution": "GAUSSIAN_DISTRIBUTION",
      "Duration_Before_Leaving_Gaussian_Mean": 14,
      "Duration_Before_Leaving_Gaussian_Std_Dev": 3}

add_migration_event(camp, node_to=5, start_day=1, coverage=0.75, duration_at_node = dan,
                   duration_before_leaving = dbl, repetitions=1, tsteps_btwn=90, target='Everyone',
                   nodesfrom={"class": "NodeSetAll"}, node_property_restrictions=[{"Place": "Rural"}])
```

emod_api.interventions.node_multiplier module

```
emod_api.interventions.node_multiplier.new_intervention(camp, new_infectivity=1.0,
                                                         profile='CONST', **kwargs)
```

Create new NodeInfectivityModifying intervention.

Parameters

- **profile** – multiplier options include:
 - **CONST(ANT)**
 - * **new_infectivity** lasts forever (or until replaced).
 - **TRAP(EZOID)**
 - * **rise_dur(ation)**
 - * **peak_dur(ation)**
 - * **fall_dur(ation)**
 - **EXP(ONENTIAL) (not implemented yet)**
 - * **rise duration**
 - * **rise rate**
 - **SIN(USOIDAL) (not implemented yet)**
 - * **period**
- **durations** (*To do boxcar, specify 0 rise and fall*) –

Returns new NodeInfectivityMult intervention dictionary.

```
emod_api.interventions.node_multiplier.new_scheduled_event(camp, start_day=1,
                                                            new_infectivity=1.0, profile='CONST',
                                                            node_ids=None, recurring=True,
                                                            **kwargs)
```

Create new NodeInfectivityModifying intervention as scheduled campaign event.

`emod_api.interventions.node_multiplier.new_intervention_as_file(camp, timestep, filename=None)`
 Create new NodeInfectivityModifying intervention as sole scheduled campaign event inside working campaign json file.

emod_api.interventions.outbreak module

`emod_api.interventions.outbreak.seed(camp, Start_Day: int, Coverage: float, Target_Props=None, Node_Ids=None, Tot_Rep: int = 1, Rep_Interval: int = - 1, Target_Age_Min: float = 0, Target_Age_Max: float = 125, Target_Gender: str = 'All', Honor_Immunity: bool = False)`

Distribute an outbreak (via prevalence increase of existing agents) to individuals based on inclusion criteria.

Parameters

- **camp** – Central campaign builder object.
- **Start_Day** – Simulation timestep when outbreak should occur. Required.
- **Coverage** – Fraction of population to reach. No default.
- **Target_Props** – Individual Properties to limit the seeding to.
- **Node_Ids** – Nodes to target. Optional. Defaults to all.
- **Tot_Rep** – Number of times to “re-seed”. Optional. Defaults to just once.
- **Rep_Interval** – Number of timesteps between re-seeding events. Optional. Use with Rep_Num.
- **Target_Age_Min** – Minimum age in years. Optional. Defaults to 0.
- **Target_Age_Max** – Maximum age in years. Optional. Defaults to AGE_MAX.
- **Target_Gender** – Optional sex-targeting param (Male or Female if you don’t want “All”).
- **Honor_Immunity** – Set to True if you want to infect people regardless of their immunity.

`emod_api.interventions.outbreak.seed_by_coverage(campaign_builder, timestep, coverage=0.01, node_ids=[], properties=None, ignore_immunity=None, intervention_only=False)`

This simple function provides a very common piece of functionality to seed an infection. A future version will support targeted nodesets.

`emod_api.interventions.outbreak.new_intervention(campaign_builder, timestep, cases=1)`

Create EMOD-ready Outbreak intervention.

Parameters

- **timestep** (*float*) – timestep at which outbreak should occur.
- **cases** (*integer*) – new paramamter that specifies maximum number of cases. May not be supported.

Returns event as dict (json)

Return type event (json)

`emod_api.interventions.outbreak.new_intervention_as_file(camp, timestep, cases=1, filename=None)`

emod_api.interventions.simple_vaccine module

```
emod_api.interventions.simple_vaccine.new_intervention(timestep, v_type='Generic', efficacy=1.0,  
                                                         sv_name='Vaccine', waning_duration=100,  
                                                         d_a_d=None, cost_to_consumer=None,  
                                                         e_i_r=None, intervention_only=False)
```

This is mostly an example but also potentially useful. With this you get a Vaccine with working defaults but 2 configurables: type and efficacy. The duration is fixed at box. You of course must specify the timestep and you can add a vaccine name which is mostly useful if you're managing a duplicate policy.

```
emod_api.interventions.simple_vaccine.new_intervention2(timestep)
```

This version lets you invoke the function sans-parameters. You get the module-level params which you can set before calling this. This is designed to support a more data-oriented way of using this API, with everything like "a.b=c", and avoid "churn" on the API itself (constantly changing function signature). TBD: Make sure that if this is called twice, we understand whether we have copies or references going on.

```
emod_api.interventions.simple_vaccine.new_intervention_as_file(timestep, filename=None)
```

emod_api.interventions.utils module

```
emod_api.interventions.utils.do_nodes(schema_path, node_ids)
```

Create and return a NodeSetConfig based on node_ids list.

```
emod_api.interventions.utils.get_waning_from_params(schema_path, initial=1.0, box_duration=365,  
                                                    decay_rate=0, decay_time_constant=None)
```

Get well configured waning structure. Default is 1-year full efficacy box. Note that an infinite decay rate (0 or even -1) is same as Box. Note that an infinite box duration (-1) is same as constant. Note that a zero box duration is same as Exponential.

Parameters

- **schema_path** – Path to schema.json file.
- **initial** – Initial efficacy value, defaults to 1.0.
- **box_duration** – Number of timesteps efficacy remains at initial before decay. Defaults to 365.
- **decay_rate** – Rate at which efficacy decays after box_duration. Defaults to 0.
- **decay_time_constant** – 1/decay_rate. Defaults to None. Use this or decay_rate, not both. If this is specified, decay_rate is ignored.

emod_api.migration package

Subpackages

emod_api.migration.client package

Submodules

emod_api.migration.client.client module

`emod_api.migration.client.client.run(input_file: pathlib.Path, parameters: dict) → None`

Run a client that tries to connect the url given in parameters. The client will do a Post operation with the parameters given in parameters.

Parameters

- **input_file** – Path to the demographics file.
- **parameters** – Dictionary containing the server url and the parameters for model calculation.

Submodules

emod_api.migration.migration module

class `emod_api.migration.migration.Layer`

Bases: `dict`

The Layer object represents a mapping from source node (IDs) to destination node (IDs) for a particular age, gender, age+gender combination, or all users if no age or gender dependence. Users will not generally interact directly with Layer objects.

property `DatavalueCount: int`

Get (maximum) number of data values for any node in this layer

Returns Maximum number of data values for any node in this layer

property `NodeCount: int`

Get the number of (source) nodes with rates in this layer

Returns Number of (source) nodes with rates in this layer

class `emod_api.migration.migration.Migration`

Bases: `object`

Represents migration data in a mapping from source node (IDs) to destination node (IDs) with rates for each pairing.

Migration data may be age dependent, gender dependent, both, or the same for all ages and genders. A migration file (along with JSON metadata) can be loaded from the static method `Migration.from_file()` and inspected and/or modified. Migration objects can be started from scratch with `Migration()`, and populated with appropriate source-dest rate data and saved to a file with the `to_file()` method. Given `migration = Migration()`, syntax is as follows:

age and gender agnostic: `migration[source_id][dest_id]` age dependent: `migration[source_id:age]` # age should be ≥ 0 , ages $>$ last bucket value use last bucket value gender dependent: `migration[source_id:gender]` # gender one of `Migration.MALE` or `Migration.FEMALE` age and gender dependent: `migration[source_id:gender:age]` # gender one of `Migration.MALE` or `Migration.FEMALE`

EMOD/DTK format migration files (and associated metadata files) can be written with `migration.to_file(<filename>)`. EMOD/DTK format migration files (with associated metadata files) can be read with `migration.from_file(<filename>)`.

SAME_FOR_BOTH_GENDERS = 0

ONE_FOR_EACH_GENDER = 1

LINEAR_INTERPOLATION = 0

PIECEWISE_CONSTANT = 1

```
LOCAL = 1
AIR = 2
REGIONAL = 3
SEA = 4
FAMILY = 5
INTERVENTION = 6
IDREF_LEGACY = 'Legacy'
IDREF_GRUMP30ARCSEC = 'Gridded world grump30arcsec'
IDREF_GRUMP2PT5ARCMIN = 'Gridded world grump2.5arcmin'
IDREF_GRUMP1DEGREE = 'Gridded world grump1degree'
MALE = 0
FEMALE = 1
MAX_AGE = 125

property AgesYears: list
    List of ages - ages < first value use first bucket, ages > last value use last bucket.

property Author: str
    str: Author value for metadata for this migration datafile

property DatavalueCount: int
    int: Maximum data value count for any layer in this migration datafile

property DateCreated: datetime.datetime
    datetime: date/time stamp of this datafile

property GenderDataType: int
    int: gender data type for this datafile - SAME_FOR_BOTH_GENDERS or ONE_FOR_EACH_GENDER

property IdReference: str
    str: ID reference metadata value

property InterpolationType: int
    int: interpolation type for this migration data file - LINEAR_INTERPOLATION or PIECE-
    WISE_CONSTANT

property MigrationType: int
    int: migration type for this migration data file - LOCAL | AIR | REGIONAL | SEA | FAMILY | INTER-
    VENTION

property Nodes: list

property NodeCount: int
    int: maximum number of source nodes in any layer of this migration data file

get_node_offsets(limit: int = 100) → dict

property NodeOffsets: dict
    dict: mapping from source node id to offset to destination and rate data in binary data

property Tool: str
    str: tool metadata value

to_file(binaryfile: pathlib.Path, metafile: Optional[pathlib.Path] = None, value_limit: int = 100)
    Write current data to given file (and .json metadata file)
```


Parameters

- **binaryfile** (*Path*) – path to output file (metadata will be written to same path with “.json” appended)
- **metafile** (*Path*) – override standard metadata file naming
- **value_limit** (*int*) – limit on number of destination values to write for each source node (default = 100)

Returns path to binary file

Return type (*Path*)

`emod_api.migration.migration.from_file(binaryfile: pathlib.Path, metafile: Optional[pathlib.Path] = None)`

Reads migration data file from given binary (and associated JSON metadata file)

Parameters

- **binaryfile** (*Path*) – path to binary file (metadata file is assumed to be at same location with “.json” suffix)
- **metafile** (*Path*) – use given metafile rather than inferring metafile name from the binary file name

Returns Migration object representing binary data in the given file.

`emod_api.migration.migration.examine_file(filename)`

`emod_api.migration.migration.from_params(demographics_file_path=None, pop=1000000.0, num_nodes=100, mig_factor=1.0, frac_rural=0.3, id_ref='from_params', migration_type=1)`

This function is for creating a migration file that goes with a (multinode) demographics file created from a few parameters, as opposed to one from real-world data. Note that the ‘demographics_file_path’ input param is not used at this time but in future will be exploited to ensure nodes, etc., match.

`emod_api.migration.migration.from_demog_and_param_gravity_webservice(demographics_file_path: str, params: str, id_ref: str, migration_type=1) → emod_api.migration.migration.Migration`

Calls a webservice (running on a GPU) to calculate the migration patterns quickly.

Parameters

- **demographics_file_path** – Path to the demographics file.
- **params** – Path to the json file with parameters for gravity calculation and server url.
- **id_ref** – Metadata tag that needs to match corresponding value in demographics file.
- **migration_type** – Migration type.

Returns Migration object

`emod_api.migration.migration.from_demog_and_param_gravity(demographics_file_path, gravity_params, id_ref, migration_type=1)`

Create migration files from a gravity model and an input demographics file.

`emod_api.migration.migration.to_csv(filename: pathlib.Path)`

`emod_api.migration.migration.from_csv(filename: pathlib.Path)`

Create migration from csv file. The file should have columns ‘source’ for the source node, ‘destination’ for the destination node, and ‘rate’ for the migration rate.

Parameters `filename` – csv file

Returns Migration object

emod_api.schema package

Submodules

emod_api.schema.dtk_post_process_schema module

`emod_api.schema.dtk_post_process_schema.recurser(in_json)`

`emod_api.schema.dtk_post_process_schema.application(schema_file)`

emod_api.schema.get_schema module

`emod_api.schema.get_schema.dtk_to_schema(path_to_binary, path_to_write_schema='schema.json')`

Runs /path/to/Eradication -get-schema -schema-path=schema.json and then post-processes the schema into something more useful. Error cases handled: - schema.json file already exists in cwd; does not overwrite. Asks users to move and retry. - Specified binary fails to run to completion. - Specified binary fails to produce a schema.json

emod_api.serialization package

Submodules

emod_api.serialization.CensusAndModPop module

`emod_api.serialization.CensusAndModPop.change_ser_pop(input_serpop_path, mod_fn=None, save_file_path=None)`

This function loads a serialization population file, iterates over each person, calls a user-provided callback with each individuals, and saves the population as manipulated by the user.

The mod function can act at will on the population object. There are no checks.

The new file is saved to a name provided by user. Interactive if none provided to function.

Assuming a single node file for now.

emod_api.serialization.SerializedPopulation module

Class to load and manipulate a saved population.

class `emod_api.serialization.SerializedPopulation.SerializedPopulation(file: str)`

Bases: `object`

Opens the passed file and reads in all the nodes.

Parameters `file` – serialized population file

Examples

Create an instance of SerializedPopulation:

```
import emod_api.serialization.SerializedPopulation as SerPop
ser_pop = SerPop.SerializedPopulation('state-00001.dtk')
```

property nodes

All nodes.

Examples

Delete number_of_ind individuals from node 0:

```
node = ser_pop.nodes[0]
del node.individualHumans[0:number_of_ind]
```

Only keep individuals with a certain condition:

```
node.individualHumans = [ind for ind in node.individualHumans if keep_fct(ind)]
```

Change susceptibility of an individual:

```
print(node.individualHumans[0].susceptibility)
new_susceptibility = {"age": 101.01, "mod_acquire": 0}
node.individualHumans[0].susceptibility.update(new_susceptibility)
```

Copy individual[0] from node 0, change properties and add individual as new individual:

```
import copy
individual_properties={"m_age": 1234}
individual = copy.deepcopy(node.individualHumans[0])
individual["suid"] = ser_pop.get_next_individual_suid(0)
individual.update(individual_properties)
ser_pop.nodes[0].individualHumans.append(individual)
```

Infect an individual with an infection copied from another individual:

```
infection = node["individualHumans"][0]["infections"][0]
infection["suid"] = self.get_next_infection_suid()
node["individualHumans"][1]["infections"].append(infection)
node["individualHumans"][1].m_is_infected = True
```

flush()

Save all made changes to the node(s).

write(output_file: str = 'my_sp_file.dtk')

Write the population to a file.

Parameters output_file – output file

get_next_infection_suid()

Each infection needs a unique identifier, this function returns one.

get_next_individual_suid(node_id: int) → dict

Each individual needs a unique identifier, this function returns one.

Parameters `node_id` – The first parameter.

Returns The return value. True for success, False otherwise.

Examples

To get a unique id for an individual:

```
print(sp.get_next_individual_suid(0))
{'id': 2}
```

`emod_api.serialization.SerializedPopulation.find(name: str, handle, currentlevel='dtk.nodes')`

Recursively searches for a paramters that matches or is close to name and prints out where to find it in the file.

Parameters

- **name** – the paramter you are looking for e.g. “age”, “gender”.
- **handle** – some iterable data structure, can be a list of nodes, a node, list of individuals, etc
currentlevel: just a string to print out where the found item is located e.g. “dtk.nodes” or “dtk.node.individuals”

Examples

What is the exact paramteter name used for the age of an individual?:

```
SerPop.find("age", node)
...
1998 Found in: dtk.nodes.individualHumans[999].m_age
1999 Found in: dtk.nodes.individualHumans[999].susceptibility.age
2000 Found in: dtk.nodes.m_vectorpopulations[0].EggQueues[0].age
2001 Found in: dtk.nodes.m_vectorpopulations[0].EggQueues[1].age
...
```

`emod_api.serialization.SerializedPopulation.get_parameters(handle, currentlevel='dtk.nodes')`

Return a set of all parameters in the serialized population file. Helpful to get an overview about what is in the serialized population file.

Parameters

- **handle** – some iterable data structure, can be a list of nodes, a node, list of individuals, etc
- **currentlevel** – just a string to print out where the found item is located e.g. “dtk.nodes”or “dtk.node.individuals

Examples

Print all parameters in serialized population file:

```
for n in sorted(SerPop.get_parameters(node)):
    print(n)
```

emod_api.serialization.dtkFileSupport module

```

class emod_api.serialization.dtkFileSupport.Uncompressed
    Bases: object
        classmethod compress(data)
        classmethod uncompress(data)
class emod_api.serialization.dtkFileSupport.EllZeeFour
    Bases: object
        classmethod compress(data)
        classmethod uncompress(data)
class emod_api.serialization.dtkFileSupport.Snappy
    Bases: object
        classmethod compress(data)
        classmethod uncompress(data)
class emod_api.serialization.dtkFileSupport.SerialObject(dictionary={})
    Bases: dict
class emod_api.serialization.dtkFileSupport.NullPtr
    Bases: emod_api.serialization.dtkFileSupport.SerialObject

```

emod_api.serialization.dtkFileTools module

Support for three formats of serialized population files: 1. “Original version”: single payload chunk with simulation and all nodes, uncompressed or snappy or LZ4 2. “First chunked version”: multiple payload chunks, one for simulation and one each for nodes 3. “Second chunked version”: multiple payload chunks, simulation and node objects are “root” objects in each chunk 4. “Metadata update”: compressed: true|false + engine: NONE|LZ4|SNAPPY replaced with compression: NONE|LZ4|SNAPPY

```

emod_api.serialization.dtkFileTools.uncompress(data, engine)
emod_api.serialization.dtkFileTools.compress(data, engine)
class emod_api.serialization.dtkFileTools.DtkHeader(dictionary={'author': 'unknown', 'bytecount': 0,
                                                                'chunkcount': 0, 'chunksizes': [], 'compressed':
                                                                True, 'date': 'Fri May 20 00:34:57 2022',
                                                                'engine': 'LZ4', 'tool': 'dtkFileTools.py', 'version':
                                                                1})
    Bases: emod_api.serialization.dtkFileSupport.SerialObject
class emod_api.serialization.dtkFileTools.DtkFile(header)
    Bases: object
        class Contents(parent)
            Bases: object
                append(item)
        class Objects(parent)
            Bases: object
                append(item)
        property header

```

property compressed
property compression
property byte_count
property chunk_count
property chunk_sizes
property author
property date
property tool
property version
property chunks
property nodes

```
class emod_api.serialization.dtkFileTools.DtkFileV1(header=None, filename="", handle=None)
    Bases: emod\_api.serialization.dtkFileTools.DtkFile
```

property simulation

```
class emod_api.serialization.dtkFileTools.DtkFileV2(header=None, filename="", handle=None)
    Bases: emod\_api.serialization.dtkFileTools.DtkFile
```

```
class NodesV2(parent)
    Bases: object
```

property simulation

```
class emod_api.serialization.dtkFileTools.DtkFileV3(header=None, filename="", handle=None)
    Bases: emod\_api.serialization.dtkFileTools.DtkFile
```

```
class NodesV3(parent)
    Bases: object
```

property simulation

```
class emod_api.serialization.dtkFileTools.DtkFileV4(header=None, filename="", handle=None)
    Bases: emod\_api.serialization.dtkFileTools.DtkFileV3
```

```
emod_api.serialization.dtkFileTools.read(filename)
```

```
emod_api.serialization.dtkFileTools.write(dtk_file, filename)
```

emod_api.serialization.dtkFileUtility module

emod_api.spatialreports package

Submodules

emod_api.spatialreports.spatial module

emod-api spatial report module. Exposes SpatialReport and SpatialNode objects.

```
class emod_api.spatialreports.spatial.SpatialNode(node_id: int, data)
    Bases: object
```

Class representing a single node of a spatial report.

property id: `int`

Node ID

property data

Time series data for this node.

```
class emod_api.spatialreports.spatial.SpatialReport(filename: Optional[str] = None, node_ids:
Optional[List[int]] = None, data:
Optional[numpy.array] = None, start: int = 0,
interval: int = 1)
```

Bases: `object`

Class for reading (and, optionally, writing) spatial reports in EMOD/DTK format. “Filtered” reports will have start > 0 and/or reporting interval > 1.

property data: `numpy.array`

Returns full 2 dimensional NumPy array with report data. Shape is (#values, #nodes).

property node_ids: `List[int]`

Returns list of node IDs (integers) for nodes in the report.

property nodes: `Dict[int, emod_api.spatialreports.spatial.SpatialNode]`

Returns dictionary of SpatialNodes keyed on node ID.

property node_count: `int`

Number of nodes in the report.

property time_steps: `int`

Number of samples in the report.

property start: `int`

Time step of first sample.

property interval: `int`

Interval, in time steps, between samples.

write_file(filename: *str*)

Save current nodes and timeseries data to given file.

emod_api.tabularoutput package

emod_api.weather package

Submodules

emod_api.weather.weather module

emod-api Weather module - Weather, Metadata, and WeatherNode objects along with IDREF and CLIMATE_UPDATE constants.

```
class emod_api.weather.weather.WeatherNode(node_id: int, data)
```

Bases: `object`

Represents information for a single node: ID and timeseries data.

property id: `int`

Node ID

property data

Time series data for this node.

```
class emod_api.weather.weather.Metadata(node_ids: List[int], datavalue_count: int, author: Optional[str]
                                         = None, created: Optional[datetime.datetime] = None,
                                         frequency: Optional[str] = None, provenance: Optional[str] =
                                         None, reference: Optional[str] = None)
```

Bases: `object`

Metadata:

- [DateCreated]
- [Author]
- [OriginalDataYears]
- [StartDayOfYear]
- [DataProvenance]
- IdReference
- NodeCount
- DatavalueCount
- UpdateResolution
- NodeOffsets

property author: `str`

Author of this file.

property creation_date: `datetime.datetime`

Creation date of this file.

property datavalue_count: `int`

Number of data values in each timeseries, should be > 0.

property id_reference: `str`

‘Schema’ for node IDs. Commonly *Legacy*, *Gridded world grump2.5arcmin*, and *Gridded world grump30arcsec*.

Legacy usually indicates a 0 or 1 based scheme with increasing ID numbers.

Gridded world grump2.5arcmin and *Gridded world grump30arcsec* encode latitude and longitude values in the node ID with the following formula:

```
latitude = (((nodeid - 1) & 0xFFFF) * resolution) - 90
longitude = ((nodeid >> 16) * resolution) - 180
# nodeid = 90967271 @ 2.5 arcmin resolution
# longitude = -122.1667, latitude = 47.5833
```

property node_count: `int`

property node_ids: `List[int]`

property provenance: `str`

property update_resolution: `str`

property nodes: `Dict[int, int]`

WeatherNodes offsets keyed by node id.

write_file(filename: str) → None

```

classmethod from_file(filename: str)
    Read weather metadata file. Metadata' and 'NodeOffsets' keys required. DatavalueCount', 'UpdateResolution', and 'IdReference' required in 'Metadata'.

class emod_api.weather.weather.Weather(filename: Optional[str] = None, node_ids: Optional[List[int]] =
    None, datavalue_count: Optional[int] = None, author:
    Optional[str] = None, created: Optional[datetime.datetime] =
    None, frequency: Optional[str] = None, provenance:
    Optional[str] = None, reference: Optional[str] = None, data:
    Optional[numpy.array] = None)

Bases: object

property data: numpy.array
    Raw data as numpy array[node index, time step].

property metadata: emod_api.weather.weather.Metadata

property author: str

property creation_date: datetime.datetime

property datavalue_count: int
    >= 1

property id_reference: str

property node_count: int
    >= 1

property node_ids: List[int]

property provenance: str

property update_resolution: str

property nodes: Dict[int, emod_api.weather.weather.WeatherNode]
    WeatherNodes indexed by node id.

write_file(filename: str) → None
    Writes data to filename and metadata to filename.json.

classmethod from_csv(filename: str, var_column: str = 'airtemp', id_column: str = 'node_id',
    step_column: str = 'step', author: Optional[str] = None, provenance: Optional[str]
    = None)
    Create weather from CSV file with specified variable column, node id column, and time step column.

```

Note:

- Column order in the CSV file is not significant, but columns names must match what is passed to this function.
 - Because a CSV might hold air temperature (may be negative and well outside 0-1 values), relative humidity (must `_not_` be negative, must be in the interval [0-1]), or rainfall (must `_not_` be negative, likely > 1), this function does not validate incoming data.
-

3.1.2 Submodules

emod_api.campaign module

You use this simple campaign builder by importing it, adding valid events via “add”, and writing it out with “save”.

`emod_api.campaign.reset()`

`emod_api.campaign.set_schema(schema_path_in)`

Set the (path to) the schema file. And reset all campaign variables. This is essentially a “start_building_campaign” function. :param schema_path_in. The path to a schema.json.:

Returns N/A.

`emod_api.campaign.add(event, name=None, first=False)`

Add a complete campaign event to the campaign builder. The new event is assumed to be a Python dict, and a valid event. The new event is not validated here. Set the first flag to True if this is the first event in a campaign because it functions as an accumulator and in some situations like sweeps it might have been used recently.

`emod_api.campaign.get_trigger_list()`

`emod_api.campaign.save(filename='campaign.json')`

Save ‘camapign_dict’ as ‘filename’.

`emod_api.campaign.get_adhocs()`

`emod_api.campaign.get_schema()`

`emod_api.campaign.get_recv_trigger(trigger, old=False)`

Get the correct representation of a trigger (also called signal or even event) that is being listened to.

`emod_api.campaign.get_send_trigger(trigger, old=False)`

Get the correct representation of a trigger (also called signal or even event) that is being broadcast.

`emod_api.campaign.get_event(event, old=False)`

Basic placeholder functionality for now. This will map new ad-hoc events to GP_EVENTS and manage that ‘cache’ If event in built-ins, return event, else if in adhoc map, return mapped event, else add to adhoc_map and return mapped event.

emod_api.peek_camp module

`emod_api.peek_camp.decorate_actual_iv(iv, signal=None)`

`emod_api.peek_camp.decorate_actual_iv_impl(iv, signal=None)`

This function converts json interventions to their CCDL versions. This relies on a lot of special-casing.

`emod_api.peek_camp.handle_di(iv)`

`emod_api.peek_camp.get_ip(coord)`

`emod_api.peek_camp.get_ages(coord)`

`emod_api.peek_camp.decode(camp_path, config_path=None)`

`emod_api.peek_camp.params_to_dict(start_day, reps=None, gap=None, nodes=None, frac=None, sex=None, minage=None, maxage=None, ips=None, signal=None, iv_name=None, payload=None, delay=None)`

Take all the CCDL params (When? Where? Who? What? Why?) and create a dictionary from them.

`emod_api.peek_camp.encode(encoded_path)`

The encode function takes a CCDL files as input and returns a list of campaign events as dictionaries that can be used to create a campaign json from it using emod-api/emodpy functions. This is early code, use at your own risk, or contribute to its improvement. :)

emod_api.schema_to_class module

class `emod_api.schema_to_class.ReadOnlyDict`

Bases: `collections.OrderedDict`

set_schema(*schema*)

Add schema node.

to_file(*config_name='config.json'*)

Write 'clean' config file out to disk as json. Param: *config_name* (defaults to 'config.json')

finalize()

Remove all params that are disabled by depends-on param being off and schema node.

`emod_api.schema_to_class.get_default_for_complex_type(schema, idmtype)`

This function used to be more involved and dumb but now it's a passthrough to `get_class_with_defaults`. If this approach proves robust, it can probably be deprecated. Depends a bit on completeness of schema.

`emod_api.schema_to_class.get_class_with_defaults(classname, schema_path=None)`

Returns the default config for a datatype in the schema.

GLOSSARY

The following terms describe both the features and functionality of the emod-api software, as well as information relevant to using emod-api.

asset collection The set of specific input files (such as input parameters, weather or migration data, or other configuration settings) required for running a simulation.

assets See asset collection.

builder TBD

experiment A collection of multiple simulations, typically sent to an HPC.

high-performance computing (HPC) The use of parallel processing for running advanced applications efficiently, reliably, and quickly.

task TBD

template TBD

PYTHON MODULE INDEX

e

`emod_api`, 7
`emod_api.campaign`, 54
`emod_api.channelreports`, 7
`emod_api.channelreports.channels`, 7
`emod_api.config`, 8
`emod_api.config.default_from_schema`, 8
`emod_api.config.default_from_schema_no_validation`, 9
`emod_api.config.dtk_post_process_adhocevents`, 10
`emod_api.config.dtk_pre_process_adhocevents`, 10
`emod_api.config.dtk_pre_process_w5ml`, 10
`emod_api.config.from_overrides`, 10
`emod_api.config.from_poi_and_binary`, 10
`emod_api.config.from_schema`, 11
`emod_api.config.schema_to_config`, 11
`emod_api.demographics`, 11
`emod_api.demographics.BaseInputFile`, 11
`emod_api.demographics.Demographics`, 12
`emod_api.demographics.demographics_utils`, 25
`emod_api.demographics.DemographicsGenerator`, 16
`emod_api.demographics.DemographicsInputDataParsers`, 19
`emod_api.demographics.DemographicsTemplates`, 19
`emod_api.demographics.grid_construction`, 26
`emod_api.demographics.Node`, 21
`emod_api.demographics.PreDefinedDistributions`, 22
`emod_api.demographics.PropertiesAndAttributes`, 22
`emod_api.demographics.Updateable`, 24
`emod_api.interventions`, 27
`emod_api.interventions.ccdl`, 27
`emod_api.interventions.ccdl_viz`, 27
`emod_api.interventions.common`, 27
`emod_api.interventions.import_pressure`, 38
`emod_api.interventions.migration`, 39
`emod_api.interventions.node_multiplier`, 40
`emod_api.interventions.outbreak`, 41
`emod_api.interventions.simple_vaccine`, 42
`emod_api.interventions.utils`, 42
`emod_api.migration`, 42
`emod_api.migration.client`, 42
`emod_api.migration.client.client`, 43
`emod_api.migration.migration`, 43
`emod_api.peak_camp`, 54
`emod_api.schema`, 46
`emod_api.schema.dtk_post_process_schema`, 46
`emod_api.schema.get_schema`, 46
`emod_api.schema_to_class`, 55
`emod_api.serialization`, 46
`emod_api.serialization.CensusAndModPop`, 46
`emod_api.serialization.dtkFileSupport`, 49
`emod_api.serialization.dtkFileTools`, 49
`emod_api.serialization.dtkFileUtility`, 50
`emod_api.serialization.SerializedPopulation`, 46
`emod_api.spatialreports`, 50
`emod_api.spatialreports.spatial`, 50
`emod_api.tabularoutput`, 51
`emod_api.weather`, 51
`emod_api.weather.weather`, 51

INDEX

A

`add()` (*emod_api.demographics.PropertiesAndAttributes.IndividualProperties* method), 22
`add()` (in module *emod_api.campaign*), 54
`add_migration_event()` (in module *emod_api.interventions.migration*), 39
`add_parameter()` (*emod_api.demographics.PropertiesAndAttributes.IndividualProperties* method), 22
`add_parameter()` (*emod_api.demographics.Updateable.Updateable* method), 24
`AddAgeDependentTransmission()` (*emod_api.demographics.Demographics.Demographics* method), 13
`AddIndividualPropertyAndHINT()` (*emod_api.demographics.Demographics.Demographics* method), 13
`AgeStructureUNWPP()` (in module *emod_api.demographics.DemographicsTemplates*), 21
`AgesYears` (*emod_api.migration.migration.Migration* property), 44
`AIR` (*emod_api.migration.migration.Migration* attribute), 44
`append()` (*emod_api.serialization.dtkFileTools.DtkFile.Comments* method), 49
`append()` (*emod_api.serialization.dtkFileTools.DtkFile.Objects* method), 49
`application()` (in module *emod_api.config.dtk_post_process_adhoc_events*), 10
`application()` (in module *emod_api.config.dtk_pre_process_adhoc_events*), 10
`application()` (in module *emod_api.config.dtk_pre_process_w5ml*), 10
`application()` (in module *emod_api.schema.dtk_post_process_schema*), 46
`apply_overlay()` (*emod_api.demographics.Demographics.Demographics* method), 12
`apply_to_defaults_or_nodes()` (in module *emod_api.demographics.demographics_utils*), 25
`arcsec_to_deg()` (in module *emod_api.demographics.DemographicsGenerator*), 16
`as_dataframe()` (*emod_api.channelreports.channels.ChannelReport* method), 8
`as_dictionary()` (*emod_api.channelreports.channels.Channel* method), 8
`as_dictionary()` (*emod_api.channelreports.channels.Header* method), 7
`asset collection`, 57
`assets`, 57
`Author` (*emod_api.migration.migration.Migration* property), 44
`author` (*emod_api.serialization.dtkFileTools.DtkFile* property), 50
`author` (*emod_api.weather.weather.Metadata* property), 52
`author` (*emod_api.weather.weather.Weather* property), 53

B

`BaseInputFile` (class in *emod_api.demographics.BaseInputFile*), 11
`basicNode()` (in module *emod_api.demographics.Node*), 22
`birth_rate` (*emod_api.demographics.Node.Node* property), 21
`BroadcastEvent()` (in module *emod_api.interventions.common*), 27
`BroadcastEventToOtherNodes()` (in module *emod_api.interventions.common*), 27
`builder`, 57
`byte_count` (*emod_api.serialization.dtkFileTools.DtkFile* property), 50

C

`change_individual_property()` (in module *emod_api.interventions.common*), 37

`change_individual_property_at_age()` (in module `emod_api.interventions.common`), 34
`change_individual_property_scheduled()` (in module `emod_api.interventions.common`), 36
`change_individual_property_triggered()` (in module `emod_api.interventions.common`), 35
`change_ser_pop()` (in module `emod_api.serialization.CensusAndModPop`), 46
Channel (class in `emod_api.channelreports.channels`), 7
`channel_names` (`emod_api.channelreports.channels.ChannelReport` property), 8
ChannelReport (class in `emod_api.channelreports.channels`), 8
`channels` (`emod_api.channelreports.channels.ChannelReport` property), 8
`chunk_count` (`emod_api.serialization.dtkFileTools.DtkFile` property), 50
`chunk_sizes` (`emod_api.serialization.dtkFileTools.DtkFile` property), 50
`chunks` (`emod_api.serialization.dtkFileTools.DtkFile` property), 50
`compress()` (`emod_api.serialization.dtkFileSupport.EllZeeDataTool` class method), 49
`compress()` (`emod_api.serialization.dtkFileSupport.SnappyDataTool` class method), 49
`compress()` (`emod_api.serialization.dtkFileSupport.UncompressedDataTool` class method), 49
`compress()` (in module `emod_api.serialization.dtkFileTools`), 49
`compressed` (`emod_api.serialization.dtkFileTools.DtkFile` property), 49
`compression` (`emod_api.serialization.dtkFileTools.DtkFile` property), 50
ConstantDistribution (class in `emod_api.demographics.PreDefinedDistributions`), 22
`construct()` (in module `emod_api.demographics.grid_construction`), 26
`ConstructNodesFromDataFrame()` (in module `emod_api.demographics.DemographicsInputDataParsers`), 19
`copy()` (`emod_api.demographics.PreDefinedDistributions.ConstantDistribution` method), 22
`creation_date` (`emod_api.weather.weather.Metadata` property), 52
`creation_date` (`emod_api.weather.weather.WeatherNode` property), 53
D
`data` (`emod_api.channelreports.channels.Channel` property), 8
`data` (`emod_api.spatialreports.spatial.SpatialNode` property), 51
`data` (`emod_api.spatialreports.spatial.SpatialReport` property), 51
`data` (`emod_api.weather.weather.Weather` property), 53
`data` (`emod_api.weather.weather.WeatherNode` property), 51
`datavalue_count` (`emod_api.weather.weather.Metadata` property), 52
`datavalue_count` (`emod_api.weather.weather.Weather` property), 53
DataValueCount (`emod_api.migration.migration.Layer` property), 43
DataValueCount (`emod_api.migration.migration.Migration` property), 44
`date` (`emod_api.serialization.dtkFileTools.DtkFile` property), 50
`DateCreated` (`emod_api.migration.migration.Migration` property), 44
`decode()` (in module `emod_api.peek_camp`), 54
`decorate_actual_iv()` (in module `emod_api.peek_camp`), 54
`decorate_actual_iv_impl()` (in module `emod_api.peek_camp`), 54
`default_population` (`emod_api.demographics.Node.Node` attribute), 21
`DefaultSusceptibilityDistribution()` (in module `emod_api.demographics.DemographicsTemplates`), 20
DelayedIntervention() (in module `emod_api.interventions.common`), 28
Demographics (class in `emod_api.demographics.Demographics`), 12
DemographicsGenerator (class in `emod_api.demographics.DemographicsGenerator`), 16
DemographicsOverlay (class in `emod_api.demographics.Demographics`), 15
DemographicsType (class in `emod_api.demographics.DemographicsGenerator`), 16
`derive_variables_from_events()` (in module `emod_api.config.dtk_pre_process_adhoc_events`), 10
`do_nodes()` (in module `emod_api.interventions.utils`), 42
`dtk_to_schema()` (in module `emod_api.schema.get_schema`), 46
`dtk_version` (`emod_api.channelreports.channels.ChannelReport` property), 8
`dtk_version` (`emod_api.channelreports.channels.Header` property), 7
DtkFile (class in `emod_api.serialization.dtkFileTools`), 12

49
 DtkFile.Contents (class in *emod_api.serialization.dtkFileTools*), 49
 DtkFile.Objects (class in *emod_api.serialization.dtkFileTools*), 49
 DtkFileV1 (class in *emod_api.serialization.dtkFileTools*), 50
 DtkFileV2 (class in *emod_api.serialization.dtkFileTools*), 50
 DtkFileV2.NodesV2 (class in *emod_api.serialization.dtkFileTools*), 50
 DtkFileV3 (class in *emod_api.serialization.dtkFileTools*), 50
 DtkFileV3.NodesV3 (class in *emod_api.serialization.dtkFileTools*), 50
 DtkFileV4 (class in *emod_api.serialization.dtkFileTools*), 50
 DtkHeader (class in *emod_api.serialization.dtkFileTools*), 49
 duplicate_nodeID_check() (in module *emod_api.demographics.DemographicsInputDataParser*), 19
E
 EllZeeFour (class in *emod_api.serialization.dtkFileSupporters*), 49
 emod_api
 module, 7
 emod_api.campaign
 module, 54
 emod_api.channelreports
 module, 7
 emod_api.channelreports.channels
 module, 7
 emod_api.config
 module, 8
 emod_api.config.default_from_schema
 module, 8
 emod_api.config.default_from_schema_no_validation
 module, 9
 emod_api.config.dtk_post_process_adhoc_events
 module, 10
 emod_api.config.dtk_pre_process_adhoc_events
 module, 10
 emod_api.config.dtk_pre_process_w5ml
 module, 10
 emod_api.config.from_overrides
 module, 10
 emod_api.config.from_poi_and_binary
 module, 10
 emod_api.config.from_schema
 module, 11
 emod_api.config.schema_to_config
 module, 11
 emod_api.demographics
 module, 11
 emod_api.demographics.BaseInputFile
 module, 11
 emod_api.demographics.Demographics
 module, 12
 emod_api.demographics.demographics_utils
 module, 25
 emod_api.demographics.DemographicsGenerator
 module, 16
 emod_api.demographics.DemographicsInputDataParsers
 module, 19
 emod_api.demographics.DemographicsTemplates
 module, 19
 emod_api.demographics.grid_construction
 module, 26
 emod_api.demographics.Node
 module, 21
 emod_api.demographics.PreDefinedDistributions
 module, 22
 emod_api.demographics.PropertiesAndAttributes
 module, 22
 emod_api.demographics.Updateable
 module, 24
 emod_api.interventions
 module, 27
 emod_api.interventions.ccdl
 module, 27
 emod_api.interventions.ccdl_viz
 module, 27
 emod_api.interventions.common
 module, 27
 emod_api.interventions.import_pressure
 module, 38
 emod_api.interventions.migration
 module, 39
 emod_api.interventions.node_multiplier
 module, 40
 emod_api.interventions.outbreak
 module, 41
 emod_api.interventions.simple_vaccine
 module, 42
 emod_api.interventions.utils
 module, 42
 emod_api.migration
 module, 42
 emod_api.migration.client
 module, 42
 emod_api.migration.client.client
 module, 43
 emod_api.migration.migration
 module, 43
 emod_api.peek_camp
 module, 54

`emod_api.schema`
 module, 46
`emod_api.schema.dtk_post_process_schema`
 module, 46
`emod_api.schema.get_schema`
 module, 46
`emod_api.schema_to_class`
 module, 55
`emod_api.serialization`
 module, 46
`emod_api.serialization.CensusAndModPop`
 module, 46
`emod_api.serialization.dtkFileSupport`
 module, 49
`emod_api.serialization.dtkFileTools`
 module, 49
`emod_api.serialization.dtkFileUtility`
 module, 50
`emod_api.serialization.SerializedPopulation`
 module, 46
`emod_api.spatialreports`
 module, 50
`emod_api.spatialreports.spatial`
 module, 50
`emod_api.tabularoutput`
 module, 51
`emod_api.weather`
 module, 51
`emod_api.weather.weather`
 module, 51
`encode()` (in module `emod_api.peek_camp`), 54
`EveryoneInitiallySusceptible()` (in module
 `emod_api.demographics.DemographicsTemplates`), 20
`examine_file()` (in module
 `emod_api.migration.migration`), 45
`experiment`, 57

F

`FAMILY` (`emod_api.migration.migration.Migration`
 attribute), 44
`FEMALE` (`emod_api.migration.migration.Migration`
 attribute), 44
`fill_nodes_legacy()` (in module
 `emod_api.demographics.DemographicsInputDataParsers`), 19
`finalize()` (`emod_api.schema_to_class.ReadOnlyDict`
 method), 55
`find()` (in module `emod_api.serialization.SerializedPopulation`), 48
`flattenConfig()` (in module
 `emod_api.config.from_overrides`), 10
`flush()` (`emod_api.serialization.SerializedPopulation.SerializedPopulation`
 method), 47

`from_csv()` (`emod_api.weather.weather.Weather` class
 method), 53
`from_csv()` (in module
 `emod_api.demographics.Demographics`), 12
`from_csv()` (in module `emod_api.migration.migration`), 45
`from_data()` (`emod_api.demographics.Node.Node` class
 method), 21
`from_dataframe()` (in module
 `emod_api.demographics.DemographicsGenerator`), 17
`from_demog_and_param_gravity()` (in module
 `emod_api.migration.migration`), 45
`from_demog_and_param_gravity_webservice()` (in
 module `emod_api.migration.migration`), 45
`from_dict()` (`emod_api.demographics.PropertiesAndAttributes.Individual`
 method), 23
`from_dict()` (`emod_api.demographics.PropertiesAndAttributes.Individual`
 method), 23
`from_dict()` (`emod_api.demographics.PropertiesAndAttributes.Individual`
 method), 23
`from_dict()` (`emod_api.demographics.PropertiesAndAttributes.NodeAttri`
 method), 24
`from_file()` (`emod_api.weather.weather.Metadata`
 class method), 52
`from_file()` (in module
 `emod_api.demographics.Demographics`), 12
`from_file()` (in module
 `emod_api.demographics.DemographicsGenerator`), 18
`from_file()` (in module
 `emod_api.migration.migration`), 45
`from_params()` (in module
 `emod_api.demographics.Demographics`), 12
`from_params()` (in module
 `emod_api.migration.migration`), 45
`from_pop_csv()` (in module
 `emod_api.demographics.Demographics`), 12
`from_template_node()` (in module
 `emod_api.demographics.Demographics`), 12
`FullRisk()` (in module
 `emod_api.demographics.DemographicsTemplates`), 19
G
`GenderDataType` (`emod_api.migration.migration.Migration`
 property), 44
`generate_demographics()`
 (`emod_api.demographics.DemographicsGenerator.DemographicsGenerator`
 method), 17

IndividualAttributes (class in Layer (class in *emod_api.migration.migration*), 43
emod_api.demographics.PropertiesAndAttributes, 23
 IndividualAttributes.AgeDistribution (class in *emod_api.demographics.PropertiesAndAttributes*, 23
 IndividualAttributes.MortalityDistribution (class in *emod_api.demographics.PropertiesAndAttributes*, 23
 IndividualAttributes.SusceptibilityDistribution (class in *emod_api.demographics.PropertiesAndAttributes*, 23
 IndividualProperties (class in *emod_api.demographics.PropertiesAndAttributes*, 22
 IndividualProperty (class in *emod_api.demographics.PropertiesAndAttributes*, 22
 infer_natural_mortality() (method in *emod_api.demographics.Demographics.Demographics*, 15
 init_resolution_from_file() (method in *emod_api.demographics.Node.Node* class, 21
 InitAgeUniform() (in *emod_api.demographics.DemographicsTemplates*, 21
 InitPrevUniform() (in *emod_api.demographics.DemographicsTemplates*, 20
 InitRiskExponential() (in *emod_api.demographics.DemographicsTemplates*, 20
 InitRiskLogNormal() (in *emod_api.demographics.DemographicsTemplates*, 20
 InitRiskUniform() (in *emod_api.demographics.DemographicsTemplates*, 19
 InitSusceptConstant() (in *emod_api.demographics.DemographicsTemplates*, 20
 InterpolationType (*emod_api.migration.migration.Migration* property), 44
 interval (*emod_api.spatialreports.spatial.SpatialReport* property), 51
 INTERVENTION (*emod_api.migration.migration.Migration* attribute), 44
 InvalidResolution, 16
 L
 lat (*emod_api.demographics.Node.Node* property), 21
 lat_lon_from_nodeid() (in *emod_api.demographics.Node*), 22
 LINEAR_INTERPOLATION (in *emod_api.migration.migration.Migration* attribute), 43
 load_default_config_as_rod() (in *emod_api.config.default_from_schema_no_validation*), 9
 LOCAL (in *emod_api.migration.migration.Migration* attribute), 44
 lon (*emod_api.demographics.Node.Node* property), 21
 lon_lat_2_point() (in *emod_api.demographics.grid_construction*), 26
 M
 make_config_from_poi() (in *emod_api.config.from_poi_and_binary*), 10
 make_config_from_poi_and_config_dict() (in *emod_api.config.from_poi_and_binary*), 10
 make_config_from_poi_and_config_file() (in *emod_api.config.from_poi_and_binary*), 10
 make_config_from_poi_and_schema() (in *emod_api.config.from_poi_and_binary*), 10
 MALE (in *emod_api.migration.migration.Migration* attribute), 44
 MAX_AGE (in *emod_api.migration.migration.Migration* attribute), 44
 Metadata (class in *emod_api.weather.weather*), 51
 metadata (*emod_api.weather.weather.Weather* property), 53
 Migration (class in *emod_api.migration.migration*), 43
 MigrationType (*emod_api.migration.migration.Migration* property), 44
 module
emod_api, 7
emod_api.campaign, 54
emod_api.channelreports, 7
emod_api.channelreports.channels, 7
emod_api.config, 8
emod_api.config.default_from_schema, 8
emod_api.config.default_from_schema_no_validation, 9
emod_api.config.dtk_post_process_adhocevents, 10
emod_api.config.dtk_pre_process_adhocevents, 10
emod_api.config.dtk_pre_process_w5ml, 10
emod_api.config.from_overrides, 10
emod_api.config.from_poi_and_binary, 10
emod_api.config.from_schema, 11
emod_api.config.schema_to_config, 11

emod_api.demographics, 11
 emod_api.demographics.BaseInputFile, 11
 emod_api.demographics.Demographics, 12
 emod_api.demographics.demographics_utils, 25
 emod_api.demographics.DemographicsGenerator, 16
 emod_api.demographics.DemographicsInputDataParsers, 19
 emod_api.demographics.DemographicsTemplates, 19
 emod_api.demographics.grid_construction, 26
 emod_api.demographics.Node, 21
 emod_api.demographics.PreDefinedDistributions, 22
 emod_api.demographics.PropertiesAndAttributes, 22
 emod_api.demographics.Updateable, 24
 emod_api.interventions, 27
 emod_api.interventions.ccdl, 27
 emod_api.interventions.ccdl_viz, 27
 emod_api.interventions.common, 27
 emod_api.interventions.import_pressure, 38
 emod_api.interventions.migration, 39
 emod_api.interventions.node_multiplier, 40
 emod_api.interventions.outbreak, 41
 emod_api.interventions.simple_vaccine, 42
 emod_api.interventions.utils, 42
 emod_api.migration, 42
 emod_api.migration.client, 42
 emod_api.migration.client.client, 43
 emod_api.migration.migration, 43
 emod_api.peek_camp, 54
 emod_api.schema, 46
 emod_api.schema.dtk_post_process_schema, 46
 emod_api.schema.get_schema, 46
 emod_api.schema_to_class, 55
 emod_api.serialization, 46
 emod_api.serialization.CensusAndModPop, 46
 emod_api.serialization.dtkFileSupport, 49
 emod_api.serialization.dtkFileTools, 49
 emod_api.serialization.dtkFileUtility, 50
 emod_api.serialization.SerializedPopulation, 46
 emod_api.spatialreports, 50
 emod_api.spatialreports.spatial, 50
 emod_api.tabularoutput, 51
 emod_api.weather, 51
 emod_api.weather.weather, 51

MortalityRateByAge() (in module *emod_api.demographics.DemographicsTemplates*), 20
 MortalityStructureNigeriaDHS() (in module *emod_api.demographics.DemographicsTemplates*), 20
 MultiInterventionDistributor() (in module *emod_api.interventions.common*), 28
 new_intervention() (in module *emod_api.interventions.import_pressure*), 38
 new_intervention() (in module *emod_api.interventions.node_multiplier*), 40
 new_intervention() (in module *emod_api.interventions.outbreak*), 41
 new_intervention() (in module *emod_api.interventions.simple_vaccine*), 42
 new_intervention2() (in module *emod_api.interventions.simple_vaccine*), 42
 new_intervention_as_file() (in module *emod_api.interventions.import_pressure*), 38
 new_intervention_as_file() (in module *emod_api.interventions.node_multiplier*), 40
 new_intervention_as_file() (in module *emod_api.interventions.outbreak*), 41
 new_intervention_as_file() (in module *emod_api.interventions.simple_vaccine*), 42
 new_scheduled_event() (in module *emod_api.interventions.node_multiplier*), 40
 NLHTI() (in module *emod_api.interventions.common*), 29
 Node (class in *emod_api.demographics.Node*), 21
 node_count (*emod_api.demographics.Demographics.Demographics* property), 13
 node_count (*emod_api.spatialreports.spatial.SpatialReport* property), 51
 node_count (*emod_api.weather.weather.Metadata* property), 52
 node_count (*emod_api.weather.weather.Weather* property), 53
 node_ID_from_lat_long() (in module *emod_api.demographics.DemographicsInputDataParsers*), 19
 node_ids (*emod_api.demographics.Demographics.Demographics* property), 13

- node_ids (*emod_api.spatialreports.spatial.SpatialReport* property), 51
- node_ids (*emod_api.weather.weather.Metadata* property), 52
- node_ids (*emod_api.weather.weather.Weather* property), 53
- NodeAttributes (class in *emod_api.demographics.PropertiesAndAttributes*), 24
- NodeCount (*emod_api.migration.migration.Layer* property), 43
- NodeCount (*emod_api.migration.migration.Migration* property), 44
- nodeid_from_lat_lon() (in module *emod_api.demographics.Node*), 22
- NodeOffsets (*emod_api.migration.migration.Migration* property), 44
- nodes (*emod_api.demographics.Demographics.Demographics* property), 13
- Nodes (*emod_api.migration.migration.Migration* property), 44
- nodes (*emod_api.serialization.dtkFileTools.DtkFile* property), 50
- nodes (*emod_api.serialization.SerializedPopulation.SerializedPopulation* property), 47
- nodes (*emod_api.spatialreports.spatial.SpatialReport* property), 51
- nodes (*emod_api.weather.weather.Metadata* property), 52
- nodes (*emod_api.weather.weather.Weather* property), 53
- nodes_for_DTK() (in module *emod_api.demographics.Node*), 22
- NoInitialPrevalence() (in module *emod_api.demographics.DemographicsTemplates*), 20
- NoRisk() (in module *emod_api.demographics.DemographicsTemplates*), 19
- NullPtr (class in *emod_api.serialization.dtkFileSupport*), 49
- num_channels (*emod_api.channelreports.channels.ChannelReport* property), 8
- num_channels (*emod_api.channelreports.channels.Header* property), 7
- num_time_steps (*emod_api.channelreports.channels.ChannelReport* property), 8
- num_time_steps (*emod_api.channelreports.channels.Header* property), 7
- ## O
- ONE_FOR_EACH_GENDER (*emod_api.migration.migration.Migration* attribute), 43
- OverlayNode (class in *emod_api.demographics.Node*), 21
- ## P
- params_to_dict() (in module *emod_api.peek_camp*), 54
- PIECEWISE_CONSTANT (*emod_api.migration.migration.Migration* attribute), 43
- point_2_grid_cell_id_lookup() (in module *emod_api.demographics.grid_construction*), 26
- pop (*emod_api.demographics.Node.Node* property), 21
- PropertyValueChanger() (in module *emod_api.interventions.common*), 29
- provenance (*emod_api.weather.weather.Metadata* property), 52
- provenance (*emod_api.weather.weather.Weather* property), 53
- ## R
- read() (in module *emod_api.serialization.dtkFileTools*), 50
- ReadOnlyDict (class in *emod_api.schema_to_class*), 55
- recurser() (in module *emod_api.schema.dtk_post_process_schema*), 46
- REGIONAL (*emod_api.migration.migration.Migration* attribute), 44
- report_type (*emod_api.channelreports.channels.ChannelReport* property), 8
- report_type (*emod_api.channelreports.channels.Header* property), 7
- report_version (*emod_api.channelreports.channels.ChannelReport* property), 8
- report_version (*emod_api.channelreports.channels.Header* property), 7
- res_in_degrees (*emod_api.demographics.Node.Node* attribute), 21
- reset() (in module *emod_api.campaign*), 54
- run() (in module *emod_api.migration.client.client*), 43
- ## S
- SAME_FOR_BOTH_GENDERS (*emod_api.migration.migration.Migration* attribute), 43
- save() (in module *emod_api.campaign*), 54
- ScheduledCampaignEvent() (in module *emod_api.interventions.common*), 30
- schema_to_config() (in module *emod_api.config.from_poi_and_binary*), 10
- schema_to_config_subnode() (in module *emod_api.config.default_from_schema_no_validation*), 9
- SchemaConfigBuilder (class in *emod_api.config.from_schema*), 11

SchemaConfigBuilder	(class in <i>emod_api.config.schema_to_config</i>), 11		(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
SEA	(<i>emod_api.migration.migration.Migration</i> attribute), 44	SetDefaultNodeAttributes()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
seed()	(in module <i>emod_api.interventions.outbreak</i>), 41	SetDefaultProperties()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
seed_by_coverage()	(in module <i>emod_api.interventions.outbreak</i>), 41	SetDefaultPropertiesFertMort()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
SerializedPopulation	(class in <i>emod_api.serialization.SerializedPopulation</i>), 46	SetEquilibriumAgeDistFromBirthAndMortRates()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
SerialObject	(class in <i>emod_api.serialization.dtkFileSupport</i>), 49	SetFertilityOverTimeFromParams()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15
set_beautifiers()	(in module <i>emod_api.interventions.ccdl_viz</i>), 27	SetHeteroRiskExponDist()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15
set_demog_distributions()	(in module <i>emod_api.demographics.demographics_utils</i>), 25	SetHeteroRiskLognormalDist()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15
set_growing_demographics()	(in module <i>emod_api.demographics.demographics_utils</i>), 26	SetHeteroRiskUniformDist()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15
set_immune_mod()	(in module <i>emod_api.demographics.demographics_utils</i>), 25	SetIndividualAttributesWithFertMort()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 13
set_resolution()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 16	SetInitialAgeExponential()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
set_risk_mod()	(in module <i>emod_api.demographics.demographics_utils</i>), 25	SetInitialAgeLikeSubSaharanAfrica()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
set_schema()	(<i>emod_api.schema_to_class.ReadOnlyDict</i> method), 55	SetInitPrevFromUniformDraw()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15
set_schema()	(in module <i>emod_api.campaign</i>), 54	SetMinimalNodeAttributes()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 13
set_schema()	(in module <i>emod_api.config.from_poi_and_binary</i>), 10	SetMortalityDistribution()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 13
set_static_demographics()	(in module <i>emod_api.demographics.demographics_utils</i>), 26	SetMortalityOverTimeFromData()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
SetAgeDistribution()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14	SetMortalityRate()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 13
SetBirthRate()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 13	SetNodeDefaultFromTemplate()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
SetConstantRisk()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15	SetOverdispersion()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14
SetConstantSusceptibility()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 15		
SetDefaultFromTemplate()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 14		
SetDefaultIndividualAttributes()	(<i>emod_api.demographics.Demographics.Demographics</i> method), 13		
SetDefaultIndividualProperties()			

method), 14

SimpleSusceptibilityDistribution() (in module *emod_api.demographics.DemographicsTemplates*), 20

simulation(*emod_api.serialization.dtkFileTools.DtkFileVto_dict()* (*emod_api.demographics.PropertiesAndAttributes.IndividualA* property), 50 method), 23

simulation(*emod_api.serialization.dtkFileTools.DtkFileVto_dict()* (*emod_api.demographics.PropertiesAndAttributes.IndividualA* property), 50 method), 23

simulation(*emod_api.serialization.dtkFileTools.DtkFileVto_dict()* (*emod_api.demographics.PropertiesAndAttributes.IndividualPr* property), 50 method), 22

Snappy (class in *emod_api.serialization.dtkFileSupport*), 49

SpatialNode (class in *emod_api.spatialreports.spatial*), 50

SpatialReport (class in *emod_api.spatialreports.spatial*), 51

StandardDiagnostic() (in module *emod_api.interventions.common*), 32

start (*emod_api.spatialreports.spatial.SpatialReport* property), 51

start_time(*emod_api.channelreports.channels.ChannelReport* property), 8

start_time(*emod_api.channelreports.channels.Header* property), 7

STATIC (*emod_api.demographics.DemographicsGenerator.DemographicsGenerator* attribute), 16

step_size(*emod_api.channelreports.channels.ChannelReport* property), 8

step_size (*emod_api.channelreports.channels.Header* property), 7

StepFunctionSusceptibility() (in module *emod_api.demographics.DemographicsTemplates*), 20

to_dict() (*emod_api.demographics.Demographics.Demographics* method), 12

to_dict() (*emod_api.demographics.Demographics.DemographicsOverlay* method), 15

to_dict() (*emod_api.demographics.Node.Node* method), 21

to_dict() (*emod_api.demographics.PreDefinedDistributions.ConstantDistribution* method), 22

to_dict() (*emod_api.demographics.PropertiesAndAttributes.IndividualA* method), 23

to_dict() (*emod_api.demographics.PropertiesAndAttributes.IndividualA* method), 23

to_dict() (*emod_api.demographics.PropertiesAndAttributes.IndividualPr* method), 22

to_dict() (*emod_api.demographics.PropertiesAndAttributes.NodeAttribu* method), 24

to_dict() (*emod_api.demographics.Updateable.Updateable* method), 24

to_file() (*emod_api.demographics.Demographics.DemographicsOverlay* method), 15

to_file() (*emod_api.migration.migration.Migration* method), 44

to_file() (*emod_api.schema_to_class.ReadOnlyDict* method), 55

to_tuple() (*emod_api.demographics.Node.Node* method), 21

tool (*emod_api.serialization.dtkFileTools.DtkFile* property), 50

triggered_campaign_delay_event() (in module *emod_api.interventions.common*), 33

triggered_campaign_event_with_optional_delay() (in module *emod_api.interventions.common*), 33

TriggeredCampaignEvent() (in module *emod_api.interventions.common*), 31

T

task, 57

template, 57

time_stamp(*emod_api.channelreports.channels.ChannelReport* property), 8

time_stamp(*emod_api.channelreports.channels.Header* property), 7

time_steps(*emod_api.spatialreports.spatial.SpatialReport* property), 51

title (*emod_api.channelreports.channels.Channel* property), 8

to_csv() (in module *emod_api.migration.migration*), 45

to_dict() (*emod_api.demographics.Demographics.Demographics* method), 12

to_dict() (*emod_api.demographics.Demographics.DemographicsOverlay* method), 15

to_dict() (*emod_api.demographics.Node.Node* method), 21

to_dict() (*emod_api.demographics.PreDefinedDistributions.ConstantDistribution* method), 22

U

uncompress() (*emod_api.serialization.dtkFileSupport.EllZeeFour* class method), 49

uncompress() (*emod_api.serialization.dtkFileSupport.Snappy* class method), 49

uncompress() (*emod_api.serialization.dtkFileSupport.Uncompressed* class method), 49

uncompress() (in module *emod_api.serialization.dtkFileTools*), 49

Uncompressed (class in *emod_api.serialization.dtkFileSupport*), 49

units (*emod_api.channelreports.channels.Channel* property), 8

update() (*emod_api.demographics.Updateable.Updateable* method), 24

update_resolution(*emod_api.weather.weather.Metadata* property), 2

`update_resolution(emod_api.weather.weather.Weather
property)`, 53
`Updateable` (class in `emod_api.demographics.Updateable`),
 24

V

`validate_res_in_arcsec()` (in module
`emod_api.demographics.DemographicsGenerator`),
 16
`version` (`emod_api.serialization.dtkFileTools.DtkFile
 property`), 50
`viz()` (in module `emod_api.interventions.ccdl_viz`), 27

W

`Weather` (class in `emod_api.weather.weather`), 53
`WeatherNode` (class in `emod_api.weather.weather`), 51
`write()` (`emod_api.serialization.SerializedPopulation.SerializedPopulation
 method`), 47
`write()` (in module `emod_api.serialization.dtkFileTools`),
 50
`write_config_from_default_and_params()` (in
 module `emod_api.config.default_from_schema_no_validation`),
 9
`write_default_from_schema()` (in module
`emod_api.config.default_from_schema`), 8
`write_default_from_schema()` (in module
`emod_api.config.default_from_schema_no_validation`),
 9
`write_file()` (`emod_api.channelreports.channels.ChannelReport
 method`), 8
`write_file()` (`emod_api.spatialreports.spatial.SpatialReport
 method`), 51
`write_file()` (`emod_api.weather.weather.Metadata
 method`), 52
`write_file()` (`emod_api.weather.weather.Weather
 method`), 53

X

`xpix_ypix_from_lat_lon()` (in module
`emod_api.demographics.Node`), 22