

---

**emod-api**

**Institute for Disease Modeling**

**Jan 15, 2021**



# CONTENTS

<b>1</b>	<b>emod_api</b>	<b>3</b>
1.1	emod_api package . . . . .	3
1.1.1	Subpackages . . . . .	3
1.1.2	Submodules . . . . .	29
<b>2</b>	<b>Glossary</b>	<b>31</b>
	<b>Python Module Index</b>	<b>33</b>
	<b>Index</b>	<b>35</b>



EMOD-API is the interface for Epidemiological MODeling software (EMOD) that users of idmttools interact with to create and modify EMOD simulations.



## 1.1 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/Eradiation.[exe]> ...

**To generate a schema.json:** python -m emod\_api.schema.get\_schema </path/to/Eradiation[.exe]>

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

### 1.1.1 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
    property dtk_version
    property time_stamp
    property report_type
    property report_version
    property step_size
        >= 1
    property start_time
        >= 0
    property num_time_steps
        >= 1
    as_dictionary() → dict
```

```
class emod_api.channelreports.channels.Channel (title, units, data)
    Bases: object

    property title
    property units
    property data
    as_dictionary() → dict

class emod_api.channelreports.channels.ChannelReport (filename: str = None, **kwargs)
    Bases: object

    property header
    property dtk_version
    property time_stamp
    property report_type
    property report_version
        major.minor
    property step_size
        >= 1
    property start_time
        >= 0
    property num_time_steps
        > 0
    property num_channels
    property channel_names
    property channels
        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( "de-  
fault_config.json")
```

emod\_api.config.default\_from\_schema\_no\_validation.**write\_default\_from\_schema** (*path\_to\_schema*,  
*schema\_node=True*)

This very simple function takes a DTK schema json file and creates a default config.json 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*,  
*set\_fn*)

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.

**Returns** dict

**Return type** config

emod\_api.config.default\_from\_schema\_no\_validation.**write\_config\_from\_default\_and\_params** (*config*,  
*set\_fn*,  
*config\_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, ad-  
hoc_events)`

Given a config file, a campaign file, and a list of `ad_hoc_events`, do the mappings. The `ad_hoc_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',
                                                             con-
                                                             fig_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.fromBasicNode** (*lat=0, lon=0, pop=1000000.0, name=1, forced\_id=1*)

This function creates a single-node Demographics instance from the params you give it.

**emod\_api.demographics.Demographics.from\_file** (*base\_file*)

**emod\_api.demographics.Demographics.get\_node\_ids\_from\_file** (*demographics\_file*)

**emod\_api.demographics.Demographics.get\_node\_pops\_from\_params** (*tot\_pop, num\_nodes, frac\_rural*)

**emod\_api.demographics.Demographics.from\_synth\_pop** (*tot\_pop=1000000.0, num\_nodes=100, frac\_rural=0.3, id\_ref='from\_synth\_pop'*)

This function creates an EMOD-compatible Demographics object with the population and number of nodes specified, distributing per the additional parameters.

**emod\_api.demographics.Demographics.from\_csv** (*input\_file, res=0.008333333333333333*)

This function uses a csv population-by-node file to create an EMOD-compatible Demographics object.

**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.json type file or from an array of valid Nodes.

**generate\_file** (*name='demographics.json'*)

**property node\_ids**

**property node\_count**

**get\_node** (*nodeid*)

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

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

**AddIndividualPropertyAndHINT** (*Property, Values, InitialDistribution=None, TransmissionMatrix=None*)

This function takes the HINT configuration json, which are presumed to be correct, and inserts them into the right place in the demoagrphics json struture.

**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]]*)

**SetDefaultIndividualAttributes** ()

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

**SetMinimalNodeAttributes** ()

**SetDefaultNodeAttributes** (*birth=True*)

**SetDefaultIndividualProperties** ()

```

SetDefaultProperties ()
SetDefaultPropertiesFertMort (CrudeBirthRate=0.04, CrudeMortRate=0.02)
    Set defaults assuming constant birth and mortality rate
SetDefaultFromTemplate (template, setter_fn=None)
SetNodeDefaultFromTemplate (template, setter_fn)
SetOverdispersion (new_overdispersion_value, nodes=[])
SetConstantSusceptibility ()

```

## emod\_api.demographics.DemographicsGenerator module

```

exception emod_api.demographics.DemographicsGenerator.InvalidResolution
    Bases: 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,
                                                                    con-
                                                                    cerns:
                                                                    Op-
                                                                    tional[Union[emod_api.dtk
                                                                    List[emod_api.dtk_tools.de
                                                                    =
                                                                    None,
                                                                    res_in_arcsec='custom',
                                                                    node_id_from_lat_long=Fa

```

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
- **apply**. If not specified, we use the **DefaultWorldBankEquilibriumConcern** (we) –
- **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.DemographicsGenerator.from_file(population_input_file:
    str,
    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 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
- **apply**. If not specified, we use the DefaultWorldBankEquilibriumConcern (we) –
- **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.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,
                                                                          De-
                                                                          moDf,
                                                                          res=0.008333333333333333)
emod_api.demographics.DemographicsInputDataParsers.ConstructNodesFromDataFrame (node_info,
                                                                                    ex-
                                                                                    tra_data_column
                                                                                    res=0.008333333333333333)
```

## emod\_api.demographics.DemographicsTemplates module

```
emod_api.demographics.DemographicsTemplates.set_suscept_complex (config)
emod_api.demographics.DemographicsTemplates.set_suscept_simple (config)
emod_api.demographics.DemographicsTemplates.set_age_simple (config)
emod_api.demographics.DemographicsTemplates.set_age_complex (config)
emod_api.demographics.DemographicsTemplates.set_init_prev (config)
emod_api.demographics.DemographicsTemplates.set_boxcar_seasonal (config)
```

`emod_api.demographics.DemographicsTemplates.NoRisk()`

NoRisk puts everyone at 0 risk.

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

FullRisk puts everyone at 100% risk.

`emod_api.demographics.DemographicsTemplates.InitRiskUniform(demog, min=0, max=1)`

InitRiskUniform puts everyone at somewhere between 0% risk and 100% risk, drawn uniformly.

**Parameters**

- **min** (*float*) – Low end of uniform distribution. Must be  $\geq 0$ ,  $< 1$ .
- **max** (*float*) – High end of uniform distribution. Must be  $\geq \text{min}$ ,  $\leq 1$ .

**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, prevalence)`

`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, mean-AgeAt-In-fec-tion=2.5)`

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

`emod_api.demographics.DemographicsTemplates.MortalityRateByAge(AgeBins, MortRate)`

Mortality Rate by Age

`emod_api.demographics.DemographicsTemplates.ConstantMortality(CrudeMortRate)`

`emod_api.demographics.DemographicsTemplates.MortalityStructureNigeriaDHS()`

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

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

```
emod_api.demographics.DemographicsTemplates.EquilibriumAgeDistFromBirthAndMortRates (demog,  

Crude-  

BirthRate  

Crude-  

Mor-  

tRate=0.0)
```

```
emod_api.demographics.DemographicsTemplates.AddSeasonalForcing (demog,  

start=100,  

end=330, fac-  

tor=1.0)
```

enable seasonal forcing on the default node attributes via boxcar scaling

Requires config enables infectivity scaling: e.g.: 'Enable\_Infectivity\_Scaling': 1, 'Enable\_Infectivity\_Scaling\_Boxcar': 1,

```
emod_api.demographics.DemographicsTemplates.NoMigrationHeterogeneity()
```

## emod\_api.demographics.Node module

```
class emod_api.demographics.Node.Node (lat, lon, pop, name="", area=None, forced_id=None,  

extra_attributes={}, meta={})
```

Bases: object

**default\_density** = 200

**default\_population** = 1000

**res\_in\_degrees** = 0.041666666666666664

**to\_dict**()

**to\_tuple**()

**property id**

**classmethod init\_resolution\_from\_file** (*fn*)

**classmethod from\_data** (*data*)

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

```
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,  

res_in_deg=0.041666666666666664)
```

```
emod_api.demographics.Node.nodeid_from_lat_lon (lat,  

res_in_deg=0.041666666666666664)
```

```
emod_api.demographics.Node.nodes_for_DTK (filename, nodes)
```

```
emod_api.demographics.Node.basicNode (lat=0, lon=0, pop=1000000.0, name=1, forced_id=1)
```

## 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.import\_pressure module**

```
emod_api.interventions.import_pressure.new_intervention (timestep, durs=[], dips=[],
                                                         nods=[])
emod_api.interventions.import_pressure.new_intervention_as_file (timestep, file-
                                                                name=None)
```

#### **emod\_api.interventions.outbreak module**

```
emod_api.interventions.outbreak.seed_by_coverage (timestep, campaign_hook, cover-
                                                    age=0.01, ignore_immunity=None)
```

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 (timestep, campaign_hook, cases=1)
    Create EMOD-ready Outbreak intervention.
```

##### **Parameters**

- **timestep** (*float*) – timestep at which outbreak should occur.
- **cases** (*integer*) – new paramter 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 (timestep, camp, 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, e_i_r=None)
```

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 fixet 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, file-
                                                                name=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, de-
                                                    cay_rate=0)
```

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.

## emod\_api.migration package

### Submodules

### emod\_api.migration.Migration module

```
class emod_api.migration.Migration.Layer
```

Bases: dict

**property DatavalueCount**

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**

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

**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 of ages - ages < first value use first bucket, ages > last value use last bucket.

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

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

property DateCreated
    datetime: date/time stamp of this datafile

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

property IdReference
    str: ID reference metadata value

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

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

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

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

property Tool
    str: tool metadata value

to_file (binaryfile: pathlib.Path, 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 to filename)
- **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*)



**static from\_file** (*binaryfile: pathlib.Path*)

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)

**Returns** Migration object representing binary data in the given file.

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

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

## 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
```

### 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 Jan 15 21:43:37  
                                                                    2021', 'engine': 'LZ4', 'tool': 'dtk-  
                                                                    FileTools.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={'author': 'unknown',
    'bytecount': 0, 'chunkcount': 0,
    'chunksizes': [], 'compressed':
    True, 'date': 'Fri Jan 15 21:43:37
    2021', 'engine': 'LZ4', 'tool':
    'dtkFileTools.py', 'version': 1},
    filename="", handle=None)
    Bases: emod_api.serialization.dtkFileTools.DtkFile
    property simulation

class emod_api.serialization.dtkFileTools.DtkFileV2(header={'author': 'unknown',
    'bytecount': 0, 'chunkcount': 0,
    'chunksizes': [], 'compressed':
    True, 'date': 'Fri Jan 15 21:43:37
    2021', 'engine': 'LZ4', 'tool':
    'dtkFileTools.py', 'version': 1},
    filename="", handle=None)
    Bases: emod_api.serialization.dtkFileTools.DtkFile

    class NodesV2(parent)
        Bases: object
        property simulation

```

```
class emod_api.serialization.dtkFileTools.DtkFileV3 (header={'author': 'unknown',
                                                             'bytecount': 0, 'chunkcount': 0,
                                                             'chunksizes': [], 'compressed':
                                                             True, 'date': 'Fri Jan 15 21:43:37
                                                             2021', 'engine': 'LZ4', 'tool':
                                                             'dtkFileTools.py', 'version': 1},
                                                             filename="", handle=None)

    Bases: emod_api.serialization.dtkFileTools.DtkFile

class NodesV3 (parent)
    Bases: object

    property simulation

class emod_api.serialization.dtkFileTools.DtkFileV4 (header={'author': 'unknown',
                                                             'bytecount': 0, 'chunkcount': 0,
                                                             'chunksizes': [], 'compressed':
                                                             True, 'date': 'Fri Jan 15 21:43:37
                                                             2021', 'engine': 'LZ4', 'tool':
                                                             'dtkFileTools.py', 'version': 1},
                                                             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
    Node ID
```

```
property data
    Time series data for this node.
```

```
class emod_api.spatialreports.spatial.SpatialReport (filename: str = None, node_ids:
                                                         List[int] = None, data:
                                                         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
    Returns full 2 dimensional NumPy array with report data. Shape is (#values, #nodes).
```

**property node\_ids**  
Returns list of node IDs (integers) for nodes in the report.

**property nodes**  
Returns dictionary of SpatialNodes keyed on node ID.

**property node\_count**  
Number of nodes in the report.

**property time\_steps**  
Number of samples in the report.

**property start**  
Time step of first sample.

**property interval**  
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**  
Node ID

**property data**  
Time series data for this node.

**class** emod\_api.weather.weather.**Metadata** (*node\_ids: List[int], datavalue\_count: int, author: str = None, created: datetime.datetime = None, frequency: str = None, provenance: str = None, reference: str = None*)

Bases: object

Metadata:

- [DateCreated]
- [Author]
- [OriginalDataYears]
- [StartDayOfYear]
- [DataProvenance]
- IdReference

- NodeCount
- DatavalueCount
- UpdateResolution
- NodeOffsets

**property author**

Author of this file.

**property creation\_date**

Creation date of this file.

**property datavalue\_count**

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

**property id\_reference**

'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****property node\_ids****property provenance****property update\_resolution****property nodes**

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: str = None, node_ids: List[int] = None,
                                         datavalue_count: int = None, author: str = None,
                                         created: datetime.datetime = None, frequency: str
                                         = None, provenance: str = None, reference: str =
                                         None, data: numpy.array = None)
```

Bases: object

**property data**

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

**property metadata****property author****property creation\_date**



---

```

property datavalue_count
    >= 1

property id_reference

property node_count
    >= 1

property node_ids

property provenance

property update_resolution

property nodes
    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: str = None, provenance: 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.
- 

## 1.1.2 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.add(event, name=None, first=False)
```

Adds an event to the campaign. event is assumed to be a dict, and a valid event. Not checked here.

```
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_event(event)
```

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.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`, 3  
`emod_api.campaign`, 29  
`emod_api.channelreports`, 3  
`emod_api.channelreports.channels`, 3  
`emod_api.config`, 4  
`emod_api.config.default_from_schema`, 4  
`emod_api.config.default_from_schema_no_validation`, 5  
`emod_api.config.dtk_post_process_adhoc_events`, 6  
`emod_api.config.dtk_pre_process_adhoc_events`, 6  
`emod_api.config.dtk_pre_process_w5ml`, 6  
`emod_api.config.from_overrides`, 6  
`emod_api.config.from_poi_and_binary`, 6  
`emod_api.config.from_schema`, 7  
`emod_api.config.schema_to_config`, 7  
`emod_api.demographics`, 7  
`emod_api.demographics.BaseInputFile`, 7  
`emod_api.demographics.Demographics`, 8  
`emod_api.demographics.demographics_utils`, 16  
`emod_api.demographics.DemographicsGenerator`, 9  
`emod_api.demographics.DemographicsInputDataParsers`, 13  
`emod_api.demographics.DemographicsTemplates`, 13  
`emod_api.demographics.grid_construction`, 17  
`emod_api.demographics.Node`, 15  
`emod_api.interventions`, 18  
`emod_api.interventions.import_pressure`, 18  
`emod_api.interventions.outbreak`, 18  
`emod_api.interventions.simple_vaccine`, 18  
`emod_api.interventions.utils`, 19  
`emod_api.migration`, 19  
`emod_api.migration.Migration`, 19  
`emod_api.schema`, 21  
`emod_api.schema.dtk_post_process_schema`, 21  
`emod_api.schema.get_schema`, 21  
`emod_api.schema_to_class`, 30  
`emod_api.serialization`, 21  
`emod_api.serialization.CensusAndModPop`, 21  
`emod_api.serialization.dtkFileSupport`, 24  
`emod_api.serialization.dtkFileTools`, 24  
`emod_api.serialization.dtkFileUtility`, 26  
`emod_api.serialization.SerializedPopulation`, 22  
`emod_api.spatialreports`, 26  
`emod_api.spatialreports.spatial`, 26  
`emod_api.tabularoutput`, 27  
`emod_api.weather`, 27  
`emod_api.weather.weather`, 27



## INDEX

### A

`add()` (in module *emod\_api.campaign*), 29

`AddAgeDependentTransmission()`  
(*emod\_api.demographics.Demographics.Demographics*  
method), 8

`AddIndividualPropertyAndHINT()`  
(*emod\_api.demographics.Demographics.Demographics*  
method), 8

`AddSeasonalForcing()` (in module  
*emod\_api.demographics.DemographicsTemplates*),  
15

`AgeStructureUNWPP()` (in module  
*emod\_api.demographics.DemographicsTemplates*),  
14

`AgesYears()` (*emod\_api.migration.Migration.Migration*  
property), 20

`AIR` (*emod\_api.migration.Migration.Migration* at-  
tribute), 19

`append()` (*emod\_api.serialization.dtkFileTools.DtkFile.Contents*  
method), 25

`append()` (*emod\_api.serialization.dtkFileTools.DtkFile.Objects*  
method), 25

`application()` (in module  
*emod\_api.config.dtk\_post\_process\_adhocevents*),  
6

`application()` (in module  
*emod\_api.config.dtk\_pre\_process\_adhocevents*),  
6

`application()` (in module  
*emod\_api.config.dtk\_pre\_process\_w5ml*),  
6

`application()` (in module  
*emod\_api.schema.dtk\_post\_process\_schema*),  
21

`apply_to_defaults_or_nodes()` (in module  
*emod\_api.demographics.demographics\_utils*),  
16

`arcsec_to_deg()` (in module  
*emod\_api.demographics.DemographicsGenerator*),  
9

`as_dataframe()` (*emod\_api.channelreports.channels.ChannelReport*  
method), 4

`as_dictionary()` (*emod\_api.channelreports.channels.Channel*  
method), 4

`as_dictionary()` (*emod\_api.channelreports.channels.Header*  
method), 3

asset collection, 31

assets, 31

`author()` (*emod\_api.migration.Migration.Migration*  
property), 20

`author()` (*emod\_api.serialization.dtkFileTools.DtkFile*  
property), 25

`author()` (*emod\_api.weather.weather.Metadata* prop-  
erty), 28

`author()` (*emod\_api.weather.weather.Weather* prop-  
erty), 28

### B

`BaseInputFile` (class in  
*emod\_api.demographics.BaseInputFile*),

`basicNode()` (in module  
*emod\_api.demographics.Node*), 15

builder, 31

`byte_count()` (*emod\_api.serialization.dtkFileTools.DtkFile*  
property), 25

### C

`change_ser_pop()` (in module  
*emod\_api.serialization.CensusAndModPop*),  
21

`Channel` (class in *emod\_api.channelreports.channels*),  
3

`channel_names()` (*emod\_api.channelreports.channels.ChannelReport*  
property), 4

`ChannelReport` (class in  
*emod\_api.channelreports.channels*), 4

`channels()` (*emod\_api.channelreports.channels.ChannelReport*  
property), 4

`chunk_count()` (*emod\_api.serialization.dtkFileTools.DtkFile*  
property), 25

`chunk_sizes()` (*emod\_api.serialization.dtkFileTools.DtkFile*  
property), 25

chunks() (*emod\_api.serialization.dtkFileTools.DtkFile* *default\_density* (*emod\_api.demographics.Node.Node* *property*), 25 *attribute*), 15

compress() (*emod\_api.serialization.dtkFileSupport.EllZeeFour* *default\_population* *class method*), 24 (*emod\_api.demographics.Node.Node* *at-* *tribute*), 15

compress() (*emod\_api.serialization.dtkFileSupport.Snappy* *DefaultSusceptibilityDistribution()* (*in* *emod\_api.serialization.dtkFileSupport.Uncompressed* *module* *emod\_api.demographics.DemographicsTemplates*), *class method*), 24 14

compress() (*in* *module* *Demographics* (*class* *in* *emod\_api.serialization.dtkFileTools*), 24 *emod\_api.demographics.Demographics*),

compressed() (*emod\_api.serialization.dtkFileTools.DtkFile* 8 *DemographicsGenerator* (*class* *in* *emod\_api.serialization.dtkFileTools.DtkFile* *emod\_api.demographics.DemographicsGenerator*), *property*), 25 9

compression() (*emod\_api.serialization.dtkFileTools.DtkFile* *emod\_api.demographics.DemographicsGenerator*), *property*), 25 9

ConstantMortality() (*in* *module* *DemographicsType* (*class* *in* *emod\_api.demographics.DemographicsTemplates*), *emod\_api.demographics.DemographicsGenerator*), 14 9

construct() (*in* *module* *do\_mapping\_from\_events()* (*in* *module* *emod\_api.demographics.grid\_construction*), *emod\_api.config.dtk\_pre\_process\_adhocevents*), 17 6

ConstructNodesFromDataFrame() (*in* *module* *do\_nodes()* (*in* *module* *emod\_api.interventions.utils*), *emod\_api.demographics.DemographicsInputDataParsers*), 19 13

creation\_date() (*emod\_api.weather.weather.Metadata* *emod\_api.schema.get\_schema*), 21 *property*), 28

creation\_date() (*emod\_api.weather.weather.Weather* *emod\_api.channelreports.channels.ChannelReport* *property*), 28 4

creation\_date() (*emod\_api.weather.weather.Weather* *emod\_api.channelreports.channels.Header* *property*), 28 3

D

DtkFile (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFile.Contents (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFile.Objects (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFileV1 (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFileV2 (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFileV2.NodesV2 (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFileV3 (*class* *in* *emod\_api.serialization.dtkFileTools*), 25

DtkFileV3.NodesV3 (*class* *in* *emod\_api.serialization.dtkFileTools*), 26

DtkFileV4 (*class* *in* *emod\_api.serialization.dtkFileTools*), 26

DtkHeader (*class* *in* *emod\_api.serialization.dtkFileTools*), 24

DuplicateNodeIDCheck() (*in* *module* *emod\_api.demographics.DemographicsInputDataParsers*), 13

data() (*emod\_api.channelreports.channels.Channel* *property*), 4

data() (*emod\_api.spatialreports.spatial.SpatialNode* *property*), 26

data() (*emod\_api.spatialreports.spatial.SpatialReport* *property*), 26

data() (*emod\_api.weather.weather.Weather* *property*), 28

data() (*emod\_api.weather.weather.WeatherNode* *property*), 27

datavalue\_count() (*emod\_api.weather.weather.Metadata* *prop-* *erty*), 28

datavalue\_count() (*emod\_api.weather.weather.Weather* *prop-* *erty*), 28

DatavalueCount() (*emod\_api.migration.Migration.Layer* *property*), 19

DatavalueCount() (*emod\_api.migration.Migration.Migration* *property*), 20

date() (*emod\_api.serialization.dtkFileTools.DtkFile* *property*), 25

DateCreated() (*emod\_api.migration.Migration.Migration* *property*), 20

E

EllZeeFour (*class* *in* *emod\_api.serialization.dtkFileTools*), 24



`emod_api.serialization.dtkFileSupport`), 24  
`emod_api`  
  `module`, 3  
`emod_api.campaign`  
  `module`, 29  
`emod_api.channelreports`  
  `module`, 3  
`emod_api.channelreports.channels`  
  `module`, 3  
`emod_api.config`  
  `module`, 4  
`emod_api.config.default_from_schema`  
  `module`, 4  
`emod_api.config.default_from_schema_no_validation`  
  `module`, 5  
`emod_api.config.dtk_post_process_adhoc_events`  
  `module`, 6  
`emod_api.config.dtk_pre_process_adhoc_events`  
  `module`, 6  
`emod_api.config.dtk_pre_process_w5ml`  
  `module`, 6  
`emod_api.config.from_overrides`  
  `module`, 6  
`emod_api.config.from_poi_and_binary`  
  `module`, 6  
`emod_api.config.from_schema`  
  `module`, 7  
`emod_api.config.schema_to_config`  
  `module`, 7  
`emod_api.demographics`  
  `module`, 7  
`emod_api.demographics.BaseInputFile`  
  `module`, 7  
`emod_api.demographics.Demographics`  
  `module`, 8  
`emod_api.demographics.demographics_util`  
  `module`, 16  
`emod_api.demographics.DemographicsGeneralEquilibriumAgeDistFromBirthAndMortRates()`  
  `module`, 9  
`emod_api.demographics.DemographicsInputDataParsers`  
  `module`, 13  
`emod_api.demographics.DemographicsTemplates`  
  `module`, 13  
`emod_api.demographics.grid_construction_experiment`  
  `module`, 17  
`emod_api.demographics.Node`  
  `module`, 15  
`emod_api.interventions`  
  `module`, 18  
`emod_api.interventions.import_pressure`  
  `module`, 18  
`emod_api.interventions.outbreak`  
  `module`, 18  
`emod_api.interventions.simple_vaccine`  
  `module`, 18  
`emod_api.interventions.utils`  
  `module`, 19  
`emod_api.migration`  
  `module`, 19  
`emod_api.migration.Migration`  
  `module`, 19  
`emod_api.schema`  
  `module`, 21  
`emod_api.schema.dtk_post_process_schema`  
  `module`, 21  
`emod_api.schema.get_schema`  
  `module`, 21  
`emod_api.schema.schema_to_class`  
  `module`, 30  
`emod_api.serialization`  
  `module`, 21  
`emod_api.serialization.CensusAndModPop`  
  `module`, 21  
`emod_api.serialization.dtkFileSupport`  
  `module`, 24  
`emod_api.serialization.dtkFileTools`  
  `module`, 24  
`emod_api.serialization.dtkFileUtility`  
  `module`, 26  
`emod_api.serialization.SerializedPopulation`  
  `module`, 22  
`emod_api.spatialreports`  
  `module`, 26  
`emod_api.spatialreports.spatial`  
  `module`, 26  
`emod_api.tabularoutput`  
  `module`, 27  
`emod_api.weather`  
  `module`, 27  
`emod_api.weather.weather`  
  `module`, 27  
`emod_api.weather.weather.EquilibriumAgeDistFromBirthAndMortRates()`  
  `(in module emod_api.demographics.DemographicsTemplates)`,  
  14  
`emod_api.weather.weather.EveryoneInitiallySusceptible()` *(in module*  
  `emod_api.demographics.DemographicsTemplates)`,  
  14  
**F**  
**FAMILY** *(emod\_api.migration.Migration.Migration attribute)*, 19  
**FEMALE** *(emod\_api.migration.Migration.Migration attribute)*, 20  
`fill_nodes_legacy()` *(in module*  
  `emod_api.demographics.DemographicsInputDataParsers)`,  
  13

[finalize\(\)](#) (*emod\_api.schema\_to\_class.ReadOnlyDict* [method](#)), 30  
[find\(\)](#) (*in module emod\_api.serialization.SerializedPopulation*), 23  
[flattenConfig\(\)](#) (*in module emod\_api.config.from\_overrides*), 6  
[flush\(\)](#) (*emod\_api.serialization.SerializedPopulation.SerializedPopulation* [method](#)), 23  
[from\\_csv\(\)](#) (*emod\_api.weather.weather.Weather class* [method](#)), 29  
[from\\_csv\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[from\\_data\(\)](#) (*emod\_api.demographics.Node.Node class* [method](#)), 15  
[from\\_dataframe\(\)](#) (*in module emod\_api.demographics.DemographicsGenerator*), 10  
[from\\_demog\\_and\\_param\\_gravity\(\)](#) (*in module emod\_api.migration.Migration*), 21  
[from\\_file\(\)](#) (*emod\_api.migration.Migration.Migration static method*), 20  
[from\\_file\(\)](#) (*emod\_api.weather.weather.Metadata class* [method](#)), 28  
[from\\_file\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[from\\_file\(\)](#) (*in module emod\_api.demographics.DemographicsGenerator*), 12  
[from\\_pop\\_csv\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[from\\_synth\\_pop\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[from\\_synth\\_pop\(\)](#) (*in module emod\_api.migration.Migration*), 21  
[fromBasicNode\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[FullRisk\(\)](#) (*in module emod\_api.demographics.DemographicsTemplates*), 14  
**G**  
[GenderDataType\(\)](#) (*emod\_api.migration.Migration.Migration property*), 20  
[generate\\_demographics\(\)](#) (*emod\_api.demographics.DemographicsGenerator.DemographicsGenerator* [method](#)), 10  
[generate\\_file\(\)](#) (*emod\_api.demographics.BaseInputFile.BaseInputFile* [method](#)), 7  
[generate\\_file\(\)](#) (*emod\_api.demographics.Demographics.Demographics* [method](#)), 8  
[generate\\_headers\(\)](#) (*emod\_api.demographics.BaseInputFile.BaseInputFile* [method](#)), 7  
[generate\\_metadata\(\)](#) (*emod\_api.demographics.DemographicsGenerator.DemographicsGenerator* [method](#)), 10  
[generate\\_nodes\(\)](#) (*emod\_api.demographics.DemographicsGenerator.DemographicsGenerator* [method](#)), 10  
[get\\_adhocs\(\)](#) (*in module emod\_api.campaign*), 29  
[get\\_bbox\(\)](#) (*in module emod\_api.demographics.grid\_construction*), 18  
[get\\_class\\_with\\_defaults\(\)](#) (*in module emod\_api.schema\_to\_class*), 30  
[get\\_config\\_from\\_default\\_and\\_params\(\)](#) (*in module emod\_api.config.default\_from\_schema\_no\_validation*), 5  
[get\\_default\\_for\\_complex\\_type\(\)](#) (*in module emod\_api.schema\_to\_class*), 30  
[get\\_event\(\)](#) (*in module emod\_api.campaign*), 29  
[get\\_grid\\_cell\\_id\(\)](#) (*in module emod\_api.demographics.grid\_construction*), 17  
[get\\_next\\_individual\\_suid\(\)](#) (*emod\_api.serialization.SerializedPopulation.SerializedPopulation* [method](#)), 23  
[get\\_next\\_infection\\_suid\(\)](#) (*emod\_api.serialization.SerializedPopulation.SerializedPopulation* [method](#)), 23  
[get\\_node\(\)](#) (*emod\_api.demographics.Demographics.Demographics* [method](#)), 8  
[get\\_node\\_ids\\_from\\_file\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[get\\_node\\_pops\\_from\\_params\(\)](#) (*in module emod\_api.demographics.Demographics*), 8  
[get\\_parameters\(\)](#) (*in module emod\_api.serialization.SerializedPopulation*), 23  
[get\\_schema\(\)](#) (*in module emod\_api.campaign*), 29  
[get\\_waning\\_from\\_params\(\)](#) (*in module emod\_api.interventions.utils*), 19  
[get\\_xpix\\_ypix\(\)](#) (*in module emod\_api.demographics.Node*), 15  
**H**  
[Header](#) (*class in emod\_api.channelreports.channels*), 3  
[Header](#) (*emod\_api.channelreports.channels.ChannelReport* [property](#)), 4  
[Header](#) (*emod\_api.serialization.dtkFileTools.DtkFile* [property](#)), 25  
[high-performance computing \(HPC\)](#), 31

## I

[id\(\)](#) (*emod\_api.demographics.Node.Node* property), 15  
[id\(\)](#) (*emod\_api.spatialreports.spatial.SpatialNode* property), 26  
[id\(\)](#) (*emod\_api.weather.weather.WeatherNode* property), 27  
[id\\_reference\(\)](#) (*emod\_api.weather.weather.Metadata* property), 28  
[id\\_reference\(\)](#) (*emod\_api.weather.weather.Weather* property), 29  
[IDREF\\_GRUMP1DEGREE](#) (*emod\_api.migration.Migration.Migration* attribute), 20  
[IDREF\\_GRUMP2PT5ARCMIN](#) (*emod\_api.migration.Migration.Migration* attribute), 20  
[IDREF\\_GRUMP30ARCSEC](#) (*emod\_api.migration.Migration.Migration* attribute), 20  
[IDREF\\_LEGACY](#) (*emod\_api.migration.Migration.Migration* attribute), 20  
[IdReference\(\)](#) (*emod\_api.migration.Migration.Migration* property), 20  
[init\\_resolution\\_from\\_file\(\)](#) (*emod\_api.demographics.Node.Node* class method), 15  
[InitAgeUniform\(\)](#) (in module *emod\_api.demographics.DemographicsTemplates*), 14  
[InitPrevUniform\(\)](#) (in module *emod\_api.demographics.DemographicsTemplates*), 14  
[InitRiskUniform\(\)](#) (in module *emod\_api.demographics.DemographicsTemplates*), 14  
[InitSusceptConstant\(\)](#) (in module *emod\_api.demographics.DemographicsTemplates*), 14  
[InterpolationType\(\)](#) (*emod\_api.migration.Migration.Migration* property), 20  
[interval\(\)](#) (*emod\_api.spatialreports.spatial.SpatialReport* property), 27  
[INTERVENTION](#) (*emod\_api.migration.Migration.Migration* attribute), 19  
[InvalidResolution](#), 9

## L

[lat\\_lon\\_from\\_nodeid\(\)](#) (in module *emod\_api.demographics.Node*), 15  
[Layer](#) (class in *emod\_api.migration.Migration*), 19  
[LINEAR\\_INTERPOLATION](#) (*emod\_api.migration.Migration.Migration* attribute), 19

[load\\_default\\_config\\_as\\_rod\(\)](#) (in module *emod\_api.config.default\_from\_schema\_no\_validation*), 5  
[LOCAL](#) (*emod\_api.migration.Migration.Migration* attribute), 19  
[lon\\_lat\\_2\\_point\(\)](#) (in module *emod\_api.demographics.grid\_construction*), 18

## M

[make\\_config\\_from\\_poi\(\)](#) (in module *emod\_api.config.from\_poi\_and\_binary*), 6  
[make\\_config\\_from\\_poi\\_and\\_config\\_dict\(\)](#) (in module *emod\_api.config.from\_poi\_and\_binary*), 6  
[make\\_config\\_from\\_poi\\_and\\_config\\_file\(\)](#) (in module *emod\_api.config.from\_poi\_and\_binary*), 6  
[make\\_config\\_from\\_poi\\_and\\_schema\(\)](#) (in module *emod\_api.config.from\_poi\_and\_binary*), 6  
[MALE](#) (*emod\_api.migration.Migration.Migration* attribute), 20  
[MAX\\_AGE](#) (*emod\_api.migration.Migration.Migration* attribute), 20  
[Metadata](#) (class in *emod\_api.weather.weather*), 27  
[Metadata\(\)](#) (*emod\_api.weather.weather.Weather* property), 28  
[Migration](#) (class in *emod\_api.migration.Migration*), 19  
[MigrationType\(\)](#) (*emod\_api.migration.Migration.Migration* property), 20  
[module](#)  
*emod\_api*, 3  
*emod\_api.campaign*, 29  
*emod\_api.channelreports*, 3  
*emod\_api.channelreports.channels*, 3  
*emod\_api.config*, 4  
*emod\_api.config.default\_from\_schema*, 4  
*emod\_api.config.default\_from\_schema\_no\_validation*, 5  
*emod\_api.config.dtk\_post\_process\_adhoc\_events*, 6  
*emod\_api.config.dtk\_pre\_process\_adhoc\_events*, 6  
*emod\_api.config.dtk\_pre\_process\_w5ml*, 6  
*emod\_api.config.from\_overrides*, 6  
*emod\_api.config.from\_poi\_and\_binary*, 6  
*emod\_api.config.from\_schema*, 7  
*emod\_api.config.schema\_to\_config*, 7

emod\_api.demographics, 7  
 emod\_api.demographics.BaseInputFile, 7  
 emod\_api.demographics.Demographics, 8  
 emod\_api.demographics.demographics\_utils, 16  
 emod\_api.demographics.DemographicsGenerator, 9  
 emod\_api.demographics.DemographicsInputDataParsers, 13  
 emod\_api.demographics.DemographicsTemplates, 13  
 emod\_api.demographics.grid\_construction, 17  
 emod\_api.demographics.Node, 15  
 emod\_api.interventions, 18  
 emod\_api.interventions.import\_pressure, 18  
 emod\_api.interventions.outbreak, 18  
 emod\_api.interventions.simple\_vaccine, 18  
 emod\_api.interventions.utils, 19  
 emod\_api.migration, 19  
 emod\_api.migration.Migration, 19  
 emod\_api.schema, 21  
 emod\_api.schema.dtk\_post\_process\_schema, 21  
 emod\_api.schema.get\_schema, 21  
 emod\_api.schema\_to\_class, 30  
 emod\_api.serialization, 21  
 emod\_api.serialization.CensusAndModPop, 21  
 emod\_api.serialization.dtkFileSupport, 24  
 emod\_api.serialization.dtkFileTools, 24  
 emod\_api.serialization.dtkFileUtility, 26  
 emod\_api.serialization.SerializedPopulation, 22  
 emod\_api.spatialreports, 26  
 emod\_api.spatialreports.spatial, 26  
 emod\_api.tabularoutput, 27  
 emod\_api.weather, 27  
 emod\_api.weather.weather, 27  
 MortalityRateByAge() (in module *emod\_api.demographics.DemographicsTemplates*), 14  
 MortalityStructureNigeriaDHS() (in module *emod\_api.demographics.DemographicsTemplates*), 14  
 new\_intervention() (in module *emod\_api.interventions.import\_pressure*), 18  
 new\_intervention() (in module *emod\_api.interventions.outbreak*), 18  
 new\_intervention() (in module *emod\_api.interventions.simple\_vaccine*), 18  
 new\_intervention2() (in module *emod\_api.interventions.simple\_vaccine*), 18  
 new\_intervention\_as\_file() (in module *emod\_api.interventions.import\_pressure*), 18  
 new\_intervention\_as\_file() (in module *emod\_api.interventions.outbreak*), 18  
 new\_intervention\_as\_file() (in module *emod\_api.interventions.simple\_vaccine*), 19  
 Node (class in *emod\_api.demographics.Node*), 15  
 node\_count() (*emod\_api.demographics.Demographics.Demographics* property), 8  
 node\_count() (*emod\_api.spatialreports.spatial.SpatialReport* property), 27  
 node\_count() (*emod\_api.weather.weather.Metadata* property), 28  
 node\_count() (*emod\_api.weather.weather.Weather* property), 29  
 node\_ID\_from\_lat\_long() (in module *emod\_api.demographics.DemographicsInputDataParsers*), 13  
 node\_ids() (*emod\_api.demographics.Demographics.Demographics* property), 8  
 node\_ids() (*emod\_api.spatialreports.spatial.SpatialReport* property), 26  
 node\_ids() (*emod\_api.weather.weather.Metadata* property), 28  
 node\_ids() (*emod\_api.weather.weather.Weather* property), 29  
 NodeCount() (*emod\_api.migration.Migration.Layer* property), 19  
 NodeCount() (*emod\_api.migration.Migration.Migration* property), 20  
 nodeid\_from\_lat\_lon() (in module *emod\_api.demographics.Node*), 15  
 NodeOffsets() (*emod\_api.migration.Migration.Migration* property), 20  
 nodes() (*emod\_api.serialization.dtkFileTools.DtkFile* property), 25  
 nodes() (*emod\_api.serialization.SerializedPopulation.SerializedPopulation* property), 22  
 nodes() (*emod\_api.spatialreports.spatial.SpatialReport* property), 27  
 nodes() (*emod\_api.weather.weather.Metadata* property), 28

`nodes()` (*emod\_api.weather.weather.Weather* property), 29  
`nodes_for_DTK()` (in module *emod\_api.demographics.Node*), 15  
`NoInitialPrevalence()` (in module *emod\_api.demographics.DemographicsTemplates*), 14  
`NoMigrationHeterogeneity()` (in module *emod\_api.demographics.DemographicsTemplates*), 15  
`NoRisk()` (in module *emod\_api.demographics.DemographicsTemplates*), 13  
`num_channels()` (*emod\_api.channelreports.channels.ChannelReport* property), 4  
`num_channels()` (*emod\_api.channelreports.channels.Header* property), 3  
`num_time_steps()` (*emod\_api.channelreports.channels.ChannelReport* property), 4  
`num_time_steps()` (*emod\_api.channelreports.channels.Header* property), 3  
**O**  
`ONE_FOR_EACH_GENDER` (*emod\_api.migration.Migration.Migration* attribute), 19  
**P**  
`PIECEWISE_CONSTANT` (*emod\_api.migration.Migration.Migration* attribute), 19  
`point_2_grid_cell_id_lookup()` (in module *emod\_api.demographics.grid\_construction*), 18  
`provenance()` (*emod\_api.weather.weather.Metadata* property), 28  
`provenance()` (*emod\_api.weather.weather.Weather* property), 29  
**R**  
`read()` (in module *emod\_api.serialization.dtkFileTools*), 26  
`ReadOnlyDict` (class in *emod\_api.schema\_to\_class*), 30  
`recurser()` (in module *emod\_api.schema.dtk\_post\_process\_schema*), 21  
`REGIONAL` (*emod\_api.migration.Migration.Migration* attribute), 19  
`report_type()` (*emod\_api.channelreports.channels.ChannelReport* property), 4  
`report_type()` (*emod\_api.channelreports.channels.Header* property), 3  
`report_version()` (*emod\_api.channelreports.channels.ChannelReport* property), 4  
`report_version()` (*emod\_api.channelreports.channels.Header* property), 3  
`res_in_degrees` (*emod\_api.demographics.Node.Node* attribute), 15  
**S**  
`SAME_FOR_BOTH_GENDERS` (*emod\_api.migration.Migration.Migration* attribute), 19  
`save()` (in module *emod\_api.campaign*), 29  
`schema_to_config()` (in module *emod\_api.config.from\_poi\_and\_binary*), 6  
`schema_to_config_subnode()` (in module *emod\_api.config.default\_from\_schema\_no\_validation*), 6  
`SchemaConfigBuilder` (class in *emod\_api.config.from\_schema*), 7  
`SchemaConfigBuilder` (class in *emod\_api.config.schema\_to\_config*), 7  
`SEA` (*emod\_api.migration.Migration.Migration* attribute), 19  
`seed_by_coverage()` (in module *emod\_api.interventions.outbreak*), 18  
`SerializedPopulation` (class in *emod\_api.serialization.SerializedPopulation*), 22  
`SerialObject` (class in *emod\_api.serialization.dtkFileSupport*), 24  
`set_age_complex()` (in module *emod\_api.demographics.DemographicsTemplates*), 13  
`set_age_simple()` (in module *emod\_api.demographics.DemographicsTemplates*), 13  
`set_boxcar_seasonal()` (in module *emod\_api.demographics.DemographicsTemplates*), 13  
`set_demog_distributions()` (in module *emod\_api.demographics.demographics\_utils*), 16  
`set_growing_demographics()` (in module *emod\_api.demographics.demographics\_utils*), 17  
`set_immune_mod()` (in module *emod\_api.demographics.demographics\_utils*), 16  
`set_init_prev()` (in module *emod\_api.demographics.DemographicsTemplates*), 13  
`set_resolution()` (*emod\_api.demographics.DemographicsGenerator* method), 9



[set\\_risk\\_mod\(\)](#) (in module [property](#)), 25  
[emod\\_api.demographics.demographics\\_utils](#), 16  
[simulation\(\)](#) ([emod\\_api.serialization.dtkFileTools.DtkFileV2](#) property), 25  
[set\\_schema\(\)](#) ([emod\\_api.schema\\_to\\_class.ReadOnlyDict](#) method), 30  
[simulation\(\)](#) ([emod\\_api.serialization.dtkFileTools.DtkFileV3](#) property), 26  
[set\\_schema\(\)](#) (in module [Snappy](#) (class in [emod\\_api.serialization.dtkFileSupport](#)), 24  
[emod\\_api.config.from\\_poi\\_and\\_binary](#), 6  
[SpatialNode](#) (class in [emod\\_api.spatialreports.spatial](#)), 26  
[set\\_static\\_demographics\(\)](#) (in module [emod\\_api.demographics.demographics\\_utils](#), 17  
[SpatialReport](#) (class in [emod\\_api.spatialreports.spatial](#)), 26  
[set\\_suscept\\_complex\(\)](#) (in module [start\(\)](#) ([emod\\_api.spatialreports.spatial.SpatialReport](#) property), 27  
[emod\\_api.demographics.DemographicsTemplates](#), 13  
[start\\_time\(\)](#) ([emod\\_api.channelreports.channels.ChannelReport](#) property), 4  
[set\\_suscept\\_simple\(\)](#) (in module [start\\_time\(\)](#) ([emod\\_api.channelreports.channels.Header](#) property), 3  
[emod\\_api.demographics.DemographicsTemplates](#), 13  
[SetConstantSusceptibility\(\)](#) [STATIC](#) ([emod\\_api.demographics.DemographicsGenerator.Demographics](#) attribute), 9  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 9  
[step\\_size\(\)](#) ([emod\\_api.channelreports.channels.ChannelReport](#) property), 4  
[SetDefaultFromTemplate\(\)](#) [step\\_size\(\)](#) ([emod\\_api.channelreports.channels.Header](#) property), 3  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 9  
[SetDefaultIndividualAttributes\(\)](#) [StepFunctionSusceptibility\(\)](#) (in module [emod\\_api.demographics.DemographicsTemplates](#), 14  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 8  
[SetDefaultIndividualProperties\(\)](#) [T](#)  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 8  
[task](#), 31  
[SetDefaultNodeAttributes\(\)](#) [template](#), 31  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 8  
[time\\_stamp\(\)](#) ([emod\\_api.channelreports.channels.ChannelReport](#) property), 4  
[SetDefaultProperties\(\)](#) [time\\_stamp\(\)](#) ([emod\\_api.channelreports.channels.Header](#) property), 3  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 8  
[time\\_steps\(\)](#) ([emod\\_api.spatialreports.spatial.SpatialReport](#) property), 27  
[SetDefaultPropertiesFertMort\(\)](#) [time\(\)](#) ([emod\\_api.channelreports.channels.Channel](#) property), 4  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 9  
[to\\_dict\(\)](#) ([emod\\_api.demographics.Node.Node](#) method), 15  
[SetIndividualAttributtsWithFertMort\(\)](#) [to\\_file\(\)](#) ([emod\\_api.migration.Migration.Migration](#) method), 20  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 8  
[to\\_file\(\)](#) ([emod\\_api.schema\\_to\\_class.ReadOnlyDict](#) method), 30  
[SetMinimalNodeAttributes\(\)](#) [to\\_tuple\(\)](#) ([emod\\_api.demographics.Node.Node](#) method), 15  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 9  
[Tool\(\)](#) ([emod\\_api.migration.Migration.Migration](#) property), 20  
[SetNodeDefaultFromTemplate\(\)](#) [uncompress\(\)](#) ([emod\\_api.serialization.dtkFileTools.DtkFile](#) property), 25  
[\(emod\\_api.demographics.Demographics.Demographics](#) method), 9  
[SimpleSusceptibilityDistribution\(\)](#) (in module [emod\\_api.demographics.DemographicsTemplates](#), 14  
[simulation\(\)](#) ([emod\\_api.serialization.dtkFileTools.DtkFileV1](#) class method), 24

```

uncompress() (emod_api.serialization.dtkFileSupport.Snappy
               class method), 24
uncompress() (emod_api.serialization.dtkFileSupport.Uncompressed
               class method), 24
uncompress() (in module
               emod_api.serialization.dtkFileTools), 24
Uncompressed (class in
               emod_api.serialization.dtkFileSupport), 24
units() (emod_api.channelreports.channels.Channel
          property), 4
update_resolution()
    (emod_api.weather.weather.Metadata prop-
     erty), 28
update_resolution()
    (emod_api.weather.weather.Weather prop-
     erty), 29

```

## V

```

validate_res_in_arcsec() (in module
                          emod_api.demographics.DemographicsGenerator),
                          9
version() (emod_api.serialization.dtkFileTools.DtkFile
           property), 25

```

## W

```

Weather (class in emod_api.weather.weather), 28
WeatherNode (class in emod_api.weather.weather), 27
write() (emod_api.serialization.SerializedPopulation.SerializedPopulation
         method), 23
write() (in module
         emod_api.serialization.dtkFileTools), 26
write_config_from_default_and_params()
    (in module emod_api.config.default_from_schema_no_validation),
    5
write_default_from_schema() (in module
                             emod_api.config.default_from_schema), 4
write_default_from_schema() (in module
                             emod_api.config.default_from_schema_no_validation),
                             5
write_file() (emod_api.channelreports.channels.ChannelReport
              method), 4
write_file() (emod_api.spatialreports.spatial.SpatialReport
              method), 27
write_file() (emod_api.weather.weather.Metadata
              method), 28
write_file() (emod_api.weather.weather.Weather
              method), 29

```

## X

```

xpix_ypix_from_lat_lon() (in module
                           emod_api.demographics.Node), 15

```