
emodpy

Institute for Disease Modeling

Apr 21, 2021

CONTENTS

1	Installation	3
1.1	Prerequisites	3
1.1.1	Basic installation	3
1.1.2	Developer installation	4
2	Create simulations	7
3	Create input files	9
4	Run simulations	11
5	Calibrate simulations	13
6	Parameter sweeps and model iteration	15
6.1	Parameter sweeps for model calibration	15
6.2	Parameter sweeps and stochasticity	15
6.3	How to do parameter sweeps	15
7	Introduction to analyzers	17
8	Output reports	19
9	Serialization	21
10	emodpy	23
10.1	emodpy package	23
10.1.1	Subpackages	23
10.1.2	Submodules	32
11	Glossary	51
12	Changelog	53
12.1	1.1.0	53
12.1.1	Additional Changes	53
12.1.2	Bugs	53
12.1.3	Developer/Test	54
12.1.4	Documentation	54
12.1.5	Feature Request	54
12.1.6	Models	54
12.1.7	Platforms	54
12.1.8	User Experience	54
12.2	1.2.0	55

12.2.1	Additional Changes	55
12.2.2	Bugs	55
12.2.3	Documentation	55
12.2.4	Feature Request	55
12.2.5	User Experience	55
Python Module Index		57
Index		59

emodpy is a collection of Python scripts and utilities created to streamline user interactions with EMOD and idmtools. Additional functionality for interacting with EMOD is provided in the [emod_api](#) and [idmtools](#) packages.

INSTALLATION

You can install emodpy in two different ways. If you intend to use emodpy as IDM builds it, follow the instructions in *Basic installation*. However, if you intend to modify the emodpy source code to add new functionality, follow the instructions in *Developer installation*. Whichever installation method you choose, the prerequisites are the same.

1.1 Prerequisites

- Windows 10 Pro or Enterprise
- Python 3.6 or 3.7 64-bit (<https://www.python.org/downloads/release>)

Warning: Do not install Python 3.8, which includes breaking changes.

- Python virtual environments

Python virtual environments enable you to isolate your Python environments from one another and give you the option to run multiple versions of Python on the same computer. When using a virtual environment, you can indicate the version of Python you want to use and the packages you want to install, which will remain separate from other Python environments. You may use `virtualenv`, which requires a separate installation, but `venv` is recommended and included with Python 3.3+.

1.1.1 Basic installation

Follow the steps below if you will use `idmtools` to run and analyze simulations, but will not make source code changes.

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

```
python -m venv emodpy
```

2. Activate the virtual environment:

- On Windows, enter the following:

```
emodpy\Scripts\activate
```

- On Linux, enter the following:

```
source emodpy/bin/activate
```

3. Install `idmtools` packages:

```
pip install emodpy --index-url=https://packages.idmod.org/api/pypi/pypi-  
→production/simple
```

If you are on Python 3.6, also run:

```
pip install dataclasses
```

4. Verify installation by pulling up idmttools help:

```
emodpy --help
```

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

```
deactivate
```

1.1.2 Developer installation

Follow the steps below if you will make changes to the idmttools source code to add new functionality.

Install idmttools

1. Install a Git client such as Git Bash or the Git GUI.
2. Open a command prompt and clone the idmttools GitHub repository to a local directory using the following command:

```
git clone https://github.com/InstituteForDiseaseModeling/emodpy-idmttools.git  
  
To work from the latest approved code, work from the "master" branch. To work from the latest code under active development, work from the "dev" branch.
```

3. Open a command prompt and create a virtual environment in any directory you choose. The command below names the environment “emodpy”, but you may use any desired name:

```
python -m venv emodpy
```

4. Activate the virtual environment:

- On Windows, enter the following:

```
emodpy\Scripts\activate
```

- On Linux, enter the following:

```
source emodpy/bin/activate
```

5. In the base directory of the cloned GitHub repository, run the setup script.

- On Windows, enter the following:

```
pip install py-make  
pymake setup-dev
```

- On Linux, enter the following:


```
make setup-dev
```

6. To verify that idmtools is installed, enter the following command:

```
emodpy --help
```

You should see a list of available cookie cutter projects and command-line options.

Run tests

If you want to run tests on the code, do the following. You can add new tests to the GitHub repository and they will be run using the same commands. Note that COMPS access is generally restricted to IDM employees.

1. Login to COMPS by navigating to the idmtools root directory and entering the following at a command prompt:

```
python dev_scripts\create_auth_token_args.py --comps_url https://comps2.idmod.org_  
↪--username yourcomps_user --password yourcomps_password
```

2. If you are running the local platform with the nightly idmtools build, enter the following to log in to Docker:

```
docker login idm-docker-staging.packages.idmod.org
```

3. Navigate to the directory containing the code you want to test, such as the root directory or a subdirectory like emodpy_platform_comps, enter the following command:

```
pymake test-all
```


CREATE SIMULATIONS

CREATE INPUT FILES

RUN SIMULATIONS

CALIBRATE SIMULATIONS

PARAMETER SWEEPS AND MODEL ITERATION

Contents

- *Parameter sweeps for model calibration*
- *Parameter sweeps and stochasticity*
- *How to do parameter sweeps*

6.1 Parameter sweeps for model calibration

(more info) For more information on model calibration, see *Calibrate simulations*.

6.2 Parameter sweeps and stochasticity

With a stochastic model (such as EMOD), it is especially important to utilize parameter sweeps, not only for calibration to data or parameter selection, but to fully explore the stochasticity in output. Single model runs may appear to provide good fits to data, but variation will arise and multiple runs are necessary to determine the appropriate range of parameter values necessary to achieve desired outcomes. Multiple iterations of a single set of parameter values should be run to determine trends in simulation output: a single simulation output could provide results that are due to random chance.

6.3 How to do parameter sweeps

INTRODUCTION TO ANALYZERS

OUTPUT REPORTS

SERIALIZATION

10.1 emodpy package

10.1.1 Subpackages

`emodpy.analyzers` package

Submodules

`emodpy.analyzers.adult_vectors_analyzer` module

class `emodpy.analyzers.adult_vectors_analyzer.AdultVectorsAnalyzer` (*name='hi'*)

Bases: `idmtools.entities.ianalyzer.IAnalyzer`

initialize ()

Call once after the analyzer has been added to the `AnalyzeManager`.

Add everything depending on the working directory or unique ID here instead of in `__init__`.

map (*data: Any, item: idmtools.core.interfaces.item.IItem*) → Any

In parallel for each simulation, consume raw data from filenames and emit selected data.

Parameters

- **data** – A dictionary associating filename with content for simulation data.
- **item** – `IItem` object that the passed data is associated with.

Returns Selected data for the given item.

reduce (*all_data: dict*) → Any

Combine the `map()` data for a set of items into an aggregate result.

Parameters **all_data** – A dictionary with entries for the item ID and selected data.

emodpy.analyzers.population_analyzer module

```
class emodpy.analyzers.population_analyzer.PopulationAnalyzer (name='idm')
    Bases: idmtools.entities.ianalyzer.IAnalyzer
```

```
initialize ()
```

Call once after the analyzer has been added to the AnalyzeManager.

Add everything depending on the working directory or unique ID here instead of in `__init__`.

```
map (data: Any, item: idmtools.core.interfaces.iitem.IItem) → Any
```

In parallel for each simulation, consume raw data from filenames and emit selected data.

Parameters

- **data** – A dictionary associating filename with content for simulation data.
- **item** – `IItem` object that the passed data is associated with.

Returns Selected data for the given item.

```
reduce (all_data: dict) → Any
```

Combine the `map()` data for a set of items into an aggregate result.

Parameters **all_data** – A dictionary with entries for the item ID and selected data.

emodpy.analyzers.timeseries_analyzer module

```
class emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer (filenames=['output/InsetChart.json'],
                                                                chan-
                                                                nels=('Statistical
                                                                Population',
                                                                'Infectious
                                                                Population',
                                                                'Infected', 'Wan-
                                                                ing Population'),
                                                                save_output=True)

    Bases: idmtools.entities.ianalyzer.IAnalyzer
```

```
data_group_names = ['group', 'sim_id', 'channel']
```

```
ordered_levels = ['channel', 'group', 'sim_id']
```

```
output_file = 'timeseries.csv'
```

```
initialize ()
```

Call once after the analyzer has been added to the AnalyzeManager.

Add everything depending on the working directory or unique ID here instead of in `__init__`.

```
default_select_fn (ts)
```

```
default_group_fn (k, v)
```

```
default_plot_fn (df, ax)
```

```
default_filter_fn (md)
```

```
filter (simulation)
```

Decide whether the analyzer should process a simulation.

Parameters **item** – An `IItem` to be considered for processing with this analyzer.

Returns A Boolean indicating whether simulation should be analyzed by this analyzer.

get_channel_data (*data_by_channel*, *selected_channels*)

map (*data*, *simulation*)

In parallel for each simulation, consume raw data from filenames and emit selected data.

Parameters

- **data** – A dictionary associating filename with content for simulation data.
- **item** – `Item` object that the passed data is associated with.

Returns Selected data for the given item.

plot_by_channel (*channels*, *plot_fn*)

reduce (*all_data*)

Combine the `map()` data for a set of items into an aggregate result.

Parameters **all_data** – A dictionary with entries for the item ID and selected data.

emodpy.defaults package

Subpackages

emodpy.defaults.ep4 package

Submodules

emodpy.defaults.ep4.dtk_in_process module

`emodpy.defaults.ep4.dtk_in_process.application` (*timestep*)

emodpy.defaults.ep4.dtk_post_process module

`emodpy.defaults.ep4.dtk_post_process.application` (*output_path*)

emodpy.defaults.ep4.dtk_pre_process module

`emodpy.defaults.ep4.dtk_pre_process.convert_plugin_reports` (*config_json*)

`emodpy.defaults.ep4.dtk_pre_process.application` (*json_config_path*)

Submodules

emodpy.defaults.emod_sir module

class `emodpy.defaults.emod_sir.EMODSir`

Bases: `emodpy.defaults.iemod_default.IEMODDefault`

static config (*erad_path*) → Dict

static campaign () → `emodpy.emod_campaign.EMODCampaign`

```
static demographics() → Dict
```

emodpy.defaults.iemod_default module

```
class emodpy.defaults.iemod_default.IEMODDefault
    Bases: object

    config(erad_path) → Dict
    campaign() → Dict
    demographics() → Dict
    process_simulation(simulation)
```

emodpy.generic package

Submodules

emodpy.generic.serialization module

```
emodpy.generic.serialization.enable_serialization(task:
                                                    emodpy.emod_task.EMODTask,
                                                    use_absolute_times: bool = False)
    Enable serialization either by TIME or TIMESTEP based on use_absolute_times :param task: Task to enable
    :param use_absolute_times: When true, Serialization_Type will be set to TIME, otherwise it will be set to :param
    *TIMESTEP*:
```

Returns:

```
emodpy.generic.serialization.add_serialization_timesteps(task:
                                                          emodpy.emod_task.EMODTask,
                                                          timesteps: List[int],
                                                          end_at_final:
                                                          bool = False,
                                                          use_absolute_times:
                                                          bool = False)
```

Serialize the population of this simulation at specified time steps.

If the simulation is run on multiple cores, multiple files will be created.

Parameters

- **task** (*EMODTask*) – An EMODSimulation
- **timesteps** (*List[int]*) – Array of integers representing the time steps to use
- **end_at_final** (*bool*) – False means set the simulation duration such that the last serialized_population file ends the simulation. NOTE- may not work if time step size is not 1
- **use_absolute_times** (*bool*) – False means the method will define simulation times instead of time steps see documentation on *Serialization_Type* for details

Returns None

```
emodpy.generic.serialization.load_serialized_population (task:
                                                    emodpy.emod_task.EMODTask,
                                                    population_path: str,
                                                    population_filenames:
                                                    List[str])
```

Sets simulation to load a serialized population from the filesystem

Parameters

- **task** (*EMODTask*) – An EMODSimulation
- **population_path** (*str*) – relative path from the working directory to the location of the serialized population files.
- **population_filenames** (*List[str]*) – names of files in question

Returns None

emodpy.interventions package

Submodules

emodpy.interventions.emod_empty_campaign module

```
class emodpy.interventions.emod_empty_campaign.EMODEmptyCampaign
    Bases: emodpy.defaults.iemod_default.IEMODDefault
    static campaign() → emodpy.emod_campaign.EMODCampaign
```

emodpy.reporters package

Submodules

emodpy.reporters.base module

```
class emodpy.reporters.base.BaseReporter
    Bases: object
    abstract to_dict()
    from_dict(data)
        Function allowing to initialize a Reporter instance with data. This function is called when reading a
        custom_reports.json file.

class emodpy.reporters.base.CustomReporter (name: str = None, Enabled: bool = True, Re-
                                           ports: list = <factory>, dll_file: str = None)
    Bases: emodpy.reporters.base.BaseReporter
```

This class represents a custom reporter. - name: Name that will be added to the custom_reports.json file and should match the DLL's class name - Enabled: True/False to enable/disable the reporter - Reports: Default section present in the custom_reports.json file allowing to configure the reporter - dll_file: Filename of the dll containing the reporter. This file will be searched in the dll folder specified by the user on the *EMODTask.reporters*.

```
name: str = None
Enabled: bool = True
```

```
Reports: list
dll_file: str = None
to_dict() → Dict
    Export the reporter to a dictionary. This function is called when serializing the reporter before writing the
    custom_reports.json file.
enable()
disable()

class emodpy.reporters.base.BuiltInReporter(class_name: str = None, parameters: dict
                                           = <factory>, Enabled: bool = True,
                                           Pretty_Format: bool = True)
    Bases: emodpy.reporters.base.BaseReporter
    class_name: str = None
    parameters: dict
    Enabled: bool = True
    Pretty_Format: bool = True
    to_dict()
    from_dict(data)
        Function allowing to initialize a Reporter instance with data. This function is called when reading a
        custom_reports.json file.

class emodpy.reporters.base.Reporters(relative_path='reporter_plugins')
    Bases: emodpy.emod_file.InputFilesList
    add_reporter(reporter)
    property json
    property empty
    add_dll(dll_path: str)
        Add a dll file from a path
        Parameters dll_path – Path to file
        Returns:
    add_dll_folder(dll_folder: str)
        Add all the dll files from a folder
        Parameters dll_folder – Folder to add the dll file from
        Returns:
    read_custom_reports_file(custom_reports_path, extra_classes=[]) → NoReturn
        Read from a custom reporter file
        Parameters custom_reports_path – The custom reports file to add(single file).
    set_task_config(task: EMODTask) → NoReturn
        Set task config
        Parameters task – Task to configure
        Returns:
```


gather_assets (**kwargs) → List[idmtools.assets.asset.Asset]
 Gather input files for Input File List
 Returns:

emodpy.reporters.builtin module

```
class emodpy.reporters.builtin.ReportNodeDemographics (class_name: str = 'ReportNodeDemographics',
parameters: dict = <factory>, Enabled: bool = True,
Pretty_Format: bool = True,
Stratify_By_Gender: bool = False, Age_Bins: list =
<factory>)
```

Bases: *emodpy.reporters.base.BuiltInReporter*

Stratify_By_Gender: bool = False

Age_Bins: list

class_name: str = 'ReportNodeDemographics'

```
class emodpy.reporters.builtin.ReportHumanMigrationTracking (class_name: str =
None, parameters:
dict = <factory>,
Enabled: bool =
True, Pretty_Format:
bool = True)
```

Bases: *emodpy.reporters.base.BuiltInReporter*

config (config_builder, manifest)

parameters: dict

emodpy.reporters.custom module

```
class emodpy.reporters.custom.Report_TBHIV_ByAge (name: str = 'Report_TBHIV_ByAge',
Enabled: bool = True, Reports:
list = <factory>, dll_file: str =
'lib_customreport_TBHIV_ReportByAge.dll')
```

Bases: *emodpy.reporters.base.CustomReporter*

name: str = 'Report_TBHIV_ByAge'

dll_file: str = 'lib_customreport_TBHIV_ReportByAge.dll'

add_report (max_age_yrs, min_age_yrs, start_year, stop_year, events=[])

```
class emodpy.reporters.custom.ReportPluginAgeAtInfectionHistogram(name: str
                                                                    = 'ReportPluginAgeAtInfectionHistogram',
                                                                    Enabled: bool = True,
                                                                    Reports: list = <factory>,
                                                                    dll_file: str = 'libReportAgeAtInfectionHistogram_plugin.dll',
                                                                    age_bins: list = <factory>,
                                                                    interval_years: int = <factory>)

Bases: emodpy.reporters.base.CustomReporter

name: str = 'ReportPluginAgeAtInfectionHistogram'
dll_file: str = 'libReportAgeAtInfectionHistogram_plugin.dll'
Reports: list
age_bins: list
interval_years: int

class emodpy.reporters.custom.ReportHumanMigrationTracking(name: str = 'ReportHumanMigrationTracking',
                                                            Enabled: bool = True,
                                                            Reports: list = <factory>,
                                                            dll_file: str = 'libhumanmigrationtracking.dll')

Bases: emodpy.reporters.base.CustomReporter

name: str = 'ReportHumanMigrationTracking'
dll_file: str = 'libhumanmigrationtracking.dll'
Reports: list

class emodpy.reporters.custom.ReportNodeDemographics(name: str = 'ReportNodeDemographics',
                                                        Enabled: bool = True,
                                                        Reports: list = <factory>,
                                                        dll_file: str = 'libReportNodeDemographics.dll',
                                                        age_bins: list = <factory>)
```

```

Bases: emodpy.reporters.base.CustomReporter

name: str = 'ReportNodeDemographics'

age_bins: list

dll_file: str = 'libReportNodeDemographics.dll'

from_dict(data)
    Function allowing to initialize a Reporter instance with data. This function is called when reading a
    custom_reports.json file.

add_report(age_bins)

to_dict()
    Export the reporter to a dictionary. This function is called when serializing the reporter before writing the
    custom_reports.json file.

class emodpy.reporters.custom.MalariaSummaryReport(name: str = 'MalariaSummaryReport', Enabled: bool =
    True, Reports: list = <factory>, dll_file: str = 'libmalariasummary_report_plugin.dll')

Bases: emodpy.reporters.base.CustomReporter

name: str = 'MalariaSummaryReport'

dll_file: str = 'libmalariasummary_report_plugin.dll'

add_report(age_bins=None, duration_days=365, event_trigger_list=None, individual_property_filter="",
    infectiousness_bins=None, max_number_reports=10000, parasitemia_bins=None, report_description="", reporting_interval=30, start_day=0)

class emodpy.reporters.custom.ReportEventCounter(name: str = 'ReportEventCounter', Enabled: bool = True, Reports: list
    = <factory>, dll_file: str = 'libreporteventcounter.dll')

Bases: emodpy.reporters.base.CustomReporter

name: str = 'ReportEventCounter'

dll_file: str = 'libreporteventcounter.dll'

add_report(duration_days=10000, event_trigger_list=None, nodeset_config=None, report_description="", start_day=0)

class emodpy.reporters.custom.ReportMalariaFiltered(name: str = 'ReportMalariaFiltered', Enabled: bool =
    True, Reports: list = <factory>, dll_file: str = 'libReportMalariaFiltered.dll')

Bases: emodpy.reporters.base.CustomReporter

name: str = 'ReportMalariaFiltered'

dll_file: str = 'libReportMalariaFiltered.dll'

add_report(end_day=0, node_ids_of_interest=None, report_file_name='ReportMalariaFiltered.json',
    start_day=0)

class emodpy.reporters.custom.MalariaImmunityReport(name: str = 'MalariaImmunityReport', Enabled: bool =
    True, Reports: list = <factory>, dll_file: str = 'libmalariainmunity_report_plugin.dll')

```

```
Bases: emodpy.reporters.base.CustomReporter

name: str = 'MalariaImmunityReport'

dll_file: str = 'libmalariaimmunity_report_plugin.dll'

add_report (pretty_format=1, age_bins=[], start_day=0, duration_days=10000, nodeset_config={'class': 'NodeSetAll'}, event_trigger_list=['EveryUpdate'], max_number_reports=15, report_description="", reporting_interval=15)

class emodpy.reporters.custom.MalariaSurveyJSONAnalyzer (name: str = 'Malaria-SurveyJSONAnalyzer', Enabled: bool = True, Reports: list = <factory>, dll_file: str = 'libmalariaurveyJSON_analyzer_plugin.dll')

Bases: emodpy.reporters.base.CustomReporter

name: str = 'MalariaSurveyJSONAnalyzer'

dll_file: str = 'libmalariaurveyJSON_analyzer_plugin.dll'

add_report (pretty_format=1, start_day=0, duration_days=10000, nodeset_config={'class': 'NodeSetAll'}, event_trigger_list=['NewClinicalCase'], max_number_reports=15, report_description='Day0', reporting_interval=73)

class emodpy.reporters.custom.MalariaTransmissionReport (name: str = 'MalariaTransmissionReport', Enabled: bool = True, Reports: list = <factory>, dll_file: str = 'libReportMalariaTransmissions.dll')

Bases: emodpy.reporters.base.CustomReporter

name: str = 'MalariaTransmissionReport'

dll_file: str = 'libReportMalariaTransmissions.dll'

add_report (pretty_format=1, start_day=0, duration_days=365, nodeset_config={'class': 'NodeSetAll'}, event_trigger_list=['NewInfection'], report_description="")
```

10.1.2 Submodules

emodpy.bamboo module

```
emodpy.bamboo.get_model_files (plan, manifest, scheduled_builds_only=True)
```

emodpy.bamboo_api_utils module

`emodpy.bamboo_api_utils.bamboo_connection()`

class `emodpy.bamboo_api_utils.BambooConnection`

Bases: `object`

Bamboo API config and basic functionality/connectivity wrapper.

Automatically probes the most likely endpoint locations (with and without https, with and without port numbers).

Important functions:

- `login`: logs into the bamboo api, caches the login token so you don't have to pass creds for every req. in a session
- `get_bamboo_api_url`: translate a relative API URL into a fully qualified URL
- `normalize_url`: detect whether a URL is relative or not, translate relative URLs to fully qualified ones
- `make_get_request`: makes a request to the specified API url, adds some convenient error and login handling
- `download_file`: downloads a file from the specified artifacts url to a location on disk

property `server`

`str`: Keeps track of a single instance of the server base url. (e.g. <http://idm-bamboo:8085>)

property `session_cookie`

`str`: Automatically load and instance the login session cookie jar.

get_server_url (*ssl: bool = False, useport: bool = False*) → `str`

Get a particular variant of the server url w/ or w/o ssl and port (e.g. False/False -> <http://idm-bamboo>)

Parameters

- **ssl** (*bool*) – whether to use ssl
- **useport** (*bool*) – whether to use the port

Returns endpoint url

Return type `str`

find_server () → `str`

Explore all possible server urls, return the first one found to exist.

Returns server url

Return type `str`

url_exists (*url: str*) → `bool`

Try a simple get request given an endpoint url, return whether it was successful (code 200).

Parameters **url** (*str*) – url to issue a test request to

Returns whether or not a request to the url succeeds (w/ status 200)

Return type `bool`

property `session_cookie_filename`

File where bamboo session cookie is stored.

Returns fully qualified file path of session cookie file

Return type `str`

load_session_cookie () → <module 'requests.cookies' from
'/home/docs/checkouts/readthedocs.org/user_builds/institute-
for-disease-modeling-emodpy/envs/v1.11.0/lib/python3.7/site-
packages/requests/cookies.py'>

Load api login session cookies from disk.

Returns session cookie jar

Return type requests.cookies

write_session_cookie (cookies: <module 'requests.cookies' from
'/home/docs/checkouts/readthedocs.org/user_builds/institute-
for-disease-modeling-emodpy/envs/v1.11.0/lib/python3.7/site-
packages/requests/cookies.py'>)

Write post-login cookies for session to disk.

get_bamboo_url (relative_url: str) → str

Add bamboo server, port, and protocol to bamboo url.

Parameters **relative_url** (str) – relative url (artifact link or api url)

Returns fully qualified url

Return type str

get_bamboo_api_url (relative_url: str, json: bool = False, params: dict = {}) → str

Get fully qualified bamboo api url from a relative url w/ given json mode and appending all parameters.

Parameters

- **relative_url** (str) – api url (e.g. project/<project-key>)
- **json** (bool) – whether to get results in json format (otherwise, default is xml)
- **params** (dict) – name/value dictionary of query parameters

Returns fully qualified url that a request can be issued against

Return type str

save_credentials (username: str, password: str)

Save bamboo api login credentials using keyring.

Parameters

- **username** (str) – bamboo api login username (e.g. somebody@idmod.org)
- **password** (str) – bamboo api login password

ensure_logged_in ()

Check if a login session exists using saved cookies, if not login using keyring stored creds.

login_session_exists () → bool

Test whether an existing session cookie exists and an active login session exists.

Returns whether an active login session exists

Return type bool

login (username: str = None, password=None) → bool

Login to the bamboo api. If username or password are not provided, use stored credentials from keyring.

Parameters

- **username** (str) – bamboo api login username (e.g. somebody@idmod.org)
- **password** (str) – bamboo api login password

Returns success/failure

Return type `bool`

normalize_url (*url*: `str`) → `str`

Determine whether a url is relative or fully qualified, translate relative urls to fully qualified versions.

Parameters **url** (`str`) – relative or fully qualified url

Returns fully qualified url

Return type `str`

make_get_request (*url*: `str`, *retries*: `int` = 3) → `requests.models.Response`

Make a get request against the bamboo server.

Parameters **url** (`str`) – relative or fully qualified url

Returns request object returned from `requests.get()`

Return type `requests.Response`

make_api_get_request (*relative_url*: `str`, *json*: `bool` = `False`, *params*: `dict` = `{}`) → `requests.models.Response`

Translate relative api url to the fully qualified bamboo api url, make a get request against it.

Parameters

- **relative_url** (`str`) – url relative to the bamboo api endpoint (e.g. 'result/MYPROJ-MYPLAN/123') to make the request against
- **json** (`bool`) – whether to return results in json
- **params** (`dict`) – name/value dictionary of additional parameters to pass

Returns request object returned from `requests.get()`

Return type `requests.Response`

download_file (*url*: `str`, *destination*: `str`) → `list`

Download a specific artifact file (from the full artifact url provided) to disk.

Streams the download to avoid common 'gotchas' with downloading via http.

Parameters

- **url** (`str`) – url to download
- **destination** (`str`) – destination path or filename where the artifact is to be downloaded to

Returns local filename of file that has been downloaded

Return type (`str`)

class `emodpy.bamboo_api_utils.BuildInfo`

Bases: `object`

A collection of methods for getting data on build results.

classmethod **build_passed** (*plan_key*: `str`, *build_num*: `int`) → `bool`

Determine whether a given build succeeded or not.

Parameters

- **plan_key** (`str`) – bamboo plan key (including project key)
- **build_num** (`int`) – build number to retrieve results for

Returns whether the build succeeded

Return type `bool`

static `successful_build_result` (*result*) → `bool`

Analyze a build result json object and determine if it corresponds to a successful build

Parameters `result` – json build result

Returns whether the build was successful

Return type `bool`

static `get_build_info` (*plan_key*: `str`, *index*: `int`)

Retrieve the build info in json format for a given build plan with a relative index (0=latest)

Parameters

- `plan_key` (`str`) – bamboo plan key (including project key)
- `index` (`int`) – index of build to retrieve info for (0=latest, 1=2nd most recent, etc.)

Returns build info results json

classmethod `get_latest_successful_build` (*plan_key*: `str`, *scheduled_only*: `bool` = `True`,
max_iterations: `int` = `100`)

Find the latest successful build within the last max_iterations builds for a given plan.

Parameters

- `plan_key` (`str`) – bamboo plan key (including project key)
- `scheduled_only` (`bool`) – only count automatically run scheduled or triggered builds as successful
- `max_iterations` (`int`) – maximum number of older builds to look through

Returns

tuple containing: `build_num` (`str`): build number of last successful build `build_info`: json data structure of build info for that build

Return type (`tuple`)

classmethod `get_latest_build` (*plan_key*: `str`)

Get the build info for the most recently run build for a given plan.

Parameters `plan_key` (`str`) – bamboo plan key (including project key)

Returns

tuple containing: `build_num` (`str`): build number of last successful build `build_info`: json data structure of build info for that build

Return type (`tuple`)

class `emodpy.bamboo_api_utils.BuildArtifacts`

Bases: `object`

A collection of methods for finding and interacting with build artifacts.

`ERADICATION_EXE` = `'Eradication.exe'`

`SCHEMA_JSON` = `'schema.json'`

`REPORTER_PLUGINS` = `'Reporter-Plugins'`

classmethod `find_artifacts_by_name` (*plan_key*: `str`, *build_num*: `int`, *artifact*: `str`) → `list`

Find all urls for files of an artifact of a given name for a specific build.

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for
- **artifact** (*str*) – artifact name/id

Returns list of artifact urls that can be downloaded

Return type (*list of str*)

classmethod find_artifacts (*plan_key: str, build_num: int, artifact_list: list*) → *list*

Find all urls for files of a list of artifacts for a specific build.

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for
- **artifact_list** (*list*) – list of artifact names/ids

Returns list of artifact urls that can be downloaded

Return type (*list of str*)

classmethod find_build_essential_artifacts (*plan_key: str, build_num: int*) → *list*

Find all ‘build essential’ artifact urls (Eradication, schema, reporters) for a specific build

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for

Returns list of artifact urls that can be downloaded

Return type (*list of str*)

classmethod find_all_artifacts (*plan_key: str, build_num: int*) → *list*

Find all artifact urls (Eradication, schema, reporters) for a specific build

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for

Returns list of artifact urls that can be downloaded

Return type (*list of str*)

classmethod find_all_artifact_names (*plan_key: str, build_num: int*) → *list*

Find all artifact names (e.g. ‘Eradication.exe’) for a specific build (can be plugged into find_artifacts() to get actual urls that can be downloaded)

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for

Returns list of artifact names that can be downloaded

Return type (*list of str*)

classmethod `download_artifact_to_file` (*plan_key*: *str*, *build_num*: *int*, *artifact*, *destination*: *str*) → *list*

Download files found for a named artifact to the filepath provided.

Additional files found will be downloaded as `_2`, `_3`, `_4`, etc. For example, if there are 3 files for 'Eradication.exe' the first will be `Eradication.exe`, the second will be `Eradication_2.exe`, the third `Eradication_3.exe`.

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for
- **artifact** (*list* or *str*) – list (or string) of artifact names
- **destination** (*str*) – destination path or filename where the artifact is to be downloaded to

Returns list of local filenames of files that have been downloaded

Return type (*list* of *str*)

classmethod `download_artifacts_to_path` (*plan_key*: *str*, *build_num*: *int*, *artifact*, *destination_path*: *str*) → *list*

Download all the files for a given artifact and build to a specific folder, using their original filenames.

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **build_num** (*int*) – build number to retrieve artifact urls for
- **artifact** (*list* or *str*) – list (or string) of artifact names
- **destination_path** (*str*) – path to destination folder where files are to be downloaded

Returns list of local filenames of files that have been downloaded

Return type (*list* of *str*)

classmethod `download_latest_good_Eradication_exe` (*plan_key*: *str*, *destination*: *str*) → *str*

Find the latest successful build for a specified plan, download the `Eradication.exe` artifact to a specified path.

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **destination** (*str*) – destination path or filename where the artifact is to be downloaded to

Returns build number of build that was found and had its artifact downloaded

Return type *str*

classmethod `download_latest_good_schema_json` (*plan_key*: *str*, *destination*: *str*) → *str*

Find the latest successful build for a specified plan, download the `schema.json` artifact to a specified path.

Parameters

- **plan_key** (*str*) – bamboo plan key (including project key)
- **destination** (*str*) – destination path or filename where the artifact is to be downloaded to

Returns build number of build that was found and had its artifact downloaded

Return type `str`

classmethod `download_eradication_exe` (*plan_key*: `str`, *build_num*: `str`, *destination*: `str`) → `str`
 Download Eradication.exe artifact from a specific build.

Parameters

- **plan_key** (`str`) – bamboo plan key (including project key)
- **build_num** (`str`) – build number to download from
- **destination** (`str`) – destination path or filename where the artifact is to be downloaded to

classmethod `make_exe_executable` (*file_path*: `str`)
 On linux change the file permissions on a binary to make it executable

Parameters **file_path** (`str`) – binary file to mark as executable

classmethod `download_schema_json` (*plan_key*: `str`, *build_num*: `str`, *destination*: `str`) → `str`
 Download schema.json artifact from a specific build.

Parameters

- **plan_key** (`str`) – bamboo plan key (including project key)
- **build_num** (`str`) – build number to download from
- **destination** (`str`) – destination path or filename where the artifact is to be downloaded to

classmethod `download_from_bamboo_url` (*url*: `str`, *destination*: `str`)
 Download Eradication.exe/Eradication directly from bamboo url Assume you already done login

Parameters

- **url** –
- **destination** (`str`) – destination path or filename where the artifact is to be downloaded to

Returns local file path that have been downloaded

Return type `str`

class `emodpy.bamboo_api_utils.BuildPlans`

Bases: `object`

Collection of methods for getting information on build plans.

static `export_spec` (*plan_key*: `str`) → `str`
 Export a specific build plan to java specs.

Parameters **plan_key** (`str`) – bamboo plan key (including project key)

Returns full text of the .java file for the plan spec, if the plan was found (empty string if not)

Return type `str`

static `get_plans_for_project` (*project_key*: `str`) → `list`
 Return a list of all the build plans for every plan in the project.

Parameters **project_key** (`str`) – bamboo project key

Returns list of plan keys for each plan that was found in the project

Return type (`list` of `str`)

`emodpy.bamboo_api_utils.login(username=None, password=None)`
Pass through to `BambooConnection.login()`

`emodpy.bamboo_api_utils.save_credentials(username, password)`
Pass through to `BambooConnection.save_credentials()`

emodpy.collections_utils module

`emodpy.collections_utils.cut_iterable_to(obj: Iterable, to: int) → Tuple[Union[List, Mapping], int]`

Cut an iterable to a certain length.

Parameters

- **obj** – The iterable to cut.
- **to** – The number of elements to return.

Returns A list or dictionary (depending on the type of object) of elements and the remaining elements in the original list or dictionary.

`emodpy.collections_utils.deep_get(d, key, default: callable = None, getter: callable = None, sep: str = '.')`

`emodpy.collections_utils.deep_set(d, key, value, default: callable = None, getter: callable = None, setter: callable = None, sep: str = '.')`

`emodpy.collections_utils.deep_del(d: dict, key, getter: callable = None, deleter: callable = None, sep: str = '.')`

emodpy.emod_campaign module

class `emodpy.emod_campaign.EMODCampaign(name='Campaign', events=None, use_defaults=True, **kwargs)`

Bases: `object`

Class representing an EMOD Campaign. It contains: - **events**: a list of events for the given campaign - **name**: campaign name - **use_defaults**: EMOD flag to use defaults for unspecified parameters - **extra_parameters**: parameters set by the user that will be added to the campaign JSON

property json

Property to transform the object in JSON

static `load_from_file(filename: str) → object`
Load a campaign from a JSON file.

Parameters **filename** – Path to the campaign file

Returns: an initialized *EMODCampaign* instance

static `load_from_dict(data: Dict) → object`
Create a campaign object from a dict. :param data: The dictionary containing the data

Returns: an initialized *EMODCampaign* instance

clear() → NoReturn
Clear all campaign events

get_events_at (*timestep: int*) → List[Dict]

Get a list of events happening at the specified timestep. Does not take into account recurrence and only consider start timestep. :param timestep: selected timestep

Returns: list of events

get_events_with_name (*name: str*) → List[Dict]

Get a list of events with the given name. This search is based on the *Event_Name* key of events. :param name: Name of the events

Returns: list of events

add_event (*event: Dict*) → NoReturn

Add the given event to the campaign event. :param event: The event to add

add_events (*events: List[Dict]*) → NoReturn

Add a list of events to the campaign events. :param events: List of events to add

emodpy.emod_file module

class emodpy.emod_file.InputFilesList (*relative_path=None*)

Bases: idmtools.assets.asset_collection.AssetCollection

abstract set_task_config (*simulation*)

gather_assets () → List[idmtools.assets.asset.Asset]

Gather input files for Input File List

Returns:

class emodpy.emod_file.MigrationTypes (*value*)

Bases: enum.Enum

An enumeration.

LOCAL = 'Local'

AIR = 'Air'

FAMILY = 'Family'

REGIONAL = 'Regional'

SEA = 'Sea'

class emodpy.emod_file.MigrationModel (*value*)

Bases: enum.Enum

An enumeration.

NO_MIGRATION = 'NO_MIGRATION'

FIXED_RATE_MIGRATION = 'FIXED_RATE_MIGRATION'

class emodpy.emod_file.MigrationPattern (*value*)

Bases: enum.Enum

An enumeration.

RANDOM_WALK_DIFFUSION = 'RANDOM_WALK_DIFFUSION'

SINGLE_ROUND_TRIPS = 'SINGLE_ROUND_TRIPS'

WAYPOINTS_HOME = 'WAYPOINTS_HOME'

```
class emodpy.emod_file.MigrationFiles (relative_path=None)
    Bases: emodpy.emod_file.InputFilesList

    enable_migration ()
        Enables migration and sets the pattern if defined. If there are not other other parameters, it also set Enable_Migration_Heterogeneity to 0

    update_migration_pattern (migration_pattern:          emodpy.emod_file.MigrationPattern,
                             **kwargs) → NoReturn
        Update migration pattern

        Parameters

        • migration_pattern – Migration Pattern to use

        • **kwargs –

        Returns NoReturn

    add_migration_from_file (migration_type: emodpy.emod_file.MigrationTypes, file_path: str,
                             multiplier: float = 1)
        Add migration info from a file

        Parameters

        • migration_type – Type of migration

        • file_path – Path to file

        • multiplier – Multiplier

        Returns:

    set_task_config (task: EMODTask)
        Update the task with the migration configuration

        Parameters task – Task to update

        Returns:

    gather_assets ()
        Gather assets for Migration files. Called by EMODTask Returns:

    set_all_persisted ()
        Set all migration assets as persisted

        Returns:

    merge_with (mf: emodpy.emod_file.MigrationFiles, left_precedence: bool = True) → NoReturn
        Merge migration file with other Migration file

        Parameters

        • mf – Other migration file to merge with

        • left_precedence – Does the current object have precedence or the other object?

        Returns:

    read_config_file (config_path, asset_path)
        Try to recreate the migration based on a given config file and an asset path :param config_path: path to the
        config :param asset_path: path containing the assets

class emodpy.emod_file.DemographicsFiles (relative_path=None)
    Bases: emodpy.emod_file.InputFilesList
```

set_task_config (*task: EMODTask, extend: bool = False*)

Set the simulation level config. If extend is true, the demographics files are appended to the list :param task: :param extend:

Returns:

add_demographics_from_file (*absolute_path: str, filename: Optional[str] = None*)

Add demographics from a file

Parameters

- **absolute_path** – Path to file
- **filename** – Optional filename. If not provided, the file name of source file will be used

Returns:

add_demographics_from_dict (*content: Dict, filename: str*)

Add demographics from a dictionary object

Parameters

- **content** – Dictionary Content
- **filename** – Filename to call demographics file

Returns:

class emodpy.emod_file.ClimateFileType (*value*)

Bases: `enum.Enum`

An enumeration.

AIR_TEMPERATURE = 'Air_Temperature'

LAND_TEMPERATURE = 'Land_Temperature'

RELATIVE_HUMIDITY = 'Relative_Humidity'

RAINFALL = 'Rainfall'

class emodpy.emod_file.ClimateModel (*value*)

Bases: `enum.Enum`

An enumeration.

CLIMATE_OFF = 'CLIMATE_OFF'

CLIMATE_CONSTANT = 'CLIMATE_CONSTANT'

CLIMATE_KOPPEN = 'CLIMATE_KOPPEN'

CLIMATE_BY_DATA = 'CLIMATE_BY_DATA'

class emodpy.emod_file.ClimateFiles

Bases: `emodpy.emod_file.InputFilesList`

set_task_config (*task: EMODTask*)

Set the task Config. Set all the correct files for the climate.

Parameters **task** – Task to config

add_climate_files (*file_type, file_path*)

gather_assets ()

Gather assets for Climate files. Called by EMODTask

set_climate_constant (*Base_Air_Temperature, Base_Rainfall, Base_Land_Temperature=None, Base_Relative_Humidity=None*)

```
read_config_file (config_path, asset_path)
```

Try to recreate the climate based on a given config file and an asset path :param config_path: path to the config :param asset_path: path containing the assets

emodpy.emod_task module

```
emodpy.emod_task.logger = <Logger emodpy.emod_task (DEBUG)>
```

Note that these 3 functions could be member functions of EMODTask but Python modules are already pretty good at being 'static classes'.

```
emodpy.emod_task.add_ep4_from_path (task, ep4_path)
```

```
emodpy.emod_task.default_ep4_fn (task)
```

```
class emodpy.emod_task.EMODTask (command: Union[str, idm-  
tools.entities.command_line.CommandLine] =  
<property object>, platform_requirements:  
Set[idmtools.entities.platform_requirements.PlatformRequirements]  
= <factory>, _ITask__pre_creation_hooks:  
List[Callable[[Union[Simulation, IWorkflowItem], IPlatform],  
NoReturn]] = <factory>, _ITask__post_creation_hooks:  
List[Callable[[Union[Simulation, IWorkflowItem], IPlatform],  
NoReturn]] = <factory>, common_assets:  
idmtools.assets.asset_collection.AssetCollection  
= <factory>, transient_assets: idm-  
tools.assets.asset_collection.AssetCollection = <fac-  
tory>, eradication_path: str = None, demographics:  
emodpy.emod_file.DemographicsFiles = <factory>, mi-  
grations: emodpy.emod_file.MigrationFiles = <factory>,  
reporters: emodpy.reporters.base.Reporters = <factory>,  
climate: emodpy.emod_file.ClimateFiles = <factory>, config:  
dict = <factory>, config_file_name: str = 'config.json',  
campaign: emodpy.emod_campaign.EMODCampaign  
= <factory>, simulation_demographics:  
emodpy.emod_file.DemographicsFiles = <factory>, sim-  
ulation_migrations: emodpy.emod_file.MigrationFiles =  
<factory>, use_embedded_python: bool = True, is_linux: bool  
= False, implicit_configs: list = <factory>)
```

Bases: idmtools.entities.itask.ITask

EMODTask allows easy running and configuration of EMOD Experiments and Simulations

eradication_path: `str = None`

Eradication path. Can also be set through config file

demographics: `emodpy.emod_file.DemographicsFiles`

Common Demographics

migrations: `emodpy.emod_file.MigrationFiles`

Common Migrations

reporters: `emodpy.reporters.base.Reporters`

Common Reports

climate: `emodpy.emod_file.ClimateFiles`

Common Climate


```

config: dict
    Represents config.json

config_file_name: str = 'config.json'

campaign: emodpy.emod_campaign.EMODCampaign
    Campaign configuration

simulation_demographics: emodpy.emod_file.DemographicsFiles
    Simulation level demographics such as overlays

simulation_migrations: emodpy.emod_file.MigrationFiles
    Simulation level migrations

use_embedded_python: bool = True
    Add -python-script-path to command line

is_linux: bool = False

implicit_configs: list

create_campaign_from_callback (builder)

create_demog_from_callback (builder, from_sweep=False)

classmethod from_default2 (eradication_path, schema_path, param_custom_cb, con-
                           fig_path='new_config.json', campaign_builder=None,
                           ep4_custom_cb=<function default_ep4_fn>, de-
                           mog_builder=None, plugin_report=None, serial_pop_files=None,
                           **kwargs) → emodpy.emod_task.EMODTask

```

Create a task from emod-api Defaults

Parameters

- **config_path** – /path/to/new_config.json
- **eradication_path** – Path to Eradication binary
- **param_custom_cb** – Function that sets parameters for config
- **ep4_custom_cb** – Function that sets EP4 assets
- **plugin_report** – Custom reports file

Returns *EMODTask*

```

classmethod from_files (eradication_path=None, config_path=None, campaign_path=None,
                        demographics_paths=None, ep4_path=None, cus-
                        tom_reports_path=None, asset_path=None, **kwargs)

```

Load custom EMOD files when creating *EMODTask*.

Parameters

- **asset_path** – If an asset path is passed, the climate, dlls, and migrations will be searched there
- **eradication_path** – The eradication.exe path.
- **config_path** – The custom configuration file.
- **campaign_path** – The custom campaign file.
- **demographics_paths** – The custom demographics files (single file or a list).
- **custom_reports_path** – Custom reports file

Returns: An initialized experiment

load_files (*config_path=None, campaign_path=None, custom_reports_path=None, demographics_paths=None, asset_path=None*) → NoReturn
Load files in the experiment/base_simulation.

Parameters

- **asset_path** – Path to find assets
- **config_path** – Configuration file path
- **campaign_path** – Campaign file path
- **demographics_paths** – Demographics file path
- **custom_reports_path** – Path for the custom reports file

pre_creation (*parent: Union[idmtools.entities.simulation.Simulation, tools.entities.iworkflow_item.IWorkflowItem], platform: tools.entities.iplatform.IPlatform*) → *idm-idm-*

Call before a task is executed. This ensures our configuration is properly done

set_command_line () → NoReturn
Builds and sets the command line object

Returns:

gather_common_assets () → *idmtools.assets.asset_collection.AssetCollection*
Gather Experiment Level Assets Returns:

gather_transient_assets () → *idmtools.assets.asset_collection.AssetCollection*
Gather assets that are per simulation Returns:

copy_simulation (*base_simulation: idmtools.entities.simulation.Simulation*) → *idm-tools.entities.simulation.Simulation*
Called when making copies of a simulation.

Here we deep copy parts of the simulation to ensure we don't accidentally update objects :param
base_simulation: Base Simulation

Returns:

set_parameter (*name: str, value: any*) → dict
Set a value in the EMOD config.json file. This will be deprecated in the future in favour of
emod_api.config.

Parameters

- **name** – Name of parameter to set
- **value** – Value to set

Returns Tags to set

static set_parameter_sweep_callback (*simulation: idmtools.entities.simulation.Simulation, param: str, value: Any*) → Dict[str, Any]

Convenience callback for sweeps

Parameters

- **simulation** – Simulation we are updating
- **param** – Parameter
- **value** – Value

Returns Tags to set on simulation

classmethod `set_parameter_partial` (*parameter: str*)

Convenience callback for sweeps

Parameters `parameter` – Parameter to set

Returns:

get_parameter (*name: str, default: Optional[Any] = None*)

Get a parameter in the simulation.

Parameters

- **name** – The name of the parameter.
- **default** – Optional, the default value.

Returns The value of the parameter.

update_parameters (*params*)

Bulk update the configuration parameter values. This will be deprecated in the future in favour of `emod_api.config`.

Parameters `params` – A dictionary with new values.

Returns None

reload_from_simulation (*simulation: idmtools.entities.simulation.Simulation*)

Optional hook that is called when loading simulations from a platform

class `emodpy.emod_task.EMODTaskSpecification`

Bases: `idmtools.registry.task_specification.TaskSpecification`

get (*configuration: dict*) → `emodpy.emod_task.EMODTask`

Return an EMODTask object using provided configuration :param configuration: Configuration for Task

Returns EMODTask for configuration

get_description () → `str`

Defines a description of the plugin

Returns Plugin description

get_example_urls () → `List[str]`

Return a list of examples. This is used by the examples cli command to allow users to quickly load examples locally

Returns List of urls to examples

get_type () → `Type[emodpy.emod_task.EMODTask]`

Returns the Task type defined by specification

Returns:

get_version () → `str`

Return the version string for EMODTask. This should be the module version so return that

Returns Version

emodpy.utils module

```
class emodpy.utils.EradicationPlatformExtension(value)
    Bases: enum.Enum

    An enumeration.

    LINUX = ''

    Windows = '.exe'

class emodpy.utils.EradicationBambooBuilds(value)
    Bases: enum.Enum

    An enumeration.

    GENERIC_LINUX = 'DTKGENCI-SCONSLNXGEN'
    GENERIC_WIN = 'DTKGENCI-SCONSWINGEN'
    GENERIC = 'DTKGENCI-SCONSLNXGEN'
    TBHIV_LINUX = 'DTKTBHIVCI-SCONSRELLNXTBHIV'
    TBHIV_WIN = 'DTKTBHIVCI-SCONSWINTBHIV'
    TBHIV = 'DTKTBHIVCI-SCONSRELLNXTBHIV'
    MALARIA_LINUX = 'DTKMALCI-SCONSLNXMAL'
    MALARIA_WIN = 'DTKMALCI-SCONSWINMAL'
    MALARIA = 'DTKMALCI-SCONSLNXMAL'
    HIV_LINUX = 'DTKHIVCI-SCONSRELLNXHIV'
    HIV_WIN = 'DTKHIVCI-RELWINHIV'
    HIV = 'DTKHIVCI-SCONSRELLNXHIV'
    DENGUE_LINUX = 'DTKDENGCI-SCONSRELLNX'
    DENGUE_WIN = 'DTKDENGCI-VSRELWINALL'
    DENGUE = 'DTKDENGCI-SCONSRELLNX'
    FP_LINUX = 'DTKFPCI-SCONSRELLNX'
    FP_WIN = 'DTKFPCI-SCONSWINFP'
    FP = 'DTKFPCI-SCONSRELLNX'
    TYPHOID_LINUX = 'DTKTYPHCI-SCONSRELLNX'
    TYPHOID_WIN = 'DTKTYPHCI-SCONSWINENV'
    TYPHOID = 'DTKTYPHCI-SCONSRELLNX'
    EMOD_RELEASE = 'EMODREL-SCONSRELLNX'
    RELEASE = 'DTKREL-SCONSRELLNX'

class emodpy.utils.BambooArtifact(value)
    Bases: enum.Flag

    An enumeration.

    ERADICATION = 1
    SCHEMA = 2
```

PLUGINS = 4

ALL = 7

```
emodpy.utils.get_github_eradication_url(version: str, extension:
                                         emodpy.utils.EradicationPlatformExtension =
                                         <EradicationPlatformExtension.LINUX: ">") →
                                         str
```

Get the github eradication url for specified release

Parameters

- **version** – Release to fetch
- **extension** – Optional extensions. Defaults to Linux(None)

Returns Url of eradication release

```
emodpy.utils.save_bamboo_credentials(username, password)
Save bamboo api login credentials using keyring.
```

Parameters

- **username** (*str*) – bamboo api login username (e.g. `somebody@idmod.org`)
- **password** (*str*) – bamboo api login password

```
emodpy.utils.bamboo_api_login()
Automatically login to bamboo, prompt for credentials if none are cached or there's no login session.
```

```
emodpy.utils.download_bamboo_artifacts(plan_key: str, build_num: str = None, sched-
                                         uled_builds_only: bool = True, artifact:
                                         emodpy.utils.BambooArtifact = <BambooArti-
                                         fact.ERADICATION: 1>, out_path: str = None) →
                                         list
```

Downloads artifact(s) for a DTK Bamboo build plan to the specified path

Parameters

- **plan_key** (*str*) –
- **build_num** (*str*) –
- **scheduled_builds_only** (*bool*) –
- **artifact** (*BambooArtifact*) –
- **out_path** (*str*) – Output path to save file (default to current directory)

Returns Returns list of downloaded files on filesystem

```
emodpy.utils.download_latest_bamboo(plan: emodpy.utils.EradicationBambooBuilds, sched-
                                         uled_builds_only: bool = True, out_path: str = None)
                                         → str
```

Downloads the Eradication binary for the latest successful build for a Bamboo Plan to specified path. Exists for backward compatibility, just a pass-thru to `download_latest_eradication()`.

Parameters

- **plan** – Bamboo Plan key. for supported build
- **out_path** – Output path to save file (default to current directory)

Returns Returns local filename of downloaded file

`emodpy.utils.download_latest_eradication` (*plan:* `emodpy.utils.EradicationBambooBuilds`,
scheduled_builds_only: `bool = True`, *out_path:* `str = None`) \rightarrow `str`

Downloads the Eradication binary for the latest successful build for a Bamboo Plan to specified path.

Parameters

- **plan** – Bamboo Plan key. for supported build
- **out_path** – Output path to save file (default to current directory)

Returns Returns local filename of downloaded file

`emodpy.utils.download_latest_reporters` (*plan:* `emodpy.utils.EradicationBambooBuilds`,
scheduled_builds_only: `bool = True`, *out_path:* `str = None`) \rightarrow `list`

Downloads the reporter plugins for the latest successful build for a Bamboo Plan to specified path.

Parameters

- **plan** – Bamboo Plan key. for supported build
- **out_path** – Output path to save file (default to current directory)

Returns Returns list of local filenames of downloaded files

`emodpy.utils.download_latest_schema` (*plan:* `emodpy.utils.EradicationBambooBuilds`, *scheduled_builds_only:* `bool = True`, *out_path:* `str = None`)
 \rightarrow `str`

Downloads the schema.json for the latest successful build for a Bamboo Plan to specified path.

Parameters

- **plan** – Bamboo Plan key. for supported build
- **out_path** – Output path to save file (default to current directory)

Returns Returns local filename of downloaded file

`emodpy.utils.download_from_url` (*url*, *out_path:* `str = None`) \rightarrow `str`

`emodpy.utils.download_eradication` (*url:* `str`, *cache_path:* `str = None`, *spinner=None*)

Downloads Eradication binary

Useful for downloading binaries from Bamboo or Github

Parameters

- **url** – Url to binary
- **cache_path** – Optional output directory
- **spinner** – Spinner object

Returns Full path to output file

GLOSSARY

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

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

CHANGELOG

12.1 1.1.0

12.1.1 Additional Changes

- #0001 - Fix emod tests
- #0024 - Support of Kurt's workflows in idmttools
- #0070 - Custom_reporters.json does not get automatically added?

12.1.2 Bugs

- #0011 - task with simulation level demographics not work
- #0012 - How to add custom simulation tags from task?
- #0040 - examples- emod_model- serialization- 03_parameter_reload getting wrong campaign
- #0042 - We should make EMODSir default work with eradication
- #0043 - Wired campaign format error
- #0044 - Examples- create_sims_pre_and_post_process.py should import config_update_parameters correctly
- #0055 - Creation of campaign.json will fail in AC in COMPS - cannot overwrite AC files
- #0059 - EmodTask.pre_post_process should be renamed
- #0069 - Fix create_serialized_sims_reload and create_sims_from_default_run_analyzer examples
- #0072 - custom_reports.json - not all of them have "enabled", but code assumes they do
- #0073 - Climate_Model should be set to whatever it is set in config.json when from_files is used.
- #0075 - custom_reports: when reading my ReportNodeDemographics report, one of the parameters is not read in

12.1.3 Developer/Test

- #0015 - Add changelog script
- #0039 - Rename repo to emodpy

12.1.4 Documentation

- #0007 - Automate docs
- #0008 - Document a simple example of running DTK in idmtools
- #0045 - examples- emod_model- post_process_command_task- needs some mortality
- #0061 - make docs failed

12.1.5 Feature Request

- #0028 - We should implement reload_from_simulation() for EMODTask
- #0030 - Support of a list of campaign events
- #0032 - Utility function to create a campaign event
- #0033 - Support of reporters for EMOD
- #0034 - Support of schema defaults
- #0063 - Support of climate files

12.1.6 Models

- #0014 - Need to add --python-script-path option to EMODTask arguments
- #0029 - Modifications of base config parameters

12.1.7 Platforms

- #0021 - SSMT Build as part of GithubActions

12.1.8 User Experience

- #0037 - Add examples url to plugins specifications and then each plugin if they have examples
- #0049 - Add system tags for EMODTask

12.2 1.2.0

12.2.1 Additional Changes

- #0091 - Eradication.exe can't consume emodpy_covid installed in a virtual environment (Windows)

12.2.2 Bugs

- #0054 - examplescreate_sims_eradication_from_github_url.py failed
- #0098 - Few migration bugs

12.2.3 Documentation

- #0060 - Help with repro: dtk_pre_process executed twice before simulation attempted

12.2.4 Feature Request

- #0036 - Creation of migration file from code
- #0090 - We should have utils to download Eradication by giving url

12.2.5 User Experience

- #0047 - Directly use Eradication.exe from bamboo url seems not working
- #0068 - emodpyutils.py needs more robust solution for getting Eradication.exe paths

PYTHON MODULE INDEX

e

- `emodpy`, [23](#)
- `emodpy.analyzers`, [23](#)
- `emodpy.analyzers.adult_vectors_analyzer`,
[23](#)
- `emodpy.analyzers.population_analyzer`,
[24](#)
- `emodpy.analyzers.timeseries_analyzer`,
[24](#)
- `emodpy.bamboo`, [32](#)
- `emodpy.bamboo_api_utils`, [33](#)
- `emodpy.collections_utils`, [40](#)
- `emodpy.defaults`, [25](#)
- `emodpy.defaults.emod_sir`, [25](#)
- `emodpy.defaults.ep4`, [25](#)
- `emodpy.defaults.ep4.dtk_in_process`, [25](#)
- `emodpy.defaults.ep4.dtk_post_process`,
[25](#)
- `emodpy.defaults.ep4.dtk_pre_process`, [25](#)
- `emodpy.defaults.iemod_default`, [26](#)
- `emodpy.emod_campaign`, [40](#)
- `emodpy.emod_file`, [41](#)
- `emodpy.emod_task`, [44](#)
- `emodpy.generic`, [26](#)
- `emodpy.generic.serialization`, [26](#)
- `emodpy.interventions`, [27](#)
- `emodpy.interventions.emod_empty_campaign`,
[27](#)
- `emodpy.reporters`, [27](#)
- `emodpy.reporters.base`, [27](#)
- `emodpy.reporters.builtin`, [29](#)
- `emodpy.reporters.custom`, [29](#)
- `emodpy.utils`, [48](#)

INDEX

A

add_climate_files() (emodpy.emod_file.ClimateFiles method), 43
 add_demographics_from_dict() (emodpy.emod_file.DemographicsFiles method), 43
 add_demographics_from_file() (emodpy.emod_file.DemographicsFiles method), 43
 add_dll() (emodpy.reporters.base.Reporters method), 28
 add_dll_folder() (emodpy.reporters.base.Reporters method), 28
 add_ep4_from_path() (in module emodpy.emod_task), 44
 add_event() (emodpy.emod_campaign.EMODCampaign method), 41
 add_events() (emodpy.emod_campaign.EMODCampaign method), 41
 add_migration_from_file() (emodpy.emod_file.MigrationFiles method), 42
 add_report() (emodpy.reporters.custom.MalariaImmunityReport method), 32
 add_report() (emodpy.reporters.custom.MalariaSummaryReport method), 31
 add_report() (emodpy.reporters.custom.MalariaSurveyJSONAnalyzer method), 32
 add_report() (emodpy.reporters.custom.MalariaTransmissionReport method), 32
 add_report() (emodpy.reporters.custom.Report_TBHIV_ByAge method), 29
 add_report() (emodpy.reporters.custom.ReportEventCounter method), 31
 add_report() (emodpy.reporters.custom.ReportMalariaFiltered method), 31
 add_report() (emodpy.reporters.custom.ReportNodeDemographics method), 31
 add_reporter() (emodpy.reporters.base.Reporters method), 28
 add_serialization_timesteps() (in module emodpy.generic.serialization), 26
 AdultVectorsAnalyzer (class in emodpy.analyzers.adult_vectors_analyzer), 23
 Age_Bins (emodpy.reporters.builtin.ReportNodeDemographics attribute), 29
 age_bins (emodpy.reporters.custom.ReportNodeDemographics attribute), 31
 age_bins (emodpy.reporters.custom.ReportPluginAgeAtInfectionHistogram attribute), 30
 AIR (emodpy.emod_file.MigrationTypes attribute), 41
 AIR_TEMPERATURE (emodpy.emod_file.ClimateFileType attribute), 43
 ALL (emodpy.utils.BambooArtifact attribute), 49
 application() (in module emodpy.defaults.ep4.dtk_in_process), 25
 application() (in module emodpy.defaults.ep4.dtk_post_process), 25
 application() (in module emodpy.defaults.ep4.dtk_pre_process), 25
 asset collection, 51
 assets, 51
B
 bamboo_api_login() (in module emodpy.utils), 49
 bamboo_connection() (in module emodpy.bamboo_api_utils), 33
 BambooArtifact (class in emodpy.utils), 48
 BambooConnection (class in emodpy.bamboo_api_utils), 33
 BaseReporter (class in emodpy.reporters.base), 27
 build_passed() (emodpy.bamboo_api_utils.BuildInfo class method), 35
 BuildArtifacts (class in emodpy.bamboo_api_utils), 36
 builder, 51
 BuildInfo (class in emodpy.bamboo_api_utils), 35
 BuildPlans (class in emodpy.bamboo_api_utils), 39
 BuiltInReporter (class in emodpy.reporters.base), 28
C
 campaign (emodpy.emod_task.EMODTask attribute), 45

[campaign\(\)](#) (*emodpy.defaults.emod_sir.EMODSir static method*), 25
[campaign\(\)](#) (*emodpy.defaults.iemod_default.IEMODDefault static method*), 26
[campaign\(\)](#) (*emodpy.interventions.emod_empty_campaign.EMODEmptyCampaign static method*), 27
[class_name](#) (*emodpy.reporters.base.BuiltInReporter attribute*), 28
[class_name](#) (*emodpy.reporters.builtin.ReportNodeDemographics attribute*), 29
[clear\(\)](#) (*emodpy.emod_campaign.EMODCampaign method*), 40
[climate](#) (*emodpy.emod_task.EMODTask attribute*), 44
[CLIMATE_BY_DATA](#) (*emodpy.emod_file.ClimateModel attribute*), 43
[CLIMATE_CONSTANT](#) (*emodpy.emod_file.ClimateModel attribute*), 43
[CLIMATE_KOPPEN](#) (*emodpy.emod_file.ClimateModel attribute*), 43
[CLIMATE_OFF](#) (*emodpy.emod_file.ClimateModel attribute*), 43
[ClimateFiles](#) (*class in emodpy.emod_file*), 43
[ClimateFileType](#) (*class in emodpy.emod_file*), 43
[ClimateModel](#) (*class in emodpy.emod_file*), 43
[config](#) (*emodpy.emod_task.EMODTask attribute*), 44
[config\(\)](#) (*emodpy.defaults.emod_sir.EMODSir static method*), 25
[config\(\)](#) (*emodpy.defaults.iemod_default.IEMODDefault static method*), 26
[config\(\)](#) (*emodpy.reporters.builtin.ReportHumanMigrationTracking static method*), 29
[config_file_name](#) (*emodpy.emod_task.EMODTask attribute*), 45
[convert_plugin_reports\(\)](#) (*in module emodpy.defaults.ep4.dtk_pre_process*), 25
[copy_simulation\(\)](#) (*emodpy.emod_task.EMODTask method*), 46
[create_campaign_from_callback\(\)](#) (*emodpy.emod_task.EMODTask method*), 45
[create_demog_from_callback\(\)](#) (*emodpy.emod_task.EMODTask method*), 45
[CustomReporter](#) (*class in emodpy.reporters.base*), 27
[cut_iterable_to\(\)](#) (*in module emodpy.collections_utils*), 40
D
[data_group_names](#) (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer attribute*), 24
[deep_del\(\)](#) (*in module emodpy.collections_utils*), 40
[deep_get\(\)](#) (*in module emodpy.collections_utils*), 40
[deep_set\(\)](#) (*in module emodpy.collections_utils*), 40
[default_ep4_fn\(\)](#) (*in module emodpy.emod_task*), 44
[default_filter_fn\(\)](#) (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer method*), 24
[default_group_fn\(\)](#) (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer method*), 24
[default_plot_fn\(\)](#) (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer method*), 24
[default_select_fn\(\)](#) (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer method*), 24
[demographics](#) (*emodpy.emod_task.EMODTask attribute*), 44
[demographics\(\)](#) (*emodpy.defaults.emod_sir.EMODSir static method*), 25
[demographics\(\)](#) (*emodpy.defaults.iemod_default.IEMODDefault static method*), 26
[DemographicsFiles](#) (*class in emodpy.emod_file*), 42
[DENGUE](#) (*emodpy.utils.EradicationBambooBuilds attribute*), 48
[DENGUE_LINUX](#) (*emodpy.utils.EradicationBambooBuilds attribute*), 48
[DENGUE_WIN](#) (*emodpy.utils.EradicationBambooBuilds attribute*), 48
[disable\(\)](#) (*emodpy.reporters.base.CustomReporter method*), 28
[dll_file](#) (*emodpy.reporters.custom.MalariaImmunityReport attribute*), 32
[dll_file](#) (*emodpy.reporters.custom.MalariaSummaryReport attribute*), 31
[dll_file](#) (*emodpy.reporters.custom.MalariaSurveyJSONAnalyzer attribute*), 32
[dll_file](#) (*emodpy.reporters.custom.MalariaTransmissionReport attribute*), 32
[dll_file](#) (*emodpy.reporters.custom.Report_TBHIV_ByAge attribute*), 29
[dll_file](#) (*emodpy.reporters.custom.ReportEventCounter attribute*), 31
[dll_file](#) (*emodpy.reporters.custom.ReportHumanMigrationTracking attribute*), 30
[dll_file](#) (*emodpy.reporters.custom.ReportMalariaFiltered attribute*), 31
[dll_file](#) (*emodpy.reporters.custom.ReportNodeDemographics attribute*), 31
[dll_file](#) (*emodpy.reporters.custom.ReportPluginAgeAtInfectionHistogram attribute*), 30
[download_artifact_to_file\(\)](#) (*emodpy.bamboo_api_utils.BuildArtifacts class method*), 37

[download_artifacts_to_path\(\)](#)
(emodpy.bamboo_api_utils.BuildArtifacts class method), 38

[download_bamboo_artifacts\(\)](#) *(in module emodpy.utils)*, 49

[download_eradication\(\)](#) *(in module emodpy.utils)*, 50

[download_eradication_exe\(\)](#)
(emodpy.bamboo_api_utils.BuildArtifacts class method), 39

[download_file\(\)](#) *(emodpy.bamboo_api_utils.BambooConnection module, 33)*, 35

[download_from_bamboo_url\(\)](#)
(emodpy.bamboo_api_utils.BuildArtifacts class method), 39

[download_from_url\(\)](#) *(in module emodpy.utils)*, 50

[download_latest_bamboo\(\)](#) *(in module emodpy.utils)*, 49

[download_latest_eradication\(\)](#) *(in module emodpy.utils)*, 49

[download_latest_good_Eradication_exe\(\)](#)
(emodpy.bamboo_api_utils.BuildArtifacts class method), 38

[download_latest_good_schema_json\(\)](#)
(emodpy.bamboo_api_utils.BuildArtifacts class method), 38

[download_latest_reporters\(\)](#) *(in module emodpy.utils)*, 50

[download_latest_schema\(\)](#) *(in module emodpy.utils)*, 50

[download_schema_json\(\)](#)
(emodpy.bamboo_api_utils.BuildArtifacts class method), 39

E

[EMOD_RELEASE](#) *(emodpy.utils.EradicationBambooBuilds attribute)*, 48

[EMODCampaign](#) *(class in emodpy.emod_campaign)*, 40

[EMODEmptyCampaign](#) *(class in emodpy.interventions.emod_empty_campaign)*, 27

[emodpy](#)
 module, 23

[emodpy.analyzers](#)
 module, 23

[emodpy.analyzers.adult_vectors_analyzer](#)
 module, 23

[emodpy.analyzers.population_analyzer](#)
 module, 24

[emodpy.analyzers.timeseries_analyzer](#)
 module, 24

[emodpy.bamboo](#)
 module, 32

[emodpy.bamboo_api_utils](#)
 module, 33

[emodpy.collections_utils](#)
 module, 40

[emodpy.defaults](#)
 module, 25

[emodpy.defaults.emod_sir](#)
 module, 25

[emodpy.defaults.ep4](#)
 module, 25

[emodpy.defaults.ep4.dtk_in_process](#)
 module, 25

[emodpy.defaults.ep4.dtk_post_process](#)
 module, 25

[emodpy.defaults.ep4.dtk_pre_process](#)
 module, 25

[emodpy.defaults.iemod_default](#)
 module, 26

[emodpy.emod_campaign](#)
 module, 40

[emodpy.emod_file](#)
 module, 41

[emodpy.emod_task](#)
 module, 44

[emodpy.generic](#)
 module, 26

[emodpy.generic.serialization](#)
 module, 26

[emodpy.interventions](#)
 module, 27

[emodpy.interventions.emod_empty_campaign](#)
 module, 27

[emodpy.reporters](#)
 module, 27

[emodpy.reporters.base](#)
 module, 27

[emodpy.reporters.builtin](#)
 module, 29

[emodpy.reporters.custom](#)
 module, 29

[emodpy.utils](#)
 module, 48

[EMODSir](#) *(class in emodpy.defaults.emod_sir)*, 25

[EMODTask](#) *(class in emodpy.emod_task)*, 44

[EMODTaskSpecification](#) *(class in emodpy.emod_task)*, 47

[empty\(\)](#) *(emodpy.reporters.base.Reporters property)*, 28

[enable\(\)](#) *(emodpy.reporters.base.CustomReporter method)*, 28

[enable_migration\(\)](#)
(emodpy.emod_file.MigrationFiles method), 42

[enable_serialization\(\)](#) *(in module emodpy.generic.serialization)*, 26

- Enabled (*emodpy.reporters.base.BuiltInReporter* attribute), 28
- Enabled (*emodpy.reporters.base.CustomReporter* attribute), 27
- ensure_logged_in() (*emodpy.bamboo_api_utils.BambooConnection* method), 34
- ERADICATION (*emodpy.utils.BambooArtifact* attribute), 48
- ERADICATION_EXE (*emodpy.bamboo_api_utils.BuildArtifacts* attribute), 36
- eradication_path (*emodpy.emod_task.EMODTask* attribute), 44
- EradicationBambooBuilds (class in *emodpy.utils*), 48
- EradicationPlatformExtension (class in *emodpy.utils*), 48
- experiment, 51
- export_spec() (*emodpy.bamboo_api_utils.BuildPlans* static method), 39
- ## F
- FAMILY (*emodpy.emod_file.MigrationTypes* attribute), 41
- filter() (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* method), 24
- find_all_artifact_names() (*emodpy.bamboo_api_utils.BuildArtifacts* class method), 37
- find_all_artifacts() (*emodpy.bamboo_api_utils.BuildArtifacts* class method), 37
- find_artifacts() (*emodpy.bamboo_api_utils.BuildArtifacts* class method), 37
- find_artifacts_by_name() (*emodpy.bamboo_api_utils.BuildArtifacts* class method), 36
- find_build_essential_artifacts() (*emodpy.bamboo_api_utils.BuildArtifacts* class method), 37
- find_server() (*emodpy.bamboo_api_utils.BambooConnection* method), 33
- FIXED_RATE_MIGRATION (*emodpy.emod_file.MigrationModel* attribute), 41
- FP (*emodpy.utils.EradicationBambooBuilds* attribute), 48
- FP_LINUX (*emodpy.utils.EradicationBambooBuilds* attribute), 48
- FP_WIN (*emodpy.utils.EradicationBambooBuilds* attribute), 48
- from_default2() (*emodpy.emod_task.EMODTask* class method), 45
- from_dict() (*emodpy.reporters.base.BaseReporter* method), 27
- from_dict() (*emodpy.reporters.base.BuiltInReporter* method), 28
- from_dict() (*emodpy.reporters.custom.ReportNodeDemographics* method), 31
- from_files() (*emodpy.emod_task.EMODTask* class method), 45
- ## G
- gather_assets() (*emodpy.emod_file.ClimateFiles* method), 43
- gather_assets() (*emodpy.emod_file.InputFilesList* method), 41
- gather_assets() (*emodpy.emod_file.MigrationFiles* method), 42
- gather_assets() (*emodpy.reporters.base.Reporters* method), 28
- gather_common_assets() (*emodpy.emod_task.EMODTask* method), 46
- gather_transient_assets() (*emodpy.emod_task.EMODTask* method), 46
- get_analyzer() (*emodpy.utils.EradicationBambooBuilds* attribute), 48
- GENERIC_LINUX (*emodpy.utils.EradicationBambooBuilds* attribute), 48
- GENERIC_WIN (*emodpy.utils.EradicationBambooBuilds* attribute), 48
- get() (*emodpy.emod_task.EMODTaskSpecification* method), 47
- get_bamboo_api_url() (*emodpy.bamboo_api_utils.BambooConnection* method), 34
- get_bamboo_url() (*emodpy.bamboo_api_utils.BambooConnection* method), 34
- get_build_info() (*emodpy.bamboo_api_utils.BuildInfo* static method), 36
- get_channel_data() (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* method), 25
- get_description() (*emodpy.emod_task.EMODTaskSpecification* method), 47
- get_events_at() (*emodpy.emod_campaign.EMODCampaign* method), 40
- get_events_with_name() (*emodpy.emod_campaign.EMODCampaign* method), 41
- get_example_urls() (*emodpy.emod_task.EMODTaskSpecification* method), 47

- get_github_eradication_url() (in module *emodpy.utils*), 49
- get_latest_build() (in module *emodpy.bamboo_api_utils.BuildInfo* class method), 36
- get_latest_successful_build() (in module *emodpy.bamboo_api_utils.BuildInfo* class method), 36
- get_model_files() (in module *emodpy.bamboo*), 32
- get_parameter() (in module *emodpy.emod_task.EMODTask* method), 47
- get_plans_for_project() (in module *emodpy.bamboo_api_utils.BuildPlans* static method), 39
- get_server_url() (in module *emodpy.bamboo_api_utils.BambooConnection* method), 33
- get_type() (in module *emodpy.emod_task.EMODTaskSpecification* method), 47
- get_version() (in module *emodpy.emod_task.EMODTaskSpecification* method), 47
- ## H
- high-performance computing (HPC), 51
- HIV (in module *emodpy.utils.EradicationBambooBuilds* attribute), 48
- HIV_LINUX (in module *emodpy.utils.EradicationBambooBuilds* attribute), 48
- HIV_WIN (in module *emodpy.utils.EradicationBambooBuilds* attribute), 48
- ## I
- IEMODDefault (class in *emodpy.defaults.iemod_default*), 26
- implicit_configs (in module *emodpy.emod_task.EMODTask* attribute), 45
- initialize() (in module *emodpy.analyzers.adult_vectors_analyzer.AdultVectorsAnalyzer* method), 23
- initialize() (in module *emodpy.analyzers.population_analyzer.PopulationAnalyzer* method), 24
- initialize() (in module *emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* method), 24
- InputFilesList (class in *emodpy.emod_file*), 41
- interval_years (in module *emodpy.reporters.custom.ReportPlugin* attribute), 30
- is_linux (in module *emodpy.emod_task.EMODTask* attribute), 45
- ## J
- json() (in module *emodpy.emod_campaign.EMODCampaign* property), 40
- json() (in module *emodpy.reporters.base.Reporters* property), 28
- ## L
- LAND_TEMPERATURE (in module *emodpy.emod_file.ClimateFileType* attribute), 43
- LINUX (in module *emodpy.utils.EradicationPlatformExtension* attribute), 48
- load_files() (in module *emodpy.emod_task.EMODTask* method), 45
- load_from_dict() (in module *emodpy.emod_campaign.EMODCampaign* static method), 40
- load_from_file() (in module *emodpy.emod_campaign.EMODCampaign* static method), 40
- load_serialized_population() (in module *emodpy.generic.serialization*), 26
- load_session_cookie() (in module *emodpy.bamboo_api_utils.BambooConnection* method), 33
- LOCAL (in module *emodpy.emod_file.MigrationTypes* attribute), 41
- logger (in module *emodpy.emod_task*), 44
- login() (in module *emodpy.bamboo_api_utils.BambooConnection* method), 34
- login() (in module *emodpy.bamboo_api_utils*), 40
- login_session_exists() (in module *emodpy.bamboo_api_utils.BambooConnection* method), 34
- ## M
- make_api_get_request() (in module *emodpy.bamboo_api_utils.BambooConnection* method), 35
- make_exe_executable() (in module *emodpy.bamboo_api_utils.BuildArtifacts* class method), 39
- make_get_request() (in module *emodpy.bamboo_api_utils.BambooConnection* method), 35
- MALARIA (in module *emodpy.utils.EradicationBambooBuilds* attribute), 48
- MALARIA_LINUX (in module *emodpy.utils.EradicationBambooBuilds* attribute), 48
- MALARIA_WIN (in module *emodpy.utils.EradicationBambooBuilds* attribute), 48
- MalariaImmunityReport (class in *emodpy.reporters.custom*), 31
- MalariaInfectionHistogram (class in *emodpy.reporters.custom*), 31
- MalariaSurveyJSONAnalyzer (class in *emodpy.reporters.custom*), 32
- MalariaTransmissionReport (class in *emodpy.reporters.custom*), 32
- map() (in module *emodpy.analyzers.adult_vectors_analyzer.AdultVectorsAnalyzer* method), 23
- map() (in module *emodpy.analyzers.population_analyzer.PopulationAnalyzer* method), 24

`map()` (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* attribute), 25
`merge_with()` (*emodpy.emod_file.MigrationFiles* attribute), 42
MigrationFiles (class in *emodpy.emod_file*), 41
MigrationModel (class in *emodpy.emod_file*), 41
MigrationPattern (class in *emodpy.emod_file*), 41
`migrations` (*emodpy.emod_task.EMODTask* attribute), 44
MigrationTypes (class in *emodpy.emod_file*), 41
module
 emodpy, 23
 emodpy.analyzers, 23
 emodpy.analyzers.adult_vectors_analyzer, 23
 emodpy.analyzers.population_analyzer, 24
 emodpy.analyzers.timeseries_analyzer, 24
 emodpy.bamboo, 32
 emodpy.bamboo_api_utils, 33
 emodpy.collections_utils, 40
 emodpy.defaults, 25
 emodpy.defaults.emod_sir, 25
 emodpy.defaults.ep4, 25
 emodpy.defaults.ep4.dtk_in_process, 25
 emodpy.defaults.ep4.dtk_post_process, 25
 emodpy.defaults.ep4.dtk_pre_process, 25
 emodpy.defaults.iemod_default, 26
 emodpy.emod_campaign, 40
 emodpy.emod_file, 41
 emodpy.emod_task, 44
 emodpy.generic, 26
 emodpy.generic.serialization, 26
 emodpy.interventions, 27
 emodpy.interventions.emod_empty_campaign, 27
 emodpy.reporters, 27
 emodpy.reporters.base, 27
 emodpy.reporters.builtin, 29
 emodpy.reporters.custom, 29
 emodpy.utils, 48
N
 name (*emodpy.reporters.base.CustomReporter* attribute), 27
 name (*emodpy.reporters.custom.MalariaImmunityReport* attribute), 32
 name (*emodpy.reporters.custom.MalariaSummaryReport* attribute), 31
O
 ordered_levels (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* attribute), 24
 output_file (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* attribute), 24
P
 parameters (*emodpy.reporters.base.BuiltInReporter* attribute), 28
 parameters (*emodpy.reporters.builtin.ReportHumanMigrationTracking* attribute), 29
 plot_by_channel() (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* method), 25
 PLUGINS (*emodpy.utils.BambooArtifact* attribute), 48
 PopulationAnalyzer (class in *emodpy.analyzers.population_analyzer*), 24
 pre_creation() (*emodpy.emod_task.EMODTask* method), 46
 Pretty_Format (*emodpy.reporters.base.BuiltInReporter* attribute), 28
 process_simulation() (*emodpy.defaults.iemod_default.IEMODDefault* method), 26
R
 RAINFALL (*emodpy.emod_file.ClimateFileType* attribute), 43
 RANDOM_WALK_DIFFUSION (*emodpy.emod_file.MigrationPattern* attribute), 41

read_config_file() (*emodpy.emod_file.ClimateFiles* method), 44
 read_config_file() (*emodpy.emod_file.MigrationFiles* method), 42
 read_custom_reports_file() (*emodpy.reporters.base.Reporters* method), 28
 reduce() (*emodpy.analyzers.adult_vectors_analyzer.AdultVectorsAnalyzer* method), 23
 reduce() (*emodpy.analyzers.population_analyzer.PopulationAnalyzer* method), 24
 reduce() (*emodpy.analyzers.timeseries_analyzer.TimeseriesAnalyzer* method), 25
 REGIONAL (*emodpy.emod_file.MigrationTypes* attribute), 41
 RELATIVE_HUMIDITY (*emodpy.emod_file.ClimateFileType* attribute), 43
 RELEASE (*emodpy.utils.EradicationBambooBuilds* attribute), 48
 reload_from_simulation() (*emodpy.emod_task.EMODTask* method), 47
 Report_TBHIV_ByAge (class in *emodpy.reporters.custom*), 29
 REPORTER_PLUGINS (*emodpy.bamboo_api_utils.BuildArtifacts* attribute), 36
 Reporters (class in *emodpy.reporters.base*), 28
 reporters (*emodpy.emod_task.EMODTask* attribute), 44
 ReportEventCounter (class in *emodpy.reporters.custom*), 31
 ReportHumanMigrationTracking (class in *emodpy.reporters.builtin*), 29
 ReportHumanMigrationTracking (class in *emodpy.reporters.custom*), 30
 ReportMalariaFiltered (class in *emodpy.reporters.custom*), 31
 ReportNodeDemographics (class in *emodpy.reporters.builtin*), 29
 ReportNodeDemographics (class in *emodpy.reporters.custom*), 30
 ReportPluginAgeAtInfectionHistogram (class in *emodpy.reporters.custom*), 29
 Reports (*emodpy.reporters.base.CustomReporter* attribute), 27
 Reports (*emodpy.reporters.custom.ReportHumanMigrationTracking* attribute), 30
 Reports (*emodpy.reporters.custom.ReportPluginAgeAtInfectionHistogram* attribute), 30
 save_bamboo_credentials() (in module *emodpy.emod_file.ClimateFiles* method), 44
 save_credentials() (*emodpy.bamboo_api_utils.BambooConnection* method), 34
 save_credentials() (in module *emodpy.bamboo_api_utils*), 40
 SCHEMA (*emodpy.utils.BambooArtifact* attribute), 48
 SCHEMA_JSON (*emodpy.bamboo_api_utils.BuildArtifacts* attribute), 36
 SEA (*emodpy.emod_file.MigrationTypes* attribute), 41
 session_cookie() (*emodpy.bamboo_api_utils.BambooConnection* property), 33
 session_cookie_filename() (*emodpy.bamboo_api_utils.BambooConnection* property), 33
 set_all_persisted() (*emodpy.emod_file.MigrationFiles* method), 42
 set_climate_constant() (*emodpy.emod_file.ClimateFiles* method), 43
 set_command_line() (*emodpy.emod_task.EMODTask* method), 46
 set_parameter() (*emodpy.emod_task.EMODTask* method), 46
 set_parameter_partial() (*emodpy.emod_task.EMODTask* class method), 46
 set_parameter_sweep_callback() (*emodpy.emod_task.EMODTask* static method), 46
 set_task_config() (*emodpy.emod_file.ClimateFiles* method), 43
 set_task_config() (*emodpy.emod_file.DemographicsFiles* method), 42
 set_task_config() (*emodpy.emod_file.InputFilesList* method), 41
 set_task_config() (*emodpy.emod_file.MigrationFiles* method), 42
 set_task_config() (*emodpy.reporters.base.Reporters* method), 28
 simulation_demographics (*emodpy.emod_task.EMODTask* attribute), 45
 simulation_migrations (*emodpy.emod_task.EMODTask* attribute), 45
 SINGLE_ROUND_TRIPS

S

(*emodpy.emod_file.MigrationPattern* attribute), *write_session_cookie()*
41 (*emodpy.bamboo_api_utils.BambooConnection*
Stratify_By_Gender (*emodpy.reporters.builtin.ReportNodeDemographics*
attribute), 29
successful_build_result()
(*emodpy.bamboo_api_utils.BuildInfo* static
method), 36

T

task, 51
TBHIV (*emodpy.utils.EradicationBambooBuilds* at-
tribute), 48
TBHIV_LINUX (*emodpy.utils.EradicationBambooBuilds*
attribute), 48
TBHIV_WIN (*emodpy.utils.EradicationBambooBuilds*
attribute), 48
template, 51
TimeseriesAnalyzer (class in
emodpy.analyzers.timeseries_analyzer), 24
to_dict() (*emodpy.reporters.base.BaseReporter*
method), 27
to_dict() (*emodpy.reporters.base.BuiltInReporter*
method), 28
to_dict() (*emodpy.reporters.base.CustomReporter*
method), 28
to_dict() (*emodpy.reporters.custom.ReportNodeDemographics*
method), 31
TYPHOID (*emodpy.utils.EradicationBambooBuilds* at-
tribute), 48
TYPHOID_LINUX (*emodpy.utils.EradicationBambooBuilds*
attribute), 48
TYPHOID_WIN (*emodpy.utils.EradicationBambooBuilds*
attribute), 48

U

update_migration_pattern()
(*emodpy.emod_file.MigrationFiles* method), 42
update_parameters()
(*emodpy.emod_task.EMODTask* method),
47
url_exists() (*emodpy.bamboo_api_utils.BambooConnection*
method), 33
use_embedded_python
(*emodpy.emod_task.EMODTask* attribute),
45

W

WAYPOINTS_HOME (*emodpy.emod_file.MigrationPattern*
attribute), 41
Windows (*emodpy.utils.EradicationPlatformExtension*
attribute), 48