
emod-api

Institute for Disease Modeling

Apr 22, 2021

CONTENTS

1	emod_api	3
1.1	emod_api package	3
1.1.1	Subpackages	3
1.1.2	Submodules	35
2	Glossary	37
	Python Module Index	39
	Index	41

EMOD-API is the interface for Epidemiological MODeling software (EMOD) that users of idmttools interact with to create and modify EMOD simulations. Additional functionality for interacting with EMOD is provided in the [emodpy](#) and [idmttools](#) packages.

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.from_template_node` (*lat=0, lon=0, pop=1000000.0, name='Erewhon', 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_params` (*tot_pop=1000000.0, num_nodes=100, frac_rural=0.3, id_ref='from_params'*)

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:

SetIndividualAttributesWithFertMort (*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 ()
SetInfectivityMultiplierByNode (node_id_to_multiplier)

```

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 arsecond. 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.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 ≥ 0 , < 1 .
- **max** (*float*) – High end of uniform distribution. Must be $\geq \text{min}$, ≤ 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_Infection=2.5)`

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

`emod_api.demographics.DemographicsTemplates.MortalityRateByAge (demog, age_bins, mort_rates)`

Mortality Rate by Age

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

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

`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
- **"15-19", etc. up to "45-49", and a column named "Years" with values like "1950-1955". (labelled)** –

- can be extra columns and the columns can be anywhere. (There) –

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.DemographicsTemplates.EquilibriumAgeDistFromBirthAndMortRates(demog, Crude-BirthRate, Crude-MortalityRate=0.0)`

For the moment, this is assuming constant mortality, should update with an interpolator based on demographics file to improve Reuse of some of these demographics variables and passing them around indicates this may be better as a class. May be best to wait however, until the idmtools Demographics class is around, so that we can inherit from that and only add functionality. Should maybe allow user to specify the “fineness” of the age-binning for the final demographics distribution Having to do this conversion of crude birth rates to per-day per-person rates frequently. Should find the right place to do this only once.

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, 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=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.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`
- **but without the distribution, which is inferred. E.g.,**
(*parameters,*) –
- **"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
- **intervention management.** (*duplicate*) –

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_Demographic='Everyone', Target_Age_Min=0, Target_Age_Max=45625, Target_Gender='All', Target_Residents_Only=1, Duration=- 1, Blackout_Event_Trigger='Blackout_Event_Trigger', Blackout_Period=0, Blackout_On_First_Occurrence=0, 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_Demographic** – Everyone, ExplicitAgeRanges, etc.
- **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,  
New_Property_Value,  
Daily_Probability=1.0, Maximum_Duration=1, Revert=False,  
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.

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, Node-
set_Config, Number_Repetitions=1,
Timesteps_Between_Repetitions=1,
Event_Name='Scheduled_Campaign_Event',
Node_Property_Restrictions=None,
Property_Restrictions=None, Target_Demographic='Everyone',
Demographic_Coverage=1.0,
Target_Age_Min=0, Target_Age_Max=45625, Target_Gender='All', 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 not functional meaning.
- **Nodeset_Config** – Nodes to target with this intervention, return from utils.do_nodes().
- **Node_Property_Restrictions** – Optional property restriction for node-targeted interventions.
- **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_Demographic** – Everyone, ExplicitAgeRanges, etc.
- **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, Event_Name,
Nodeset_Config, Triggers, Intervention_List,
Node_Property_Restrictions=None,
Property_Restrictions=None,
Number_Repetitions=1,
Timesteps_Between_Repetitions=-1,
Demographic_Coverage=1.0, Target_Demographic='Everyone',
Target_Age_Min=0, Target_Age_Max=45625,
Target_Gender='All', Target_Residents_Only=1,
Duration=-1, Blackout_Event_Trigger='Blackout_Event_Trigger',
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 not functional meaning.
- **Nodeset_Config** – Nodes to target with this intervention, return from utils.do_nodes().
- **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_Demographic** – Everyone, ExplicitAgeRanges, etc.
- **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.
- **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=1,
                                                  Base_Specificity=1,
                                                  Days_To_Diagnosis=0,
                                                  Treatment_Fraction=1,
                                                  Event_Or_Config='Config')
```

Wrapper function to create and return a StandardDiagnostic 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.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.outbreak module

```
emod_api.interventions.outbreak.seed_by_coverage(timestep, campaign_hook, coverage=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 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(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 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, 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

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

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
```

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 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 Nodes**property NodeCount**

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

get_node_offsets (*limit: int = 100*) → dict**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, metafile: 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: 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(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.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': 'Thu Apr 22  
22:05:32 2021', 'engine': 'LZ4',  
                                                                    'tool': 'dtkFileTools.py', 'ver-  
sion': 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': [], 'com-
    pressed': True, 'date': 'Thu Apr
    22 22:05:32 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': [], 'com-
    pressed': True, 'date': 'Thu Apr
    22 22:05:32 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': [], 'com-
pressed': True, 'date': 'Thu Apr
22 22:05:32 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': [], 'com-
pressed': True, 'date': 'Thu Apr
22 22:05:32 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.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_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.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`, 35
`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.common`, 18
`emod_api.interventions.import_pressure`, 23
`emod_api.interventions.outbreak`, 23
`emod_api.interventions.simple_vaccine`, 24
`emod_api.interventions.utils`, 24
`emod_api.migration`, 24
`emod_api.migration.migration`, 24
`emod_api.schema`, 27
`emod_api.schema.dtk_post_process_schema`, 27
`emod_api.schema.get_schema`, 27
`emod_api.schema_to_class`, 36
`emod_api.serialization`, 27
`emod_api.serialization.CensusAndModPop`, 27
`emod_api.serialization.dtkFileSupport`, 30
`emod_api.serialization.dtkFileTools`, 30
`emod_api.serialization.dtkFileUtility`, 32
`emod_api.serialization.SerializedPopulation`, 28
`emod_api.spatialreports`, 32
`emod_api.spatialreports.spatial`, 32
`emod_api.tabularoutput`, 33
`emod_api.weather`, 33
`emod_api.weather.weather`, 33

A

`add()` (in module *emod_api.campaign*), 35

`AddAgeDependentTransmission()`
(*emod_api.demographics.Demographics.Demographics*
method), 8

`AddIndividualPropertyAndHINT()`
(*emod_api.demographics.Demographics.Demographics*
method), 8

`AgeStructureUNWPP()` (in module
emod_api.demographics.DemographicsTemplates),
15

`AgessYears()` (*emod_api.migration.migration.Migration*
property), 25

`AIR` (*emod_api.migration.migration.Migration* at-
tribute), 25

`append()` (*emod_api.serialization.dtkFileTools.DtkFile.Contents*
method), 30

`append()` (*emod_api.serialization.dtkFileTools.DtkFile.Objects*
method), 31

`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),
27

`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, 37

assets, 37

`author()` (*emod_api.migration.migration.Migration*
property), 25

`author()` (*emod_api.serialization.dtkFileTools.DtkFile*
property), 31

`author()` (*emod_api.weather.weather.Metadata* prop-
erty), 34

`author()` (*emod_api.weather.weather.Weather* prop-
erty), 34

B

`BaseInputFile` (class in
emod_api.demographics.BaseInputFile),

`basicNode()` (in module
emod_api.demographics.Node), 15

`BroadcastEvent()` (in module
emod_api.interventions.common), 18

`BroadcastEventToOtherNodes()` (in module
emod_api.interventions.common), 18

builder, 37

`byte_count()` (*emod_api.serialization.dtkFileTools.DtkFile*
property), 31

C

`change_ser_pop()` (in module
emod_api.serialization.CensusAndModPop),
27

`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), 31

`chunk_sizes()` (*emod_api.serialization.dtkFileTools.DtkFile*
property), 31

chunks () (emod_api.serialization.dtkFileTools.DtkFile default_density (emod_api.demographics.Node.Node property), 31 attribute), 15

compress () (emod_api.serialization.dtkFileSupport.EllzFile default_population (emod_api.demographics.Node.Node class method), 30 attribute), 15

compress () (emod_api.serialization.dtkFileSupport.Snappy (class method), 30 DefaultSusceptibilityDistribution () (in module emod_api.demographics.DemographicsTemplates), 14

compress () (emod_api.serialization.dtkFileSupport.Uncompressed module emod_api.demographics.DemographicsTemplates), 14

compress () (in module DelayedIntervention () (in module emod_api.serialization.dtkFileTools), 30 emod_api.interventions.common), 19

compressed () (emod_api.serialization.dtkFileTools.DtkFileDemographics (class in emod_api.demographics.Demographics), 31

compression () (emod_api.serialization.dtkFileTools.DtkFile 8

property), 31 DemographicsGenerator (class in emod_api.demographics.DemographicsGenerator), 9

ConstantMortality () (in module emod_api.demographics.DemographicsTemplates), 14

DemographicsType (class in emod_api.demographics.DemographicsGenerator), 9

construct () (in module emod_api.demographics.grid_construction), 17

do_mapping_from_events () (in module emod_api.config.dtk_pre_process_adhoc_events), 6

ConstructNodesFromDataFrame () (in module emod_api.demographics.DemographicsInputDataParsers), 13

do_nodes () (in module emod_api.interventions.utils), 24

creation_date () (emod_api.weather.weather.Metadata property), 34

dtk_to_schema () (in module emod_api.schema.get_schema), 27

creation_date () (emod_api.weather.weather.Weather property), 34

dtk_version () (emod_api.channelreports.channels.ChannelReport property), 4

D dtk_version () (emod_api.channelreports.channels.Header property), 3

data () (emod_api.channelreports.channels.Channel property), 4

DtkFile (class in emod_api.serialization.dtkFileTools), 30

data () (emod_api.spatialreports.spatial.SpatialNode property), 32

DtkFile.Contents (class in emod_api.serialization.dtkFileTools), 30

data () (emod_api.spatialreports.spatial.SpatialReport property), 32

DtkFile.Objects (class in emod_api.serialization.dtkFileTools), 31

data () (emod_api.weather.weather.Weather property), 34

DtkFileV1 (class in emod_api.serialization.dtkFileTools), 31

data () (emod_api.weather.weather.WeatherNode property), 33

DtkFileV2 (class in emod_api.serialization.dtkFileTools), 31

datavalue_count ()

DtkFileV2.NodesV2 (class in emod_api.serialization.dtkFileTools), 31

(emod_api.weather.weather.Metadata property), 34

DtkFileV3 (class in emod_api.serialization.dtkFileTools), 31

datavalue_count ()

DtkFileV3.NodesV3 (class in emod_api.serialization.dtkFileTools), 32

(emod_api.weather.weather.Weather property), 35

DtkFileV4 (class in emod_api.serialization.dtkFileTools), 32

DatavalueCount () (emod_api.migration.migration.Layer property), 24

DtkHeader (class in emod_api.serialization.dtkFileTools), 30

DatavalueCount () (emod_api.migration.migration.Migration property), 25

date () (emod_api.serialization.dtkFileTools.DtkFile property), 31

duplicate_nodeID_check () (in module emod_api.demographics.DemographicsInputDataParsers), 13

DateCreated () (emod_api.migration.migration.Migration property), 25

E

EllZeeFour (class in *emod_api.serialization.dtkFileSupport*), 30
 emod_api
 module, 3
 emod_api.campaign
 module, 35
 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_utils
 module, 16
 emod_api.demographics.DemographicsGenerator
 module, 9
 emod_api.demographics.DemographicsInputDataParsers
 module, 13
 emod_api.demographics.DemographicsTemplates
 module, 13
 emod_api.demographics.grid_construction
 module, 17
 emod_api.demographics.Node
 module, 15
 emod_api.interventions
 module, 18
 emod_api.interventions.common
 module, 18
 emod_api.interventions.import_pressure
 module, 23
 emod_api.interventions.outbreak
 module, 23
 emod_api.interventions.simple_vaccine
 module, 24
 emod_api.interventions.utils
 module, 24
 emod_api.migration
 module, 24
 emod_api.migration.migration
 module, 24
 emod_api.schema
 module, 27
 emod_api.schema.dtk_post_process_schema
 module, 27
 emod_api.schema.get_schema
 module, 27
 emod_api.schema_to_class
 module, 36
 emod_api.serialization
 module, 27
 emod_api.serialization.CensusAndModPop
 module, 27
 emod_api.serialization.dtkFileSupport
 module, 30
 emod_api.serialization.dtkFileTools
 module, 30
 emod_api.serialization.dtkFileUtility
 module, 32
 emod_api.serialization.SerializedPopulation
 module, 28
 emod_api.spatialreports
 module, 32
 emod_api.spatialreports.spatial
 module, 32
 emod_api.tabularoutput
 module, 33
 emod_api.weather
 module, 33
 emod_api.weather.weather
 module, 33
 EquilibriumAgeDistFromBirthAndMortRates()
 (in module *emod_api.demographics.DemographicsTemplates*),
 15
 EveryoneInitiallySusceptible() (in module
 emod_api.demographics.DemographicsTemplates),
 14
 examine_file() (in module
 emod_api.migration.migration), 26
 experiment, 37

F

FAMILY (*emod_api.migration.migration.Migration* attribute), 25

FEMALE (*emod_api.migration.migration.Migration* attribute), 25
 fill_nodes_legacy() (in module *emod_api.demographics.DemographicsInputDataParsers*), method), 7
 13
 finalize() (*emod_api.schema_to_class.ReadOnlyDict* method), 36
 find() (in module *emod_api.serialization.SerializedPopulation*), 29
 flattenConfig() (in module *emod_api.config.from_overrides*), 6
 flush() (*emod_api.serialization.SerializedPopulation.SerializedPopulation* method), 28
 from_csv() (*emod_api.weather.weather.Weather* class method), 35
 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*), 27
 from_file() (*emod_api.weather.weather.Metadata* class method), 34
 from_file() (in module *emod_api.demographics.Demographics*), 8
 from_file() (in module *emod_api.demographics.DemographicsGenerator*), 12
 from_file() (in module *emod_api.migration.migration*), 26
 from_params() (in module *emod_api.demographics.Demographics*), 8
 from_params() (in module *emod_api.migration.migration*), 26
 from_pop_csv() (in module *emod_api.demographics.Demographics*), 8
 from_template_node() (in module *emod_api.demographics.Demographics*), 8
 FullRisk() (in module *emod_api.demographics.DemographicsTemplates*), 13
 G
 GenderDataType() (*emod_api.migration.migration.Migration* property), 25
 generate_demographics() (*emod_api.demographics.DemographicsGenerator.DemographicsGenerator* method), 10
 generate_file() (*emod_api.demographics.BaseInputFile.BaseInputFile* method), 7
 generate_file() (*emod_api.demographics.Demographics.DemographicsGenerator* 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*), 35
 get_bbox() (in module *emod_api.demographics.grid_construction*), 18
 get_class_with_defaults() (in module *emod_api.schema_to_class*), 36
 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*), 36
 get_event() (in module *emod_api.campaign*), 35
 get_fert_dist() (in module *emod_api.demographics.DemographicsTemplates*), 14
 get_grid_cell_id() (in module *emod_api.demographics.grid_construction*), 17
 get_next_individual_suid() (*emod_api.serialization.SerializedPopulation.SerializedPopulation* method), 29
 get_next_infection_suid() (*emod_api.serialization.SerializedPopulation.SerializedPopulation* method), 29
 get_node() (*emod_api.demographics.Demographics.Demographics* method), 8
 get_node_ids_from_file() (in module *emod_api.demographics.Demographics*), 8
 get_node_offsets() (*emod_api.migration.migration.Migration* method), 26
 get_node_pops_from_params() (in module *emod_api.demographics.Demographics*), 8
 get_parameters() (in module *emod_api.serialization.SerializedPopulation*), 29
 get_schema() (in module *emod_api.campaign*), 35
 get_trigger_list() (in module *emod_api.campaign*), 35

get_waning_from_params() (in module *emod_api.interventions.utils*), 24
 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), 31
 high-performance computing (HPC), 37
 HSB() (in module *emod_api.interventions.common*), 19
I
 id() (*emod_api.demographics.Node.Node* property), 15
 id() (*emod_api.spatialreports.spatial.SpatialNode* property), 32
 id() (*emod_api.weather.weather.WeatherNode* property), 33
 id_reference() (*emod_api.weather.weather.Metadata* property), 34
 id_reference() (*emod_api.weather.weather.Weather* property), 35
 IDREF_GRUMP1DEGREE (*emod_api.migration.migration.Migration* attribute), 25
 IDREF_GRUMP2PT5ARCMIN (*emod_api.migration.migration.Migration* attribute), 25
 IDREF_GRUMP30ARCSEC (*emod_api.migration.migration.Migration* attribute), 25
 IDREF_LEGACY (*emod_api.migration.migration.Migration* attribute), 25
 IdReference() (*emod_api.migration.migration.Migration* property), 26
 init_resolution_from_file() (*emod_api.demographics.Node.Node* class method), 15
 InitAgeUniform() (in module *emod_api.demographics.DemographicsTemplates*), 15
 InitPrevUniform() (in module *emod_api.demographics.DemographicsTemplates*), 14
 InitRiskUniform() (in module *emod_api.demographics.DemographicsTemplates*), 13
 InitSusceptConstant() (in module *emod_api.demographics.DemographicsTemplates*), 14
 InterpolationType() (*emod_api.migration.migration.Migration* property), 26
 interval() (*emod_api.spatialreports.spatial.SpatialReport* property), 33
 INTERVENTION (*emod_api.migration.migration.Migration* attribute), 25
 InvalidResolution, 9
L
 lat_lon_from_nodeid() (in module *emod_api.demographics.Node*), 15
 Layer (class in *emod_api.migration.migration*), 24
 LINEAR_INTERPOLATION (*emod_api.migration.migration.Migration* attribute), 25
 load_default_config_as_rod() (in module *emod_api.config.default_from_schema_no_validation*), 5
 LOCAL (*emod_api.migration.migration.Migration* attribute), 25
 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), 25
 MAX_AGE (*emod_api.migration.migration.Migration* attribute), 25
 Metadata (class in *emod_api.weather.weather*), 33
 metadata() (*emod_api.weather.weather.Weather* property), 34
 Migration (class in *emod_api.migration.migration*), 24
 MigrationType() (*emod_api.migration.migration.Migration* property), 26
 module
 emod_api, 3
 emod_api.campaign, 35
 emod_api.channelreports, 3
 emod_api.channelreports.channels, 3
 emod_api.config, 4

emod_api.config.default_from_schema, 32
 4 emod_api.serialization.SerializedPopulation,
 emod_api.config.default_from_schema_no_validation, 28
 5 emod_api.spatialreports, 32
 emod_api.config.dtk_post_process_adhoc_events, 32
 6 emod_api.spatialreports.spatial, 32
 emod_api.config.dtk_pre_process_adhoc_events, 33
 6 emod_api.tabularoutput, 33
 emod_api.config.dtk_pre_process_w5ml, 33
 6 emod_api.weather, 33
 emod_api.config.dtk_pre_process_w5ml, MortalityRateByAge() (in module
 6 emod_api.weather.weather, 33
 emod_api.config.dtk_pre_process_w5ml, MortalityRateByAge() (in module
 6 emod_api.demographics.DemographicsTemplates),
 emod_api.config.from_overrides, 6 14
 emod_api.config.from_poi_and_binary, MortalityStructureNigeriaDHS() (in module
 6 emod_api.demographics.DemographicsTemplates),
 emod_api.config.from_schema, 7 14
 emod_api.config.schema_to_config, 7 MultiInterventionDistributor() (in module
 emod_api.demographics, 7 emod_api.interventions.common), 18
 emod_api.demographics.BaseInputFile, 7 **N**
 emod_api.demographics.Demographics, new_intervention() (in module
 8 emod_api.interventions.import_pressure),
 emod_api.demographics.demographics_utils, 23
 16 new_intervention() (in module
 emod_api.demographics.DemographicsGenerator, emod_api.interventions.outbreak), 23
 9 new_intervention() (in module
 emod_api.demographics.DemographicsInputDataParsers, emod_api.interventions.simple_vaccine),
 13 24
 emod_api.demographics.DemographicsTemplates, new_intervention2() (in module
 13 emod_api.interventions.simple_vaccine),
 emod_api.demographics.grid_construction, 24
 17 new_intervention_as_file() (in module
 emod_api.demographics.Node, 15 emod_api.interventions.import_pressure), 23
 emod_api.interventions, 18 new_intervention_as_file() (in module
 emod_api.interventions.common, 18 emod_api.interventions.outbreak), 23
 emod_api.interventions.import_pressure, new_intervention_as_file() (in module
 23 emod_api.interventions.simple_vaccine), 24
 emod_api.interventions.outbreak, 23 NLHTI() (in module emod_api.interventions.common),
 emod_api.interventions.simple_vaccine, 19
 24 Node (class in emod_api.demographics.Node), 15
 emod_api.interventions.utils, 24 node_count() (emod_api.demographics.Demographics.Demographics
 emod_api.migration, 24 property), 8
 emod_api.migration.migration, 24 node_count() (emod_api.spatialreports.spatial.SpatialReport
 emod_api.schema, 27 property), 33
 emod_api.schema.dtk_post_process_schema, node_count() (emod_api.weather.weather.Metadata
 27 property), 34
 emod_api.schema.get_schema, 27 node_count() (emod_api.weather.weather.Weather
 emod_api.schema_to_class, 36 property), 35
 emod_api.serialization, 27 node_ID_from_lat_long() (in module
 emod_api.serialization.CensusAndModPop, emod_api.demographics.DemographicsInputDataParsers),
 27 13
 emod_api.serialization.dtkFileSupport, node_ids() (emod_api.demographics.Demographics.Demographics
 30 property), 8
 emod_api.serialization.dtkFileTools, node_ids() (emod_api.spatialreports.spatial.SpatialReport
 30 property), 33
 emod_api.serialization.dtkFileUtility,

`node_ids()` (*emod_api.weather.weather.Metadata* property), 34
`node_ids()` (*emod_api.weather.weather.Weather* property), 35
`NodeCount()` (*emod_api.migration.migration.Layer* property), 24
`NodeCount()` (*emod_api.migration.migration.Migration* property), 26
`nodeid_from_lat_lon()` (in module *emod_api.demographics.Node*), 15
`NodeOffsets()` (*emod_api.migration.migration.Migration* property), 26
`Nodes()` (*emod_api.migration.migration.Migration* property), 26
`nodes()` (*emod_api.serialization.dtkFileTools.DtkFile* property), 31
`nodes()` (*emod_api.serialization.SerializedPopulation.SerializedPopulation* property), 25
`nodes()` (*emod_api.spatialreports.spatial.SpatialReport* property), 33
`nodes()` (*emod_api.weather.weather.Metadata* property), 34
`nodes()` (*emod_api.weather.weather.Weather* property), 35
`nodes_for_DTK()` (in module *emod_api.demographics.Node*), 15
`NoInitialPrevalence()` (in module *emod_api.demographics.DemographicsTemplates*), 14
`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), 25
P
`PIECEWISE_CONSTANT` (*emod_api.migration.migration.Migration* attribute), 25
`point_2_grid_cell_id_lookup()` (in module *emod_api.demographics.grid_construction*), 18
`PropertyValueChanger()` (in module *emod_api.interventions.common*), 20
`provenance()` (*emod_api.weather.weather.Metadata* property), 34
`provenance()` (*emod_api.weather.weather.Weather* property), 35
R
`read()` (in module *emod_api.serialization.dtkFileTools*), 32
`ReadOnlyDict` (class in *emod_api.schema_to_class*), 36
`recursor()` (in module *emod_api.schema.dtk_post_process_schema*), 27
`REGIONAL` (*emod_api.migration.migration.Migration* attribute), 25
`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), 25
`save()` (in module *emod_api.campaign*), 35
`ScheduledCampaignEvent()` (in module *emod_api.interventions.common*), 20
`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*), 5
`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), 25
`seed_by_coverage()` (in module *emod_api.interventions.outbreak*), 23
`SerializedPopulation` (class in *emod_api.serialization.SerializedPopulation*), 28
`SerialObject` (class in *emod_api.serialization.dtkFileSupport*), 30

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_resolution() (*emod_api.demographics.DemographicsGenerator.DemographicsGenerator*), 9
 set_risk_mod() (in module *emod_api.demographics.demographics_utils*), 16
 set_schema() (*emod_api.schema_to_class.ReadOnlyDict*), 36
 set_schema() (in module *emod_api.config.from_poi_and_binary*), 6
 set_static_demographics() (in module *emod_api.demographics.demographics_utils*), 17
 SetConstantSusceptibility() (*emod_api.demographics.Demographics.Demographics*), 9
 SetDefaultFromTemplate() (*emod_api.demographics.Demographics.Demographics*), 9
 SetDefaultIndividualAttributes() (*emod_api.demographics.Demographics.Demographics*), 8
 SetDefaultIndividualProperties() (*emod_api.demographics.Demographics.Demographics*), 8
 SetDefaultNodeAttributes() (*emod_api.demographics.Demographics.Demographics*), 8
 SetDefaultProperties() (*emod_api.demographics.Demographics.Demographics*), 9
 SetDefaultPropertiesFertMort() (*emod_api.demographics.Demographics.Demographics*), 9
 SetIndividualAttributesWithFertMort() (*emod_api.demographics.Demographics.Demographics*), 8
 SetInfectivityMultiplierByNode() (*emod_api.demographics.Demographics.Demographics*), 9
 SetMinimalNodeAttributes() (*emod_api.demographics.Demographics.Demographics*), 8
 SetNodeDefaultFromTemplate() (*emod_api.demographics.Demographics.Demographics*), 26
 SetOverdispersion() (*emod_api.demographics.Demographics.Demographics*), 9
 SimpleSusceptibilityDistribution() (in module *emod_api.demographics.DemographicsTemplates*), 14
 simulation() (*emod_api.serialization.dtkFileTools.DtkFileV1*), 31
 simulation() (*emod_api.serialization.dtkFileTools.DtkFileV2*), 31
 simulation() (*emod_api.serialization.dtkFileTools.DtkFileV3*), 32
 Snappy (class in *emod_api.serialization.dtkFileSupport*), 30
 SpatialNode (class in *emod_api.spatialreports.spatial*), 32
 SpatialReport (class in *emod_api.spatialreports.spatial*), 32
 StandardDiagnostic() (in module *emod_api.interventions.common*), 23
 start() (*emod_api.spatialreports.spatial.SpatialReport*), 33
 start_time() (*emod_api.channelreports.channels.ChannelReport*), 4
 start_time() (*emod_api.channelreports.channels.Header*), 3
 STATIC (*emod_api.demographics.DemographicsGenerator.DemographicsGenerator*), 9
 time_size() (*emod_api.channelreports.channels.ChannelReport*), 4
 step_size() (*emod_api.channelreports.channels.Header*), 3
 StepFunctionSusceptibility() (in module *emod_api.demographics.DemographicsTemplates*), 14
 template, 37
 time_stamp() (*emod_api.channelreports.channels.ChannelReport*), 4
 time_stamp() (*emod_api.channelreports.channels.Header*), 3
 time_steps() (*emod_api.spatialreports.spatial.SpatialReport*), 33
 title() (*emod_api.channelreports.channels.ChannelReport*), 4
 to_csv() (in module *emod_api.migration.migration*), 27
 to_dict() (*emod_api.demographics.Node.Node*), 15
 to_file() (*emod_api.migration.migration.Migration*), 26

T

`to_file()` (*emod_api.schema_to_class.ReadOnlyDict* method), 36
`to_tuple()` (*emod_api.demographics.Node.Node* method), 15
`Tool()` (*emod_api.migration.migration.Migration* property), 26
`tool()` (*emod_api.serialization.dtkFileTools.DtkFile* property), 31
`TriggeredCampaignEvent()` (in module *emod_api.interventions.common*), 21

U

`uncompress()` (*emod_api.serialization.dtkFileSupport.EllZeeFour* class method), 30
`uncompress()` (*emod_api.serialization.dtkFileSupport.Snappy* class method), 30
`uncompress()` (*emod_api.serialization.dtkFileSupport.Uncompressed* class method), 30
`uncompress()` (in module *emod_api.serialization.dtkFileTools*), 30
`Uncompressed` (class in *emod_api.serialization.dtkFileSupport*), 30
`units()` (*emod_api.channelreports.channels.Channel* property), 4
`update_resolution()` (*emod_api.weather.weather.Metadata* property), 34
`update_resolution()` (*emod_api.weather.weather.Weather* property), 35

V

`validate_res_in_arcsec()` (in module *emod_api.demographics.DemographicsGenerator*), 9
`version()` (*emod_api.serialization.dtkFileTools.DtkFile* property), 31

W

`Weather` (class in *emod_api.weather.weather*), 34
`WeatherNode` (class in *emod_api.weather.weather*), 33
`write()` (*emod_api.serialization.SerializedPopulation.SerializedPopulation* method), 29
`write()` (in module *emod_api.serialization.dtkFileTools*), 32
`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

X

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