International Institute for Applied Systems Analysis

message-ix-models

IIASA Energy, Climate, and Environment (ECE) Program

Apr 22, 2024

USER GUIDE

`message_ix_models` provides tools for research using the **MESSAGEix-GLOBIOM family of models** developed by the IIASA Energy, Climate, and Environment (ECE) Program and its collaborators. This 'family' includes single-country and other models derived from the main, global model; all built in the MESSAGEix framework and on the ix modeling platform (ixmp).

Among other tasks, the tools allow modelers to:

- retrieve input data from various upstream sources,
- process/transform upstream data into model input parameters,
- create, populate, modify, and parametrize scenarios,
- conduct model runs,
- set up model *variants* with additional details or features, and
- report quantities computed from model outputs.

CHAPTER

ONE

INSTALLATION

Note: *message_ix_models* requires message_ix to run. Please ensure your system has their required dependencies installed.

message_ix_models is structured as a Python package and is published to the PyPI public code repository. Hence, there are two options for the installation:

1.1 From PyPI

This option is only recommended for users who do not wish to make any changes to the source code.

1. Run:

```
$ pip install message-ix-models[<extra_dependencies>]
```

1.2 From source

Use this option if you intend to make changes to the source code. We value your contributions via pull requests to the main repository. Please consider contributing your changes.

1. Fork the the main repository. This will create a new repository <user>/message-ix-models.
2. Clone your fork; using the Github Desktop client, or the command line:

```
$ git clone git@github.com:USER/message-ix-models.git
```

3. Add the main repository as a remote git repository. This will allow keeping up to date with changes there and importing tags, which also needs to be done for the install tests to succeed:

```
$ git remote add upstream git@github.com:iiasa/message-ix-models.git
$ git fetch upstream --tags
```

4. Inside the message-ix-models directory, run:

```
$ pip install --editable .[<extra_dependencies>]
```

1.3 Dependencies

See `pyproject.toml`. The following sets of extra dependencies are available; per the user guide linked above, they can be installed along with the mandatory dependencies by adding (for instance) `extra_name` to the package spec **`pip install message_data[extra_name]`**.

`docs`
: Minimum requirements for building the docs.

`report`
: For running the *Reporting (report)* functionality.

`tests`
: Minimal requirements for the test suite.

1.4 Check that installation was successful

Verify that the version installed corresponds to the latest release by running the following commands on the command line:

```
# Show versions of message_ix, message-ix-models, and key dependencies
$ message-ix show-versions

# Show the list of modelling platforms that have been installed and
# the path to the database config file
# By default, just the local database should appear in the list
$ message-ix platform list
$ mix-models config show
```

The above commands will work as of `message_ix` version 3.0 and in subsequent versions. Please read through the output of the *mix-models command* to understand the different CLI options and what you can do with them.

CHAPTER

TWO

DATA, METADATA, AND CONFIGURATION

Many, varied kinds of data are used to prepare and modify MESSAGEix-GLOBIOM scenarios. Other data are produced by code as incidental or final output.

These can be categorized in several ways. One is by the purpose they serve:

- **data**—actual numerical values—used or produced by code,
- **metadata**, information describing where data is, how to manipulate it, how it is structured, etc.;
- **configuration** that otherwise affects how code works.

Another is by whether the data are **input**, **output**, or both.

This page describes how to store and handle such files in *message_ix_models* and message_data.[1]

[1] Unless specifically distinguished in the text, all of the following applies to *both* *message_ix_models* and message_data.

2.1 Choose locations for data

These are listed in order of preference.

2.1.1 (1) Not in `message_ix_models`

Data that are available from public, stable sources **should not** be added to the *`message_ix_models`* repository. Instead:

1. Fetch the code from their original location. If possible, this **should** be done by extending or using *`message_ix_models.util.pooch`*.
2. If *`message_ix_models`* relies on certain adjustments to the data, *do not* commit the adjusted data. Instead:
   a. Commit code that performs the adjustments. This makes methods for data transformation (and any assumptions involved) transparent.
   b. If necessary, cache the result—see below.

2.1.2 (2) `message_ix_models/data/`

- Files in this directory are **public**.
- In standard Python terms, these are "package data".
- This is the preferred location for:
  - General-purpose metadata for the MESSAGEix-GLOBIOM base global model or variants.
  - Configuration.
  - Data for publicized model variants and completed/published projects.
- These files are packaged, published, and installable from PyPI with *`message_ix_models`*—*unless* specifically excluded via `MANIFEST.in` (see *Large/binary input data*, below).
- These data can be reached with *`package_data_path()`*, *`load_package_data()`*, or other, more specialized code.
- Documentation files like `doc/pkg-data/*.rst` describe the contents of these files, and appear in the automatically-built documentation. For example: *Node code lists*.

2.1.3 (3) `data/` directory in the `message_data` repository

- Files in this directory are **private** and not installable from PyPI (because `message_data` is not packaged for or installable from PyPI).
- This is the preferred location for:
  - Data for model variants and projects under current development.
  - Specific data files that cannot (currently, or ever) be made public, for instance because of restrictive licenses.
- These data can be reached with *`private_data_path()`*, *`load_private_data()`* or other, more specialized code.

2.1.4 (4) Other, system-specific ("local") directories

These are the preferred location for:

- Outputs, such as data or plot files generated by reporting.
- Data files not distributable with *message_ix_models*, for instance those with access conditions (registration, payment, etc.).
- Caches: temporary data files used to speed up other code by avoiding repeat of slow operations.

These kinds of data **must not** be committed to *message_ix_models*. Caches and output **should not** be committed to `message_data`.

(4A) Local data

Each user **may** configure a location for these data, appropriate to their system, and then use *Context.get_local_path()* and/or *local_data_path()* to construct paths under this directory.

This setting can be made in multiple ways. From lowest to highest precedence:

1. The default location is the *current working directory*: the directory in which the **mix-models** *Command-line interface* is invoked, or in which Python code is run that imports and uses *message_ix_models*.
2. The `ixmp` configuration file setting `message local data`.
3. The `MESSAGE_LOCAL_DATA` environment variable.
4. The `--local-data` CLI option and related options such as the `--output` option to the `report` command.
5. Code that directly modifies the `local_data` setting on *Context*.

This location **should** be outside the Git-controlled directories for *message_ix_models* or `message_data`. In other words, users **should** at least use (2) or (3) to specify such directories. If not, they **may** use `.gitignore` files to hide these from Git.

(4B) Cache data

Code **should** use `platformdirs.user_cache_path()` to identify a system-specific path to a cache directory. For example:

```
from platformdirs import user_cache_path

# Always use "message-ix-models" as the `appname` parameter
ucp = user_cache_path("message-ix-models")

# Construct the sub-directory for the current module
dir_ = ucp.joinpath("my-project", "subdir")
dir_.mkdir(parents=True, exist_ok=True)

# Construct a file path within this directory
p = dir_.joinpath("data-file-name.csv")
```

2.2 General guidelines

Always consider: "Will this code work on another researcher's computer?"

Prefer text formats
…such as CSV, over binary formats like Excel. CSV files up to several thousand lines are compressed by Git automatically, and Git can handle diffs to these files easily.

***Do not* hard-code paths**
Data stored with (2–4) above can be retrieved with the utility functions mentioned, instead of hard-coded paths.

For system-specific paths (4) only, get a `Context` object and use it to get an appropriate `Path` object pointing to a file:

```
# Store a base path
project_path = context.get_local_path("myproject", "output")

# Use the Path object to generate a subpath
run_id = "foo"
output_file = project_path.joinpath("reporting", run_id, "all.xlsx")
```

Keep input and output data separate
Where possible, use (1–3) above for input data, and (4A) for output data.

Use a consistent scheme for data locations
For a submodule for a specific model variant or project named, for instance, `message_ix_models.model.[name]` or `message_ix_models.project.[name]`, keep input data in a well-organized directory under:

- `[base]/[name]/` —preferred, flatter,
- `[base]/model/[name]/`,
- `[base]/project/[name]/`,
- or similar,

where `[base]` is (2) or (3), above.

Keep *project-specific configuration files* in the same locations, or (less preferable) alongside Python code files:

```
# Located in `message_ix_models/data/`:
config = load_package_data("myproject", "config.yaml")

# Located in `data/` in the message_data repo:
config = load_private_data("myproject", "config.yaml")

# Located in the same directory as the code
config = yaml.safe_load(open(Path(__file__).with_name("config.yaml")))
```

Use a similar scheme for output data, except under (4A).

Re-use configuration
Configuration to run a set of scenarios or to prepare reported submissions **should** re-use or extend existing, general-purpose code. Do not duplicate code or configuration. Instead, adjust or selectively overwrite its behaviour via project-specific configuration read from a file.

2.3 Large/binary input data

These data, such as Microsoft Excel spreadsheets, **must not** be committed as ordinary Git objects. This is because the entire file is re-added to the Git history for even small modifications, making it very large (see issue #37).

Instead, use one or more of the following patterns, in order of preference. Whichever pattern is used, code for handling large input data **must** be in *message_ix_models*, even if the data itself is private, for instance in `message_data` or another location.

2.3.1 Fetch directly from a remote source

This corresponds to section (1) above. Preferably, do this via *message_ix_models.util.pooch*:

- Extend *pooch.SOURCE* to store the Internet location, file name(s), and hash(es) of the file(s).
- Call *pooch.fetch()* to retrieve the file and cache it locally.
- Write code in *message_ix_models* that processes the data into a common format, for instance by sub-classing *ExoDataSource*.

This pattern is preferred because it can be replicated by anyone, and the reference data is public.

This pattern may be applied to:

- Data published and maintained by others, or
- Data created by the IIASA ECE program to be used in *message_ix_models*, such as Zenodo records.

2.3.2 Use Git Large File Storage (LFS)

Git LFS is a Git extension that allows for storing large, binary files without bloating the commit history. Essentially, Git stores a 3-line text file with a hash of the full file, and the full file is stored separately. The IIASA GitHub organization has up to 300 GB of space for such LFS objects.

To use this pattern, simply **`git add ...`** and **`git commit`** files in an appropriate location (above). New or unusual binary file extensions may require a **`git lfs`** command or modification to `.gitattributes` to ensure they are tracked by LFS and not by Git itself. See the Git LFS documentation at the link above for more detail.

For large files stored in `message_ix_models/data/` (2, above) using Git LFS, these:

- **must** be added to `MANIFEST.in`. This avoids including the files in Python distributions published on PyPI.
- **should** be added to *util.pooch*. This allows users who install *message_ix_models* from PyPI to easily retrieve the data. This usage **must** be included in the documentation that describes the data files.

2.3.3 Retrieve data from existing databases

These include the same IIASA ENE ixmp databases that are used to store scenarios. Documentation **must** be provided that ensures this data is reproducible: that is, any original sources and code to create the database used by `message_data`.

2.3.4 Other patterns

Some other patterns exist, but should not be repeated in new code, and should be migrated to one of the above patterns.

- SQL queries against a Oracle/JDBC database. See message_data:data-iea (in `message_data`) and issue #53 for a description of how to replace/simplify this code.

2.4 Configuration

Context objects are used to carry configuration, environment information, and other data between parts of the code. Scripts and user code can also store values in a Context object.

```
# Get an existing instance of Context. There is always at
# least 1 instance available
c = Context.get_instance()

# Store a value using attribute syntax
c.foo = 42

# Store a value with spaces in the name using item syntax
c["PROJECT data source"] = "Source A"

# my_function() responds to 'foo' or 'PROJECT data source'
my_function(c)

# Store a sub-dictionary of values
c["PROJECT2"] = {"setting A": 123, "setting B": 456}

# Create a subcontext with all the settings of `c`
c2 = deepcopy(c)

# Modify one setting
c2.foo = 43

# Run code with this alternate setting
my_function(c2)
```

For the CLI, every command decorated with `@click.pass_obj` gets a first positional argument `context`, which is an instance of this class. The settings are populated based on the command-line parameters given to `mix-models` or (sub)commands.

2.4.1 Top-level settings

These are defined by *message_ix_models.Config*.

Specific modules for model variants, projects, etc. **should**:

- Define a single `dataclass` to express the configuration options they understand. See for example:
  - *model.Config* for describing existing models or constructing new models,
  - *report.Config* for reporting,
  - `message_data.model.buildings.Config` (for the MESSAGEix-Buildings model variant / linkage).
- Store this on the *Context* at a simple key. For example *model.Config* is stored at `context.model` or `context["model"]`.
- Retrieve and respect configuration from existing objects, i.e. only duplicate settings with the same meaning when strictly necessary.

- Communicate to other modules by setting the appropriate configuration values.

class message_ix_models.**Config**(*local_data: ~pathlib.Path = <factory>, platform_info: ~typing.MutableMapping[str, str] = <factory>, scenario_info: ~typing.MutableMapping[str, str] = <factory>, scenarios: ~typing.List[~message_ix_models.util.scenarioinfo.ScenarioInfo] = <factory>, dest_platform: ~typing.MutableMapping[str, str] = <factory>, dest_scenario: ~typing.MutableMapping[str, str] = <factory>, url: str | None = None, dest: str | None = None, cache_path: str | None = None, debug_paths: ~typing.Sequence[str] = <factory>, dry_run: bool = False, verbose: bool = False*)

Top-level configuration for *message_ix_models* and message_data.

cache_path: str | None = None

Base path for cached data, e.g. as given by the **--cache-path** CLI option. Default: the directory message-ix-models within the directory given by platformdirs.user_cache_path().

debug_paths: Sequence[str]

Paths of files containing debug outputs. See *Context.write_debug_archive()*.

dest: str | None = None

Like *url*, used by e.g. *clone_to_dest()*.

dest_platform: MutableMapping[str, str]

Like *platform_info*, used by e.g. *clone_to_dest()*.

dest_scenario: MutableMapping[str, str]

Like *scenario_info*, used by e.g. *clone_to_dest()*.

dry_run: bool = False

Whether an operation should be carried out, or only previewed. Different modules will respect *dry_run* in distinct ways, if at all, and **should** document behaviour.

local_data: Path

Base path for *system-specific data*, i.e. as given by the **--local-data** CLI option or *message local data* key in the ixmp configuration file.

platform_info: MutableMapping[str, str]

Keyword arguments—especially *name*—for the ixmp.Platform constructor, from the **--platform** or **--url** CLI option.

scenario_info: MutableMapping[str, str]

Keyword arguments—*model*, *scenario*, and optionally *version*—for the ixmp.Scenario constructor, as given by the **--model**/ **--scenario** or **--url** CLI options.

scenarios: List[*ScenarioInfo*]

Like *scenario_info*, but a list for operations affecting multiple scenarios.

url: str | None = None

A scenario URL, e.g. as given by the **--url** CLI option.

verbose: bool = False

Flag for causing verbose output to logs or stdout. Different modules will respect *verbose* in distinct ways.

CHAPTER

THREE

COMMAND-LINE INTERFACE

This page describes how to use the **mix-models** command-line interface (CLI) to perform common tasks. **mix-models** is organized into **commands** and **subcommands**, sometimes in multiple levels. Our goal is that the *semantics* of all commands are similar, so that interacting with each command feels similar.

3.1 Controlling CLI behaviour

To support a variety of complex use-cases, the MESSAGEix stack takes configuration and inputs from several places:

3.1.1 ixmp configuration file: config.json

ixmp keeps track of named Platforms and their associated databases, and stores information in its config.json file. See Configuration in the documentation. List existing platforms:

```
$ ixmp platform list
```

To add a specific database, you can use the ixmp CLI[1]:

```
$ ixmp platform add <PLATFORMNAME> jdbc oracle <COMPUTER>:<PORT>/<PATH> <USERNAME>
↪<PASSWORD>
```

You may also want to make this the *default* platform. Unless told otherwise, *message_ix_models* creates Platform objects without any arguments (mp = ixmp.Platform()); this loads the default platform. Set the default:

```
$ ixmp platform add default <PLATFORMNAME>
```

message_ix recognizes the following config.json value:

[1] <COMPUTER> is in this case either the hostname or the IP address.

message_model_dir
Path to the GAMS model files. Most code in MESSAGEix-GLOBIOM expects the GAMS model files from the current `message_ix` `main` branch, so you should not set this, or unset it when using *`message_ix_models`*.

`message_ix_models` recognizes the following 2 `config.json` values:

message_local_data
Path to local data, if it is set and not overridden.

no_message_data
If not set or `False`, then the CLI displays a warning message if the private `message_data` package is not installed:

```
Warning: message_data is not installed or cannot be imported; see the␣
↪documentation via --help
```

If set to `True`, then the message is suppressed:

```
$ mix-models config set no_message_data true
```

3.1.2 Environment variables

Some code responds to environment variables. For example, ixmp responds to `IXMP_DATA`, which tells it where to find the file `config.json`.

`message_ix_models` responds to `MESSAGE_LOCAL_DATA`; see *the discussion of local data*.

3.1.3 CLI parameters (arguments and options)

Each command has zero or more arguments and options. **Arguments** are mandatory and follow the command name in a certain order. **Options**, as the name implies, are not required. If an option is omitted, a default value is used; the code and `--help` text make clear what the default behaviour is.

Arguments and options are **hierarchical**. Consider the following examples:

```
$ mix-data --opt0=foo cmd1 --opt1=bar arg1 cmd2 --opt2=baz arg2
$ mix-data --opt0=foo cmd1            arg1 cmd3 --opt3=baz arg3a arg3b
```

In these examples:

- **`--opt0`** is an option that (potentially) affects **any** command, including the subcommands **`cmd2`** or **`cmd3`**.
- **`--opt1`** and **`arg1`** are an option and mandatory argument to the command **`cmd1`**. They might not have any relevance to other **`mix-models`** commands.
- **`cmd2`** and **`cmd3`** are distinct subcommands of **`cmd1`**.
  - They *may* respond to **`--opt1`** and **`arg1`**, and to **`--opt0`**; at least, they *must* not contradict them.
  - They each may have their own options and arguments, which can be distinct.

Tip: Use `--help` for any (sub)command to read about its behaviour. If the help text does not make the behaviour clear, file an issue.

3.1.4 Configuration files and metadata

For some features of the code, the default behaviour is very elaborate and serves for most uses; but we also provide the option to override it. This default behaviour or optional behaviour is defined by reading an input file. These are stored in the *package data* directory.

For example, **mix-models report** loads reporting configuration from `message_ix_models/data/report/global.yaml`, a YAML file with hundreds of lines. Optionally, a different file can be used:

```
$ mix-models report --config other
```

…looks for a file `other.yaml` in the *local data* directory or current working directory. Or:

```
$ mix-models report --config /path/to/another/file.yaml
```

…can be used to point to a file in a different directory.

3.2 Important CLI options and commands

3.2.1 Top-level options and commands

mix-models --help describes these:

```
Usage: mix-models [OPTIONS] COMMAND [ARGS]...

  Command-line interface for MESSAGEix-GLOBIOM model tools.

  Every tool and script in this repository is accessible through this CLI.
  Scripts are grouped into commands and sub-commands. For help on specific
  (sub)commands, use --help, for instance:

          mix-models report --help
          mix-models ssp gen-structures --help

  The top-level options --platform, --model, and --scenario are used by
  commands that access specific MESSAGEix scenarios in a specific ixmp
  platform/database; these can also be specified with --url.

  For complete documentation, see
  https://docs.messageix.org/projects/models/en/latest/cli.html

Options:
  --url ixmp://PLATFORM/MODEL/SCENARIO[#VERSION]
                                  Scenario URL.
  --platform PLATFORM             ixmp platform name.
  --model MODEL                   Model name for some commands.
  --scenario SCENARIO             Scenario name for some commands.
  --version INTEGER               Scenario version for some commands.
  --local-data PATH               Base path for local data.
  -v, --verbose                   Print DEBUG-level log messages.
  --help                          Show this message and exit.

Commands:
  buildings         MESSAGEix-Buildings model.
  cd-links          CD-LINKS project.
  config            Get and set configuration keys.
  covid             COVID project.
  engage            ENGAGE project.
  export-test-data  Prepare data for testing.
  fetch             Retrieve data from primary sources.
```

(continues on next page)

(continued from previous page)

```
  iiasapp           Import power plant capacity.
  last-log          Show the location of the last log file, if any.
  material          Model with materials accounting.
  model             MESSAGEix-GLOBIOM reference energy system (RES).
  navigate          NAVIGATE project.
  prep-submission   Prepare scenarios for submission to an IIASA Scenario...
  report            Postprocess results.
  res               MESSAGEix-GLOBIOM reference energy system (RES).
  ssp               Shared Socioeconomic Pathways (SSP) project.
  techs             Export metadata to technology.csv.
  testing           Manipulate test data.
  transport         MESSAGEix-Transport variant.
  water-ix          MESSAGEix-Water and Nexus variant.
```

Further information about the top-level options:

`--platform PLATFORM` or `--url`
: By default, message_data connects to the default ixmp Platform. These options direct it to work with a different Platform.

`--model MODEL --scenario SCENARIO` or `--url`
: Many commands use an *existing* `Scenario` as a starting point, and begin by cloning that Scenario to a new (model name, scenario name). For any such command, these top-level options define the starting point/initial Scenario to clone/'baseline'.

  In contrast, see **`--output-model`**, below.

3.2.2 Common options

Since *`message_ix_models.model`* and `message_ix_models.project` codes often perform similar tasks, their CLI options and arguments are provided in *`util.click`* for easy re-use. These include:

`SSP` argument
: This takes one of the values 'SSP1', 'SSP2', or 'SSP3'.

  Commands that will not work for one or more of the SSPs should check the argument value given by the user and raise `NotImplementedError`.

`--output-model NAME` option
: This option is a counterpart to the top-level **`--url`**, **`--model`**, or **`--scenario`** options. A command that starts from one Scenario, and builds one or more Scenarios from it will clone *to* a new (model name, scenario name); **`--output-model`** gives the model name.

  Current code generates a variety of fixed (non-configurable) scenario names; use `--help` for each command to see which.

To employ these in new code, refer to the example of existing code.

CHAPTER

FOUR

QUICK-START GUIDE TO RUNNING A MESSAGEIX-GLOBIOM BASELINE MODEL

This starting guide is for researchers at IIASA or external collaborators, who are not constant developers of `message_ix` and related software. It should support running scenarios of different models of the MESSAGEix-GLOBIOM family.

Note:

- This is only a guide, based on some users' experiences. It is **very possible** that the following steps for running a scenario **will not work directly** on your machine.
- This guide **does not** cover how to contribute to *`message_ix_models`*. If you wish to contribute, please read and understand our contribution guide completely.

4.1 Prerequisites

The following prerequisites are necessary to be able to follow the steps of this guide and be able to run scenarios. Most of these prerequisites are analogous to the prerequisites of using `message_ix`.

4.1.1 Knowledge & skills

Please go through the Prerequisite knowledge & skills in the `message_ix` documentation. You should understand **all** of the basic pre-requisites, and **most** of the advanced pre-requisites. Understanding the concept of virtual environments is especially important. If needed, please educate yourself in the specific areas.

Going through this guide, some error messages might occur. Most of them can be fixed by understanding the error message itself. Understanding the Python Traceback might support with reading and interpreting an error message in Python. If an error message is not directly understandable, a web search is often very helpful. Stack Overflow is a great source of support, and most likely someone else already had the same question on how to solve an error.

If you couldn't find a solution on the internet, please open an issue on the main repository or another repository of the `message_ix` software stack, as appropriate.

4.2 Installation

Go through the installation process of *message_ix_models* via *Installation*.

Tip: If you run into issues related to installing message_ix, please also check Common issues in its installation docs.

4.3 Preparation

When using *message_ix_models*, you will store and access data for distinct scenarios. ixmp provides this storage service via Platforms. Each platform corresponds to a database (DB), either local to your system or remote. To receive the data for a global model, you can either configure access to an existing, remote DB; or download a model 'snapshot' file from Zenodo and load it into your local DB.

4.3.1 Configure access to existing DBs

Note: The existing DBs at IIASA facilities are not publicly accessible. If you are interested in collaborating, please reach out to the MESSAGE*ix* team.

Every model variant and project documented with *message_ix_models* and message_data should contain information about the platform where associated scenarios are stored. (If not, reach out to the person(s) responsible for those variants/projects and ask them.) ixmp keeps track of these platforms. See the *message_ix_models Command-line interface*, and the ixmp Configuration documentation for how to configure these.[1] To access the IIASA DBs:

- <Computer> is x8oda.iiasa.ac.at,
- <PORT> is 1521,
- <PATH> will be pIXMP2.iiasa.ac.at for most modern DBs, but could also be pIXMP1.iiasa.ac.at on older instances.

Note: If you are a collaborator, but do not know the <USERNAME> and/or <PASSWORD> of the platform you want to use, please reach out to the MESSAGE*ix* Community Manager, currently @glatterf42.

If you do not have access to the IIASA DBs, you need to configure a local DB. This is appropriate for small applications (i.e. less than hundreds of scenarios). Per the ixmp "Configuration" docs linked above, in your call to **mix-models config ...** you can:

- replace oracle with hsqldb,
- replace the URL <COMPUTER>:<PORT>/<PATH> with a path on your system, and
- omit <USERNAME> and <PASSWORD>.

For more complex modelling needs and required infrastructure, please reach out to the MESSAGE*ix* team.

[1] The **mix-models config ...** CLI is identical to the **ixmp config ...** CLI, and has exactly the same behaviour.

4.3.2 Download snapshot from Zenodo

The latest version of a MESSAGEix-GLOBIOM baseline model can be found on Zenodo. For convenience, the function *snapshot.load()* (documented at *Load model snapshots (model.snapshot)*) automates the following steps, but you can also perform them manually:

- Fetch the snapshot from Zenodo. To do this manually, do one of the following:
  - Use a CLI command like:

    ```
    mix-models fetch snapshot-1
    ```

  - Call *pooch.fetch()*.
  - Download it from Zenodo via your browser and extract the `.xlsx` data file from the `.zip` archive.
- Pass the file to `read_excel()` to read it into a `Scenario` and store it in a local DB via `commit()`.

CHAPTER

FIVE

REPRODUCIBILITY

On this page:

Elsewhere:

- A high-level introduction, to how testing supports validity, reproducibility, interoperability, and reusability, in `message_ix_models` and related packages.
- *Test utilities and fixtures (testing)* (`message_ix_models.testing`), on a separate page.

5.1 Strategy

The code in `model.bare` generates a "bare" reference energy system. This is a Scenario that has the same *structure* (ixmp 'sets') as actual instances of the MESSAGEix-GLOBIOM global model, but contains no *data* (ixmp 'parameter' values). Code that operates on the global model can be tested on the bare RES; if it works on that scenario, this is one indication (necessary, but not always sufficient) that it should work on fully-populated scenarios. Such tests are faster and lighter than testing on fully-populated scenarios, and make it easier to isolate errors in the code that is being tested.

5.2 Test suite (`message_ix_models.tests`)

`message_ix_models.tests` contains a suite of tests written using Pytest.

The following is automatically generated documentation of all modules, test classes, functions, and fixtures in the test suite. Each test **should** have a docstring explaining what it checks.

`tests`	Test suite for `message_ix_models`.

5.2.1 message_ix_models.tests

Test suite for *message_ix_models*.

Modules

message_ix_models.tests.model	Tests of *message_ix_models.model* and submodules.
message_ix_models.tests.project	
message_ix_models.tests.report	
message_ix_models.tests.test_cli	Basic tests of the command line.
message_ix_models.tests.test_import	
message_ix_models.tests.test_report	Tests for *message_ix_models.report*.
message_ix_models.tests.test_testing	
message_ix_models.tests.test_util	Tests of *message_ix_models.util*.
message_ix_models.tests.test_workflow	
message_ix_models.tests.tools	
message_ix_models.tests.util	Tests of submodules of *message_ix_models.util*.

message_ix_models.tests.model

Tests of *message_ix_models.model* and submodules.

Modules

message_ix_models.tests.model.test_bare	
message_ix_models.tests.model.test_build	
message_ix_models.tests.model.test_cli	
message_ix_models.tests.model.test_config	
message_ix_models.tests.model.test_disutility	Tests of *model.disutility*.
message_ix_models.tests.model.test_emissions	
message_ix_models.tests.model.test_macro	
message_ix_models.tests.model.test_snapshot	
message_ix_models.tests.model.test_structure	

message_ix_models.tests.model.test_bare

Module Attributes

SET_SIZE	Number of items in the respective YAML files.

message_ix_models.tests.model.test_bare.SET_SIZE

message_ix_models.tests.model.test_bare.**SET_SIZE = {'commodity': 18, 'level': 6, 'node': 15, 'relation': 20, 'technology': 377, 'year': 28}**

Number of items in the respective YAML files.

Functions

test_create_res(request, test_context, ...)	

message_ix_models.tests.model.test_bare.test_create_res

message_ix_models.tests.model.test_bare.**test_create_res**(*request*, *test_context*, *settings*, *expected*)

Classes

TestConfig()	

message_ix_models.tests.model.test_bare.TestConfig

class message_ix_models.tests.model.test_bare.**TestConfig**

Bases: object

__init__()

Methods

__init__()	
test_init()	

message_ix_models.tests.model.test_build

Functions

scenario(test_context)	
spec()	
test_apply_spec0(caplog, scenario, spec)	Require missing element raises ValueError.
test_apply_spec1(caplog, scenario, spec)	Add data using the data= argument.
test_apply_spec2(caplog, scenario, spec)	Remove an element, with fast=True.
test_apply_spec3(caplog, scenario, spec)	Actually remove data.

message_ix_models.tests.model.test_build.scenario

message_ix_models.tests.model.test_build.**scenario**(*test_context*)

message_ix_models.tests.model.test_build.spec

message_ix_models.tests.model.test_build.**spec**()

message_ix_models.tests.model.test_build.test_apply_spec0

message_ix_models.tests.model.test_build.**test_apply_spec0**(*caplog*, *scenario*, *spec*)

Require missing element raises ValueError.

message_ix_models.tests.model.test_build.test_apply_spec1

message_ix_models.tests.model.test_build.**test_apply_spec1**(*caplog*, *scenario*, *spec*)

Add data using the data= argument.

message_ix_models.tests.model.test_build.test_apply_spec2

message_ix_models.tests.model.test_build.**test_apply_spec2**(*caplog*, *scenario*, *spec*)

Remove an element, with fast=True.

message_ix_models.tests.model.test_build.test_apply_spec3

message_ix_models.tests.model.test_build.**test_apply_spec3**(*caplog*, *scenario*, *spec*)

Actually remove data.

message_ix_models.tests.model.test_cli

Functions

test_create_bare(mix_models_cli)	The `res create-bare` CLI command can be invoked.

message_ix_models.tests.model.test_cli.test_create_bare

message_ix_models.tests.model.test_cli.**test_create_bare**(*mix_models_cli*)

The `res create-bare` CLI command can be invoked.

message_ix_models.tests.model.test_config

Classes

TestConfig()	

message_ix_models.tests.model.test_config.TestConfig

class message_ix_models.tests.model.test_config.**TestConfig**

Bases: `object`

__init__()

Methods

__init__()	
test_check(values)	

message_ix_models.tests.model.test_disutility

Tests of *model.disutility*.

Functions

groups()	Fixture: list of 2 consumer groups.
minimal_test_data(scenario)	Generate data for *test_minimal()*.
scenario(request, test_context, techs)	Fixture: a Scenario with technologies given by *techs()*.
spec(groups, techs, template)	Fixture: a prepared spec for the minimal test case.
techs()	Fixture: list of 2 technologies for which groups can have disutility.
template()	Fixture: :class:.`Code` with annotations, for *disutility.get_spec()*.
test_add(scenario, groups, techs, template)	*disutility.add()* runs on the bare RES; the result solves.
test_data_conversion(scenario, spec)	*data_conversion()* runs.
test_data_source(scenario, spec)	*data_source()* runs.
test_get_data(scenario, spec)	*get_data()* runs.
test_get_spec(groups, techs, template)	*get_spec()* runs and produces expected output.
test_minimal(scenario, groups, techs, template)	Expected results are generated from a minimal test case.

message_ix_models.tests.model.test_disutility.groups

message_ix_models.tests.model.test_disutility.**groups**()

Fixture: list of 2 consumer groups.

message_ix_models.tests.model.test_disutility.minimal_test_data

message_ix_models.tests.model.test_disutility.**minimal_test_data**(*scenario*)

Generate data for *test_minimal()*.

message_ix_models.tests.model.test_disutility.scenario

message_ix_models.tests.model.test_disutility.**scenario**(*request*, *test_context*, *techs*)

Fixture: a Scenario with technologies given by *techs()*.

message_ix_models.tests.model.test_disutility.spec

message_ix_models.tests.model.test_disutility.**spec**(*groups*, *techs*, *template*)

Fixture: a prepared spec for the minimal test case.

message_ix_models.tests.model.test_disutility.techs

message_ix_models.tests.model.test_disutility.**techs**()

Fixture: list of 2 technologies for which groups can have disutility.

message_ix_models.tests.model.test_disutility.template

message_ix_models.tests.model.test_disutility.**template**()

Fixture: :class:.`Code` with annotations, for *disutility.get_spec()*.

message_ix_models.tests.model.test_disutility.test_add

message_ix_models.tests.model.test_disutility.**test_add**(*scenario*, *groups*, *techs*, *template*)

disutility.add() runs on the bare RES; the result solves.

message_ix_models.tests.model.test_disutility.test_data_conversion

message_ix_models.tests.model.test_disutility.**test_data_conversion**(*scenario*, *spec*)

data_conversion() runs.

message_ix_models.tests.model.test_disutility.test_data_source

message_ix_models.tests.model.test_disutility.**test_data_source**(*scenario*, *spec*)

data_source() runs.

message_ix_models.tests.model.test_disutility.test_get_data

message_ix_models.tests.model.test_disutility.**test_get_data**(*scenario*, *spec*)

get_data() runs.

message_ix_models.tests.model.test_disutility.test_get_spec

message_ix_models.tests.model.test_disutility.**test_get_spec**(*groups*, *techs*, *template*)

get_spec() runs and produces expected output.

message_ix_models.tests.model.test_disutility.test_minimal

message_ix_models.tests.model.test_disutility.**test_minimal**(*scenario*, *groups*, *techs*, *template*)

Expected results are generated from a minimal test case.

message_ix_models.tests.model.test_emissions

Functions

add_test_data(scenario)	
test_add_tax_emission(request, caplog, ...)	
test_get_emission_factors(units, exp_coal)	

message_ix_models.tests.model.test_emissions.add_test_data

message_ix_models.tests.model.test_emissions.**add_test_data**(*scenario*)

message_ix_models.tests.model.test_emissions.test_add_tax_emission

message_ix_models.tests.model.test_emissions.**test_add_tax_emission**(*request*, *caplog*, *test_context*)

message_ix_models.tests.model.test_emissions.test_get_emission_factors

message_ix_models.tests.model.test_emissions.**test_get_emission_factors**(*units*, *exp_coal*)

message_ix_models.tests.model.test_macro

Functions

test_generate0(test_context, parameter, value)
test_generate1(test_context)
test_load(test_context)

message_ix_models.tests.model.test_macro.test_generate0

message_ix_models.tests.model.test_macro.**test_generate0**(*test_context*, *parameter*, *value*)

message_ix_models.tests.model.test_macro.test_generate1

message_ix_models.tests.model.test_macro.**test_generate1**(*test_context*)

message_ix_models.tests.model.test_macro.test_load

message_ix_models.tests.model.test_macro.**test_load**(*test_context*)

message_ix_models.tests.model.test_snapshot

Functions

test_load(test_context, loaded_snapshot)

message_ix_models.tests.model.test_snapshot.test_load

message_ix_models.tests.model.test_snapshot.**test_load**(*test_context*, *loaded_snapshot*)

message_ix_models.tests.model.test_structure

Functions

test_cli_techs(session_context, mix_models_cli)	Test the *techs* CLI command.
test_codelists(kind, exp)	*codelists()* returns the expected IDs.
test_generate_set_elements(colour_expr, expected)	
test_process_units_anno()	

message_ix_models.tests.model.test_structure.test_cli_techs

message_ix_models.tests.model.test_structure.**test_cli_techs**(*session_context*, *mix_models_cli*)

Test the *techs* CLI command.

message_ix_models.tests.model.test_structure.test_codelists

message_ix_models.tests.model.test_structure.**test_codelists**(*kind*, *exp*)

codelists() returns the expected IDs.

message_ix_models.tests.model.test_structure.test_generate_set_elements

message_ix_models.tests.model.test_structure.**test_generate_set_elements**(*colour_expr*, *expected*)

message_ix_models.tests.model.test_structure.test_process_units_anno

message_ix_models.tests.model.test_structure.**test_process_units_anno**()

Classes

TestGetCodes()	Test *get_codes()* for different code lists.

message_ix_models.tests.model.test_structure.TestGetCodes

class message_ix_models.tests.model.test_structure.**TestGetCodes**

Bases: object

Test *get_codes()* for different code lists.

__init__()

Methods

__init__()	
test_commodities()	
test_get_codes(name)	The included code lists can be loaded.
test_hierarchy()	get_codes() returns objects with the expected hierarchical relationship.
test_levels()	
test_node_historic_country()	get_codes() handles ISO 3166 alpha-3 codes for historic countries.
test_nodes(codelist, to_check, length, member)	Tests of node codelists.
test_relation(codelist, length, expected)	
test_technologies()	
test_year(codelist, length)	Year code lists can be loaded and contain the correct number of codes.

test_get_codes(*name*)

The included code lists can be loaded.

test_hierarchy()

get_codes() returns objects with the expected hierarchical relationship.

test_node_historic_country()

get_codes() handles ISO 3166 alpha-3 codes for historic countries.

test_nodes(*codelist*, *to_check*, *length*, *member*)

Tests of node codelists.

test_year(*codelist*, *length*)

Year code lists can be loaded and contain the correct number of codes.

Seealso

TestScenarioInfo.test_year_from_codes().

message_ix_models.tests.project

Modules

message_ix_models.tests.project.test_advance
message_ix_models.tests.project.test_gea
message_ix_models.tests.project.test_shape
message_ix_models.tests.project.test_ssp

message_ix_models.tests.project.test_advance

Classes

TestADVANCE()

message_ix_models.tests.project.test_advance.TestADVANCE

class message_ix_models.tests.project.test_advance.**TestADVANCE**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, ...)

message_ix_models.tests.project.test_gea

Classes

TestGEA()

message_ix_models.tests.project.test_gea.TestGEA

class message_ix_models.tests.project.test_gea.**TestGEA**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, ...)

message_ix_models.tests.project.test_shape

Functions

test_get_shape_data(test_context, args, size)

message_ix_models.tests.project.test_shape.test_get_shape_data

message_ix_models.tests.project.test_shape.**test_get_shape_data**(*test_context*, *args*, *size*)

Classes

TestSHAPE()

message_ix_models.tests.project.test_shape.TestSHAPE

class message_ix_models.tests.project.test_shape.**TestSHAPE**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, ...)

message_ix_models.tests.project.test_ssp

Functions

test_cli(mix_models_cli)
test_enum()
test_generate(tmp_path, test_context)
test_parse(value, expected)
test_ssp_field()

message_ix_models.tests.project.test_ssp.test_cli

message_ix_models.tests.project.test_ssp.**test_cli**(*mix_models_cli*)

message_ix_models.tests.project.test_ssp.test_enum

message_ix_models.tests.project.test_ssp.**test_enum**()

message_ix_models.tests.project.test_ssp.test_generate

message_ix_models.tests.project.test_ssp.**test_generate**(*tmp_path*, *test_context*)

message_ix_models.tests.project.test_ssp.test_parse

message_ix_models.tests.project.test_ssp.**test_parse**(*value*, *expected*)

message_ix_models.tests.project.test_ssp.test_ssp_field

message_ix_models.tests.project.test_ssp.**test_ssp_field**() → None

Classes

TestSSPOriginal()
TestSSPUpdate()

message_ix_models.tests.project.test_ssp.TestSSPOriginal

class message_ix_models.tests.project.test_ssp.**TestSSPOriginal**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, source, ...)

message_ix_models.tests.project.test_ssp.TestSSPUpdate

class message_ix_models.tests.project.test_ssp.**TestSSPUpdate**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, source, ...)

message_ix_models.tests.report

Modules

message_ix_models.tests.report.test_compat	
message_ix_models.tests.report.test_config	
message_ix_models.tests.report.test_legacy	
message_ix_models.tests.report.test_operator	

message_ix_models.tests.report.test_compat

Functions

test_compat(tmp_path, test_context)	
test_prepare_techs(test_context)	
test_prepare_techs_filter_error(caplog, ...)	*prepare_techs()* logs warnings for invalid filters.

message_ix_models.tests.report.test_compat.test_compat

message_ix_models.tests.report.test_compat.**test_compat**(*tmp_path*, *test_context*)

message_ix_models.tests.report.test_compat.test_prepare_techs

message_ix_models.tests.report.test_compat.**test_prepare_techs**(*test_context*)

message_ix_models.tests.report.test_compat.test_prepare_techs_filter_error

message_ix_models.tests.report.test_compat.**test_prepare_techs_filter_error**(*caplog*, *monkeypatch*)

prepare_techs() logs warnings for invalid filters.

message_ix_models.tests.report.test_config

Classes

TestConfig()	

message_ix_models.tests.report.test_config.TestConfig

class message_ix_models.tests.report.test_config.**TestConfig**

Bases: object

__init__()

Methods

__init__()	
test_use_file(tmp_path)	

message_ix_models.tests.report.test_legacy

Functions

test_legacy_report(test_context, loaded_snapshot)	

message_ix_models.tests.report.test_legacy.test_legacy_report

message_ix_models.tests.report.test_legacy.**test_legacy_report**(*test_context*, *loaded_snapshot*)

message_ix_models.tests.report.test_operator

Functions

c()	
scenario(test_context)	
test_compound_growth()	*compound_growth()* on a 2-D quantity.
test_filter_ts()	
test_from_url(scenario)	
test_get_remove_ts(caplog, scenario)	
test_gwp_factors()	
test_make_output_path(tmp_path, c)	
test_model_periods()	
test_share_curtailment()	

message_ix_models.tests.report.test_operator.c

message_ix_models.tests.report.test_operator.**c**() → Computer

message_ix_models.tests.report.test_operator.scenario

message_ix_models.tests.report.test_operator.**scenario**(*test_context*)

message_ix_models.tests.report.test_operator.test_compound_growth

message_ix_models.tests.report.test_operator.**test_compound_growth**()
compound_growth() on a 2-D quantity.

message_ix_models.tests.report.test_operator.test_filter_ts

message_ix_models.tests.report.test_operator.**test_filter_ts**()

message_ix_models.tests.report.test_operator.test_from_url

message_ix_models.tests.report.test_operator.**test_from_url**(*scenario*)

message_ix_models.tests.report.test_operator.test_get_remove_ts

message_ix_models.tests.report.test_operator.**test_get_remove_ts**(*caplog*, *scenario*)

message_ix_models.tests.report.test_operator.test_gwp_factors

message_ix_models.tests.report.test_operator.**test_gwp_factors**()

message_ix_models.tests.report.test_operator.test_make_output_path

message_ix_models.tests.report.test_operator.**test_make_output_path**(*tmp_path*, *c*)

message_ix_models.tests.report.test_operator.test_model_periods

message_ix_models.tests.report.test_operator.**test_model_periods**()

message_ix_models.tests.report.test_operator.test_share_curtailment

message_ix_models.tests.report.test_operator.**test_share_curtailment**()

message_ix_models.tests.test_cli

Basic tests of the command line.

Functions

test_cli_debug(mix_models_cli)	The 'debug' CLI command can be invoked.
test_cli_export_test_data(monkeypatch, ...)	The **export-test-data** command can be invoked.
test_cli_help(mix_models_cli, command)	--help works for every CLI command.

message_ix_models.tests.test_cli.test_cli_debug

message_ix_models.tests.test_cli.**test_cli_debug**(*mix_models_cli*)

The 'debug' CLI command can be invoked.

message_ix_models.tests.test_cli.test_cli_export_test_data

message_ix_models.tests.test_cli.**test_cli_export_test_data**(*monkeypatch*, *tmp_path*, *test_context*, *mix_models_cli*)

The **export-test-data** command can be invoked.

message_ix_models.tests.test_cli.test_cli_help

message_ix_models.tests.test_cli.**test_cli_help**(*mix_models_cli*, *command*)

–help works for every CLI command.

message_ix_models.tests.test_import

Functions

test_import(name)	Check that modules can be imported.

message_ix_models.tests.test_import.test_import

message_ix_models.tests.test_import.**test_import**(*name*)

Check that modules can be imported.

Modules **should** have specific tests that actually run code. Where those tests are not yet written, this test checks that the modules can at least be imported without error.

Once tests are written for each module, remove it from the above list.

message_ix_models.tests.test_report

Tests for *message_ix_models.report*.

Functions

simulated_solution_reporter()	Reporter with a simulated solution for snapshot 0.
test_add_simulated_solution(test_context, ...)	
test_apply_units(request, test_context, regions)	
test_cli(mix_models_cli)	
test_collapse(input, exp)	Test *report.util.collapse()* and use of *REPLACE_VARS*.
test_prepare_reporter(test_context)	
test_register(caplog)	
test_report_bare_res(request, tmp_path, ...)	Prepare and run the standard MESSAGE-GLOBIOM reporting on a bare RES.
test_report_deprecated(caplog, request, ...)	
test_report_legacy(caplog, request, ...)	Legacy reporting can be invoked via *message_ix_models.report.report()*.

message_ix_models.tests.test_report.simulated_solution_reporter

message_ix_models.tests.test_report.**simulated_solution_reporter**()

Reporter with a simulated solution for snapshot 0.

This uses *add_simulated_solution()*, so test functions that use it should be marked with **@to_simulate**.minimum_version.

message_ix_models.tests.test_report.test_add_simulated_solution

message_ix_models.tests.test_report.**test_add_simulated_solution**(*test_context*, *test_data_path*)

message_ix_models.tests.test_report.test_apply_units

message_ix_models.tests.test_report.**test_apply_units**(*request*, *test_context*, *regions*)

message_ix_models.tests.test_report.test_cli

message_ix_models.tests.test_report.**test_cli**(*mix_models_cli*)

message_ix_models.tests.test_report.test_collapse

message_ix_models.tests.test_report.**test_collapse**(*input*, *exp*)

Test *report.util.collapse()* and use of *REPLACE_VARS*.

This test is parametrized with example input and expected output strings for the `variable` IAMC column. There should be ≥1 example for each pattern in *REPLACE_VARS*.

When adding test cases, if the pattern does not start with `^` or end with `$`, then prefix "x " or suffix " x" respectively to ensure these are handled as intended.

Todo: Extend or duplicate to also cover *REPLACE_DIMS*.

message_ix_models.tests.test_report.test_prepare_reporter

message_ix_models.tests.test_report.**test_prepare_reporter**(*test_context*)

message_ix_models.tests.test_report.test_register

message_ix_models.tests.test_report.**test_register**(*caplog*)

message_ix_models.tests.test_report.test_report_bare_res

message_ix_models.tests.test_report.**test_report_bare_res**(*request*, *tmp_path*, *test_context*)

Prepare and run the standard MESSAGE-GLOBIOM reporting on a bare RES.

message_ix_models.tests.test_report.test_report_deprecated

message_ix_models.tests.test_report.**test_report_deprecated**(*caplog*, *request*, *tmp_path*, *test_context*)

message_ix_models.tests.test_report.test_report_legacy

message_ix_models.tests.test_report.**test_report_legacy**(*caplog*, *request*, *tmp_path*, *test_context*)

Legacy reporting can be invoked via *message_ix_models.report.report()*.

message_ix_models.tests.test_testing

Functions

test_bare_res_no_request(test_context)	*bare_res()* works with *request* = None.
test_bare_res_solved(request, test_context)	*bare_res()* works with *solve* = True.
test_cli_runner(mix_models_cli)	
test_not_ci_skip()	Test not_ci(action="skip").
test_not_ci_xfail()	Test not_ci(action="skip").

message_ix_models.tests.test_testing.test_bare_res_no_request

message_ix_models.tests.test_testing.**test_bare_res_no_request**(*test_context*)

bare_res() works with *request* = None.

message_ix_models.tests.test_testing.test_bare_res_solved

message_ix_models.tests.test_testing.**test_bare_res_solved**(*request*, *test_context*)

bare_res() works with *solve* = True.

This test can be removed once this feature of the test function is used by another test.

message_ix_models.tests.test_testing.test_cli_runner

message_ix_models.tests.test_testing.**test_cli_runner**(*mix_models_cli*)

message_ix_models.tests.test_testing.test_not_ci_skip

message_ix_models.tests.test_testing.**test_not_ci_skip**()

Test not_ci(action="skip").

message_ix_models.tests.test_testing.test_not_ci_xfail

message_ix_models.tests.test_testing.**test_not_ci_xfail**()

Test not_ci(action="skip").

message_ix_models.tests.test_util

Tests of *message_ix_models.util*.

Functions

test_as_codes()	Forward reference to a child is silently dropped.
test_as_codes_invalid(data)	as_codes() rejects invalid data.
test_broadcast(caplog)	
test_check_support(test_context)	*check_support()* raises an exception for missing/non-matching values.
test_convert_units(recwarn)	*convert_units()* and *series_of_pint_quantity()* work.
test_copy_column()	
test_ffill()	
test_iter_parameters(test_context)	Parameters indexed by set 'node' can be retrieved.
test_load_package_data(path)	Existing package data can be loaded.
test_load_package_data_invalid()	load_package_data() raises an exception for an unsupported file type.
test_load_package_data_twice(caplog)	Loading the same data twice logs a message.
test_load_private_data(*parts[, suffix])	
test_local_data_path(tmp_path_factory, ...)	
test_make_source_tech0()	
test_make_source_tech1(test_mp)	Test make_source_tech() with a Scenario object as input.
test_maybe_query()	*maybe_query()* works as intended.
test_package_data_path()	
test_path_fallback(caplog)	
test_private_data_path()	
test_replace_par_data(caplog, test_context)	Test *replace_par_data()*.
test_same(name, func, col)	Test both *same_node()* and *same_time()*.
test_strip_par_data(caplog, test_context)	Test the "dry run" feature of *strip_par_data()*.

message_ix_models.tests.test_util.test_as_codes

message_ix_models.tests.test_util.**test_as_codes**()

Forward reference to a child is silently dropped.

message_ix_models.tests.test_util.test_as_codes_invalid

message_ix_models.tests.test_util.**test_as_codes_invalid**(*data*)

as_codes() rejects invalid data.

message_ix_models.tests.test_util.test_broadcast

message_ix_models.tests.test_util.**test_broadcast**(*caplog*)

message_ix_models.tests.test_util.test_check_support

message_ix_models.tests.test_util.**test_check_support**(*test_context*)

check_support() raises an exception for missing/non-matching values.

message_ix_models.tests.test_util.test_convert_units

message_ix_models.tests.test_util.**test_convert_units**(*recwarn*)

convert_units() and *series_of_pint_quantity()* work.

message_ix_models.tests.test_util.test_copy_column

message_ix_models.tests.test_util.**test_copy_column**()

message_ix_models.tests.test_util.test_ffill

message_ix_models.tests.test_util.**test_ffill**()

message_ix_models.tests.test_util.test_iter_parameters

message_ix_models.tests.test_util.**test_iter_parameters**(*test_context*)

Parameters indexed by set 'node' can be retrieved.

message_ix_models.tests.test_util.test_load_package_data

message_ix_models.tests.test_util.**test_load_package_data**(*path*)

Existing package data can be loaded.

message_ix_models.tests.test_util.test_load_package_data_invalid

message_ix_models.tests.test_util.**test_load_package_data_invalid**()

load_package_data() raises an exception for an unsupported file type.

message_ix_models.tests.test_util.test_load_package_data_twice

message_ix_models.tests.test_util.**test_load_package_data_twice**(*caplog*)

Loading the same data twice logs a message.

message_ix_models.tests.test_util.test_load_private_data

message_ix_models.tests.test_util.**test_load_private_data**(**parts*, *suffix=None*)

message_ix_models.tests.test_util.test_local_data_path

message_ix_models.tests.test_util.**test_local_data_path**(*tmp_path_factory*, *session_context*)

message_ix_models.tests.test_util.test_make_source_tech0

message_ix_models.tests.test_util.**test_make_source_tech0**()

message_ix_models.tests.test_util.test_make_source_tech1

message_ix_models.tests.test_util.**test_make_source_tech1**(*test_mp*)

Test make_source_tech() with a Scenario object as input.

message_ix_models.tests.test_util.test_maybe_query

message_ix_models.tests.test_util.**test_maybe_query**()

maybe_query() works as intended.

message_ix_models.tests.test_util.test_package_data_path

message_ix_models.tests.test_util.**test_package_data_path**()

message_ix_models.tests.test_util.test_path_fallback

message_ix_models.tests.test_util.**test_path_fallback**(*caplog*)

message_ix_models.tests.test_util.test_private_data_path

message_ix_models.tests.test_util.**test_private_data_path**()

message_ix_models.tests.test_util.test_replace_par_data

message_ix_models.tests.test_util.**test_replace_par_data**(*caplog*, *test_context*)

Test *replace_par_data()*.

message_ix_models.tests.test_util.test_same

message_ix_models.tests.test_util.**test_same**(*name*, *func*, *col*)

Test both *same_node()* and *same_time()*.

message_ix_models.tests.test_util.test_strip_par_data

message_ix_models.tests.test_util.**test_strip_par_data**(*caplog*, *test_context*)

Test the "dry run" feature of *strip_par_data()*.

message_ix_models.tests.test_workflow

Functions

changes_a(c, s)	Change a scenario by modifying structure data, but not data.
changes_b(c, s[, value])	Change a scenario by modifying parameter data, but not structure.
test_make_click_command(mix_models_cli)	
test_workflow(caplog, request, test_context, wf)	
wf(request, test_context)	

message_ix_models.tests.test_workflow.changes_a

message_ix_models.tests.test_workflow.**changes_a**(*c*, *s*) → None

Change a scenario by modifying structure data, but not data.

message_ix_models.tests.test_workflow.changes_b

message_ix_models.tests.test_workflow.**changes_b**(*c*, *s*, *value=None*) → None

Change a scenario by modifying parameter data, but not structure.

message_ix_models.tests.test_workflow.test_make_click_command

message_ix_models.tests.test_workflow.**test_make_click_command**(*mix_models_cli*) → None

message_ix_models.tests.test_workflow.test_workflow

message_ix_models.tests.test_workflow.**test_workflow**(*caplog*, *request*, *test_context*, *wf*) → None

message_ix_models.tests.test_workflow.wf

message_ix_models.tests.test_workflow.**wf**(*request*, *test_context*) → *Workflow*

Classes

TestWorkflowStep()

message_ix_models.tests.test_workflow.TestWorkflowStep

class message_ix_models.tests.test_workflow.**TestWorkflowStep**

Bases: object

__init__()

Methods

__init__()
test_call(test_context)
test_repr()

message_ix_models.tests.tools

Modules

message_ix_models.tests.tools.costs
message_ix_models.tests.tools.iea
message_ix_models.tests.tools.test_advance
message_ix_models.tests.tools.test_exo_data
message_ix_models.tests.tools.test_gfei
message_ix_models.tests.tools.test_iamc
message_ix_models.tests.tools.test_wb

message_ix_models.tests.tools.costs

Modules

message_ix_models.tests.tools.costs.test_decay	
message_ix_models.tests.tools.costs.test_gdp	
message_ix_models.tests.tools.costs.test_projections	
message_ix_models.tests.tools.costs.test_regional_differentiation	

message_ix_models.tests.tools.costs.test_decay

Functions

test_get_cost_reduction_data(module, t_exp)	
test_get_technology_reduction_scenarios_data(module)	
test_project_ref_region_inv_costs_using_reduction_rates(...)	

message_ix_models.tests.tools.costs.test_decay.test_get_cost_reduction_data

message_ix_models.tests.tools.costs.test_decay.**test_get_cost_reduction_data**(*module: str*, *t_exp*) → None

message_ix_models.tests.tools.costs.test_decay.test_get_technology_reduction_scenarios_data

message_ix_models.tests.tools.costs.test_decay.**test_get_technology_reduction_scenarios_d**

message_ix_models.tests.tools.costs.test_decay.test_project_ref_region_inv_costs_using_reduction_rates

message_ix_models.tests.tools.costs.test_decay.**test_project_ref_region_inv_costs_using_r**

message_ix_models.tests.tools.costs.test_gdp

Functions

test_adjust_cost_ratios_with_gdp(...)	
test_process_raw_ssp_data(test_context, node)	

message_ix_models.tests.tools.costs.test_gdp.test_adjust_cost_ratios_with_gdp

message_ix_models.tests.tools.costs.test_gdp.**test_adjust_cost_ratios_with_gdp**(*test_context*, *module*) → None

message_ix_models.tests.tools.costs.test_gdp.test_process_raw_ssp_data

message_ix_models.tests.tools.costs.test_gdp.**test_process_raw_ssp_data**(*test_context*, *node*) → None

message_ix_models.tests.tools.costs.test_projections

Functions

test_bare_res(request, test_context, node)	Costs data can be added to the bare RES and solved.
test_create_cost_projections(config, ...)	

message_ix_models.tests.tools.costs.test_projections.test_bare_res

message_ix_models.tests.tools.costs.test_projections.**test_bare_res**(*request*, *test_context*, *node*)

Costs data can be added to the bare RES and solved.

message_ix_models.tests.tools.costs.test_projections.test_create_cost_projections

message_ix_models.tests.tools.costs.test_projections.**test_create_cost_projections**(*config*, *exp_fix*, *exp_inv*) → None

message_ix_models.tests.tools.costs.test_regional_differentiation

Functions

test_adjust_technology_mapping(module)	
test_apply_regional_differentiation(module, ...)	Regional differentiation is applied correctly for each *module*.
test_get_intratec_data()	
test_get_raw_technology_mapping(module, ...)	
test_get_weo_data()	

message_ix_models.tests.tools.costs.test_regional_differentiation.test_adjust_technology_mapping

message_ix_models.tests.tools.costs.test_regional_differentiation.**test_adjust_technology**

message_ix_models.tests.tools.costs.test_regional_differentiation.test_apply_regional_differentiation

message_ix_models.tests.tools.costs.test_regional_differentiation.**test_apply_regional_di**

Regional differentiation is applied correctly for each *module*.

message_ix_models.tests.tools.costs.test_regional_differentiation.test_get_intratec_data

message_ix_models.tests.tools.costs.test_regional_differentiation.**test_get_intratec_data**

message_ix_models.tests.tools.costs.test_regional_differentiation.test_get_raw_technology_mapping

message_ix_models.tests.tools.costs.test_regional_differentiation.**test_get_raw_technolog**

message_ix_models.tests.tools.costs.test_regional_differentiation.test_get_weo_data

message_ix_models.tests.tools.costs.test_regional_differentiation.**test_get_weo_data**()
→ None

message_ix_models.tests.tools.iea

Modules

message_ix_models.tests.tools.iea.test_eei	
message_ix_models.tests.tools.iea.test_web	Tests of *tools*.

message_ix_models.tests.tools.iea.test_eei

Classes

TestIEA_EEI()

message_ix_models.tests.tools.iea.test_eei.TestIEA_EEI

class message_ix_models.tests.tools.iea.test_eei.**TestIEA_EEI**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, ...)

message_ix_models.tests.tools.iea.test_web

Tests of *tools*.

Functions

test_generate_code_lists(tmp_path, provider, ...)
test_load_codelists(urn, N)
test_load_data(test_context, tmp_path, ...)

message_ix_models.tests.tools.iea.test_web.test_generate_code_lists

message_ix_models.tests.tools.iea.test_web.**test_generate_code_lists**(*tmp_path*, *provider*, *edition*)

message_ix_models.tests.tools.iea.test_web.test_load_codelists

message_ix_models.tests.tools.iea.test_web.**test_load_codelists**(*urn*, *N*)

message_ix_models.tests.tools.iea.test_web.test_load_data

message_ix_models.tests.tools.iea.test_web.**test_load_data**(*test_context*, *tmp_path*, *provider*, *edition*)

Classes

TestIEA_EWEB()

message_ix_models.tests.tools.iea.test_web.TestIEA_EWEB

class message_ix_models.tests.tools.iea.test_web.**TestIEA_EWEB**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, source, ...)

message_ix_models.tests.tools.test_advance

Functions

advance_test_data(session_context)	
test_get_advance_data(session_context)	Test *get_advance_data()*.

message_ix_models.tests.tools.test_advance.advance_test_data

message_ix_models.tests.tools.test_advance.**advance_test_data**(*session_context*)

message_ix_models.tests.tools.test_advance.test_get_advance_data

message_ix_models.tests.tools.test_advance.**test_get_advance_data**(*session_context*)

Test *get_advance_data()*.

message_ix_models.tests.tools.test_exo_data

Functions

test_operator(test_context, regions, N_n)	Exogenous data calculations can be set up through Computer.add().
test_prepare_computer(test_context, regions, N_n)	*exo_data.prepare_computer()* works as intended.
test_prepare_computer_exc(test_context)	

message_ix_models.tests.tools.test_exo_data.test_operator

message_ix_models.tests.tools.test_exo_data.**test_operator**(*test_context*, *regions*, *N_n*)

Exogenous data calculations can be set up through Computer.add().

message_ix_models.tests.tools.test_exo_data.test_prepare_computer

message_ix_models.tests.tools.test_exo_data.**test_prepare_computer**(*test_context*, *regions*, *N_n*)

exo_data.prepare_computer() works as intended.

message_ix_models.tests.tools.test_exo_data.test_prepare_computer_exc

message_ix_models.tests.tools.test_exo_data.**test_prepare_computer_exc**(*test_context*)

Classes

TestExoDataSource()	

message_ix_models.tests.tools.test_exo_data.TestExoDataSource

class message_ix_models.tests.tools.test_exo_data.**TestExoDataSource**

Bases: object

__init__()

Methods

__init__()	
test_abstract()	
test_register_source()	

message_ix_models.tests.tools.test_gfei

Classes

TestGFEI()

message_ix_models.tests.tools.test_gfei.TestGFEI

class message_ix_models.tests.tools.test_gfei.**TestGFEI**

Bases: object

__init__()

Methods

__init__()
test_prepare_computer(test_context, regions, ...)

message_ix_models.tests.tools.test_iamc

Functions

test_describe(test_context)

message_ix_models.tests.tools.test_iamc.test_describe

message_ix_models.tests.tools.test_iamc.**test_describe**(*test_context*)

message_ix_models.tests.tools.test_wb

Functions

cl_ig()
test_assign_income_groups(cl_ig, nodes, ...)
test_get_income_group_codelist(cl_ig)
test_make_map()

message_ix_models.tests.tools.test_wb.cl_ig

message_ix_models.tests.tools.test_wb.**cl_ig**() → sdmx.model.common.Codelist

message_ix_models.tests.tools.test_wb.test_assign_income_groups

message_ix_models.tests.tools.test_wb.**test_assign_income_groups**(*cl_ig: sdmx.model.common.Codelist*, *nodes: str*, *method: str*, *replace: int | None*) → None

message_ix_models.tests.tools.test_wb.test_get_income_group_codelist

message_ix_models.tests.tools.test_wb.**test_get_income_group_codelist**(*cl_ig: sdmx.model.common.Codelist*) → None

message_ix_models.tests.tools.test_wb.test_make_map

message_ix_models.tests.tools.test_wb.**test_make_map**() → None

message_ix_models.tests.util

Tests of submodules of *message_ix_models.util*.

Modules

message_ix_models.tests.util.test_cache	
message_ix_models.tests.util.test_click	Basic tests of the command line.
message_ix_models.tests.util.test_config	
message_ix_models.tests.util.test_context	
message_ix_models.tests.util.test_logging	
message_ix_models.tests.util.test_node	Tests of *message_ix_models.util.node*.
message_ix_models.tests.util.test_scenarioinfo	
message_ix_models.tests.util.test_sdmx	

message_ix_models.tests.util.test_cache

Functions

test_cached(caplog, test_context, tmp_path)	*cached()* works as expected.

message_ix_models.tests.util.test_cache.test_cached

message_ix_models.tests.util.test_cache.**test_cached**(*caplog*, *test_context*, *tmp_path*)

cached() works as expected.

Todo: test behaviour when *SKIP_CACHE* is `True`

Classes

TestEncoder()	

message_ix_models.tests.util.test_cache.TestEncoder

class message_ix_models.tests.util.test_cache.**TestEncoder**

Bases: `object`

__init__()

Methods

__init__()	
test_sdmx()	*message_ix_models* configures `Encoder` for `Code`.

test_sdmx()

message_ix_models configures `Encoder` for `Code`.

message_ix_models.tests.util.test_click

Basic tests of the command line.

Functions

test_default_path_cb(session_context, ...)	Test *default_path_cb()*.
test_regions(mix_models_cli)	--regions=… used on both group and a command within the group.
test_store_context(mix_models_cli)	Test *store_context()*.
test_urls_from_file(mix_models_cli, tmp_path)	Test *urls_from_file()* callback.

message_ix_models.tests.util.test_click.test_default_path_cb

message_ix_models.tests.util.test_click.**test_default_path_cb**(*session_context*, *mix_models_cli*)

Test *default_path_cb()*.

message_ix_models.tests.util.test_click.test_regions

message_ix_models.tests.util.test_click.**test_regions**(*mix_models_cli*)

–regions=… used on both group and a command within the group.

If the option is not provided to the inner command, the value given to the outer group should persist.

message_ix_models.tests.util.test_click.test_store_context

message_ix_models.tests.util.test_click.**test_store_context**(*mix_models_cli*)

Test *store_context()*.

message_ix_models.tests.util.test_click.test_urls_from_file

message_ix_models.tests.util.test_click.**test_urls_from_file**(*mix_models_cli*, *tmp_path*)

Test *urls_from_file()* callback.

message_ix_models.tests.util.test_config

Classes

TestConfigHelper()	

message_ix_models.tests.util.test_config.TestConfigHelper

class message_ix_models.tests.util.test_config.**TestConfigHelper**

Bases: object

__init__()

Methods

__init__()	
c(cls)	
c2(cls, cls2)	
cls()	A class which inherits from ConfigHelper.
cls2(cls)	A class with an attribute.
test_canonical_name(cls)	
test_from_dict(cls, c)	
test_read_file(caplog, tmp_path, cls, cls2, ...)	
test_replace(c)	
test_update(c)	*ConfigHelper.update()* raises AttributeError.

cls() → Type

A class which inherits from ConfigHelper.

cls2(*cls*) → Type

A class with an attribute.

test_update(*c*)

ConfigHelper.update() raises AttributeError.

message_ix_models.tests.util.test_context

Classes

TestContext()

message_ix_models.tests.util.test_context.TestContext

class message_ix_models.tests.util.test_context.**TestContext**

Bases: object

__init__()

Methods

__init__()	
test_clone_to_dest(caplog, test_context)	
test_dealias(caplog)	Aliasing works with Context.__init__(), *Context.update()*.
test_deepcopy(session_context)	Paths are preserved through deepcopy().
test_default_value(test_context)	
test_get_cache_path(pytestconfig, test_context)	cache_path() returns the expected output.
test_get_instance(session_context)	
test_get_local_path(tmp_path_factory, ...)	
test_get_platform(session_context)	
test_get_scenario(test_context)	
test_handle_cli_args()	
test_only()	
test_repr()	
test_set_scenario(test_context)	
test_use_defaults(caplog)	
test_write_debug_archive(mix_models_cli)	*write_debug_archive()* works.

test_dealias(*caplog*)

Aliasing works with Context.__init__(), *Context.update()*.

test_deepcopy(*session_context*)

Paths are preserved through deepcopy().

test_get_cache_path(*pytestconfig*, *test_context*)

cache_path() returns the expected output.

test_write_debug_archive(*mix_models_cli*)

write_debug_archive() works.

message_ix_models.tests.util.test_logging

Functions

test_mark_time(caplog)	
test_silence_log(caplog)	

message_ix_models.tests.util.test_logging.test_mark_time

message_ix_models.tests.util.test_logging.**test_mark_time**(*caplog*)

message_ix_models.tests.util.test_logging.test_silence_log

message_ix_models.tests.util.test_logging.**test_silence_log**(*caplog*)

Classes

TestQueueListener()

message_ix_models.tests.util.test_logging.TestQueueListener

class message_ix_models.tests.util.test_logging.**TestQueueListener**

Bases: object

__init__()

Methods

__init__()	
test_flush(caplog, mix_models_cli, method, k)	Test logging in multiple processes, multiple threads, and with click.

Attributes

N	Number of log messages to emit.
k	Number of times to run the test.

N = 1000

Number of log messages to emit.

k = 4

Number of times to run the test.

test_flush(*caplog*, *mix_models_cli*, *method*, *k*)

Test logging in multiple processes, multiple threads, and with click.

With pytest-xdist, these *k* test cases will run in multiple processes. Each process will have its main thread, and the thread of the QueueListener. The test ensures that all *N* log records emitted by the func() are "flushed" from the queue, transferred to stdout by the *StreamHandler* and captured by the *CliRunner*.

message_ix_models.tests.util.test_node

Tests of *message_ix_models.util.node*.

Functions

input()	Fixture: test data for *adapt_R11_R14*.
test_adapt_df(input, func, N, expected, ...)	*adapt_R11_R14* handles pandas.DataFrame.
test_adapt_qty(input, func, expected, node_loc)	*adapt_R11_R14* handles genno.Quantity.
test_identify_nodes(caplog, test_context, ...)	
test_identify_nodes1(test_context)	
test_mapping_adapter()	Generic test of MappingAdapter.

message_ix_models.tests.util.test_node.input

message_ix_models.tests.util.test_node.**input**()

Fixture: test data for *adapt_R11_R14*.

message_ix_models.tests.util.test_node.test_adapt_df

message_ix_models.tests.util.test_node.**test_adapt_df**(*input*, *func*, *N*, *expected*, *target_nodes*)

adapt_R11_R14 handles pandas.DataFrame.

message_ix_models.tests.util.test_node.test_adapt_qty

message_ix_models.tests.util.test_node.**test_adapt_qty**(*input*, *func*, *expected*, *node_loc*)

adapt_R11_R14 handles genno.Quantity.

message_ix_models.tests.util.test_node.test_identify_nodes

message_ix_models.tests.util.test_node.**test_identify_nodes**(*caplog*, *test_context*, *regions*)

message_ix_models.tests.util.test_node.test_identify_nodes1

message_ix_models.tests.util.test_node.**test_identify_nodes1**(*test_context*)

message_ix_models.tests.util.test_node.test_mapping_adapter

message_ix_models.tests.util.test_node.**test_mapping_adapter**()

Generic test of MappingAdapter.

message_ix_models.tests.util.test_scenarioinfo

Classes

TestScenarioInfo()	
TestSpec()	

message_ix_models.tests.util.test_scenarioinfo.TestScenarioInfo

class message_ix_models.tests.util.test_scenarioinfo.**TestScenarioInfo**

Bases: object

__init__()

Methods

__init__()	
test_empty()	ScenarioInfo created from scratch.
test_from_scenario(test_context)	ScenarioInfo initialized from an existing Scenario.
test_from_url()	
test_iter()	
test_path(input, expected)	
test_repr()	
test_units(caplog)	Test both *io_units()* and *units_for()*.
test_update()	
test_url()	
test_year_from_codes(caplog, codelist, y0, ...)	

test_empty()

ScenarioInfo created from scratch.

test_from_scenario(*test_context*) → None

ScenarioInfo initialized from an existing Scenario.

test_units(*caplog*)

Test both *io_units()* and *units_for()*.

message_ix_models.tests.util.test_scenarioinfo.TestSpec

class message_ix_models.tests.util.test_scenarioinfo.**TestSpec**

Bases: object

__init__()

Methods

__init__()	
test_getitem()	
test_merge()	
test_setitem()	

message_ix_models.tests.util.test_sdmx

Functions

test_eval_anno(caplog, recwarn)	
test_make_enum()	*make_enum()* works with Flag and subclasses.
test_read0(urn, expected)	
test_read1()	

message_ix_models.tests.util.test_sdmx.test_eval_anno

message_ix_models.tests.util.test_sdmx.**test_eval_anno**(*caplog*, *recwarn*)

message_ix_models.tests.util.test_sdmx.test_make_enum

message_ix_models.tests.util.test_sdmx.**test_make_enum**()

make_enum() works with Flag and subclasses.

message_ix_models.tests.util.test_sdmx.test_read0

message_ix_models.tests.util.test_sdmx.**test_read0**(*urn*, *expected*)

message_ix_models.tests.util.test_sdmx.test_read1

message_ix_models.tests.util.test_sdmx.**test_read1**()

5.3 Running the test suite

Some notes for running the test suite.

cached() is used to cache the data resulting from slow operations, like parsing large input files. Data are stored in a location described by the *Context* setting `cache_path`. The test suite interacts with caches in two ways:

- `--local-cache`: Giving this option causes pytest to use whatever cache directory is configured for normal runs/usage of *message_ix_models* or **mix-models**.
- By default (without `--local-cache`), the test suite uses pytest's built-in cache fixture. This creates and uses a temporary directory, usually `.pytest_cache/d/cache/` within the repository root. This location is used *only* by tests, and not by normal runs/usage of the code.

In either case:

- The tests use existing cached data in these locations and skip over code that generates this data. If the generating code is changed, the cached data **must** be deleted in order to actually check that the code runs properly.
- Running the test suite with `--local-cache` causes the local cache to be populated, and this will affect subsequent runs.
- The continuous integration (below) services don't preserve caches, so code always runs.

5.4 Continuous testing

The test suite (*message_ix_models.tests*) is run using GitHub Actions for new commits on the `main` branch; new commits on any branch associated with a pull request; and on a daily schedule. These ensure that the code is functional and produces expected outputs, even as upstream dependencies evolve. Workflow runs and their outputs can be viewed here.

Because it is closed-source and requires access to internal IIASA resources, including databases, continuous integration for `message_data` is handled by GitHub Actions self-hosted runners running on IIASA systems.

5.5 Prepare data for testing

Use the `export-test-data` CLI command:

```
mix-models --url="ixmp://ixmp-dev/ENGAGE_SSP2_v4.1.7/baseline" export-test-data
```

See also the documentation for `export_test_data()`. Use the **`--exclude`**, **`--nodes`**, and **`--techs`** options to control the content of the resulting file.

CHAPTER

SIX

DISTRIBUTED COMPUTING

This page introduces considerations, tools, and features for using **distributed** or **high-throughput computing** with MESSAGEix-GLOBIOM.

6.1 Overview

Scenarios in the MESSAGEix-GLOBIOM global model family are characterized by:

- ca. 100 MB of data, depending on storage format (e.g. in a `JDBCBackend` local, HyperSQL database, or Scenario/model data in Excel files).
- `solve()` times of between 10 and 60 minutes, depending on hardware and configuration, plus similar amounts of time to run the legacy reporting in `message_data`.
- Memory usage of ~10 GB or more using `JDBCBackend`, currently the only supported backend.

These resource needs can be a bottleneck in applications, for example:

- where many/related scenarios must be solved.
- when iteration (repeatedly solving 1 or more scenarios) is a key approach in developing code that sets up scenarios.

To improve research productivity, researchers may choose to run scenarios or 'workflows' (a combination of solving scenarios and pre- and post-processing steps or codes) through **distributed computing**, i.e. not on their local machine. Hardware and software environments for distributed computing can vary widely, and can be categorized in multiple ways, such as:

1. More powerful single-CPU systems, accessed remotely.
2. Cloud services, e.g. Google Compute Engine; Amazon AWS; Github Actions; etc. providing access to one or more machines.
3. Dedicated cluster systems (sometimes labelled **high-throughput computing**, HTC, or **high-performance computing**, HPC, systems) for scientific computing, operated by a variety of parties.

6.2 Tooling

- *message_ix_models* and related packages (ixmp, message_ix, message_data) aim to provide *simple, general-purpose features* that allow working with a variety of distributed systems.
- Specific configuration necessarily depends on the specific system(s) in use and the researcher's application.
- These features should not duplicate features of existing tools such as Slurm, HTCondor, etc., but rather allow message_ix & co. to be used with/through those.
- The individual features, tools, utilities, etc. should each be simple, i.e. do one thing, and do it well.

Todo: Extend.

CHAPTER

SEVEN

REFERENCES

CHAPTER

EIGHT

API REFERENCE

Commonly used classes may be imported directly from `message_ix_models`.

8.1 Models and variants (`model`)

Submodules described on this page:

- `model.macro`*: MESSAGE-MACRO*
- `model.structure`*: Model structure information*

Submodules described on separate pages:

- *Reproduce the RES (model.bare)*
- *Building models (model.build)*
- *Consumer disutility (model.disutility)*
- *Emissions data (model.emissions)*
- *Load model snapshots (model.snapshot)*

Code for constructing models/scenarios in the MESSAGEix-GLOBIOM model family.

class message_ix_models.model.**Config**(*regions: str = 'R14'*, *relations: str = 'A'*, *years: str = 'B'*, *res_with_dummies: bool = False*, *units: dict = <factory>*)

Settings and valid values for *message_ix_models.model* and submodules.

For backwards compatibility, it is possible to access these on a `Context` using:

```
c = Context()
c.regions = "R14"
```

…however, it is best to access them explicitly as:

```
c.model.regions = "R14"
```

check()

Check the validity of *regions*, *relations*, *years*.

regions: str = 'R14'

The 'node' codelist (regional aggregation) to use. Must be one of the lists of nodes described at *Node code lists*.

relations: str = 'A'

The 'relations' codelist to use. Must be one of the lists of relations described at *Relations (relation/*.yaml)*.

res_with_dummies: bool = False

Create the reference energy system with dummy commodities and technologies. See *bare.get_spec()*.

units: dict

Default or preferred units for model quantities and reporting.

years: str = 'B'

The 'year' codelist (time periods) to use, Must be one of the lists of periods described at *Years or time periods (year/*.yaml)*.

8.1.1 model.macro: MESSAGE-MACRO

Tools for calibrating MACRO for MESSAGEix-GLOBIOM.

See Calibrate and tune MESSAGE-MACRO for *general* documentation on MACRO and MESSAGE-MACRO. This module contains tools specifically for using these models with MESSAGEix-GLOBIOM.

message_ix_models.model.macro.**COMMODITY = ['i_therm', 'i_spec', 'rc_spec', 'rc_therm', 'transport']**

Default set of commodities to include in *generate()*.

message_ix_models.model.macro.**generate**(*parameter: Literal['aeei', 'config', 'depr', 'drate', 'lotol']*, *context:* Context, *commodities: List[str] | List[Code] = ['i_therm', 'i_spec', 'rc_spec', 'rc_therm', 'transport']*, *value: float | None = None*) → DataFrame

Generate uniform data for one `message_ix.macro` *parameter*.

`message_ix.Scenario.add_macro()` expects as its *data* parameter a `dict` that maps certain MACRO parameter names (or the special name "config") to `pandas.DataFrame`. This function generates data for those data frames.

For the particular dimensions, generate automatically includes:

- "node": All nodes in the node code list given by *nodes_ex_world()*, for the node list indicated by *model.Config.regions*.
- "year": All periods from the period *before* the first model year.
- "commodity": The elements of *commodities*.

- "sector": If each entry of *commodities* is a `Code` and has an annotation with id="macro-sector", the value of that annotation. Otherwise, the same as *commodity*.

value supplies the parameter value, which is the same for all observations. The labels level="useful" and unit="-" are fixed.

Parameters

- **parameter** (`str`) – MACRO parameter for which to generate data.
- **context** – Used with *bare.get_spec()*.
- **commodities** (`list` of `str` or `Code`) – Commodities to include in the MESSAGE-MACRO linkage.
- **value** (`float`) – Parameter value.

Returns

The columns vary according to *parameter*:

- "aeei": node, sector, year, value, unit.
- "depr", "drate", or "lotol": node, value, unit.
- "config": node, sector, commodity, level, year.

Return type

`pandas.DataFrame`

`message_ix_models.model.macro.`**load**(*base_path: Path*) → Mapping[str, DataFrame]

Load MACRO data from CSV files.

The function reads files in the simple/long CSV format understood by `genno.operator.load_file()`. For use with `add_macro()`, the dimension names should be given in full, for instance "node" or "sector".

Parameters

base_path (`pathlib.Path`) – Directory containing zero or more CSV files.

Returns

Mapping from MACRO calibration parameter names to data; one entry for each file in *base_path*.

Return type

`dict` of `(str -> pandas.DataFrame)`

8.1.2 `model.structure`: Model structure information

codelists(kind)	Return a valid IDs for code lists of *kind*.
generate_product(data, name, template)	Generates codes using a *template* by Cartesian product along ≥1 dimensions.
generate_set_elements(data, name)	Generate elements for set *name*.
get_codes(name)	Return codes for the dimension/set *name* in MESSAGE-GLOBIOM scenarios.
get_region_codes(codelist)	Return the codes that are children of "World" in the specified *codelist*.
process_units_anno(set_name, code[, quiet])	Process an annotation on *code* with id="units".
process_commodity_codes(codes)	Process a list of codes for `commodity`.
process_technology_codes(codes)	Process a list of codes for `technology`.

`message_ix_models.model.structure.`**codelists**(*kind: str*) → List[str]

Return a valid IDs for code lists of *kind*.

Parameters
kind (str) – "node" or "year".

message_ix_models.model.structure.**generate_product**(*data: Mapping*, *name: str*, *template: Code*) → Tuple[List[Code], Dict[str, DataArray]]

Generates codes using a *template* by Cartesian product along ≥1 dimensions.

generate_set_elements() is called for each of the *dims*, and these values are used to format *base*.

Parameters

- **data** – Mapping from dimension IDs to lists of codes.
- **name** (str) – Name of the set.
- **template** (Code) – Must have Python format strings for its its id and name attributes.

message_ix_models.model.structure.**generate_set_elements**(*data: MutableMapping*, *name*) → None

Generate elements for set *name*.

This function converts lists of codes in *data*, calling *generate_product()* and *process_units_anno()* as appropriate.

Parameters

- **data** – Mapping from dimension IDs to lists of codes.
- **name** (str) – Name of the set for which to generate elements e.g. "commodity" or "technology".

message_ix_models.model.structure.**get_codelist**(*name: str*) → Codelist

Return a Codelist for *name* in MESSAGEix-GLOBIOM scenarios.

message_ix_models.model.structure.**get_region_codes**(*codelist: str*) → List[Code]

Return the codes that are children of "World" in the specified *codelist*.

message_ix_models.model.structure.**process_commodity_codes**(*codes*)

Process a list of codes for commodity.

The function warns for commodities missing units or with non-pint-compatible units.

message_ix_models.model.structure.**process_technology_codes**(*codes*)

Process a list of codes for technology.

This function ensures every code has an annotation with id "vintaged", default False.

message_ix_models.model.structure.**process_units_anno**(*set_name: str*, *code: Code*, *quiet: bool = False*) → None

Process an annotation on *code* with id="units".

The annotation text is wrapped as 'registry.Unit("{text}")', such that it can be retrieved with *eval_anno()* or *ScenarioInfo.units_for()*. If *code* has direct children, the annotation is also copied to those codes.

Parameters

- **set_name** (str) – Used in logged messages when *quiet* is False.
- **quiet** (bool, *optional*) – If False (the default), log on level WARNING if:
  - the annotation is missing, or
  - its text is not parseable with the pint application registry, i.e. iam_units.registry.

  Otherwise, log on DEBUG.

message_ix_models.model.structure.**get_codes**(*name: str*) → List[Code]

Return codes for the dimension/set *name* in MESSAGE-GLOBIOM scenarios.

The information is read from `data/name.yaml`, e.g. `data/technology.yaml`.

When *name* includes "node", then child codes are automatically populated from the ISO 3166 database via `pycountry`. For instance:

```
myregion:
  name: Custom region
  child: [AUT, SCG]
```

…results in a region with child codes for Austria (a current country) and the formerly-existing country Serbia and Montenegro.

Parameters
name (str) – Any `.yaml` file in the folder `message_ix_models/data/`.

Returns
Every Code has `id`, `name`, `description`, and `annotations` attributes. Calling `str()` on a code returns its `id`.

Return type
`list` of `Code`

The available code lists are reproduced as part of this documentation. The returned code objects have annotations that vary by set. See:

- *Other code lists*
- *Node code lists*

Also available is `cd_links/unit.yaml`. This is a project-specific list of units appearing in CD-LINKS scenarios.

Example:

```
>>> from message_ix_models.model.structure import get_codes
>>> codes = get_codes("node/R14")

# Show the codes
>>> codes
[<Code ABW: Aruba>,
 <Code AFG: Afghanistan>,
 <Code AGO: Angola>,
 ...
 <Code ZWE: Zimbabwe>,
 <Code World: World>,
 ...
 <Code R11_PAS: Other Pacific Asia>,
 <Code R11_SAS: South Asia>,
 <Code R11_WEU: Western Europe>]

# Retrieve one code matching a certain ID
>>> world = codes[codes.index("World")]

# Get its children's IDs strings, e.g. for a "node" dimension
>>> [str(c) for c in world.child]
['R11_AFR',
 'R11_CPA',
 'R11_EEU',
 'R11_FSU',
 'R11_LAM',
 'R11_MEA',
 'R11_NAM',
```

(continues on next page)

(continued from previous page)

```
 'R11_PAO',
 'R11_PAS',
 'R11_SAS',
 'R11_WEU']

# Navigate from one ISO 3166-3 country code to its parent
>>> AUT = codes[codes.index("AUT")]
>>> AUT.parent
<Code R11_WEU: Western Europe>
```

See also:

adapt_R11_R14

8.2 Reproduce the RES (`model.bare`)

In contrast to `model.create`, this module creates the RES 'from scratch'. *create_res()* begins by creating a new, totally empty `Scenario` and adding data to it (instead of cloning and modifying an existing scenario).

Note: Currently, the Scenario returned by *create_res()*…

- is not complete, nor the official/preferred version of MESSAGEix-GLOBIOM, and as such **must not** be used for actual research,
- however, it **should** be used for creating unit tests of other code that is designed to operate on MESSAGEix-GLOBIOM scenarios; code that works against the bare RES should also work against MESSAGEix-GLOBIOM scenarios.

---

bare.get_spec() can also be used directly, to get a *description* of the RES based on certain settings/options, but without any need to connect to a database, load an existing Scenario, or call *bare.create_res()*. This can be useful in code that processes data into a form compatible with MESSAGEix-GLOBIOM.

8.2.1 Configuration

The code obeys the settings on the *model.Config* instance stored at `context.model`.

8.2.2 Code reference

message_ix_models.model.bare.**SETTINGS = {'regions': ['R14', 'ADVANCE', 'B210-R11', 'ISR', 'R11', 'R12', 'R17', 'R20', 'R32', 'RCP', 'ZMB'], 'res_with_dummies': [False, True], 'years': ['B', 'A']}**

Deprecated; use *model.Config* instead.

message_ix_models.model.bare.**create_res**(*context*, *quiet=True*)

Create a 'bare' MESSAGEix-GLOBIOM reference energy system (RES).

Parameters

- **context** (*Context*) – `model.Config.scenario_info` determines the model name and scenario name of the created Scenario. If not provided, the defaults are:
  - Model name generated by *name()*.
  - Scenario name "baseline".
- **quiet** (`bool`, *optional*) – Passed to *quiet* argument of *build.apply_spec()*.

Returns
A scenario as described by *bare.get_spec()*, prepared using *apply_spec()*.

Return type
message_ix.Scenario

message_ix_models.model.bare.**name**(*context*)

Generate a candidate name for a model given *context*.

The name has a form like:

```
MESSAGEix-GLOBIOM R99 YA +D
```

where:

- "R99" is the node list/regional aggregation.
- "YA" indicates the year codelist (*Years or time periods (year/*.yaml)*).
- "+D" appears if *Config.res_with_dummies* is true.

message_ix_models.model.bare.**get_spec**(*context*) → *Spec*

Return the spec for the MESSAGE-GLOBIOM global model RES.

If *Config.res_with_dummies* is set, additional elements are added:

- commodity: "dummy"
- technology: "dummy", "dummy supply"

These **may** be used for testing purposes, but **should not** be used in production models.

Return type
dict of *ScenarioInfo* objects

Since the RES is the base for all variants of MESSAGEix-GLOBIOM, the 'require' and 'remove' portions of the spec are empty.

For the 'add' section, *message_ix_models.model.structure.get_codes()* is used to retrieve data from the YAML files in *message_ix_models*.

Settings are retrieved from *context*, as above.

message_ix_models.model.data.**get_data**(*scenario*, *context*, *spec*, ***options*)

Data for the bare RES.

Todo: With ixmp#212 merged, some *model.bare* code could be moved to a new class and method like MESSAGE_GLOBIOM.initialize().

8.3 Building models (model.build)

apply_spec() can be used to be build (compose, assemble, construct, …) models given three pieces of information:

- A message_ix.Scenario to be used as a base. This scenario may be empty.
- A specification, or *Spec*, which is trio of *ScenarioInfo* objects; see below.
- An optional function that adds or produces *data* to add to the *scenario*.

The spec is applied as follows:

1. For each ixmp set that exists in *scenario*:

a. **Required** elements from *Spec.require*, if any, are checked.

   If they are missing, `apply_spec()` raises `ValueError`. This indicates that *spec* is not compatible with the given *scenario*.

b. Elements from *Spec.remove*, if any, are **removed**.

   Any parameter values which reference these set elements are also removed, using *strip_par_data()*.

c. New set elements from *Spec.add*, if any, are **added**.

2. Elements in `spec.add.set["unit"]` are added to the Platform on which *scenario* is stored.
3. The *data* argument, a function, is called with *scenario* as the first argument, and a keyword argument *dry_run* from *apply_spec()*. *data* may either add to *scenario* directly (by calling `Scenario.add_par()` and similar methods); or it can return a `dict` that can be passed to *add_par_data()*.

The following modules use this workflow and can be examples for developing similar code:

- *model.bare*
- *model.disutility*
- `message_data.model.transport`

8.3.1 Code reference

message_ix_models.model.build.**apply_spec**(*scenario: Scenario*, *spec:* Spec | *Mapping[str,* ScenarioInfo*]*, *data: Callable* | *None* = *None*, ***options*)

Apply *spec* to *scenario*.

Parameters

- **spec** (*Spec*) – Specification of changes to make to *scenario*.
- **data** (`callable`, *optional*) – Function to add data to *scenario*. *data* can either manipulate the scenario directly, or return a `dict` compatible with *add_par_data()*.
- **dry_run** (`bool`) – Don't modify *scenario*; only show what would be done. Default `False`. Exceptions will still be raised if the elements from `spec['required']` are missing; this serves as a check that the scenario has the required features for applying the spec.
- **fast** (`bool`) – Do not remove existing parameter data; increases speed on large scenarios.
- **quiet** (`bool`) – Only show log messages at level `ERROR` and higher. If `False` (default), show log messages at level `DEBUG` and higher.
- **message** (`str`) – Commit message.

See also:

add_par_data, *strip_par_data*, `Code`, *ScenarioInfo*

message_ix_models.model.build.**ellipsize**(*elements: List*) → str

Generate a short string representation of *elements*.

If the list has more than 5 elements, only the first two and last two are shown, with "…" between.

8.4 Emissions data (`model.emissions`)

model.emissions contains codes for working with emissions data, including policies on emissions.

In general, models created with *message_ix_models*:

- Use tonnes of carbon equivalent ("t C") as units for mass of emissions.
- Use "USD / t C" as units for price of emissions. Because (as of 2022-07-20) `iam_units` treats "USD" as an alias for "USD_2005", this is implicitly USD_2005 / t C.

`message_ix_models.model.emissions.`**`add_tax_emission`**(*scen: Scenario*, *price: float*, *conversion_factor: float | None = None*, *drate_parameter='drate'*) → None

Add a global CO_2 price to *scen*.

A carbon price is implemented on node="World" by populating the MESSAGEix parameter `tax_emission`, starting from the first model year and covering the entire model horizon. The tax has an annual growth rate equal to the discount rate.

The other dimensions of `tax_emission` are filled with type_emission="TCE" and type_tec="all".

Parameters

- **`scen`** (`message_ix.Scenario`) –
- **`price`** (`float`) – Price in the first model year, in USD / tonne CO_2.
- **`conversion_factor`** (`float`, *optional*) – Factor for converting *price* into the model's internal emissions units, currently USD / tonne carbon. Optional: a default value is retrieved from `iam_units`.
- **`drate_parameter`** (`str; one of "drate" or "interestrate"`) – Name of the parameter to use for the growth rate of the carbon price.

`message_ix_models.model.emissions.`**`get_emission_factors`**(*units: str | None = None*) → AttrSeries

Return carbon emission factors.

Values are from the file `message_ix_models/data/ipcc/1996_v3_t1-2.csv`, in turn from IPCC (see Table 1-2 on page 1.6); these are the same that appear on the "Emissions from energy" page of the MESSAGEix-GLOBIOM documentation.

The fuel dimension and names in the source are mapped to a c ("commodity") dimension and labels from *commodity.yaml*, using the `ipcc-1996-name` annotations appearing in the latter. A value for "methanol" that appears in the MESSAGEix-GLOBIOM docs table but not in the source is appended.

Parameters

`unit` (`str`, *optional*) – Expression for units of the returned quantity. Tested values include:

- "tC / TJ", source units (default),
- "t CO2 / TJ", and
- "t C / kWa", internal units in MESSAGEix-GLOBIOM, for instance for "relation_activity" entries for emissions relations.

Returns

with 1 dimension (c).

Return type

`Quantity`

`message_ix_models.model.emissions.`**`split_species`**(*unit_expr: str*) → Tuple[str, str | None]

Split *unit_expr* to an expression without a unit mention, and maybe species.

8.5 Load model snapshots (`model.snapshot`)

This code allows to fetch *snapshots* containing completely parametrized MESSAGEix-GLOBIOM model instances, and load these into `Scenarios`.

8.5.1 Usage

From the command line, download data for a single snapshot:

```
$ mix-models fetch snapshot-0
```

…where **0** is the ID of a snapshot; see `SNAPSHOTS`.

In code, use *`snapshot.load()`*:

```
from message_ix import Scenario
from message_ix_models.model import snapshot

scenario = Scenario(...)

snapshot.load(scenario, 0)
```

Note: For snapshot 0, contrary to the description of the Zenodo item, the file cannot be loaded using `Scenario.read_excel()`. This limitation will be fixed in subsequent snapshots.

8.5.2 Code reference

Prepare base models from snapshot data.

`message_ix_models.model.snapshot.`**`load`**(*scenario: Scenario*, *snapshot_id: int*, *extra_cache_path: str | None = None*) → None

Fetch and load snapshot with ID *snapshot_id* into *scenario*.

See also:

`SNAPSHOTS`

`message_ix_models.model.snapshot.`**`read_excel`**(*scenario: Scenario*, *path: Path*) → None

Similar to `Scenario.read_excel()`, but using *`unpack()`*.

`message_ix_models.model.snapshot.`**`unpack`**(*path: Path*) → Path

Unpack ixmp-format Excel file at *path*.

The file is unpacked into a directory with the same name stem as the file (that is, without the `.xlsx` suffix). In this directory are created:

- One `.csv.gz` file for each MESSAGE and/or MACRO parameter.
- One file `sets.xlsx` with only the `ixmp` sets, and no parameter data.

If the files exist, they are not updated. To force re-unpacking, delete the files.

Returns
Path to the directory containing the unpacked files.

Return type
`Path`

8.6 Consumer disutility (`model.disutility`)

This module provides a generalized consumer disutility formulation, currently used by `message_data.model.transport`. The formulation rests on the concept of "consumer groups"; each consumer group may have a distinct disutility associated with using the outputs of each technology. A set of 'pseudo-'/'virtual'/non-physical "usage technologies" converts the outputs of the actual technologies into the commodities demanded by each group, while also requiring input of a costly "disutility" commodity.

8.6.1 Method & usage

Use this code by calling *`add()`*, which takes arguments that describe the concrete usage:

Consumer groups
: This is a list of `Code` objects describing the consumer groups. The list must be 1-dimensional, but can be composed (as in `message_data.model.transport`) from multiple dimensions.

Technologies
: This is a list of `Code` objects describing the technologies for which the consumers in the different groups experience disutility. Each object must be have 'input' and 'output' annotations (`annotations`); each of these is a `dict` with the keys 'commodity', 'input', and 'unit', describing the source or sink for the technology.

Template
: This is also a `Code` object, similar to those in `technologies`; see below.

The code creates a source technology for the "disutility" commodity. The code does *not* perform the following step(s) needed to completely parametrize the formulation:

- Set consumer group-specific `demand` parameter values for new commodities.
- Set the amounts of "disutility" commodities used as `input` to the new usage technologies.

These must be parametrized based on the particular application.

8.6.2 Detailed example

This example is similar to the one used in *`test_disutility.test_minimal()`*:

```
# Two consumer groups
groups = [Code(id="g0"), Code(id="g1")]

# Two technologies, for which groups may have different disutilities.
techs = [Code(id="t0"), Code(id="t1")]

# Add generalized disutility formulation to some technologies
disutility.add(
    scenario,
    groups=groups,
    technologies=techs,

    template=Code(
        # Template for IDs of conversion technologies
        id="usage of {technology} by {group}",

        # Templates for inputs of conversion technologies
        input=dict(
            # Technology-specific output commodity
            commodity="output of {technology}",
            level="useful",
            unit="kg",
        ),
```

(continues on next page)

(continued from previous page)

```
        # Templates for outputs of conversion technologies
        output=dict(
            # Consumer-group-specific demand commodity
            commodity="demand of group {group}",
            level="useful",
            unit="kg",
        ),
    ),
    **options,
)
```

add() uses *get_spec()* to generate a specification that adds the following:

- For the set `commodity`:
  - The single element "disutility".
  - One element per *technologies*, using the *template* "input" annotation, e.g. "output of t0" generated from `output of {technology}` and the id "t0". These **may** already be present in the *scenario*; if not, the spec causes them to be added.
  - One elements per *groups*, using the *template* "output" annotation, e.g. "demand of group g1" generated from `demand of group {group}` and the id "g1". These **may** already be present in the *scenario*; if not, the spec causes them to be added.
- For the set `technology`:
  - The single element "disutility source".
  - One element per each combination of disutility-affected technology (*technologies*) and consumer group (*groups*). For example, "usage of t0 by g1" generated from `usage of {technology} by {group}`, and the ids "t0" and "g1".

The spec is applied to the target scenario using *model.build.apply_spec()*. If the arguments produce a spec that is inconsistent with the target scenario, an exception will by raised at this point.

Next, *add()* uses *data_conversion()* and *data_source()* to generate:

- `output` and `var_cost` parameter data for "disutility source". This technology outputs the unitless commodity "disutility" at a cost of 1.0 per unit.
- `input` and `output` parameter data for the new usage technologies. For example, the new technology "usage of t0 by g1"…
  - …takes input from the *technology-specific* commodity "output of t0".
  - …takes input from the common commodity "disutility", in an amount specific to group "g1".
  - …outputs to a *group-specific* commodity "demand of group g1".

Note that the *technologies* towards which the groups have disutility are assumed to already be configured to `output` to the corresponding commodities. For example, the technology "t0" outputs to the commodity "output of t0"; the `output` values for this technology are **not** added/introduced by *add()*.

(Dis)utility is generally dimensionless. In `pint` and thus also *message_ix_models*, this should be represented by `""`. However, to work around iiasa/ixmp#425, *data_conversion()* and *data_source()* return data with `"-"` as units. See GH #45 for more information.

8.6.3 Code reference

See also *message_ix_models.tests.model.test_disutility*.

message_ix_models.model.disutility.**add**(*scenario: Scenario*, *groups: Sequence[Code]*, *technologies: Sequence[Code]*, *template: Code*, ***options*) → *Spec*

Add disutility formulation to *scenario*.

message_ix_models.model.disutility.**data_conversion**(*info*, *spec: Spec*) → MutableMapping[str, DataFrame]

Generate input and output data for disutility conversion technologies.

message_ix_models.model.disutility.**data_source**(*info*, *spec*) → Mapping[str, DataFrame]

Generate data for a technology that emits the "disutility" commodity.

message_ix_models.model.disutility.**dp_for**(*col_name: str*, *info: ScenarioInfo*) → Series

`pandas.DataFrame.assign()` helper for `duration_period`.

Returns a callable to be passed to `pandas.DataFrame.assign()`. The callable takes a data frame as the first argument, and returns a `pandas.Series` based on the `duration_period` parameter in *info*, aligned to *col_name* in the data frame.

Currently (2021-04-07) unused.

message_ix_models.model.disutility.**get_data**(*scenario*, *spec*, ***kwargs*) → Mapping[str, DataFrame]

Get data for disutility formulation.

Calls *data_conversion()* and *data_source()*.

Parameters

spec (`dict`) – The output of *get_spec()*.

message_ix_models.model.disutility.**get_spec**(*groups: Sequence[Code]*, *technologies: Sequence[Code]*, *template: Code*) → *Spec*

Get a spec for a disutility formulation.

Parameters

- **groups** (`list` of `Code`) – Identities of the consumer groups with distinct disutilities.
- **technologies** (`list` of `Code`) – The technologies to which the disutilities are applied.
- **template** (`Code`) –

8.7 Reporting (`report`)

On this page:

Elsewhere:

- `global.yaml`, the *Default reporting configuration*.
- Documentation for `genno` (genno: efficient, transparent calculation on N-D data), `ixmp.report`, and `message_ix.report`.
- Reporting for specific model variants:
    - *`water.reporting`*
    - (Private) Reporting of message_data.model.transport
- *'Legacy' reporting*.

8.7.1 Default reporting configuration

```yaml
# Configuration for reporting of the MESSAGEix-GLOBIOM global model
#
#
# EDITING
#
# - Keep lists of labels—technologies, etc.—in alphabetical order.
# - Long lists can be on a single line.


units:
  # Unit definitions are loaded from iam-units; only add units here which are
  # idiosyncrasies of MESSAGEix-GLOBIOM.
  # define: |
  #   USD = [value]

  replace:
    '???': ''
    '-': ''

  apply:
    GDP: billion USD_2005 / year
    PRICE_COMMODITY: USD_2010 / kWa
    # These were initially "carbon", which is not a unit.
    # TODO check that Mt (rather than t or kt) is correct for all values.
    PRICE_EMISSION: USD_2005 / Mt
    tax_emission: USD_2005 / Mt
    # Inconsistent units. These quantities (and others computed from them)
    # must be split apart into separate quantities with consistent units.
    # Also applies to "emi". This value preserved because it works for the
    # reporting of CH4 emissions.
    emission_factor: "Mt / year"

# Files with external data
# files:
# - path: ./foo.csv
#   key: gwp:e-gwp_source
```

(continues on next page)

(continued from previous page)

```
#   dims:
#     message_act_name: e
#     gwp: gwp_source


# Filters
#
# These limit the data that is retrieved from the backend by ixmp.report;
# so ALL quantities in the Reporter are limited to these values. Use these for
# debugging.

# filters:
#   t: [coal_ppl, po_turbine, c_ppl_co2scr]


# Aggregate across dimensions of single quantities
#
# - The dimension `_dim` is not removed; it gains new labels that are the sum
#   of the listed members. Basically regrouping over one dimension.
# - Keep members in alphabetical order.
aggregate:
  # Quantities to aggregate
- _quantities: [in, out]
  # Added to the end of the each key (e.g. '[key]:pe' or '[key]:[tag]+pe')
  _tag: pe
  # Dimension along which to aggregate
  _dim: t

  # Mappings from group names to members along _dim
  # Coal
  coal: [coal_extr, coal_extr_ch4]
  lignite: [lignite_extr]
  gas conventional: [gas_extr_1, gas_extr_2, gas_extr_3, gas_extr_4]
  gas unconventional: [gas_extr_5, gas_extr_6, gas_extr_7, gas_extr_8]
  oil conventional: [oil_extr_1_ch4, oil_extr_2_ch4, oil_extr_3_ch4,
                     oil_extr_1, oil_extr_2, oil_extr_3]
  oil unconventional: [oil_extr_4_ch4, oil_extr_4, oil_extr_5, oil_extr_6,
                       oil_extr_7, oil_extr_8]

- _quantities:
  - in
  # This quantity (defined below in 'general:') has the non-main outputs of
  # dual-output technologies removed; e.g. for t=h2_coal, only c=hydrogen and
  # not c=electr is included.
  - out::se_1
  # Produces 'in::se' and 'out::se_1+se'
  _tag: se
  _dim: t

  # Electricity
  # Electricity missing.
  Electricity|Biomass: [bio_istig, bio_istig_ccs ,bio_ppl]
  Electricity|Biomass|w/o CCS: [bio_istig, bio_ppl]
  Electricity|Biomass|w/ CCS: [bio_istig_ccs]
  Electricity|Coal: [coal_adv, coal_adv_ccs, coal_ppl, coal_ppl_u, igcc, igcc_ccs]
  Electricity|Coal|w/o CCS: [coal_adv, coal_ppl, coal_ppl_u, igcc]
  Electricity|Coal|w/ CCS: [coal_adv_ccs, igcc_ccs]
  # Electricity|Fossil will be in 'combine:' section
  Electricity|Gas: [gas_cc, gas_cc_ccs, gas_ct, gas_htfc, gas_ppl]
  Electricity|Gas|w/o CCS: [gas_cc, gas_ct, gas_htfc, gas_ppl]
  Electricity|Gas|w/ CCS: [gas_cc_ccs]
```

(continues on next page)

(continued from previous page)

```
  Electricity|Geothermal: [geo_ppl]
  Electricity|Hydro: [hydro_hc, hydro_lc]
  # Non-biomass in 'combine:' section
  Electricity|Nuclear: [nuc_fbr, nuc_hc, nuc_lc]
  Electricity|Oil: [foil_ppl, loil_cc, loil_ppl, oil_ppl, SO2_scrub_ppl]
  Electricity|Oil|w/o CCS: [foil_ppl, loil_cc, loil_ppl, oil_ppl, SO2_scrub_ppl]
  # Electricity|Other:
  # TODO include missing and these 3 below in 'combine:' section, as an
  #      operation:
  # Electricity|Fossil: [coal_adv, coal_adv_ccs, coal_ppl, coal_ppl_u, foil_ppl,␣
↪gas_cc, gas_cc_ccs, gas_ct, gas_htfc, gas_ppl, igcc, igcc_ccs, loil_cc, loil_ppl,
↪ oil_ppl, SO2_scrub_ppl]
  # Electricity|Fossil|w/o CCS: [coal_ppl, coal_ppl_u, foil_ppl, gas_cc, gas_ct,␣
↪gas_htfc, gas_ppl, igcc, loil_cc, loil_ppl, oil_ppl, SO2_scrub_ppl]
  # Electricity|Fossil|w/ CCS: [coal_adv_ccs, gas_cc_ccs, igcc_ccs]

  # Gases
  # Gases: in 'combine:' section
  Gases|Biomass: [gas_bio]
  Gases|Coal: [coal_gas]
  # Notice NG imports are included here.
  Gases|Natural Gas: [LNG_regas, gas_imp, gas_bal]
  # TODO use an input-based calculation here.
  Gases|Other: [h2_mix]

  # Heat
  Heat: [bio_hpl, coal_hpl, gas_hpl, geo_hpl, foil_hpl, po_turbine]
  Heat|Biomass: [bio_hpl]
  Heat|Coal: [coal_hpl]
  Heat|Gas: [gas_hpl]
  Heat|Geothermal: [geo_hpl]
  Heat|Oil: [foil_hpl]
  Heat|Other: [po_turbine]

  # Hydrogen
  Hydrogen: [h2_bio, h2_bio_ccs, h2_coal, h2_coal_ccs, h2_elec, h2_smr, h2_smr_ccs]
  Hydrogen|Biomass: [h2_bio, h2_bio_ccs]  # Skip electr
  Hydrogen|Biomass|w/o CCS: [h2_bio]
  Hydrogen|Biomass|w/ CCS: [h2_bio_ccs]
  Hydrogen|Coal: [h2_coal, h2_coal_ccs]  # Skip electr
  Hydrogen|Coal|w/o CCS: [h2_coal]
  Hydrogen|Coal|w/ CCS: [h2_coal_ccs]
  Hydrogen|Electricity: [h2_elec]
  Hydrogen|Fossil: [h2_coal, h2_coal_ccs, h2_smr, h2_smr_ccs]
  Hydrogen|Fossil|w/o CCS: [h2_coal, h2_smr]
  Hydrogen|Fossil|w/ CCS: [h2_coal_ccs, h2_smr_ccs]
  Hydrogen|Gas: [h2_smr, h2_smr_ccs]
  Hydrogen|Gas|w/o CCS: [h2_smr]
  Hydrogen|Gas|w/ CCS: [h2_smr_ccs]

  # Liquids
  # Liquids missing. Do the calculation in 'combine:' section
  Liquids|Biomass: [eth_bio, eth_bio_ccs, liq_bio, liq_bio_ccs]
  Liquids|Biomass|w/o CCS: [eth_bio, liq_bio]
  Liquids|Biomass|w/ CCS: [eth_bio_ccs, liq_bio_ccs]
  Liquids|Coal: [meth_coal, meth_coal_ccs, syn_liq, syn_liq_ccs]
  Liquids|Coal|w/o CCS: [syn_liq, meth_coal]
  Liquids|Coal|w/ CCS: [syn_liq_ccs, meth_coal_ccs]
  Liquids|Fossil: [meth_coal, meth_coal_ccs, meth_ng, meth_ng_ccs, ref_hil, ref_
↪lol, syn_liq, syn_liq_ccs, SO2_scrub_ref]
  Liquids|Fossil|w/o CCS: [syn_liq, meth_coal, meth_ng, meth_ng_ccs, ref_hil, ref_
```

(continues on next page)

(continued from previous page)

```
↪lol, SO2_scrub_ref]
  Liquids|Fossil|w/ CCS: [syn_liq_ccs, meth_coal_ccs, meth_ng_ccs]
  Liquids|Gas: [meth_ng, meth_ng_ccs]
  Liquids|Gas|w/o CCS: [meth_ng]
  Liquids|Gas|w/ CCS: [meth_ng_ccs]
  Liquids|Oil: [ref_hil, ref_lol, SO2_scrub_ref]

  # Solids
  Solids: [biomass_i, biomass_nc, biomass_rc, coal_i, coal_fs, coal_rc, coal_trp]
  Solids|Biomass: [biomass_i, biomass_nc, biomass_rc]
  Solids|Coal: [coal_i, coal_fs, coal_rc, coal_trp]

  # Wind
  wind curtailment: [wind_curtailment1, wind_curtailment2, wind_curtailment3]
  wind gen onshore: [wind_res1, wind_res2, wind_res3, wind_res4]
  wind gen offshore: [wind_ref1, wind_ref2, wind_ref3, wind_ref4, wind_ref5]

  # Solar
  solar pv gen elec: [solar_res1, solar_res2, solar_res3, solar_res4, solar_res5,␣
↪solar_res6, solar_res7, solar_res8]
  solar pv gen elec RC: [solar_pv_RC]
  solar pv gen elec I: [solar_pv_I]
  solar pv curtailment: [solar_curtailment1, solar_curtailment2, solar_
↪curtailment3]
  solar csp gen elec sm1: [csp_sm1_res, csp_sm1_res1, csp_sm1_res2, csp_sm1_res3,␣
↪csp_sm1_res4, csp_sm1_res5, csp_sm1_res6, csp_sm1_res7]
  solar csp gen elec sm3: [csp_sm3_res, csp_sm3_res1, csp_sm3_res2, csp_sm3_res3,␣
↪csp_sm3_res4, csp_sm3_res5, csp_sm3_res6, csp_sm3_res7]
  solar csp gen heat rc: [solar_rc]
  solar csp gen heat i: [solar_i]

  # Storage
  storage elec: [stor_ppl]

  # Cogeneration
  cogeneration plants: [bio_istig, bio_istig_ccs, bio_ppl, coal_adv, coal_adv_ccs,␣
↪coal_ppl, coal_ppl_u, foil_ppl, gas_cc, gas_cc_ccs, gas_ct, gas_htfc, gas_ppl,␣
↪geo_ppl, igcc_ccs, igcc, loil_cc, loil_ppl, nuc_hc, nuc_lc]
  passout turbine: [po_turbine]

  # CO2 Scrubber
  coal scrubber: [c_ppl_co2scr]
  gas scrubber: [g_ppl_co2scr]


# Emissions of CH4
- _quantities: [emi]
  _tag: CH4_0
  _dim: t

  Waste: [CH4_WasteBurnEM, CH4_DomWasteWa, CH4_IndWasteWa, CH4n_landfills,␣
↪landfill_compost1, landfill_digester1, landfill_direct1, landfill_ele, landfill_
↪flaring, landfill_heatprdn, landfill_mechbio]
  Industrial Processes: [CH4_IndNonEnergyEM]
  AFOLU|Biomass Burning: [CH4_AgWasteEM]
  Heat: [bio_hpl, coal_hpl, foil_hpl, gas_hpl]
  Electricity: [bio_istig_ccs, bio_istig, bio_ppl, coal_adv_ccs, coal_adv, coal_
↪ppl_u, coal_ppl, foil_ppl, gas_cc_ccs, gas_cc, gas_ct, gas_htfc, gas_ppl, igcc_
↪ccs, igcc, loil_cc, loil_ppl, oil_ppl, SO2_scrub_ppl]

  # Fugitive - gas
```

(continues on next page)

(continued from previous page)

```
  Gases|Extraction: [flaring_CO2, gas_extr_1, gas_extr_2, gas_extr_3, gas_extr_4,␣
↪gas_extr_5, gas_extr_6, gas_extr_7, gas_extr_8]
  Gases|Transportation: [gas_t_d, gas_t_d_ch4]
  Gases|Coal: [coal_gas]
  Gases|Hydrogen: [h2_bio_ccs, h2_bio, h2_coal_ccs, h2_coal, h2_smr_ccs, h2_smr]

  # Fugitive - liquid
  Liquids|Extraction: [oil_extr_1, oil_extr_1_ch4, oil_extr_2, oil_extr_2_ch4, oil_
↪extr_3, oil_extr_3_ch4, oil_extr_4, oil_extr_4_ch4, oil_extr_5, oil_extr_6, oil_
↪extr_7, oil_extr_8]
  Liquids|Transportation: [eth_t_d, foil_t_d, loil_t_d, meth_t_d]
  Liquids|Oil: [ref_hil, ref_lol, SO2_scrub_ref]
  Liquids|Natural Gas: [meth_ng, meth_ng_ccs]
  Liquids|Coal: [meth_coal, meth_coal_ccs, syn_liq, syn_liq_ccs]
  Liquids|Biomass: [eth_bio, eth_bio_ccs, liq_bio, liq_bio_ccs]

  # Fugitive - solid
  Solids|Extraction: [coal_extr, coal_extr_ch4, lignite_extr]
  Solids|Transportation: [biomass_t_d, coal_t_d]

  # Demand
  Energy|Demand|Residential and Commercial: [biomass_nc, biomass_rc, coal_rc, elec_
↪rc, eth_rc, foil_rc, gas_rc, h2_fc_RC, h2_rc, heat_rc, hp_el_rc, hp_gas_rc, loil_
↪rc, meth_rc, other_sc, solar_pv_RC, solar_rc, sp_el_RC]
  Energy|Demand|Transportation|Road Rail and Domestic Shipping: [coal_trp, elec_
↪trp, eth_fc_trp, eth_ic_trp, foil_trp, gas_trp, h2_fc_trp, loil_trp, meth_fc_trp,
↪ meth_ic_trp]

# Second stage of CH4 aggregation
- _quantities: [emi::CH4_0]
  _tag: '1'
  _dim: t

  Fugitive: [Gases|Coal, Gases|Extraction, Gases|Hydrogen, Gases|Transportation,␣
↪Liquids|Biomass, Liquids|Coal, Liquids|Extraction, Liquids|Gas, Liquids|Oil,␣
↪Liquids|Transportation, Solids|Extraction, Solids|Transportation]

# Third stage of CH4 aggregation
- _quantities: [emi::CH4_0+1]
  _tag: '2'
  _dim: e

  CH4: [CH4_Emission, CH4_Emission_bunkers, CH4_new_Emission, CH4_nonenergy]

# Transmission & distribution
- _quantities: [in, out]
  _tag: t_d
  _dim: t

  biomass: [biomass_t_d]
  coal: [coal_t_d-rc-06p, coal_t_d-in-06p, coal_t_d-in-SO2, coal_t_d-rc-SO2, coal_
↪t_d]
  elec: [elec_t_d]
  gas: [gas_t_d, gas_t_d_ch4]
  heat: [heat_t_d]
  oil: [loil_t_d, foil_t_d]

# Bunkers
- _quantities: [in, out]
  _tag: bunker
  _dim: t
```

(continues on next page)

(continued from previous page)

```
  methanol: [methanol_bunker]
  gas: [LNG_bunker]
  lh2: [LH2_bunker]
  oil: [loil_bunker, foil_bunker]

# Imports
- _quantities: [in, out]
  _tag: import
  _dim: t

  coal: [coal_imp]
  elec: [elec_imp]
  ethanol: [eth_imp]
  # TODO check if LNG_imp should be included here. In old reporting it was not.
  gas: [LNG_imp, gas_imp]
  lh2: [lh2_imp]
  methanol: [meth_imp]
  oil: [oil_imp, loil_imp, foil_imp]

# Exports
- _quantities: [in, out]
  _tag: export
  _dim: t

  coal: [coal_exp]
  elec: [elec_exp]
  ethanol: [eth_exp]
  lh2: [lh2_exp]
  gas: [LNG_exp, gas_exp_nam, gas_exp_weu, gas_exp_eeu, gas_exp_pao,
        gas_exp_cpa, gas_exp_afr, gas_exp_sas, gas_exp_pas]
  methanol: [meth_exp]
  oil: [oil_exp, loil_exp, foil_exp]

# Aggregate emissions species
- _quantities: [emi::gwpe]
  _tag: agg
  _dim: e

  F gases: [CF4, HFC, SF6]

# Parent technologies that have addon
# - _quantities: ['addon pot']
#   _tag: addon_tecs
#   _dim: type_addon
#
#   cogeneration: [cogeneration_heat]
#   scrubber: [scrubber_CO2_coal, scrubber_CO2_gas, scrubber_CO2_bio, scrubber_CO2_
↪cement]


# Create new quantities by weighted sum across multiple quantities
combine:
  # Name and dimensions of quantity to be created
- key: coal:nl-ya
  # Inputs to sum
  inputs:
    # Input quantity. If dimensions are none ('name::tag') then the necessary
    # dimensions are inferred: the union of the dimensions of 'key:' above,
    # plus any dimensions appearing in 'select:''
  - quantity: in::pe  # e.g. 'in:nl-t-ya:pe' is inferred
```

(continues on next page)

(continued from previous page)

```
    # Values to select
    select: {t: [coal, lignite]}
    # Weight for these values in the weighted sum
  - quantity: in::import
    select: {t: coal}
  - quantity: in::export
    select: {t: coal}
    weight: -1
  # commented (PNK 2019-10-07): doesn't exist
  # - quantity: in::bunker
  #   select: {t: coal}

- key: gas:nl-ya
  inputs:
  - quantity: in::pe
    select: {t: ['gas conventional', 'gas unconventional']}
  - quantity: in::import
    select: {t: gas}
  - quantity: in::export
    select: {t: gas}
    weight: -1
  - quantity: in::bunker
    select: {t: gas}

- key: oil:nl-ya
  inputs:
  - quantity: in::pe
    select: {t: ['oil conventional', 'oil unconventional']}
  - quantity: in::import
    select: {t: oil}
  - quantity: in::export
    select: {t: oil}
  - quantity: in::bunker
    select: {t: oil}
    weight: 1

- key: solar:nl-ya
  inputs:
  - quantity: out::se_1+se
    select:
      t:
      - solar pv gen elec
      - solar pv gen elec RC
      - solar pv gen elec I
      - solar csp gen elec sm1
      - solar csp gen elec sm3
      - solar csp gen heat rc
      - solar csp gen heat i
      # c: [electr]
  - quantity: in::se
    select: {t: solar pv curtailment}  #, c: [electr]}
    weight: -1

- key: se_trade:nl-ya
  inputs:
  - quantity: out::import
    select: {t: [elec, ethanol, lh2, methanol]}
  - quantity: in::export
    select: {t: [elec, ethanol, lh2, methanol]}
    weight: -1
```

(continues on next page)

(continued from previous page)

```
- key: wind:nl-ya
  inputs:
  - quantity: out::se_1+se
    select: {t: ['wind gen onshore', 'wind gen offshore']}
  - quantity: in::se
    select: {t: wind curtailment}
    weight: -1

# TODO check if the sum of Electricity in aggregate:se should give the same
# values as the operation below
# - key: elec:nl-ya
#   inputs:
#   # Electricity going into the grid: [prod + exp - imp]
#   - quantity: in::t_d
#     select: {t: elec}
#   # Minus electricity from imports, to obtain the net electricity production
#     by conversion technologies
#   - quantity: out::import
#     select: {t: elec}
#     weight: -1
#   # Plus electricity from exports
#   - quantity: in::export
#     select: {t: elec}

- key: electr_fossil:nl-ya
  inputs:
  - quantity: out::se_1+se
    # TOO check if this should include electr from h2_coal.
    select: {t: Electricity|Coal}
  - quantity: out::se_1+se
    select: {t: Electricity|Gas}
  - quantity: out::se_1+se
    select: {t: Electricity|Oil}

- key: electr_fossil_w/_ccs:nl-ya
  inputs:
  - quantity: out::se_1+se
    select: {t: Electricity|Coal|w/ CCS}
  - quantity: out::se_1+se
    select: {t: Electricity|Gas|w/ CCS}
  # - quantity: out::se_1+se
  #   select: {t: Electricity|Oil|w/ CCS}

- key: electr_fossil_w/o_ccs:nl-ya
  inputs:
  - quantity: out::se_1+se
    select: {t: Electricity|Coal|w/o CCS}
  - quantity: out::se_1+se
    select: {t: Electricity|Gas|w/o CCS}
  - quantity: out::se_1+se
    select: {t: Electricity|Oil|w/o CCS}

- key: gases:nl-ya
  inputs:
  - quantity: out::se_1+se
    select: {t: Gases|Biomass}
  - quantity: out::se_1+se
    select: {t: Gases|Coal}
  - quantity: out::se_1+se
    select: {t: Gases|Natural Gas}
  - quantity: in::se
```

(continues on next page)

(continued from previous page)

```
    select: {t: Gases|Other}

- key: liquids:nl-ya
  inputs:
  - quantity: out::se_1+se
    select: {t: Liquids|Biomass}
  - quantity: out::se_1+se
    select: {t: Liquids|Fossil}


# Emissions

# CH4 emissions from GLOBIOM: apply a factor of 0.025 to land_out
# TODO document why this is needed
- key: land_out:n-s-y-c:CH4_0+1+2
  inputs:
  - quantity: land_out::CH4_0+1
    weight: 0.025


# Prices

- key: price_carbon:n-y
  # TODO PRICE_EMISSION has dimension "y", tax_emission has dimension
  #      "type_year". Implement a dimension rename so that the two can be
  #      combined in this way.
  inputs:
  - quantity: PRICE_EMISSION
    select: {type_emission: [TCE], type_tec: [all]}
  # - quantity: tax_emission
  #   select: {type_emission: [TCE], type_tec: [all]}

# Commodity price minus emission price
# NB This is only for illustration.
# TODO use emission factor must be used to convert the following to compatible
#      units:
# - PRICE_COMMODITY with (c, l) dimensions and units [currency] / [energy]
# - 'price emission' with (e, t) dimensions and units [currency] / [mass]
- key: price ex carbon:n-t-y-c-l-e
  inputs:
  - quantity: PRICE_COMMODITY:n-c-l-y
  - quantity: price emission:n-e-t-y
    weight: -1

# TODO remove these entries once the one-step conversion is checked.
# - The following entries subset the components of PRICE_COMMODITY used in the
#   legacy reporting. The preferred method is to convert the entire variable to
#   IAMC format in one step; see below in the "iamc:" section.
# - l: [import] is sometimes included to pick up prices at the global node(s).
# - In general, PRICE_COMMODITY has data for n=GLB and level=import OR for
#   other nodes and l=primary or secondary—but not otherwise.
- key: price_c:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [coal], l: [primary, import]}
- key: price_g_w:n-y-h
  # Only includes 11 regions; no data for c="gas" for global regions, instead
  # c="LNG" is used. The name "LNG" is replaced later.
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [gas], l: [primary]}
```

(continues on next page)

(continued from previous page)

```
- key: price_o_w:n-y-h
  # l="import" is used for the global region.
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [crudeoil], l: [primary, import]}
- key: price_b_w:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [biomass], l: [primary]}
- key: price_e_w:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [electr], l: [secondary]}
- key: price_h_w:n-y-h
  # For the global region: l="import", c="l2h".
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [hydrogen], l: [secondary]}
- key: price_liq_o_w:n-y-h
  # l="import" is used for the global region.
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [lightoil], l: [secondary, import]}
- key: price_liq_b_w:n-y-h
  # l="import" is used for the global region.
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [ethanol], l: [secondary, import]}
- key: price_final_e:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [electr], l: [final]}
- key: price_final_sol_c:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [coal], l: [final]}
- key: price_final_sol_b:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [biomass], l: [final]}
- key: price_final_liq_o:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [lightoil], l: [final]}
- key: price_final_liq_b:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [ethanol], l: [final]}
- key: price_final_gas:n-y-h
  inputs:
  - quantity: PRICE_COMMODITY
    select: {c: [gas], l: [final]}
# TODO complete or replace this
# - key: land_out:n-s-y-h
#   inputs:
#   - quantity: land_output
#     select:
#       l: [land_use_reporting]
#       c: ["Price|Agriculture|Non-Energy Crops and Livestock|Index"]
```

(continues on next page)

(continued from previous page)

```
general:
- key: Liquids:nl-ya
  comp: apply_units
  inputs: [liquids:nl-ya]
  args:
    units: 'GWa / year'

- key: Gases:nl-ya
  comp: apply_units
  inputs: [gases:nl-ya]
  args:
    units: 'GWa / year'

- key: Electricity|Fossil|w/o CCS:nl-ya
  comp: apply_units
  inputs: [electr_fossil_w/o_ccs:nl-ya]
  args:
    units: 'GWa / year'

- key: Electricity|Fossil|w/ CCS:nl-ya
  comp: apply_units
  inputs: [electr_fossil_w/_ccs:nl-ya]
  args:
    units: 'GWa / year'

- key: Electricity|Fossil:nl-ya
  comp: apply_units
  inputs: [electr_fossil:nl-ya]
  args:
    units: 'GWa / year'

- key: secondary_energy:nl-t-ya
  comp: select
  inputs: [out:nl-t-ya:se_1+se]
  args:
    indexers:
      t:
      # - Electricity
      - Electricity|Biomass
      - Electricity|Biomass|w/ CCS
      - Electricity|Biomass|w/o CCS
      - Electricity|Coal
      - Electricity|Coal|w/ CCS
      - Electricity|Coal|w/o CCS
      # - Electricity|Fossil
      # - Electricity|Fossil|w/ CCS
      # - Electricity|Fossil|w/o CCS
      - Electricity|Gas
      - Electricity|Gas|w/ CCS
      - Electricity|Gas|w/o CCS
      - Electricity|Geothermal
      - Electricity|Hydro
      # - Electricity|Non-Biomass Renewables
      - Electricity|Nuclear
      - Electricity|Oil
      - Electricity|Oil|w/o CCS
      # - Electricity|Other
      # - Electricity|Solar
      # - Electricity|Solar|CSP
      # - Electricity|Solar|PV
      # - Electricity|Storage Losses
```

(continues on next page)

(continued from previous page)

```
      # - Electricity|Transmission Losses
      # - Electricity|Wind
      # - Electricity|Wind|Offshore
      # - Electricity|Wind|Onshore
      # - Gases
      - Gases|Biomass
      - Gases|Coal
      - Gases|Natural Gas
      - Gases|Other
      - Heat
      - Heat|Biomass
      - Heat|Coal
      - Heat|Gas
      - Heat|Geothermal
      - Heat|Oil
      - Heat|Other
      - Hydrogen
      - Hydrogen|Biomass
      - Hydrogen|Biomass|w/ CCS
      - Hydrogen|Biomass|w/o CCS
      - Hydrogen|Coal
      - Hydrogen|Coal|w/ CCS
      - Hydrogen|Coal|w/o CCS
      - Hydrogen|Electricity
      - Hydrogen|Fossil
      - Hydrogen|Fossil|w/ CCS
      - Hydrogen|Fossil|w/o CCS
      - Hydrogen|Gas
      - Hydrogen|Gas|w/ CCS
      - Hydrogen|Gas|w/o CCS
      # - Liquids
      - Liquids|Biomass
      - Liquids|Biomass|w/ CCS
      - Liquids|Biomass|w/o CCS
      - Liquids|Coal
      - Liquids|Coal|w/ CCS
      - Liquids|Coal|w/o CCS
      - Liquids|Fossil
      - Liquids|Fossil|w/ CCS
      - Liquids|Fossil|w/o CCS
      - Liquids|Gas
      - Liquids|Gas|w/ CCS
      - Liquids|Gas|w/o CCS
      - Liquids|Oil
      # - Solids
      # - Solids|Biomass
      # - Solids|Coal

- key: secondary_energy2:nl-t-ya
  comp: apply_units
  inputs: [secondary_energy:nl-t-ya]
  args:
    units: 'GWa / year'

# For secondary energy, only the 'main' output of technologies that produce
# hydrogen
- key: out:*:h2
  comp: select
  inputs: [out]
  args:
    indexers:
```

(continues on next page)

(continued from previous page)

```
      t: [h2_coal, h2_coal_ccs, h2_smr, h2_smr_ccs, h2_bio, h2_bio_ccs]
      c: [hydrogen]

# All other technologies not in out::h2
- key: out:*:se_0
  comp: select
  inputs: [out]
  args:
    indexers:
      t: [h2_coal, h2_coal_ccs, h2_smr, h2_smr_ccs, h2_bio, h2_bio_ccs]
    inverse: true

# For secondary energy, only the 'main' output of technologies that produce
# ethanol
- key: out::eth
  comp: select
  inputs: [out]
  args:
    indexers:
      t: [eth_bio, eth_bio_ccs, liq_bio, liq_bio_ccs]
      c: [ethanol]

# For secondary energy, only the 'main' output of technologies that produce
# methanol
- key: out::meth
  comp: select
  inputs: [out]
  args:
    indexers:
      t: [meth_coal, meth_coal_ccs]
      c: [methanol]

# For secondary energy, only the 'main' output of technologies that produce
# lightoil
- key: out::liq
  comp: select
  inputs: [out]
  args:
    indexers:
      t: [syn_liq, syn_liq_ccs]
      c: [lightoil]

# TODO re-combine out::liq (and others?), similar to how out::h2 is handled
#      below.

# Re-combine only the 'main' outputs of technologies for SE computations
- key: out:*:se_1
  comp: concat
  inputs:
  - out::h2
  - out::se_0

- key: solids_sum:nl-t-ya
  comp: select
  inputs: [in:nl-t-ya:se]
  args:
    indexers:
      t:
      - Solids
      - Solids|Biomass
      - Solids|Coal
```

(continues on next page)

(continued from previous page)

```
- key: solids:nl-t-ya
  comp: apply_units
  inputs: [solids_sum:nl-t-ya]
  args:
    units: 'GWa / year'

- key: gdp_ppp
  comp: product
  inputs:
  - GDP
  - MERtoPPP

# CH4 emissions from GLOBIOM: select only the subset
- key: land_out:n-s-y-c-l-h:CH4_0
  comp: select
  inputs: [land_out]
  args:
    indexers:
      c:
      - Agri_CH4
      - Emissions|CH4|Land Use|Agriculture|Enteric Fermentation
      - Emissions|CH4|Land Use|Agriculture|AWM
      l:
      - land_use_reporting
  sums: true

# Auto-sum over [l, h], apply units
- key: land_out:n-s-y-c:CH4_0+1
  comp: apply_units
  inputs: [land_out:n-s-y-c:CH4_0]
  args:
    units: 'Mt / year'
  sums: true

# Remove elements from 'emi' so that the remainder have consistent units.
# 1. 'TCE' emissions. These appear to be used for some internal model purpose
#    (maybe relations?) and have units 'tC'.
# 2. Water-related emissions. These have units '-'.
#
# TODO check if this is correct. If the actual units for different 'e' values
#      are not the same, then add another 'comp: select' so that emi is split
#      into two (or more) separate quantities, each with consistent units
- key: emi::_0
  comp: select
  inputs: [emi]
  args:
    # Remove the elements below
    inverse: true
    indexers:
      e: [TCE,
          fresh_consumption, fresh_thermal_pollution, fresh_wastewater,
          instream_consumption,
          saline_consumption, saline_thermal_polution, saline_wastewater]

# <emi::_0> filters out elements with units other than 'kg / kWa'. The units
# of emission_factor are stored as 'kg / kWa', and 'ACT' is in GWa / year, so
# assigning kt / year gives correct results.
#
- key: emi:nl-t-yv-ya-m-e-h:ghg
  inputs: [emi::_0]
```

(continues on next page)

(continued from previous page)

```
  comp: apply_units
  args:
    units: 'kt / year'  # TODO check if this is correct
  sums: true

# GWP factors retrieved from the iam_units package. Dimensionless
- key: gwp factors:gwp metric-e-e equivalent
  comp: gwp_factors

# Emissions converted to GWP-equivalent species
- key: emi::gwpe
  comp: product
  inputs:
  - emi::ghg
  - gwp factors


# Groups of keys for re-use. These keys are not parsed by
# reporting.prepare_reporter; they only exist to be referenced further in
# the file.
#
# - Ending a line with '&label' defines a YAML anchor.
# - Using the YAML alias '<<: *label' re-uses the referenced keys.
_iamc formats:
  primary energy: &pe_iamc
    drop:  # Drop 'commodity', 'level', 'mode', 'node_dest', 'node_origin'
    - c
    - l
    - m
    - nd
    - 'no'  # Bare no is a special YAML value for False, so must quote here.
    - t

  price_iamc: &price_iamc
    unit: USD_2010 / GJ


iamc:
- variable: GDP|MER
  base: GDP:n-y
  unit: billion USD_2010 / year

- variable: GDP|PPP
  base: gdp_ppp:n-y
  unit: billion USD_2010 / year

- variable: Primary Energy|Coal
  base: coal:nl-ya
  <<: *pe_iamc

- variable: Primary Energy|Gas
  base: gas:nl-ya
  <<: *pe_iamc

# NB still incomplete
# - variable: Primary Energy|Geothermal
#   base: out:nl-t-ya-m-c-l
#   select: {l: [secondary], t: [geothermal elec, geothermal heat]}
#   <<: *pe_iamc

- variable: Primary Energy|Hydro
```

(continues on next page)

(continued from previous page)

```
  base: out:nl-t-ya-m-c-l:se
  select: {l: [secondary], t: [hydro]}
  <<: *pe_iamc

- variable: Primary Energy|Nuclear
  base: out:nl-t-ya-m-c-l:se
  select: {l: [secondary], t: [nuclear]}
  <<: *pe_iamc

- variable: Primary Energy|Oil
  base: oil:nl-ya
  <<: *pe_iamc

- variable: Primary Energy|Other
  base: in:nl-t-ya-m-c-l:bunker
  select: {t: [lh2]}
  <<: *pe_iamc

- variable: Primary Energy|Secondary Energy Trade
  base: se_trade:nl-ya
  <<: *pe_iamc

- variable: Primary Energy|Solar
  base: solar:nl-ya
  <<: *pe_iamc

- variable: Primary Energy|Wind
  base: wind:nl-ya
  <<: *pe_iamc

# Secondary Energy
- variable: Secondary Energy
  base: secondary_energy2:nl-t-ya
  unit: 'EJ/yr'
  var: [t]

- variable: Secondary Energy|Electricity|Fossil
  base: Electricity|Fossil:nl-ya
  unit: 'EJ/yr'

- variable: Secondary Energy|Electricity|Fossil|w/ CCS
  base: Electricity|Fossil|w/ CCS:nl-ya
  unit: 'EJ/yr'

- variable: Secondary Energy|Electricity|Fossil|w/o CCS
  base: Electricity|Fossil|w/o CCS:nl-ya
  unit: 'EJ/yr'

- variable: Secondary Energy|Gases
  base: Gases:nl-ya
  unit: 'EJ/yr'

- variable: Secondary Energy|Liquids
  base: Liquids:nl-ya
  unit: 'EJ/yr'

- variable: Secondary Energy|Solids
  base: solids:nl-t-ya
  unit: 'EJ/yr'
  var: [t]
```

(continues on next page)

(continued from previous page)

```
# Emissions

# CH4 emissions from MESSAGE technologies
- variable: Emissions|CH4
  # Auto-sum over dimensions yv, m, h
  base: emi:nl-t-ya:CH4_0+1+2
  var: [t]
  unit: 'Mt / year'  # CH4; the species is indicated by 'variable'
  select:
    t:
    - AFOLU|Biomass Burning
    - Electricity
    - Energy|Demand|Residential and Commercial
    - Energy|Demand|Transportation|Road Rail and Domestic Shipping
    - Fugitive
    - Gases|Coal
    - Gases|Extraction
    - Gases|Hydrogen
    - Gases|Transportation
    - Heat
    - Industrial Processes
    - Liquids|Biomass
    - Liquids|Coal
    - Liquids|Extraction
    - Liquids|Natural Gas
    - Liquids|Oil
    - Liquids|Transportation
    - Solids|Extraction
    - Solids|Transportation
    - Waste

# CH4 emissions from GLOBIOM
# - The variable name signals utils.collapse to make some replacements, then is
#   removed.
- variable: land_out CH4
  base: land_out:n-s-y-c:CH4_0+1+2
  rename: {y: year}
  var: [c, s]

# SF6 emissions
- variable: Emissions
  # Auto sum over t, yv, m, h
  base: emi:nl-ya-e-gwp metric-e equivalent:gwpe+agg
  var:  # Add these to 'variable' column; collapse_gwp_info() is applied
  - e
  - e equivalent
  - gwp metric
  unit: Mt / year  # Species captured in 'e equivalent'

# Prices
# Preferred method: convert all the contents of the variable at once.
- variable: Price
  base: PRICE_COMMODITY:n-c-l-y
  var: [l, c]
  <<: *price_iamc
- variable: Price|Carbon
  base: price_carbon:n-y
  # This was initially "carbon_dioxide", which is not a unit.
  # TODO check that Mt (rather than t or kt) is correct.
  # TODO check whether there is a species / GWP conversion here.
```

(continues on next page)

(continued from previous page)

```
  unit: USD_2010 / Mt
  rename: {y: year}
# commented: see above
# - variable: Price w/o carbon
#   base: price ex carbon:n-t-y-c-e
#   var: [t, c, l, e]
#   rename: {y: year}

# TODO ensure these are covered by the preferred method, above, and then
#      remove these separate conversions.
- variable: Price (legacy)|Primary Energy wo carbon price|Biomass
  base: price_b_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Primary Energy wo carbon price|Coal
  base: price_c:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Primary Energy wo carbon price|Gas
  base: price_g_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Primary Energy wo carbon price|Oil
  base: price_o_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Secondary Energy wo carbon price|Electricity
  base: price_e_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Secondary Energy wo carbon price|Hydrogen
  base: price_h_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Secondary Energy wo carbon price|Liquids|Biomass
  base: price_liq_b_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Secondary Energy wo carbon price|Liquids|Oil
  base: price_liq_o_w:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Final Energy wo carbon price|Residential|Electricity
  base: price_final_e:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Final Energy wo carbon price|Residential|Gases|Natural␣
↪Gas
  base: price_final_gas:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Final Energy wo carbon price|Residential|Liquids|Biomass
  base: price_final_liq_b:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Final Energy wo carbon price|Residential|Liquids|Oil
  base: price_final_liq_o:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Final Energy wo carbon price|Residential|Solids|Biomass
  base: price_final_sol_b:n-y-h
  <<: *price_iamc
- variable: Price (legacy)|Final Energy wo carbon price|Residential|Solids|Coal
  base: price_final_sol_c:n-y-h
  <<: *price_iamc
#- variable: Price (legacy)|Agriculture|Non-Energy Crops and Livestock|Index
#  base: price_agriculture:n-y-h
#  rename: {y: year}

report:
- key: pe test
  members:
#  - Primary Energy|Biomass::iamc
```

(continues on next page)

(continued from previous page)

```
  - Primary Energy|Coal::iamc
  - Primary Energy|Gas::iamc
  - Primary Energy|Hydro::iamc
  - Primary Energy|Nuclear::iamc
  - Primary Energy|Solar::iamc
  - Primary Energy|Wind::iamc

- key: gdp test
  members:
  - GDP|MER::iamc
  - GDP|PPP::iamc

- key: se test
  members:
  - Secondary Energy::iamc
  - Secondary Energy|Electricity|Fossil::iamc
  - Secondary Energy|Electricity|Fossil|w/ CCS::iamc
  - Secondary Energy|Electricity|Fossil|w/o CCS::iamc
  - Secondary Energy|Gases::iamc
  - Secondary Energy|Liquids::iamc
  - Secondary Energy|Solids::iamc

- key: emissions
  members:
  - Emissions::iamc

- key: CH4 emissions
  members:
  - Emissions|CH4::iamc
  - land_out CH4::iamc
  # - Emissions|CH4|Fossil Fuels and Industry::iamc
  # - Emissions|CH4|Energy|Supply|Gases|Biomass|Fugitive::iamc
  # - Emissions|CH4|Energy|Supply|Gases|Natural Gas|Fugitive::iamc
  # - Emissions|CH4|Energy|Supply|Solids|Biomass|Fugitive::iamc
  # - Emissions|CH4|Energy|Supply|Solids|Coal|Fugitive::iamc

- key: price test
  members:
  - Price::iamc
  - Price|Carbon::iamc
  # commented: see above
  # - Price w/o carbon::iamc

  # TODO ensure these are covered by the preferred method, above, then remove
  #      these
  - Price (legacy)|Primary Energy wo carbon price|Biomass::iamc
  - Price (legacy)|Primary Energy wo carbon price|Coal::iamc
  - Price (legacy)|Primary Energy wo carbon price|Gas::iamc
  - Price (legacy)|Primary Energy wo carbon price|Oil::iamc
  - Price (legacy)|Secondary Energy wo carbon price|Electricity::iamc
  - Price (legacy)|Secondary Energy wo carbon price|Hydrogen::iamc
  - Price (legacy)|Secondary Energy wo carbon price|Liquids|Biomass::iamc
  - Price (legacy)|Secondary Energy wo carbon price|Liquids|Oil::iamc
  # NB for "Price|Secondary Energy|Liquids|Oil", the legacy reporting inserts a
  #    zero matrix.
  - Price (legacy)|Final Energy wo carbon price|Residential|Electricity::iamc
  - Price (legacy)|Final Energy wo carbon price|Residential|Gases|Natural Gas::iamc
  - Price (legacy)|Final Energy wo carbon price|Residential|Liquids|Biomass::iamc
  - Price (legacy)|Final Energy wo carbon price|Residential|Liquids|Oil::iamc
  - Price (legacy)|Final Energy wo carbon price|Residential|Solids|Biomass::iamc
  - Price (legacy)|Final Energy wo carbon price|Residential|Solids|Coal::iamc
```

8.7.2 'Legacy' reporting (`report.legacy`)

report.legacy contains 'legacy' reporting code transferred from message_data ().

This code:

- is only tested to *run* against *the global model snapshot*, specifically snapshot ID `1`. This implies that the *outputs*—specific numerical values, labels, etc.—of the code are not tested or validated in any way. Users of the code should carefully validate output data, especially when the code is run against any scenario other than snapshot `1`.
- is provided primarily for reference by users of snapshot `1`.
- is not currently used in IIASA ECE program workflows or publications. As of 2024-04-04, all MESSAGEix-GLOBIOM outputs that appear in public and closed databases, associated with peer-reviewed works, etc. have been produced with other code from various branches of `message_data`.
- predates `genno` and the stack of tools built on it (*described here*); these were designed to avoid issues with performance and extensibility in the older code.[1] It is intended that *report* will eventually replace *report.legacy*, which in the meantime serves as a reference for that code (such as *report.compat*).

Usage

Set *report.Config.legacy* to include `use=True` and any other keyword arguments to *iamc_report_hackathon.report()*, then call *message_ix_models.report.report()*:

```
from message_ix_models import Context
from message_ix_models.report import Config, report

# Configure to use .report.legacy
context = Context.get_instance()
context.report.legacy.update(
    use=True,
    # Only this exact set of keyword arguments is
    # tested and known to work:
    merge_hist=True,
    ref_sol="True",
    run_config="ENGAGE_SSP2_v417_run_config.yaml",
)

# Invoke
report(context)
```

Or, call *iamc_report_hackathon.report()* directly.

Reference

[1] See a (non-public) "Reporting" project board on GitHub for details of the initial implementation of these features.

message_ix_models.report.legacy.iamc_report_hackathon.**report**(*mp*, *scen*, *ref_sol=False*, *model_out=None*, *scenario_out=None*, *out_dir=None*, *merge_hist=False*, *merge_ts=False*, *aggr_def=None*, *var_def=None*, *unit_yaml=None*, *run_config=None*, *urban_perc=None*, *kyoto_hist=None*, *lu_hist=None*, *verbose=False*, *, *context:* Context | *None* = *None*)

Main reporting function.

This function will run reporting for specific "tables" as specified in the configuration file *run_config*.

Outputs will be stored as an xlsx file in IAMC format for upload to a scenario database/explorer instance.

IMPORTANT!! If extending the variable template, please ensure NOT to overwrite the existing file as this is used for global model intercomparison projects.

Only variables defined in the variable template are reported. All other variables will be excluded.

Parameters

- **mp** (ixmp.Platform) – Database connection where scenario object is located.
- **scen** (message_ix.Scenario) – Scenario object for which reporting should be run.
- **ref_sol** (boolean (default: False)) – Option whether to process historical results or optimization results.
- **model_out** (str (default: None)) – Model name of the scenario in the output file.
- **scenario_out** (str (default: None)) – Scenario name of the scenario in the output file.
- **out_dir** (str (default: None)) – Directory where the result file should be written to.
- **merge_hist** (boolean (default: False)) – Switch to determine whether the reporting results should be merged with already processed histroical results, which are then, additionally, stored as Timeseries with the scenario object.
- **merge_ts** (boolean (default: False)) – Switch to use data stored as TS to overwrite results from reporting.
- **var_def** (str (default: None)) – Name of file to be used to define allowed variables.
- **aggr_def** (str (default=None)) – Name of file to be used to define aggregate mapping.
- **unit_yaml** (str (default: None)) – Directory incl file name of unit conversion factors from model units to output units.
- **run_config** (str (default: None)) – Directory incl file name of which reporting tables are to be run.
- **urban_perc** (str (default: None)) – Regional urban shares in %.

- **kyoto_hist** (str (default: None)) – Historic Kyoto Gas emissions excl. land-use emissions for regions.
- **lu_hist** (str (default: None)) – Historic land-use GHG emissions for regions.
- **verbose** (str (default: False)) – Option whther to print onscreen messages.
- **context** (*Context*) – Only the dry_run setting is respected. If True, configuration is read, but nothing is done.

8.7.3 Introduction

See the discussion in the MESSAGEix docs about the stack. In short, for instance:

- message_ix **must not** contain reporting code that references technology="coal_ppl", because not every model built on the MESSAGE framework will have a technology with this name.
- Any model in the MESSAGEix-GLOBIOM family—built with *message_ix_models* and/or message_data—**should**, with few exceptions, have a technology="coal_ppl", since this appears in the common list of *Technologies (technology.yaml)*. Reporting specific to this technology ID, *as it is represented* in this model family, should be in *message_ix_models* or user code.

The basic **design pattern** of *message_ix_models.report* is:

- *prepare_reporter()* populates a new Reporter for a given Scenario with many keys to report all quantities of interest in a MESSAGEix-GLOBIOM–family model.
- This function relies on *callbacks* defined in multiple submodules to add keys and tasks for general or tailored reporting calculations and actions. Additional modules **should** define callback functions and register them with register() when they are to be used. For example:

  1. The module *message_ix_models.report.plot* defines *plot.callback()* that adds standard plots to the Reporter.
  2. The module message_data.model.transport.report defines callback() that adds tasks specific to MESSAGEix-Transport.
  3. The module message_data.projects.navigate.report defines callback() that add tasks specific to the 'NAVIGATE' research project.

  The callback (1) is always registered, because these plots are always applicable and can be expected to function correctly for all models in the family. In contrast, (2) and (3) **should** only be registered and run for the specific model variants for which they are developed/intended.

  Modules with tailored reporting configuration **may** also be indicated on the *command line* by using the **-m/--modules** option: **mix-models report -m model.transport**.

- A file global.yaml file (in YAML format) contains a description of some of the reporting computations needed for a MESSAGE-GLOBIOM model. *prepare_reporter()* uses the configuration handlers built into genno (and some extensions specific to *message_ix_models*) to handle the different sections of the file.

8.7.4 Features

By combining these genno, ixmp, message_ix, and message_ix_models features, the following functionality is provided.

Note: If any of this does not appear to work as advertised, file a bug!

Units

- Are read automatically for ixmp parameters.
- Pass through calculations/are derived automatically.
- Are recognized based on the definitions of non-SI units from IAMconsortium/units.
- Are discarded when inconsistent.
- Can be overridden for entire parameters:

```
units:
  apply:
    inv_cost: USD
```

- Can be set explicitly when converting data to IAMC format:

```
iamc:
# 'value' will be in kJ; 'units' will be the string 'kJ'
- variable: Variable Name
  base: example_var:a-b-c
  units: kJ
```

8.7.5 API reference

Config(from_file, _legacy, cli_output, ...)	Settings for *message_ix_models.report*.
prepare_reporter(context[, scenario, reporter])	Return a Reporter and *key* prepared to report a Scenario.
register(name_or_callback)	Register a callback function for *prepare_reporter()*.
report(context, *args, **kwargs)	Report (post-process) solution data in a Scenario.

class message_ix_models.report.**Config**(*from_file: dataclasses.InitVar[typing.Optional[pathlib.Path]] = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/iiasa-energy-program-message-ix-models/envs/stable/lib*
_legacy: dataclasses.InitVar[typing.Optional[bool]] = False, cli_output: ~pathlib.Path | None = None, genno_config: ~typing.Dict = <factory>, key: KeyLike | None = None, output_dir: ~pathlib.Path | None = <factory>, use_scenario_path: bool = True, legacy: ~typing.Dict = <factory>)

Settings for *message_ix_models.report*.

When initializing a new instance, the *from_file* and *_legacy* parameters are respected.

cli_output: Path | None = None

Path to write reporting outputs when invoked from the command line.

from_file: dataclasses.InitVar[Optional[pathlib.Path]] = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/iiasa-energy-program-message-ix-models/envs/stable/lib/python3.10/site-packages/message_ix_models/data/report/global.yaml')

Shorthand to call *use_file()* on a new instance.

genno_config: Dict

Configuration to be handled by genno.config.

key: KeyLike | None = None

Key for the Quantity or computation to report.

legacy: Dict

Keyword arguments for *report.legacy.iamc_report_hackathon.report()*, plus the key "use", which should be `True` if legacy reporting is to be used.

mkdir() → None

Ensure the *output_dir* exists.

output_dir: Path | None

Directory for output.

set_output_dir(*arg: Path | None*) → None

Set *output_dir*, the output directory.

The value is also stored to be passed to `genno` as the "output_dir" configuration key.

use_file(*file_path: str | Path | None*) → None

Use genno configuration from a (YAML) file at *file_path*.

See `genno.config` for the format of these files. The path is stored at `.genno_config["path"]`, where it is picked up by genno's configuration mechanism.

Parameters

file_path (`os.PathLike`, *optional*) – This may be:

1. The complete path to any existing file.
2. A stem like "global" or "other". This is interpreted as referring to a file named, for instance, `global.yaml`.
3. A partial path like "project/report.yaml". This or (2) is interpreted as referring to a file within `MESSAGE_MODELS_PATH/data/report/`; that is, a file packaged and distributed with *message_ix_models*.

use_scenario_path: bool = True

`True` to use an output directory based on the scenario's model name and name.

`message_ix_models.report.`**prepare_reporter**(*context:* Context, *scenario: Scenario | None = None*, *reporter: Reporter | None = None*) → Tuple[Reporter, Key]

Return a `Reporter` and *key* prepared to report a `Scenario`.

Parameters

- **context** (*Context*) – The code responds to `context.report`, which is an instance of *report.Config*.
- **scenario** (`Scenario`, *optional*) – Scenario to report. If not given, *Context.get_scenario()* is used to retrieve a Scenario.
- **reporter** (`Reporter`, *optional*) – Existing reporter to extend with computations. If not given, it is created using `message_ix.Reporter.from_scenario()`.

Returns

- `Reporter` – Reporter prepared with MESSAGEix-GLOBIOM calculations; if *reporter* is given, this is a reference to the same object.

  If *cli_output* is given, a task with the key "cli-output" is added that writes the *Config.key* to that path.

- `Key` – Same as *Config.key* if any, but in full resolution; else either "default" or "cli-output" according to the other settings.

message_ix_models.report.**register**(*name_or_callback: Callable | str*) → str | None

Register a callback function for *prepare_reporter()*.

Each registered function is called by *prepare_reporter()*, in order to add or modify reporting keys. Specific model variants and projects can register a callback to extend the reporting graph.

Callback functions must take two arguments: the Reporter, and a *Context*:

```
from message_ix.report import Reporter
from message_ix_models import Context
from message_ix_models.report import register

def cb(rep: Reporter, ctx: Context):
    # Modify `rep` by calling its methods ...
    pass

register(cb)
```

Parameters

name_or_callback – If a string, this may be a submodule of *message_ix_models*, or `message_data`, in which case the function `{message_data, message_ix_models}.{name}.report.callback` is used. Or, it may be a fully-resolved package/module name, in which case `{name}.callback` is used. If a callable (function), it is used directly.

message_ix_models.report.**report**(*context: Context*, **args*, ***kwargs*)

Report (post-process) solution data in a `Scenario`.

This function provides a single, common interface to call both the `genno` -based (*message_ix_models.report*) and 'legacy' (*message_ix_models.report.legacy*) reporting codes.

Parameters

context (`Context`) – The code responds to:

- *dry_run*: if `True`, reporting is prepared but nothing is done.
- *scenario_info* and *platform_info*: used to retrieve the Scenario to be reported.
- `context.report`, which is an instance of *report.Config*; see there for available configuration settings.

Plots

Plots for MESSAGEix-GLOBIOM reporting.

The current set functions on time series data stored on the scenario by *message_ix_models.report* or `message_data` legacy reporting.

class message_ix_models.report.plot.**EmissionsCO2**

CO_2 Emissions.

basename = 'emission-CO2'

File name base for saving the plot.

generate(*data: DataFrame*, *scenario: Scenario*)

Generate and return the plot.

A subclass of Plot **must** implement this method.

Parameters

args (`Sequence` of `pandas.DataFrame` or `other`) – One argument is given corresponding to each of the *inputs*.

Because `plotnine` operates on pandas data structures, `save()` automatically converts any `Quantity` inputs to `pandas.DataFrame` before they are passed to *`generate()`*.

inputs: Sequence[str] = ['Emissions|CO2::iamc', 'scenario']

`Keys` referring to `Quantities` or other inputs accepted by *`generate()`*.

static: List['plotnine.typing.PlotAddable'] = [<plotnine.themes.theme.theme object>, {'x': 'year', 'y': 'value', 'color': 'region'}, <plotnine.geoms.geom_line.geom_line object>, <plotnine.geoms.geom_point.geom_point object>, <plotnine.labels.labs object>]

list of plotnine objects that are not dynamic.

Type
'Static' geoms

class message_ix_models.report.plot.**FinalEnergy0**

Final Energy.

basename = 'fe0'

File name base for saving the plot.

inputs: Sequence[str] = ['Final Energy::iamc', 'scenario']

`Keys` referring to `Quantities` or other inputs accepted by `generate()`.

class message_ix_models.report.plot.**FinalEnergy1**

Final Energy.

basename = 'fe1'

File name base for saving the plot.

generate(*data: DataFrame*, *scenario: Scenario*)

Generate and return the plot.

A subclass of Plot **must** implement this method.

Parameters
args (`Sequence` of `pandas.DataFrame` or `other`) – One argument is given corresponding to each of the *`inputs`*.

Because `plotnine` operates on pandas data structures, `save()` automatically converts any `Quantity` inputs to `pandas.DataFrame` before they are passed to *`generate()`*.

inputs: Sequence[str] = ['fe1-0::iamc', 'scenario']

`Keys` referring to `Quantities` or other inputs accepted by *`generate()`*.

inputs_regex: List[re.Pattern] = [re.compile('Final Energy\\|(Electricity|Gases|Geothermal|Heat|Hydrogen|Liquids|Solar|Solids)')]

List of regular expressions corresponding to *`inputs`*. These are passed as the *expr* argument to *`filter_ts()`* to filter the entire set of time series data.

static: List['plotnine.typing.PlotAddable'] = [<plotnine.themes.theme.theme object>, {'x': 'year', 'y': 'value', 'fill': 'variable'}, <plotnine.geoms.geom_bar.geom_bar object>, <plotnine.labels.labs object>]

list of plotnine objects that are not dynamic.

Type
'Static' geoms

message_ix_models.report.plot.**PLOTS = (<class 'message_ix_models.report.plot.EmissionsCO2'>, <class 'message_ix_models.report.plot.FinalEnergy0'>, <class 'message_ix_models.report.plot.FinalEnergy1'>, <class 'message_ix_models.report.plot.PrimaryEnergy0'>, <class 'message_ix_models.report.plot.PrimaryEnergy1'>)**

All plot classes.

class message_ix_models.report.plot.**Plot**

Base class for plots based on reported time-series data.

Subclasses should be used like:

```
class MyPlot(Plot):
    ...

c.add("plot myplot", MyPlot, "scenario")
```

…that is, giving "scenario" or another key that points to a `Scenario` object with stored time series data. See the examples in this file.

classmethod add_tasks(*c: Computer*, *key: KeyLike*, **inputs*, *strict: bool = False*) → KeyLike

Add a task to *c* to generate and save the Plot.

Analogous to `Operator.add_tasks()`.

ggtitle(*value=None*) → ggtitle

Return `plotnine.ggtitle` including the current date & time.

groupby_plot(*data: DataFrame*, **args*)

Combination of groupby and ggplot().

Groups by *args* and yields a series of `plotnine.ggplot` objects, one per group, with *`static`* geoms and *`ggtitle()`* appended to each.

inputs: Sequence[str] = []

`Keys` referring to `Quantities` or other inputs accepted by `generate()`.

inputs_regex: List[Pattern] = []

List of regular expressions corresponding to *`inputs`*. These are passed as the *expr* argument to *`filter_ts()`* to filter the entire set of time series data.

static: List[plotnine.typing.PlotAddable] = [<plotnine.themes.theme.theme object>]

list of plotnine objects that are not dynamic.

Type
'Static' geoms

title = None

Fixed plot title string. If not given, the first line of the class docstring is used.

unit = None

Units expression for plot title.

url: str | None = None

Scenario URL for plot title.

class message_ix_models.report.plot.**PrimaryEnergy0**

Primary Energy.

basename = 'pe0'

File name base for saving the plot.

inputs: Sequence[str] = ['Primary Energy::iamc', 'scenario']

Keys referring to Quantities or other inputs accepted by generate().

class message_ix_models.report.plot.**PrimaryEnergy1**

Primary Energy.

basename = 'pe1'

File name base for saving the plot.

inputs: Sequence[str] = ['pe1-0::iamc', 'scenario']

Keys referring to Quantities or other inputs accepted by generate().

inputs_regex: List[re.Pattern] = [re.compile('Primary Energy\\|((?!Fossil|Non-Biomass Renewables|Secondary Energy Trade)[^\\|]*)')]

List of regular expressions corresponding to *inputs*. These are passed as the *expr* argument to *filter_ts()* to filter the entire set of time series data.

message_ix_models.report.plot.**callback**(*c: Computer*, *context:* Context) → None

Add all *PLOTS* to *c*.

Also add a key "plot all" to triggers the generation of all plots.

Operators

Atomic reporting operations for MESSAGEix-GLOBIOM.

message_ix_models.report.operator provides the following:

codelist_to_groups(codes[, dim])	Convert *codes* into a mapping from parent items to their children.
compound_growth(qty, dim)	Compute compound growth along *dim* of *qty*.
exogenous_data	No action.
filter_ts(df, expr, *[, column])	Filter time series data in *df*.
from_url(url[, cls])	Return a ixmp.TimeSeries or subclass instance, given its *url*.
get_ts(scenario[, filters, iamc, subannual])	Retrieve timeseries data from *scenario*.
gwp_factors()	Use iam_units to generate a Quantity of GWP factors.
make_output_path(config, name)	Return a path under the "output_dir" Path from the reporter configuration.
model_periods(y, cat_year)	Return the elements of *y* beyond the firstmodelyear of *cat_year*.
remove_ts(scenario[, config, after, dump])	Remove all time series data from *scenario*.
share_curtailment(curt, *parts)	Apply a share of *curt* to the first of *parts*.

The following functions, defined elsewhere, are exposed through *operator* and so can also be referenced by name:

message_ix_models.util.add_par_data(...[, ...])	Add *data* to *scenario*.

Other operators or genno-compatible functions are provided by:

- Upstream packages:
  - message_ix.report.operator
  - ixmp.report.operator

  - genno.operator

- Other submodules:
  - *model.emissions*: *get_emission_factors()*.

Any of these can be made available for a Computer instance using require_compat(), for instance:

```
# Indicate that a certain module contains functions to
# be referenced by name
c.require_compat("message_ix_models.model.emissions")

# Add computations to the graph by referencing functions
c.add("ef:c", "get_emission_factors", units="t C / kWa")
```

message_ix_models.report.operator.**codelist_to_groups**(*codes: List[Code]*, *dim: str = 'n'*) → Mapping[str, Mapping[str, List[str]]]

Convert *codes* into a mapping from parent items to their children.

The returned value is suitable for use with genno.operator.aggregate().

If this is a list of nodes per *get_codes()*, then the mapping is from regions to the ISO 3166-1 alpha-3 codes of the countries within each region. The code for the region itself is also included in the values to be aggregated, so that already- aggregated data will pass through.

message_ix_models.report.operator.**compound_growth**(*qty: AttrSeries*, *dim: str*) → AttrSeries

Compute compound growth along *dim* of *qty*.

message_ix_models.report.operator.**filter_ts**(*df: DataFrame*, *expr: Pattern*, *, *column='variable'*) → DataFrame

Filter time series data in *df*.

1. Keep only rows in *df* where *expr* is a full match (fullmatch()) for the entry in *column*.
2. Retain only the first match group ("…(…)…") from *expr* as the *column* entry.

message_ix_models.report.operator.**from_url**(*url: str*, *cls=<class 'ixmp.core.timeseries.TimeSeries'>*) → TimeSeries

Return a ixmp.TimeSeries or subclass instance, given its *url*.

Todo: Move upstream, to ixmp.report.

Parameters

cls (type, *optional*) – Subclass to instantiate and return; for instance, Scenario.

message_ix_models.report.operator.**get_ts**(*scenario: Scenario*, *filters: dict | None = None*, *iamc: bool = False*, *subannual: bool | str = 'auto'*)

Retrieve timeseries data from *scenario*.

Corresponds to ixmp.Scenario.timeseries().

Todo: Move upstream, e.g. to ixmp alongside store_ts().

---

message_ix_models.report.operator.**gwp_factors**() → AttrSeries

Use iam_units to generate a Quantity of GWP factors.

The quantity is dimensionless, e.g. for converting [mass] to [mass], andhas dimensions:

- **'gwp metric': the name of a GWP metric, e.g. 'SAR', 'AR4', 'AR5'. All metrics are** on a 100-year basis.
- 'e': emissions species, as in MESSAGE. The entry 'HFC' is added as an alias for the species 'HFC134a' from iam_units.
- 'e equivalent': GWP-equivalent species, always 'CO2'.

message_ix_models.report.operator.**make_output_path**(*config: Mapping*, *name: str*) → Path

Return a path under the "output_dir" Path from the reporter configuration.

message_ix_models.report.operator.**model_periods**(*y: List[int]*, *cat_year: DataFrame*) → List[int]

Return the elements of *y* beyond the firstmodelyear of *cat_year*.

Todo: Move upstream, to `message_ix`.

---

message_ix_models.report.operator.**nodes_ex_world**(*nodes: Sequence[str | Code]*) → List[str | Code]

Exclude "World" and anything containing "GLB" from *nodes*.

May also be used as a genno (reporting) operator.

message_ix_models.report.operator.**remove_ts**(*scenario: Scenario*, *config: dict | None = None*, *after: int | None = None*, *dump: bool = False*) → None

Remove all time series data from *scenario*.

Note that data stored with `add_timeseries()` using `meta=True` as a keyword argument cannot be removed using `TimeSeries.remove_timeseries()`, and thus also not with this operator.

Todo: Move upstream, to `ixmp` alongside `store_ts()`.

---

message_ix_models.report.operator.**share_curtailment**(*curt*, **parts*)

Apply a share of *curt* to the first of *parts*.

If this is being used, it usually will indicate the need to split *curt* into multiple technologies; one for each of *parts*.

Utilities

add_replacements(dim, codes)	Update *REPLACE_DIMS* for dimension *dim* with values from *codes*.
as_quantity(info)	Convert values from a `dict` to Quantity.
collapse(df[, var])	Callback for the *collapse* argument to `convert_pyam()`.
collapse_gwp_info(df, var)	*collapse()* helper for emissions data with GWP dimensions.
copy_ts(rep, other, filters)	Prepare *rep* to copy time series data from *other* to *scenario*.

message_ix_models.report.util.**REPLACE_DIMS: Dict[str, Dict[str, str]] = {'c': {'Agri_Ch4': 'GLOBIOM|Emissions|CH4 Emissions Total'}, 'l': {'Final Energy': 'Final Energy|Residential'}, 't': {}}**

Replacements used in *collapse()*. These are applied using `pandas.DataFrame.replace()` with `regex=True`; see the documentation of that method.

- Applied to whole strings along each dimension.
- These columns have `str.title()` applied before these replacements.

message_ix_models.report.util.**REPLACE_VARS** =
{'(Emissions\\|CH4)\\|((Gases|Liquids|Solids|Elec|Heat)(.*))': '\\1|Energy|Supply|\\3|Fugitive\\4', '(Emissions\\|CH4)\\|Fugitive': '\\1|Energy|Supply|Fugitive', '(Secondary Energy\\|Solids)\\|Solids': '\\1', 'Import Energy\\|(Liquids\\|(Biomass|Oil))': 'Secondary Energy|\\1', 'Import Energy\\|Coal': 'Primary Energy|Coal', 'Import Energy\\|Lh2': 'Secondary Energy|Hydrogen', 'Import Energy\\|Lng': 'Primary Energy|Gas', 'Import Energy\\|Oil': 'Primary Energy|Oil', 'Residential\\|(Biomass|Coal)': 'Residential|Solids|\\1', 'Residential\\|Gas': 'Residential|Gases|Natural Gas', '^(land_out CH4.*\\|)Awm': '\\1Manure Management', '^land_out CH4\\|': '', '^land_out CH4\\|Emissions\\|Ch4\\|Land Use\\|Agriculture\\|': 'Emissions|CH4|AFOLU|Agriculture|Livestock|'}

Replacements used in *collapse()* after the 'variable' column is assembled. These are applied using `pandas.DataFrame.replace()` with `regex=True`; see the documentation of that method. For documentation of regular expressions, see https://docs.python.org/3/library/re.html and https://regex101.com.

Todo: These may be particular or idiosyncratic to a single "template". The strings used to collapse multiple conceptual dimensions into the IAMC "variable" column are known to vary in poorly-documented ways across these templates.

This setting is currently applied universally. To improve, specify a different mapping with the replacements needed for each individual template, and load the correct one when reporting scenarios to that template.

---

message_ix_models.report.util.**add_replacements**(*dim: str*, *codes: Iterable[Code]*) → None

Update *REPLACE_DIMS* for dimension *dim* with values from *codes*.

message_ix_models.report.util.**as_quantity**(*info: dict | float | str*) → AttrSeries

Convert values from a `dict` to Quantity.

Todo: move upstream, to `genno`.

---

message_ix_models.report.util.**collapse**(*df: DataFrame*, *var=[]*) → DataFrame

Callback for the *collapse* argument to `convert_pyam()`.

Replacements from *REPLACE_DIMS* and *REPLACE_VARS* are applied. The dimensions listed in the *var* arguments are automatically dropped from the returned `pyam.IamDataFrame`. If `var[0]` contains the word "emissions", then *collapse_gwp_info()* is invoked.

Adapted from `genno.compat.pyam.collapse()`.

Parameters

var (`list` of `str`, *optional*) – Strings or dimensions to concatenate to the 'Variable' column. The first of these is usually a string value used to populate the column. These are joined using the pipe ('|') character.

See also:

REPLACE_DIMS, *REPLACE_VARS*, *collapse_gwp_info*, `test_collapse`

message_ix_models.report.util.**collapse_gwp_info**(*df*, *var*)

collapse() helper for emissions data with GWP dimensions.

The dimensions 'e equivalent', and 'gwp metric' dimensions are combined with the 'e' dimension, using a format like:

```
'{e} ({e equivalent}-equivalent, {GWP metric} metric)'
```

For example:

```
'SF6 (CO2-equivalent, AR5 metric)'
```

message_ix_models.report.util.**copy_ts**(*rep: Reporter*, *other: str*, *filters: dict | None*) → Key

Prepare *rep* to copy time series data from *other* to *scenario*.

Parameters

- **other_url** (str) – URL of the other scenario from which to copy time series data.
- **filters** (dict, *optional*) – Filters; passed via store_ts() to ixmp.TimeSeries.timeseries().

Returns

Key for the copy operation.

Return type

str

Compatibility with report.legacy

Compatibility code that emulates legacy reporting.

report.compat prepares a Reporter to perform the same calculations as *report.legacy*, except using genno.

Warning: This code is **under development** and **incomplete**. It is not yet a full or exact replacement for *report.legacy*. Use with caution.

Main API:

TECH_FILTERS	Filters for determining subsets of technologies.
callback(rep, context)	Partially duplicate the behaviour of default_tables.retr_CO2emi().
prepare_techs(c, technologies)	Prepare sets of technologies in *c*.
get_techs(c, prefix[, kinds])	Return a list of technologies.

Utility functions:

inp(c, technologies, *[, name, filters, ...])	
eff(c, technologies[, filters_in, filters_out])	Throughput efficiency (input / output) for *technologies*.
emi(c, technologies, *[, name, filters, ...])	
out(c, technologies, *[, name, filters, ...])	

message_ix_models.report.compat.**TECH_FILTERS = {'gas all': "c_in == 'gas' and l_in in 'secondary final' and '_ccs' not in id", 'gas extra': 'False', 'rc gas': "sector == 'residential/commercial' and c_in == 'gas'", 'trp coal': "sector == 'transport' and c_in == 'coal'", 'trp foil': "sector == 'transport' and c_in == 'fueloil'", 'trp gas': "sector == 'transport' and c_in == 'gas'", 'trp loil': "sector == 'transport' and c_in == 'lightoil'", 'trp meth': "sector == 'transport' and c_in == 'methanol'"}**

Filters for determining subsets of technologies.

Each value is a Python expression `eval()`'d in an environment containing variables derived from the annotations on `Codes` for each technology. If the expression evaluates to `True`, then the code belongs to the set identified by the key.

See also:

get_techs, *prepare_techs*

`message_ix_models.report.compat.`**`callback`**(*rep: Reporter*, *context:* Context) → None

Partially duplicate the behaviour of `default_tables.retr_CO2emi()`.

Currently, this prepares the following keys and the necessary preceding calculations:

- "transport emissions full::iamc": data for the IAMC variable "Emissions|CO2|Energy|Demand|Transportation|Road Rail and Domestic Shipping"

`message_ix_models.report.compat.`**`eff`**(*c: Computer*, *technologies: List[str]*, *filters_in: dict | None = None*, *filters_out: dict | None = None*) → Key

Throughput efficiency (input / output) for *technologies*.

Equivalent to `PostProcess.eff()`.

Parameters

- **`filters_in`** (`dict`, *optional*) – Passed as the *filters* parameter to `inp()`.
- **`filters_out`** (`dict`, *optional*) – Passed as the *filters* parameter to `out()`.

`message_ix_models.report.compat.`**`get_techs`**(*c: Computer*, *prefix: str*, *kinds: str | None = None*) → List[str]

Return a list of technologies.

The list is assembled from lists in *c* with keys like "t::{prefix} {kind}", with one *kind* for each space-separated item in *kinds*. If no *kinds* are supplied, "t::{prefix}" is used.

See also:

prepare_techs

`message_ix_models.report.compat.`**`prepare_techs`**(*c: Computer*, *technologies: List[Code]*) → None

Prepare sets of technologies in *c*.

For each *key* → *expr* in *TECH_FILTERS* and each technology `Code` *t* in *technologies*:

- Apply the filter expression *expr* to information about *t*.
- If the expression evaluates to `True`, add it to a list in *c* at "t::{key}".

These lists of technologies can be used directly or retrieve with *get_techs()*.

8.7.6 Command-line interface

```
$ mix-models report --help

Usage: mix-models report [OPTIONS] [KEY]

  Postprocess results.

  KEY defaults to the comprehensive report 'message::default', but may also be
  the name of a specific model quantity, e.g. 'output'.

  --config can give either the absolute path to a reporting configuration
  file, or the stem (i.e. name without .yaml extension) of a file in
  data/report.
```

(continues on next page)

(continued from previous page)

```
  With --from-file, read multiple Scenario identifiers from FILE, and report
  each one. In this usage, --output-path may only be a directory.

Options:
  --dry-run             Only show what would be done.
  --config TEXT         Path or stem for reporting config file.  [default:
                        global]
  -L, --legacy          Invoke legacy reporting.
  -m, --module MODULES  Add extra reporting for MODULES.
  -o, --output PATH     Write output to file instead of console.
  --from-file FILE      Report multiple Scenarios listed in FILE.
  --help                Show this message and exit.
```

8.7.7 Testing

Simulated solution data for testing *report*.

message_ix_models.report.sim.**add_simulated_solution**(*rep: Reporter*, *info:* ScenarioInfo, *data: Dict | None = None*, *path: Path | None = None*)

Add a simulated model solution to *rep*.

Parameters

- **data** (dict or pandas.DataFrame, *optional*) – If given, a mapping from MESSAGE item (set, parameter, or variable) names to inputs that are passed to *simulate_qty()*.
- **path** (Path, *optional*) – If given, a path to a directory containing one or more files with names like ACT.csv.gz. These files are taken as containing "simulated" model solution data for the MESSAGE variable with the same name. See *data_from_file()*.

message_ix_models.report.sim.**data_from_file**(*path: Path*, *, *name: str*, *dims: Sequence[str]*) → AttrSeries

Read simulated solution data for item *name* from *path*.

For variables and equations (*name* in upper case), the file **must** have columns corresponding to *dims* followed by "Val", "Marginal", "Upper", and "Scale". The "Val" column is returned.

For parameters, the file **must** have columns corresponding to *dims* followed by "value" and "unit". The "value" column is returned.

message_ix_models.report.sim.**simulate_qty**(*name: str*, *dims: List[str]*, *item_data: dict | DataFrame*) → AttrSeries

Return simulated data for item *name*.

Parameters

- **dims** – Dimensions of the resulting quantity.
- **item_data** – Optional data for the quantity.

message_ix_models.report.sim.**to_simulate**()

Return items to be included in a simulated solution.

Continuous reporting

As part of the *Test suite (message_ix_models.tests)*, reporting is run on the same events (pushes and daily schedule) on publicly-available *model snapshots*. One goal of these tests *inter alia* is to ensure that adjustments and improvements to the reporting code do not disturb manually-verified model outputs.

As part of the (private) `message_data` test suite, multiple workflows run on regular schedules; some of these include a combination of `message_ix_models`-based and *'legacy' reporting*. These workflows:

- Operate on specific scenarios within IIASA databases.
- Create files in CSV, Excel, and/or PDF formats that are that are preserved and made available as 'build artifacts' via the GitHub Actions web interface and API.

8.8 General purpose modeling tools

"Tools" can include, *inter alia*:

- Codes for retrieving data from specific data sources and adapting it for use with `message_ix_models`.
- Codes for modifying scenarios; although tools for building models should go in `message_ix_models.model`.

On other pages:

- *Investment and fixed costs (tools.costs)*

On this page:

8.8.1 Exogenous data (`tools.exo_data`)

Generic tools for working with exogenous data sources.

`MEASURES`	Supported measures.
`SOURCES`	Known sources for data.
`DemoSource`(source, source_kw)	Example source of exogenous population and GDP data.
`ExoDataSource`(source, source_kw)	Base class for sources of exogenous data.
`iamc_like_data_for_query`(path, query, *[, ...])	Load data from *path* in IAMC-like format and transform to `Quantity`.
`prepare_computer`(context, c[, source, ...])	Prepare *c* to compute GDP, population, or other exogenous data.
`register_source`(cls)	Register `ExoDataSource` *cls* as a source of exogenous data.

class message_ix_models.tools.exo_data.**DemoSource**(*source*, *source_kw*)

Example source of exogenous population and GDP data.

Parameters

- **source** (`str`) – **Must** be like `test s1`, where "s1" is a scenario ID from ("s0"…"s4").

- **source_kw** (dict) – **Must** contain an element "measure", one of *MEASURES*.

id: str = 'DEMO'

Identifier for this particular source.

static random_data()

Generate some random data with n, y, s, and v dimensions.

message_ix_models.tools.exo_data.**MEASURES = ('GDP', 'POP')**

Supported measures. Subclasses of *ExoDataSource* may provide support for other measures.

Todo: Store this in a separate code list or concept scheme.

---

message_ix_models.tools.exo_data.**SOURCES: Dict[str, Type[*ExoDataSource*]] = {'ADVANCE': <class 'message_ix_models.project.advance.data.ADVANCE'>, 'DEMO': <class 'message_ix_models.tools.exo_data.DemoSource'>, 'GEA': <class 'message_ix_models.project.gea.data.GEA'>, 'GFEI': <class 'message_ix_models.tools.gfei.GFEI'>, 'IEA EEI': <class 'message_ix_models.tools.iea.eei.IEA_EEI'>, 'IEA_EWEB': <class 'message_ix_models.tools.iea.web.IEA_EWEB'>, 'SHAPE': <class 'message_ix_models.project.shape.data.SHAPE'>, 'SSP': <class 'message_ix_models.project.ssp.data.SSPOriginal'>, 'SSP update': <class 'message_ix_models.project.ssp.data.SSPUpdate'>}**

Known sources for data. Use *register_source()* to add to this collection.

message_ix_models.tools.exo_data.**iamc_like_data_for_query**(*path: Path*, *query: str*, *, *archive_member: str | None = None*, *drop: List[str] | None = None*, *non_iso_3166: Literal['keep', 'discard'] = 'discard'*, *replace: dict | None = None*, *unique: str = 'MODEL SCENARIO VARIABLE UNIT'*, ***kwargs*) → AttrSeries

Load data from *path* in IAMC-like format and transform to Quantity.

The steps involved are:

1. Read the data file; use pyarrow for better performance.
2. Immediately apply *query* to reduce the data to be handled in subsequent steps.
3. Assert that Model, Scenario, Variable, and Unit are unique; store the unique values. This means that *query* **must** result in data with unique values for these dimensions.
4. Transform "Region" labels to ISO 3166-1 alpha-3 codes using *iso_3166_alpha_3()*.
5. Drop entire time series without such codes; for instance "World".
6. Transform to a pd.Series with "n" and "y" index levels; ensure the latter are int.
7. Transform to Quantity with units.

The result is *cached*.

Parameters

- **archive_member** (bool, *optional*) – If given, *path* may be an archive with 2 or more members. The member named by *archive_member* is extracted and read.

- **non_iso_3166** (bool, *optional*) – If "discard" (default), "region" labels that are not ISO 3166-1 country names are discarded, along with associated data. If "keep", such labels are kept.

Data returned by this function is cached using *cached()*; see also *SKIP_CACHE*.

message_ix_models.tools.exo_data.**register_source**(*cls: Type[*ExoDataSource*]*) → Type[*ExoDataSource*]

Register *ExoDataSource* *cls* as a source of exogenous data.

message_ix_models.tools.exo_data.**prepare_computer**(*context*, *c: Computer*, *source='test'*, *source_kw: Mapping | None = None*, ***, *strict: bool = True*) → Tuple[Key, ...]

Prepare *c* to compute GDP, population, or other exogenous data.

Check each *ExoDataSource* in *SOURCES* to determine whether it recognizes and can handle *source* and *source_kw*. If a source is identified, add tasks to *c* that retrieve and process data into a Quantity with, at least, dimensions (n, y).

Parameters

- **source** (str) – Identifier of the source, possibly with other information to be handled by a *ExoDataSource*.
- **source_kw** (dict, *optional*) – Keyword arguments for a Source class. These can include indexers, selectors, or other information needed by the source class to identify the data to be returned.

  If the key "measure" is present, it **should** be one of *MEASURES*.
- **strict** (bool, *optional*) – Raise an exception if any of the keys to be added already exist.

Return type
tuple of Key

Raises
ValueError – if no source is registered which can handle *source* and *source_kw*.

The first returned key, like `{measure}:n-y`, triggers the following computations:

1. Load data by invoking a *ExoDataSource*.
2. Aggregate on the n (node) dimension according to *Config.regions*.
3. Interpolate on the y (year) dimension according to *Config.years*.

Additional key(s) include:

- `{measure}:n-y:y0 indexed`: same as `{measure}:n-y`, indexed to values as of y_0 (the first model year).

See particular data source classes, like *SSPOriginal*, for particular examples of usage.

Todo: Extend to also prepare to compute values indexed to a particular n.

---

class message_ix_models.tools.exo_data.**ExoDataSource**(*source: str*, *source_kw: Mapping*)

Base class for sources of exogenous data.

abstract **__call__**() → AttrSeries

Return the data.

The Quantity returned by this method **must** have dimensions $(n, y) \cup$

abstract __init__(*source: str*, *source_kw: Mapping*) → None

Handle *source* and *source_kw*.

An implementation **must**:

- Raise `ValueError` if it does not recognize or cannot handle the arguments in *source* or *source_kw*.
- Recognize and handle (if possible) a "measure" keyword in *source_kw* from *MEASURES*.

It **may**:

- Transform these into other values, for instance by mapping certain values to others, applying regular expressions, or other operations.
- Store those values as instance attributes for use in *__call__()*.
- Set *name* and/or *extra_dims* to control the behaviour of *prepare_computer()*.
- Log messages that give information that may help to debug a `ValueError` for *source* or *source_kw* that cannot be handled.

It **should not** actually load data or perform any time- or memory-intensive operations; these should only be triggered by *__call__()*.

aggregate: bool = True

`True` if *transform()* should aggregate data on the n dimension.

extra_dims: Tuple[str, ...] = ()

Optional additional dimensions for the returned `Key`/`Quantity`. If not set by *__init__()*, the dimensions are (n, y).

id: str = ''

Identifier for this particular source.

interpolate: bool = True

`True` if *transform()* should interpolate data on the y dimension.

name: str = ''

Optional name for the returned `Key`/`Quantity`. If not set by *__init__()*, then the "measure" keyword is used.

raise_on_extra_kw(*kwargs*) → None

Helper for subclasses to handle the *source_kw* argument.

1. Store *aggregate* and *interpolate*, if they remain in *kwargs*.
2. Raise `ValueError` if there are any other, unhandled keyword arguments in *kwargs*.

transform(*c: Computer*, *base_key: Key*) → Key

Prepare *c* to transform raw data from *base_key*.

base_key identifies the `Quantity` that is returned by *__call__()*. Before the data is returned, *transform()* allows the data source to add additional tasks or computations to *c* that further transform the data. (These operations **may** be done in *__call__()* directly, but *transform()* allows use of other `genno` operators and conveniences.)

The default implementation:

1. If *aggregate* is `True`, aggregates the data (`genno.operator.aggregate()`) on the n dimension using the key "n::groups".
2. If *interpolate* is `True`, interpolates the data (`genno.operator.interpolate()`) on the y dimension using "y::coords".

8.8.2 ADVANCE data (`tools.advance`)

Deprecated since version 2023.11: Use *project.advance* instead.

get_advance_data([query])	Return data from the ADVANCE Work Package 2 data snapshot at *LOCATION*.
advance_data(variable[, query])	Return a single ADVANCE data *variable* as a genno.Quantity.

message_ix_models.tools.advance.**LOCATION = ('advance', 'advance_compare_20171018-134445.csv.zip')**

Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

This is a location relative to a parent directory. The specific parent directory depends on whether message_data is available:

Without message_data:
The code finds the data within *(4) Other, system-specific ("local") directories* (see discussion there for how to configure this location). Users should:

1. Visit https://tntcat.iiasa.ac.at/ADVANCEWP2DB/dsd?Action=htmlpage&page=about and register for access to the data.
2. Log in.
3. Download the snapshot with the file name given in *LOCATION* to a subdirectory advance/ within their local data directory.

With message_data:
The code finds the data within *(3) data/ directory in the message_data repository*. The snapshot is stored directly in the repository using Git LFS.

Handle data from the ADVANCE project.

message_ix_models.tools.advance.**DIMS = ['model', 'scenario', 'region', 'variable', 'unit', 'year']**

Standard dimensions for data produced as snapshots from the IIASA ENE Program "WorkDB".

message_ix_models.tools.advance.**_read_workdb_snapshot**(*path: Path*, *name: str*) → Series

Read the data file.

The expected format is a ZIP archive at *path* containing a member at *name* in CSV format, with columns corresponding to *DIMS*, except for "year", which is stored as column headers ('wide' format). (This corresponds to an older version of the "IAMC format," without more recent additions intended to represent sub-annual time resolution using a separate column.)

Deprecated since version 2023.11: Use *iamc_like_data_for_query()* instead.

Data returned by this function is cached using *cached()*; see also *SKIP_CACHE*.

message_ix_models.tools.advance.**advance_data**(*variable: str*, *query: str | None = None*) → AttrSeries

Return a single ADVANCE data *variable* as a genno.Quantity.

Deprecated since version 2023.11: Use *ADVANCE* through *exo_data.prepare_computer()* instead.

Parameters
query (str, *optional*) – Passed to *get_advance_data()*.

Returns
with the dimensions *DIMS* and name *variable*. If the units of the data for *variable* are consistent and parseable by `pint`, the returned Quantity has these units; otherwise units are discarded and the returned Quantity is dimensionless.

Return type
`genno.Quantity`

`message_ix_models.tools.advance.`**`get_advance_data`**(*query: str | None = None*) → Series

Return data from the ADVANCE Work Package 2 data snapshot at *LOCATION*.

Deprecated since version 2023.11: Use *ADVANCE* through *exo_data.prepare_computer()* instead.

Parameters
query (`str`, *optional*) – Passed to `pandas.DataFrame.query()` to limit the returned values.

Returns
with a `pandas.MultiIndex` having the levels *DIMS*.

Return type
`pandas.Series`

Data returned by this function is cached using *cached()*; see also *SKIP_CACHE*.

8.8.3 IAMC data structures (`tools.iamc`)

Tools for working with IAMC-structured data.

`message_ix_models.tools.iamc.`**`describe`**(*data: DataFrame*, *extra: str | None = None*) → StructureMessage

Generate SDMX structure information from *data* in IAMC format.

Parameters

- **data** – Data in “wide” or “long” IAMC format.
- **extra** (`str`, *optional*) – Extra text added to the description of each Codelist.

Returns
The message contains one `Codelist` for each of the MODEL, SCENARIO, REGION, VARIABLE, and UNIT dimensions. Codes for the VARIABLE code list have annotations with `id="preferred-unit-measure"` that give the corresponding UNIT Code(s) that appear with each VARIABLE.

Return type
`sdmx.message.StructureMessage`

8.8.4 World Bank structures (`tools.wb`)

Tools for World Bank data.

`message_ix_models.tools.wb.`**`assign_income_groups`**(*cl_node: sdmx.model.common.Codelist*, *cl_income_group: sdmx.model.common.Codelist*, *method: str = 'population'*, *replace: Dict[str, str] | None = None*) → None

Annotate *cl_node* with income groups.

Each node is assigned an `Annotation` with `id="wb-income-group"`, according to the income groups of its children (countries), as reflected in *cl_income_group* (see *get_income_group_codelist()*).

Parameters

- **method** ("population" or "count") – Method for aggregation:
  - "population" (default): the WB World Development Indicators (WDI) 2020 population for each country is used as a weight, so that the node's income group is the income group of the plurality of the population of its children.
  - "count": each country is weighted equally, so that the node's income group is the mode (most frequently occurring value) of its childrens'.
- **replace** (dict) – Mapping from wb-income-group annotation text appearing in *cl_income_group* to texts to be attached to *cl_node*. Mapping two keys to the same value effectively combines or aggregates those groups. See *make_map()*.

Example

Annotate the R12 node list with income group information, mapping high income countries (HIC) and upper-middle income countries (UMC) into one group and aggregating by population.

```
>>> cl_node = get_codelist(f"node/R12")
>>> cl_ig = get_income_group_codelist()
>>> replace = make_map({"HIC": "HMIC", "UMC": "HMIC"})
>>> assign_income_groups(cl_node, cl_ig, replace=replace)
>>> cl_node["R12_NAM"].get_annotation(id="wb-income-group").text
HMIC
```

message_ix_models.tools.wb.**fetch_codelist**(*id: str*) → sdmx.model.common.Codelist

Retrieve code lists related to the WB World Development Indicators.

In principle this could be done with `sdmx.Client("WB_WDI").codelist(id)`, but the World Bank SDMX REST API does not support queries for a specific code list. See https://datahelpdesk.worldbank.org/knowledgebase/articles/1886701-sdmx-api-queries.

fetch_codelist() retrieves http://api.worldbank.org/v2/sdmx/rest/codelist/WB/, the structure message containing *all* code lists; and extracts and returns the one with the given *id*.

message_ix_models.tools.wb.**get_income_group_codelist**() → sdmx.model.common.Codelist

Return a `Codelist` with World Bank income group information.

The returned code list is a modified version of the one with URN `…Codelist=WB:CL_REF_AREA_WDI(1.0)`, via *fetch_codelist()*.

This is augmented with information about the income group and lending category concepts as described at https://datahelpdesk.worldbank.org/knowledgebase/articles/906519

The information is stored two ways:

- Existing codes in the list like "HIC: High income" that designate groups of countries are associated with child codes that are designated as members of that country. These can be accessed at `Code.child`.
- Existing codes in the list like "ABW: Aruba" are annotated with:
  - `id="wb-income-group"`: the URN of the income group code, for instance "urn:sdmx:org.sdmx.infomodel.codelist.Code=WB:CL_REF_AREA_WDI(1.0).HIC". This is an unambiguous reference to a code in the same list.
  - `id="wb-lending-category"`: the name of the lending category, if any.

  These can be accessed using `Code.annotations`, `Code.get_annotation`, and other methods.

message_ix_models.tools.wb.**make_map**(*source: Dict[str, str]*, *expand_key_urn: bool = True*, *expand_value_urn: bool = False*) → Dict[str, str]

Prepare the `replace` parameter of *assign_income_groups()*.

The result has one (*key*, *value*) for each in *source*.

Parameters

- **expand_key_urn** (bool) – If True (the default), replace each *key* from *source* with the URN for the code in CL_REF_AREA_WDI with id=key.
- **expand_value_urn** (bool) – If True, replace each *value* from *source* with the URN for the code in CL_REF_AREA_WDI with id=value.

8.9 Investment and fixed costs (tools.costs)

tools.costs implements methods to **project investment and fixed costs of technologies**[1] in MESSAGEix-GLOBIOM family models.

8.9.1 Methods

The tool creates distinct projected cost values for different regions, technologies, and scenarios. The costs are projected based on historical (mostly a base year) data and assumptions about future cost reductions.

The projections use the concept of a **reference region**[2] and apply distinct methods to the reference and non-reference regions:

Reference region
Costs in the reference region are projected based on the following assumption: given a cost reduction rate, the cost of the technology in the reference region experiences an exponential decay over time.

Non-reference regions
Costs for each technology in all non-reference regions may be calculated using one of three methods, specified using *Config.method*:

1. Constant cost reduction rate (*Config.method* = "constant"): the regional cost ratio (versus the reference region) that is calculated in the base year is held constant and used to project regionally-differentiated costs across all years.
2. Convergence to reference region costs by a certain year (*Config.method* = "convergence"): all other regions' costs exponentially decay until they become they same as the reference region's cost by a specified year.
3. GDP-adjusted cost reduction rate (*Config.method* = "gdp"): this method assumes that regional costs converge not based on a specified year but based on GDP per capita. All non-reference regions' costs are

[1] Fixed costs are also referred to as "operation and maintenance (O&M)" or "fixed O&M" costs. Investment and fixed costs are also collectively referred to as "techno-economic costs" or "techno-economic parameters".
[2] In message_ix, these are elements of the node set. The term 'region' is used in this documentation to mean the same thing.

adjusted based on the ratio of the GDP per capita of the region to the GDP per capita of the reference region.

8.9.2 Modules and model variants

Within the context of the tool, the term **module** (specified by `Config.module`) is used to mean input data for particular *sets of technologies*. These correspond to subsets of all the technologies in MESSAGEix-GLOBIOM models—either the base model or model variants.[3] Currently, `tools.costs` supports two module `module` settings:

"energy"
Mostly electric power technologies, as well as a few other supply-side technologies.

This can be considered the "base" module, corresponding to the "base" version of MESSAGEix-GLOBIOM, as it contains the most technologies.

"materials"
Technologies conceived as part of the materials and industry sectors.

Data and files for a particular module can refer to other modules. This allows for values or settings for "materials" and other technologies to be assumed to match the values and settings used for the referenced "energy"-module technologies.

To add a new module, the following steps are required:

- In `message_ix_models/data/costs/`, create another subdirectory with the name of the new module, for instance `message_ix_models/data/costs/[module]/`.
- Add the following files to the new directory:

  `first_year_[module].csv`
  A file with a list of technologies and the corresponding first year that the respective technology can start being deployed/modeled. The file should have the following columns:

  - "message_technology": the technology name.
  - "first_year_original": the first year the technology can start being deployed.

  `tech_map_[module].csv`
  A file with the mapping of technologies to a source of base year cost data. The file should have the following columns:

  - "message_technology": the technology name.
  - "reg_diff_source" and "reg_diff_technology": the source data for the regional differentiation of costs and the corresponding technology to map to.
    - If "reg_diff_source" is "energy", then "reg_diff_technology" should be a technology that is present in the "energy" module.
    - If "reg_diff_source" is "weo", then "reg_diff_technology" should be a technology that is present in the WEO data (refer to `tech_map_energy.csv` for the names of WEO technologies available, as all energy technologies are mapped to a WEO technology).
    - You can also add another source of regional differentiation (in the case of `module="materials"`, a newly created source called "intratec" is used). However, this method is a little more involved as it requires extending the code to read in new source data.
  - "base_year_reference_region_cost": the base year cost for the technology in the reference region.
  - "fix_ratio": the ratio of fixed O&M costs to investment costs for the technology.

- Add the new module to the allowed values of `Config.module`.

Please note that the following assumptions are made in technology costs mapping:

[3] This usage of "module" differs from the meaning of a "Python module". For instance, `message_ix_models.model.water` is a *Python module* for MESSAGEix-Nexus. If the setting `.costs.Config.module = "water"` were added, this *might* refer to input data for projecting investment and fixed costs of water technologies that are defined in `message_ix_models.model.water`—but not necessarily.

- If a technology is mapped to a technology in the "energy" module, then the cost reduction across scenarios is the same as the cost reduction of the mapped technology.
- If a "materials" (or any other non-"energy") technology has `reg_diff_source="energy"` and the "base_year_reference_region_cost" is not empty, then the "base_year_reference_region_cost" in `tech_map_[module].csv` is used as the base year cost for the technology in the reference region. If the "base_year_reference_region_cost" is empty, then the cost reduction across scenarios is the same as the cost reduction of the mapped technology.
- If using the "materials" module, if a technology that is specified in `tech_map_materials.csv` already exists in `tech_map_energy.csv`, then the reference region cost is taken from `tech_map_materials.csv`.
- If a technology in a module is not mapped to any source of regional differentiation, then no cost reduction over the years is applied to the technology.
- If a technology has a non-empty "base_year_reference_region_cost" but is not mapped to any source of regional differentiation, then assume no regional differentiation and use the reference region base year cost as the base year cost for all regions.

8.9.3 Data sources

The tool uses the following data sources for the regional differentiation of costs:

- WEO: the World Energy Outlook data from the International Energy Agency (IEA).
- Intratec: the Intratec data, which is a database of production costs for chemicals and other materials.

The tool also uses *ssp.data* (via *exo_data.prepare_computer()*) to adjust the costs of technologies based on GDP per capita.

8.9.4 Usage

create_cost_projections() is the top-level entry point.

This function takes a single *costs.Config* object as an argument. The object carries all the settings understood by *create_cost_projections()* and other functions. Those settings include the following; click each for the full description, allowable values, and defaults:

> *module*, *method*, *node*, *ref_region*, *scenario*, *scenario_version*, *base_year*, *convergence_year*, *fom_rate*, and *format*.

create_cost_projections() in turn calls the other functions in the module in the correct order, and returns a Python `dict` with the following keys mapped to `pandas.DataFrame`.

- "inv_cost": the investment costs of the technologies in each region.
- "fix_cost": the fixed O&M costs of the technologies in each region.

To use the tool with the default settings, simply create a *Config* object and pass it as an argument to *create_cost_projections()*:

```
from message_ix_models.tools.costs import Config, create_cost_projections

# Use default settings
cfg = Config()

# Compute cost projections
costs = create_cost_projections(cfg)

# Show the resulting data
costs["inv_cost"]
costs["fix_cost"]
```

These data can be further manipulated; for instance, added to a scenario using *add_par_data()*. See the file message_ix_models/tools/costs/demo.py for multiple examples using various non-default settings to control the methods and data used by *create_cost_projections()*.

8.9.5 Code reference

The top-level function and configuration class:

Config(base_year, convergence_year, ...)	Configuration for *costs*.
create_cost_projections(config)	Get investment and fixed cost projections.

The other submodules implement the supporting methods, calculations, and data handling, in roughly the following order:

1. *regional_differentiation* calculates the regional differentiation of costs for technologies.
2. *decay* projects the costs of technologies in a reference region with only a cost reduction rate applied.
3. *gdp* adjusts the regional differentiation of costs for technologies based on the GDP per capita of the region.
4. *projections* combines all the above steps and returns the projected costs for each technology in each region.

class message_ix_models.tools.costs.**Config**(*base_year: int = 2021*, *convergence_year: int = 2050*, *final_year: int = 2100*, *fom_rate: float = 0.025*, *format: ~typing.Literal['iamc', 'message'] = 'message'*, *node: ~typing.Literal['R11', 'R12', 'R20'] = 'R12'*, *method: ~typing.Literal['constant', 'convergence', 'gdp'] = 'gdp'*, *module: ~typing.Literal['energy', 'materials'] = 'energy'*, *pre_last_year_rate: float = 0.01*, *ref_region: str | None = None*, *scenario_version: ~typing.Literal['original', 'updated', 'all'] = 'updated'*, *scenario: ~typing.Literal['all', 'LED', 'SSP1', 'SSP2', 'SSP3', 'SSP4', 'SSP5'] = 'all'*, *_info: ~message_ix_models.util.scenarioinfo.ScenarioInfo = <factory>*)

Configuration for *costs*.

On creation:

- If not given, *ref_region* is set based on *node* using, for instance, ref_region="R12_NAM" for node="R12".

property Y: List[int]

List of model periods.

base_year: int = 2021

Base year for projected costs.

check()

Validate settings.

convergence_year: int = 2050

Year of convergence; used when *method* is "convergence". This is the year by which costs in all regions should converge to the reference region's costs. See *create_projections_converge()*.

final_year: int = 2100

Final year for projections. Note that the default is different from the final model year of 2110 commonly used in MESSAGEix-GLOBIOM (*Years or time periods (year/*.yaml)*).

fom_rate: float = 0.025

Rate of exponential growth (positive values) or decrease of fixed operating and maintenance costs over time. The default of 0.025 implies exponential growth at a rate of 2.5% per year; or `(1 + 0.025) ** N` for a period of length N.

format: Literal['iamc', 'message'] = 'message'

Format of output from *create_cost_projections()*. One of:

- "iamc": IAMC time series data structure.
- "message": `message_ix` parameter data.

method: Literal['constant', 'convergence', 'gdp'] = 'gdp'

Method for projecting costs in non-reference regions. One of:

- "constant": uses *create_projections_constant()*.
- "convergence": uses *create_projections_converge()*.
- "gdp": uses *create_projections_gdp()*.

module: Literal['energy', 'materials'] = 'energy'

Model variant for which to project costs.

node: Literal['R11', 'R12', 'R20'] = 'R12'

Node code list / spatial resolution for which to project costs. This should correspond to the target scenario to which data is to be added.

pre_last_year_rate: float = 0.01

Todo: Document the meaning of this setting.

---

ref_region: str | None = None

Reference region. If not given, "`{node}_NAM"` for a given *node*. This default **must** be overridden if there is no such node.

scenario: Literal['all', 'LED', 'SSP1', 'SSP2', 'SSP3', 'SSP4', 'SSP5'] = 'all'

Scenario(s) for which to project costs. "all" implies the set of all the other values, meaning that costs are projected for all scenarios.

scenario_version: Literal['original', 'updated', 'all'] = 'updated'

Set of SSPs referenced by *scenario*. One of:

- "original": `SSP_2017`
- "updated": `SSP_2024`
- "all": both of the above.

property seq_years: List[int]

Similar to *Y*.

This list of periods differs in that it:

1. Excludes periods after *final_year*.
2. Includes 5-year periods even when these are not in *Y*.

property y0: int

The first model period.

message_ix_models.tools.costs.**create_cost_projections**(*config:* Config) → Mapping[str, DataFrame]

Get investment and fixed cost projections.

This is the main function to get investment and fixed cost projections. It calls the other functions in this module, and returns the projections in the specified format.

Parameters

config (*Config*) – The function responds to, or passes on to other functions, the fields: *base_year*, *convergence_year*, *fom_rate*, *format*, *method*, *module*, *node*, *ref_region*, *scenario*, and *scenario_version*.

Returns

Keys are "fix_cost" and "inv_cost", each mapped to a DataFrame.

If *Config.format* is "message", the data frames have the same columns as required by message_ix for the respective parameter—for instance, the columns given by make_df("fix_cost", ...)—*plus* columns named "scenario" and "scenario_version".

Return type

dict

Cost reduction of technologies over time (decay)

get_cost_reduction_data(module)	Get cost reduction data from file.
get_technology_reduction_scenarios_data(...)	Read in technology first year and cost reduction scenarios.
project_ref_region_inv_costs_using_reduction_rates(...)	Project investment costs for the reference region using cost reduction rates.

message_ix_models.tools.costs.decay.**get_cost_reduction_data**(*module*) → DataFrame

Get cost reduction data from file.

Raw data on cost reduction in 2100 for technologies are read from data/[module]/cost_reduction_[module].csv, based on GEA data.

Parameters

module (str) – Model module

Returns

DataFrame with columns:

- message_technology: name of technology in MESSAGEix
- reduction_rate: the cost reduction rate (either very_low, low, medium, high,

or very_high) - cost_reduction: cost reduction in 2100 (%)

Return type

pandas.DataFrame

message_ix_models.tools.costs.decay.**get_technology_reduction_scenarios_data**(*base_year: int*, *module: str*) → DataFrame

Read in technology first year and cost reduction scenarios.

Raw data on technology first year and reduction scenarios are read from `data/costs/[module]/first_year_[module]`. The first year the technology is available in MESSAGEix is adjusted to be the base year if the original first year is before the base year.

Raw data on cost reduction scenarios are read from `data/costs/[module]/scenarios_reduction_[module].csv`.

Assumptions are made for the materials module for technologies' cost reduction scenarios that are not given.

Parameters

- **base_year** (int, *optional*) – The base year, by default set to global BASE_YEAR
- **module** (str) – Model module

Returns

DataFrame with columns:

- message_technology: name of technology in MESSAGEix
- scenario: scenario (SSP1, SSP2, SSP3, SSP4, SSP5, or LED)
- first_technology_year: first year the technology is available in MESSAGEix.
- reduction_rate: the cost reduction rate (either very_low, low, medium, high,

or very_high)

Return type

pandas.DataFrame

message_ix_models.tools.costs.decay.**project_ref_region_inv_costs_using_reduction_rates**(*r*
g
I
c
f
C
f
-
I

Project investment costs for the reference region using cost reduction rates.

This function uses the cost reduction rates for each technology under each scenario to project the capital costs for each technology in the reference region.

The returned data have the list of periods given by *Config.seq_years*.

Parameters

- **regional_diff_df** (pandas.DataFrame) – Dataframe output from `get_weo_region_differentiated_costs()`
- **config** (*Config*) – The function responds to, or passes on to other functions, the fields: *base_year*, *module*, and *ref_region*.

Returns

DataFrame with columns:

- message_technology: name of technology in MESSAGEix
- scenario: scenario (SSP1, SSP2, SSP3, SSP4, SSP5, or LED)
- reference_region: reference region
- first_technology_year: first year the technology is available in MESSAGEix.

- year: year
- inv_cost_ref_region_decay: investment cost in reference region in year.

Return type
pandas.DataFrame

GDP-adjusted costs and regional differentiation (gdp)

process_raw_ssp_data(context, config)	Retrieve SSP data as required for *tools.costs*.
adjust_cost_ratios_with_gdp(region_diff_df, ...)	Calculate adjusted region-differentiated cost ratios.

message_ix_models.tools.costs.gdp.**adjust_cost_ratios_with_gdp**(*region_diff_df*, *config:* Config)

Calculate adjusted region-differentiated cost ratios.

This function takes in a data frame with region-differentiated cost ratios and calculates adjusted region-differentiated cost ratios using GDP per capita data.

Parameters

- **region_diff_df** (pandas.DataFrame) – Output of `apply_regional_differentiation()`.
- **config** (*Config*) – The function responds to, or passes on to other functions, the fields: *base_year*, *node*, *ref_region*, *scenario*, and *scenario_version*.

Returns

DataFrame with columns:

- scenario_version: scenario version
- scenario: SSP scenario
- message_technology: message technology
- region: R11, R12, or R20 region
- year
- gdp_ratio_reg_to_reference: ratio of GDP per capita in respective region to GDP per capita in reference region.
- reg_cost_ratio_adj: adjusted region-differentiated cost ratio

Return type
pandas.DataFrame

message_ix_models.tools.costs.gdp.**process_raw_ssp_data**(*context:* Context, *config:* Config) → DataFrame

Retrieve SSP data as required for *tools.costs*.

This method uses *SSPOriginal* and *SSPUpdate* via *exo_data.prepare_computer()*

Returns

with the columns:

- scenario_version
- scenario
- region
- year

- total_gdp
- total_population
- gdp_ppp_per_capita
- gdp_ratio_reg_to_reference

Return type
pandas.DataFrame

Projection of costs given input parameters (`projections`)

create_projections_constant(config)	Create cost projections using assuming constant regional cost ratios.
create_projections_gdp(config)	Create cost projections using the GDP method.
create_projections_converge(config)	Create cost projections using the convergence method.
create_message_outputs(df_projections, config)	Create MESSAGEix outputs for investment and fixed costs.
create_iamc_outputs(msg_inv, msg_fix)	Create IAMC outputs for investment and fixed costs.

message_ix_models.tools.costs.projections.**create_cost_projections**(*config:* Config) → Mapping[str, DataFrame]

Get investment and fixed cost projections.

This is the main function to get investment and fixed cost projections. It calls the other functions in this module, and returns the projections in the specified format.

Parameters
config (*Config*) – The function responds to, or passes on to other functions, the fields: *base_year*, *convergence_year*, *fom_rate*, *format*, *method*, *module*, *node*, *ref_region*, *scenario*, and *scenario_version*.

Returns

Keys are "fix_cost" and "inv_cost", each mapped to a DataFrame.

If *Config.format* is "message", the data frames have the same columns as required by message_ix for the respective parameter—for instance, the columns given by make_df("fix_cost", ...)—*plus* columns named "scenario" and "scenario_version".

Return type
dict

message_ix_models.tools.costs.projections.**create_iamc_outputs**(*msg_inv:* *DataFrame*, *msg_fix:* *DataFrame*) → Tuple[DataFrame, DataFrame]

Create IAMC outputs for investment and fixed costs.

Parameters

- **msg_inv** (pd.DataFrame) – Dataframe containing investment costs in MESSAGEix format. Output of func:*create_message_outputs*.
- **msg_fix** (pd.DataFrame) – Dataframe containing fixed operating and maintenance costs in MESSAGEix format. Output of func:*create_message_outputs*.

Returns

- **iamc_inv** (pd.DataFrame) – Dataframe containing investment costs in IAMC format.
- **iamc_fix** (pd.DataFrame) – Dataframe containing fixed operating and maintenance costs in IAMC format.

message_ix_models.tools.costs.projections.**create_message_outputs**(*df_projections: DataFrame*, *config:* Config) → Tuple[DataFrame, DataFrame]

Create MESSAGEix outputs for investment and fixed costs.

The returned data have the model periods given by *Config.Y*.

Parameters

- **df_projections** (pd.DataFrame) – Dataframe containing the cost projections for each technology. Output of func:*create_cost_projections*.
- **config** (*Config*) – The function responds to the fields *fom_rate* and *Y*.

Returns

- **inv** (pd.DataFrame) – Dataframe containing investment costs.
- **fom** (pd.DataFrame) – Dataframe containing fixed operating and maintenance costs.

message_ix_models.tools.costs.projections.**create_projections_constant**(*config:* Config)

Create cost projections using assuming constant regional cost ratios.

Parameters
config (*Config*) – The function responds to, or passes on to other functions, the fields: *base_year*, *module*, *node*, *ref_region*, and *scenario*.

Returns

df_costs – Dataframe containing the cost projections with the columns: - scenario_version: scenario version (for constant method, only

"Not applicable")

- scenario: scenario name (SSP1, SSP2, SSP3, SSP4, SSP5, or LED)
- message_technology: technology name
- region: region name
- year: year
- inv_cost: investment cost
- fix_cost: fixed operating and maintenance cost

Return type
pd.DataFrame

message_ix_models.tools.costs.projections.**create_projections_converge**(*config:* Config)

Create cost projections using the convergence method.

Parameters
config (*Config*) – The function responds to, or passes on to other functions, the fields: *base_year*, *convergence_year*, *module*, *node*, *ref_region*, and *scenario*.

Returns

df_costs – Dataframe containing the cost projections with the columns: - scenario_version: scenario version (for convergence method, only "Not

applicable")

- scenario: scenario name (SSP1, SSP2, SSP3, SSP4, SSP5, or LED)
- message_technology: technology name
- region: region name
- year: year
- inv_cost: investment cost
- fix_cost: fixed operating and maintenance cost

Return type
`pd.DataFrame`

`message_ix_models.tools.costs.projections.`**`create_projections_gdp`**(*config:* Config)

Create cost projections using the GDP method.

Parameters
`config` (*`Config`*) – The function responds to, or passes on to other functions, the fields: *`base_year`*, *`module`*, *`node`*, *`ref_region`*, *`scenario`*, and *`scenario_version`*.

Returns

df_costs – Dataframe containing the cost projections with the columns: - scenario_version: scenario version (for gdp method, either "Review (2023)" or

"Previous (2013)"

- scenario: scenario name (SSP1, SSP2, SSP3, SSP4, SSP5, or LED)
- message_technology: technology name
- region: region name
- year: year
- inv_cost: investment cost
- fix_cost: fixed operating and maintenance cost

Return type
`pd.DataFrame`

Regional differentiation of costs (`regional_differentiation`)

get_weo_data()	Read in raw WEO investment/capital costs and O&M costs data.
get_intratec_data()	Read in raw Intratec data.
get_raw_technology_mapping(module)	Retrieve a technology mapping for *module*.
subset_materials_map(raw_map)	Subset materials mapping for only technologies that have sufficient data.
adjust_technology_mapping(module)	Adjust technology mapping based on sources and assumptions.
get_weo_regional_differentiation(config)	Apply WEO regional differentiation.
get_intratec_regional_differentiation(node, ...)	Apply Intratec regional differentiation.
apply_regional_differentiation(config)	Apply regional differentiation depending on mapping source.

message_ix_models.tools.costs.regional_differentiation.**adjust_technology_mapping**(*module: Literal['energy', 'materials']*) → DataFram

Adjust technology mapping based on sources and assumptions.

Parameters
module (str) – See *Config.module*.

Returns

DataFrame with columns:

- message_technology: MESSAGEix technology name.
- reg_diff_source: data source to map MESSAGEix technology to (e.g., WEO, Intratec).
- reg_diff_technology: technology name in the data source.
- base_year_reference_region_cost: manually specified base year cost of the technology in the reference region (in 2005 USD).

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**apply_regional_differentiation**(*co fig* Co fig → Da

Apply regional differentiation depending on mapping source.

1. Retrieve an adjusted technology mapping from *adjust_technology_mapping()*.
2. Based on the value in the `reg_diff_source` column:
   - "energy" or "weo": use WEO data via *get_weo_regional_differentiation()*.

- "intratec": use Intratec data via *get_intratec_regional_differentiation()*.
- "none": assume no regional differentiation; use the *ref_region* cost as the cost for all regions.

Parameters
config (*Config*) – The function responds to, or passes on to other functions, the fields: *module*, *node*, and *ref_region*.

Returns

DataFrame with columns:

- message_technology: MESSAGEix technology name
- reg_diff_source: data source to map MESSAGEix technology to (e.g., WEO, Intratec)
- reg_diff_technology: technology name in the data source
- region: MESSAGEix region
- base_year_reference_region_cost: manually specified base year cost of the technology in the reference region (in 2005 USD)
- reg_cost_ratio: regional cost ratio relative to reference region
- fix_ratio: ratio of fixed O&M costs to investment costs

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**get_intratec_data**() → DataFrame

Read in raw Intratec data.

Returns

DataFrame with columns:

- node: Intratec region
- value: Intratec index value

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**get_intratec_regional_differentia**

Apply Intratec regional differentiation.

1. Retrieve Intratec data using *get_intratec_data()*.
2. Map data to MESSAGEix-GLOBIOM regions according to the *Config.node*.
3. Calculate cost ratios for each region relative to the *ref_region*.

Parameters

- **node** (str) – See :attr`.Config.node`.
- **ref_region** (str) – See :attr`.Config.ref_region`.

Returns

DataFrame with columns:

- message_technology: MESSAGEix technology name
- region: MESSAGEix region
- intratec_ref_region_cost: Intratec cost in reference region
- reg_cost_ratio: regional cost ratio relative to reference region

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**get_raw_technology_mapping**(*module: Literal['energy', 'materials']*) → DataFra

Retrieve a technology mapping for *module*.

The data are read from a CSV file at `data/module/tech_map_module.csv`. The file must have the following columns:

- `message_technology`: MESSAGEix-GLOBIOM technology code
- `reg_diff_source`: data source to map MESSAGEix technology to. A string like "weo", "energy", or possibly others.
- `reg_diff_technology`: Technology code in the source data.
- `base_year_reference_region_cost`: manually specified base year cost of the technology in the reference region (in 2005 USD).
- `fix_ratio`: ???

Parameters
module (str) – See *Config.module*.

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**get_weo_data**() → DataFrame

Read in raw WEO investment/capital costs and O&M costs data.

Returns

DataFrame with columns:

- cost_type: investment or fixed O&M cost
- weo_technology: WEO technology name
- weo_region: WEO region
- year: year
- value: cost value

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**get_weo_region_map**(*regions: str*) → Mapping[str, str]

Return a mapping from MESSAGE node IDs to WEO region names.

The mapping is constructed from the `iea-weo-region` annotations on the *Node code lists*.

message_ix_models.tools.costs.regional_differentiation.**get_weo_regional_differentiation**

Apply WEO regional differentiation.

1. Retrieve WEO data using *get_weo_data()*.
2. Map data to MESSAGEix-GLOBIOM regions according to the *Config.node*.
3. Calculate cost ratios for each region relative to the *ref_region*.

Parameters
config (*Config*) – The function responds to the fields: *base_year*, *node*, and *ref_region*.

Returns

DataFrame with columns:

- message_technology: MESSAGEix technology name
- region: MESSAGEix region
- weo_ref_region_cost: WEO cost in reference region
- reg_cost_ratio: regional cost ratio relative to reference region

Return type
pandas.DataFrame

message_ix_models.tools.costs.regional_differentiation.**subset_materials_map**(*raw_map*)

Subset materials mapping for only technologies that have sufficient data.

Parameters
raw_map (pandas.DataFrame) – Output of *get_raw_technology_mapping()*

Returns

DataFrame with columns:

- message_technology: MESSAGEix technology name
- reg_diff_source: data source to map MESSAGEix technology to (e.g., WEO)
- reg_diff_technology: technology name in the data source
- base_year_reference_region_cost: manually specified base year cost of the technology in the reference region (in 2005 USD)

Return type
pandas.DataFrame

8.10 Tools for specific data sources

8.10.1 Global Fuel Economy Initiative (GFEI) (`tools.gfei`)

Handle data from the Global Fuel Economy Initiative (GFEI).

class message_ix_models.tools.gfei.**GFEI**(*source*, *source_kw*)

Provider of exogenous data from the GFEI 2017 data source.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: "GFEI".
- *source_kw* including:
  - *plot* (optional, default `False`): add a task with the key "plot GFEI debug" to generate diagnostic plot using *Plot*.
  - *aggregate*, *interpolate*: see *ExoDataSource.transform()*.

The source data:

- is derived from https://theicct.org/publications/gfei-tech-policy-drivers-2005-2017, specifically the data underlying "Figure 37. Fuel consumption range by type of powertrain and vehicle size, 2017".
- has resolution of individual countries.
- corresponds to new vehicle registrations in 2017.
- has units of megajoule / kilometre, converted from original litres of gasoline equivalent per 100 km.

Note: if py:*source_kw["aggregate"] is True*, the aggregation performed is an unweighted `sum()`. To produce meaningful values for multi-country regions, instead perform perform a weighted mean using appropriate weights; for instance the vehicle activity for each country. The class currently **does not** do this automatically.

---

aggregate: bool = False

By default, do not aggregate.

id: str = 'GFEI'

Identifier for this particular source.

interpolate: bool = False

By default, do not interpolate.

transform(*c: Computer*, *base_key: Key*) → Key

Prepare *c* to transform raw data from *base_key*.

class message_ix_models.tools.gfei.**Plot**

Diagnostic plot of processed data.

basename = 'GFEI-fuel-economy-t'

File name base for saving the plot.

generate(*data*)

Generate and return the plot.

A subclass of Plot **must** implement this method.

Parameters

args (`Sequence` of `pandas.DataFrame` or `other`) – One argument is given corresponding to each of the `inputs`.

Because `plotnine` operates on pandas data structures, `save()` automatically converts any `Quantity` inputs to `pandas.DataFrame` before they are passed to *generate()*.

8.10.2 International Energy Agency (IEA) (`tools.iea`)

The IEA publishes many kinds of data. Each distinct data source is handled by a separate submodule of *message_ix_models.tools.iea*.

Documentation for all module contents:

iea	Tools for working with IEA data and structures.

message_ix_models.tools.iea

Tools for working with IEA data and structures.

Modules

message_ix_models.tools.iea.eei	Handle data from the IEA Energy Efficiency Indicators (EEI).
message_ix_models.tools.iea.web	Tools for IEA (Extended) World Energy Balance (WEB) data.

message_ix_models.tools.iea.eei

Handle data from the IEA Energy Efficiency Indicators (EEI).

Module Attributes

WAVG_MAP	Mapping of weights to variables used as weights for weighted averaging.

message_ix_models.tools.iea.eei.WAVG_MAP

message_ix_models.tools.iea.eei.**WAVG_MAP = {'Freight load factor': 'vehicle-kilometres', 'Fuel intensity': 'vehicle-kilometres', 'Passenger load factor': 'vehicle-kilometres', 'Vehicle use': 'vehicle stock', 'Vehicle-kilometres energy intensity': 'vehicle-kilometres'}**

Mapping of weights to variables used as weights for weighted averaging.

Todo: Replace with tests showing usage of *wavg()*.

Functions

extract_measure_and_units(df)	
iea_eei_data_raw(path[, non_iso_3166])	
melt(df)	Melt on any dimensions.
wavg(measure, df, weight_data)	Perform masked & weighted average for *measure* in *df*, using *weight_data*.

message_ix_models.tools.iea.eei.extract_measure_and_units

message_ix_models.tools.iea.eei.**extract_measure_and_units**(*df: DataFrame*) → DataFrame

message_ix_models.tools.iea.eei.iea_eei_data_raw

message_ix_models.tools.iea.eei.**iea_eei_data_raw**(*path*, *non_iso_3166: Literal['keep', 'discard'] = 'discard'*)

message_ix_models.tools.iea.eei.melt

message_ix_models.tools.iea.eei.**melt**(*df: DataFrame*) → DataFrame

Melt on any dimensions.

message_ix_models.tools.iea.eei.wavg

message_ix_models.tools.iea.eei.**wavg**(*measure: str*, *df: DataFrame*, *weight_data: DataFrame*) → DataFrame

Perform masked & weighted average for *measure* in *df*, using *weight_data*.

Todo: Replace this with usage of genno; add tests.

WAVG_MAP is used to select a data from *weight_data* appropriate for weighting *measure*: either “population”, “vehicle stock” or “vehicle-kilometres*. If the measure to be used for weights is all NaNs, then “population” is used as a fallback as weight.

The weighted average is performed by grouping *df* on the “region”, “year”, and “Mode/vehicle type” dimensions, i.e. the values returned are averages weighted within these groups.

Parameters

- **measure** (str) – Name of measure contained in *df*.
- **df** (pandas.DataFrame) – Data to be aggregated.
- **weight_data** (pandas.DataFrame.) – Data source for weights.

Return type

pandas.DataFrame

Classes

IEA_EEI(source, source_kw)	Provider of exogenous data from the IEA Energy Efficiency Indicators data source.
Plot()	Diagnostic plot of processed data.

message_ix_models.tools.iea.eei.IEA_EEI

class message_ix_models.tools.iea.eei.**IEA_EEI**(*source*, *source_kw*)

Bases: *ExoDataSource*

Provider of exogenous data from the IEA Energy Efficiency Indicators data source.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: "IEA_EEI".
- *source_kw* including:
  - *measure*: name of a measure or indicator in the data.
  - *broadcast_map* (optional): name of a `Key` containing a mapping for `genno.operator.broadcast_map()`.
  - *plot* (optional, default `False`): add a task with the key "plot IEA_EEI debug" to generate diagnostic plot using *Plot*.
  - *aggregate*, *interpolate*: see *ExoDataSource.transform()*.

__init__(*source*, *source_kw*)

Handle *source* and *source_kw*.

An implementation **must**:

- Raise `ValueError` if it does not recognize or cannot handle the arguments in *source* or *source_kw*.
- Recognize and handle (if possible) a "measure" keyword in *source_kw* from `MEASURES`.

It **may**:

- Transform these into other values, for instance by mapping certain values to others, applying regular expressions, or other operations.
- Store those values as instance attributes for use in `__call__()`.
- Set *name* and/or *extra_dims* to control the behaviour of *prepare_computer()*.
- Log messages that give information that may help to debug a `ValueError` for *source* or *source_kw* that cannot be handled.

It **should not** actually load data or perform any time- or memory-intensive operations; these should only be triggered by *__call__()*.

Methods

`__init__`(source, source_kw)	Handle *source* and *source_kw*.
`raise_on_extra_kw`(kwargs)	Helper for subclasses to handle the *source_kw* argument.
`transform`(c, base_key)	Prepare *c* to transform raw data from *base_key*.

Attributes

`aggregate`	By default, do not aggregate.
`extra_dims`	Optional additional dimensions for the returned `Key`/`Quantity`.
`id`	Identifier for this particular source.
`interpolate`	By default, do not interpolate.
`name`	Optional name for the returned `Key`/`Quantity`.

aggregate: bool = False

By default, do not aggregate.

extra_dims: Tuple[str, ...] = ()

Optional additional dimensions for the returned `Key`/`Quantity`. If not set by `__init__()`, the dimensions are (n, y).

id: str = 'IEA EEI'

Identifier for this particular source.

interpolate: bool = False

By default, do not interpolate.

name: str = ''

Optional name for the returned `Key`/`Quantity`. If not set by `__init__()`, then the "measure" keyword is used.

raise_on_extra_kw(*kwargs*) → None

Helper for subclasses to handle the *source_kw* argument.

1. Store `aggregate` and `interpolate`, if they remain in *kwargs*.
2. Raise `ValueError` if there are any other, unhandled keyword arguments in *kwargs*.

transform(*c: Computer*, *base_key: Key*) → Key

Prepare *c* to transform raw data from *base_key*.

base_key identifies the `Quantity` that is returned by `__call__()`. Before the data is returned, `transform()` allows the data source to add additional tasks or computations to *c* that further transform the data. (These operations **may** be done in `__call__()` directly, but `transform()` allows use of other `genno` operators and conveniences.)

The default implementation:

1. If `aggregate` is `True`, aggregates the data (`genno.operator.aggregate()`) on the n dimension using the key "n::groups".
2. If `interpolate` is `True`, interpolates the data (`genno.operator.interpolate()`) on the y dimension using "y::coords".

message_ix_models.tools.iea.eei.Plot

class message_ix_models.tools.iea.eei.**Plot**

Bases: Plot

Diagnostic plot of processed data.

__init__()

Methods

__init__()	
add_tasks(c, key, *inputs[, strict])	Add a task to *c* to generate and save the Plot.
generate(data)	Generate and return the plot.
make_task(*inputs)	Return a task `tuple` to add to a Computer.
save(config, *args, **kwargs)	Prepare data, call *generate()*, and save to file.

Attributes

basename	File name base for saving the plot.
inputs	`Keys` referring to `Quantities` or other inputs accepted by *generate()*.
path	Path for file output.
save_args	Keyword arguments for `plotnine.ggplot.save`.
`static`	
suffix	File extension; determines file format.

classmethod add_tasks(*c: Computer*, *key: Key | str*, **inputs*, *strict: bool = False*) → Key | str

Add a task to *c* to generate and save the Plot.

Analogous to `Operator.add_tasks()`.

basename = 'IEA_EEI-data'

File name base for saving the plot.

generate(*data*)

Generate and return the plot.

A subclass of Plot **must** implement this method.

Parameters

args (`Sequence` of `pandas.DataFrame` or `other`) – One argument is given corresponding to each of the *inputs*.

Because `plotnine` operates on pandas data structures, *save()* automatically converts any `Quantity` inputs to `pandas.DataFrame` before they are passed to *generate()*.

inputs: Sequence[Hashable] = []

`Keys` referring to `Quantities` or other inputs accepted by *generate()*.

classmethod make_task(**inputs*)

Return a task `tuple` to add to a Computer.

Deprecated since version 1.18.0: Use *add_tasks()* instead.

Parameters
***inputs** (*.Key* or `str` or `hashable`, *optional*) – If provided, overrides the *inputs* property of the class.

Returns
- The first, callable element of the task is *save()*.
- The second element is `"config"`, to access the configuration of the Computer.
- The third and following elements are the *inputs*.

Return type
`tuple`

path: Path | None = None

Path for file output. If it is not set, *save()* will populate it with a value constructed from `config["output_dir"]`, *basename*, and *suffix*. The implementation of *generate()* in a Plot sub-class may assign any other value, for instance one constructed at runtime from the *inputs*.

save(*config*, **args*, ***kwargs*) → Path | None

Prepare data, call *generate()*, and save to file.

This method is used as the callable in the task generated by *add_tasks()*.

New in version 1.24.1: This method uses `disable_copy_on_write()` to work around has2k1/mizani#38. This may cause issues if other computations (for instance, of the inputs to the Plot) rely on Pandas' copy-on-write behaviour being enabled.

save_args: Dict[str, Any] = {'verbose': False}

Keyword arguments for `plotnine.ggplot.save`.

suffix = '.pdf'

File extension; determines file format.

message_ix_models.tools.iea.web

Tools for IEA (Extended) World Energy Balance (WEB) data.

Module Attributes

DIMS	Dimensions of the data.
FILES	Mapping from (provider, year, time stamp) → set of file name(s) containing data.

message_ix_models.tools.iea.web.DIMS

message_ix_models.tools.iea.web.**DIMS = ['COUNTRY', 'PRODUCT', 'TIME', 'FLOW', 'MEASURE']**

Dimensions of the data.

message_ix_models.tools.iea.web.FILES

message_ix_models.tools.iea.web.**FILES = {('IEA', '2023'): ('WBIG1.zip', 'WBIG2.zip'), ('OECD', '2021'): ('cac5fa90-en.zip',), ('OECD', '2022'): ('372f7e29-en.zip',), ('OECD', '2023'): ('8624f431-en.zip',)}**

Mapping from (provider, year, time stamp) → set of file name(s) containing data.

Functions

fwf_to_csv(path[, progress])	Convert the IEA fixed-width file format to CSV.
generate_code_lists(provider, edition[, ...])	Extract structure from the data itself.
iea_web_data_for_query(base_path, ...)	Load data from *base_path* / *filenames* in IEA WEB formats.
load_data(provider, edition[, query_expr, path])	Load data from the IEA World Energy Balances.
unpack_zip(path)	Unpack a ZIP archive.

message_ix_models.tools.iea.web.fwf_to_csv

message_ix_models.tools.iea.web.**fwf_to_csv**(*path: Path*, *progress: bool = False*) → Path

Convert the IEA fixed-width file format to CSV.

This appears to operate at about 900k lines / second, about 1 minute for the IEA 2023 .TXT files. This is faster than doing full pandas I/O, which takes 5–10 minutes depending on formats.

message_ix_models.tools.iea.web.generate_code_lists

message_ix_models.tools.iea.web.**generate_code_lists**(*provider: str*, *edition: str*, *output_path: os.PathLike | None = None*) → None

Extract structure from the data itself.

message_ix_models.tools.iea.web.iea_web_data_for_query

message_ix_models.tools.iea.web.**iea_web_data_for_query**(*base_path: Path*, **filenames: str*, *query_expr: str*) → DataFrame

Load data from *base_path* / *filenames* in IEA WEB formats.

Data returned by this function is cached using *cached()*; see also *SKIP_CACHE*.

message_ix_models.tools.iea.web.load_data

message_ix_models.tools.iea.web.**load_data**(*provider: str*, *edition: str*, *query_expr="MEASURE == 'TJ' and TIME >= 1980"*, *path: Path | None = None*) → DataFrame

Load data from the IEA World Energy Balances.

Parameters

- **provider** (str) – First entry in *FILES*.
- **edition** (str) – Second entry in *FILES*.

- **query_expr** (str, *optional*) – Used with pandas.DataFrame.query() to reduce the returned data.
- **base_path** (os.Pathlike, *optional*) – Path containing *FILES*. If not provided, locations within message_data or *message_ix_models* are used.

Returns
The data frame has one column for each of *DIMS*, plus "Value".

Return type
pandas.DataFrame

message_ix_models.tools.iea.web.unpack_zip

message_ix_models.tools.iea.web.**unpack_zip**(*path: Path*) → Path

Unpack a ZIP archive.

Classes

IEA_EWEB(source, source_kw)	Provider of exogenous data from the IEA Extended World Energy Balances.

message_ix_models.tools.iea.web.IEA_EWEB

class message_ix_models.tools.iea.web.**IEA_EWEB**(*source*, *source_kw*)

Bases: *ExoDataSource*

Provider of exogenous data from the IEA Extended World Energy Balances.

To use data from this source, call *exo_data.prepare_computer()* with the source_kw:

- "provider": Either "IEA" or "OECD". See *FILES*.
- "edition": one of "2021", "2022", or "2023". See *FILES*.
- "product": str or list of str.
- "flow": str or list of str.

The returned data have the extra dimensions "product" and "flow", and are not aggregated by year.

Example

```
>>> keys = prepare_computer(
...     context,
...     computer,
...     source="IEA_EWEB",
...     source_kw=dict(
...         provider="OECD", edition="2022", product="CHARCOAL", flow="RESIDENT
↪"
...     ),
... )
>>> result = computer.get(keys[0])
```

__init__(*source*, *source_kw*)

Initialize the data source.

Methods

__init__(source, source_kw)	Initialize the data source.
raise_on_extra_kw(kwargs)	Helper for subclasses to handle the *source_kw* argument.
transform(c, base_key)	Aggregate only; do not interpolate on "y".

Attributes

aggregate	`True` if *transform()* should aggregate data on the n dimension.
extra_dims	Optional additional dimensions for the returned `Key`/`Quantity`.
id	Identifier for this particular source.
interpolate	`True` if *transform()* should interpolate data on the y dimension.
name	Optional name for the returned `Key`/`Quantity`.

aggregate: bool = True

`True` if *transform()* should aggregate data on the n dimension.

extra_dims: Tuple[str, ...] = ('product', 'flow')

Optional additional dimensions for the returned `Key`/`Quantity`. If not set by *__init__()*, the dimensions are (n, y).

id: str = 'IEA_EWEB'

Identifier for this particular source.

interpolate: bool = True

`True` if *transform()* should interpolate data on the y dimension.

name: str = ''

Optional name for the returned `Key`/`Quantity`. If not set by *__init__()*, then the "measure" keyword is used.

raise_on_extra_kw(*kwargs*) → None

Helper for subclasses to handle the *source_kw* argument.

1. Store *aggregate* and *interpolate*, if they remain in *kwargs*.
2. Raise `ValueError` if there are any other, unhandled keyword arguments in *kwargs*.

transform(*c: genno.Computer*, *base_key: genno.Key*) → genno.Key

Aggregate only; do not interpolate on "y".

Energy efficiency indicators (`tools.iea.eei`)

See *IEA_EEI*. This data is produced by the IEA and retrieved from the Energy Efficiency Indicators database. It is proprietary.

The data:

- Has the geographic resolution of individual countries, and scope including 41 countries:
- 24 IEA member countries for which data covering most end-uses area available: Australia, Austria, Belgium, Canada, Czech Republic, Denmark, Finland, France, Germany, Greece, Hungary, Italy, Japan, Korea, Luxembourg, the Netherlands, New Zealand, Poland, Portugal, Slovak Republic, Spain, Switzerland, the United Kingdom and the United States.

- Others including Brazil, Chile, Lithuania, Morocco, Armenia, Azerbaijan, Belarus, Georgia, Kazakhstan, Kyrgyzstan, Republic of Moldova, Ukraine, Uzbekistan.
- Includes measures/variables for energy consumption, efficiency, carbon emissions, and others for four conceptual sectors: Residential, Services, Industry and Transport.
- The **December 2020 edition** covers the time periods 2000–2018 with annual resolution.

Note: Currently, *iea.eei* mainly retrieves and processes data useful for MESSAGEix-Transport. To retrieve other end-use sectoral data, the code can be extended.

(Extended) World Energy Balances (tools.iea.web)

Note: These data are **proprietary** and require a paid subscription.

The approach to handling proprietary data is the same as in *project.advance* and *project.ssp*:

- Copies of the data are stored in the (private) message_data repository using Git LFS. This respository is accessible only to users who have a license for the data.
- *message_ix_models* contains only a 'fuzzed' version of the data (same structure, random values) for testing purposes.
- Non-IIASA users must obtain their own license to access and use the data; obtain the data themselves; and place it on the system where they use *message_ix_models*.

The module *message_ix_models.tools.iea.web* attempts to detect and support both the providers/formats described below. The code supports using data from any of the above locations and formats, in multiple ways:

- Use *tools.iea.web.load_data()* to load data as pandas.DataFrame and apply further pandas processing.
- Use *IEA_EWEB* via *tools.exo_data.prepare_computer()* to use the data in genno structured calculations.

The **documentation** for the 2023 edition of the IEA source/format is publicly available.

Structure

The data have the following conceptual dimensions, each enumerated by a different list of codes:

- FLOW, PRODUCT: for both of these, the lists of codes appearing in the data are the same from 2021 and 2023 inclusive.
- COUNTRY: The data provided by IEA directly contain codes that are all caps, abbreviated country names, for instance "DOMINICANR". The data provided by the OECD contain ISO 3166-1 alpha-3 codes, for instance "DOM". In both cases, there are additional labels denoting country groupings; these are defined in the documentation linked above.

  Changes visible in these lists include:

  - 2022 → 2023:

        - New codes: ASEAN, BFA, GREENLAND, MALI, MRT, PSE, TCD.
        - Removed: MASEAN.
    - 2021 → 2022:
        - New codes: GNQ, MDG, MKD, RWA, SWZ, UGA.
        - Removed: EQGUINEA, GREENLAND, MALI, MBURKINAFA, MCHAD, MMADAGASCA, MMAURITANI, MPALESTINE, MRWANDA, MUGANDA, NORTHMACED.
- TIME: always a year.
- MEASURE: unit of measurement, either "TJ" or "ktoe".

message_ix_models is packaged with SDMX structure data (stored in `message_ix_models/data/sdmx/`) comprising code lists extracted from the raw data for the COUNTRY, FLOW, and PRODUCT dimensions. These can be used with other package utilities, for instance:

```
>>> from message_ix_models.util.sdmx import read

# Read a code list from file: codes used in the
# 2022 edition data from the OECD provider
>>> cl = read("IEA:PRODUCT_OECD(2022)")

# Show some of its elements
>>> print("\n".join(sorted(cl.items[:5])))
ADDITIVE
ANTCOAL
AVGAS
BIODIESEL
BIOGASES
```

The documentation linked above has full descriptions of each code.

IEA provider/format

From 2023 (or earlier), the data are provided directly on the IEA website at https://www.iea.org/data-and-statistics/data-product/world-energy-balances. These data are available in two formats; 'IVT' or "Beyond 20/20" format (not supported by this module) or fixed-width text files. The latter are characterized by:

- Multiple ZIP archives with names like `WBIG[12].zip`, each containing a portion of the data and typically 110–130 MiB compressed size
- …each containing a single, fixed-with TXT file with a name like `WORLDBIG[12].TXT`, typically 3–4 GiB uncompressed,
- …with no column headers, but data resembling:

  ```
  WORLD   HARDCOAL   1960   INDPROD   KTOE ..
  ```

  …that appear to correspond to, respectively, the COUNTRY, PRODUCT, TIME, FLOW, and MEASURE dimensions and "Value" column of the above data, respectively.

OECD provider/format

Up until 2023, the EWEB data were available from the OECD iLibrary with DOI 10.1787/enestats-data-en. These files were characterized by:

- Single ZIP archives with names like `cac5fa90-en.zip`; typically ~850 MiB compressed size,
- …containing a single CSV file with a name like `WBIG_2022-2022-1-EN-20230406T100006.csv`, typically >20 GiB uncompressed,
- …with a particular list of columns like: "MEASURE", "Unit", "COUNTRY", "Country", "PRODUCT", "Product", "FLOW", "Flow", "TIME", "Time", "Value", "Flag Codes", "Flags",
- …with contents that duplicated code IDs—for instance, in the "FLOW" column—with human-readable labels—for instance in the "Flow" column:

Column name	Example value
MEASURE[1]	KTOE
Unit	ktoe
COUNTRY	WLD
Country	World
PRODUCT	COAL
Product	Coal and coal products
FLOW	INDPROD
Flow	Production
TIME	2012
Time	2012
Value	1234.5678
Flag Codes	M
Flags	Missing value; data cannot exist

This source is discontinued and will not publish subsequent editions of the data.

8.11 Low-level utilities (`util`)

Submodules:

click	Command-line utilities.
context	Context and settings for *message_ix_models* code.
importlib	Load model and project code from `message_data`.
_logging	Logging utilities.
node	Utilities for nodes.
pooch	Utilities for using Pooch.
scenarioinfo	*ScenarioInfo* class.
sdmx	Utilities for handling objects from `sdmx`.

Commonly used:

[1] the column is sometimes labelled "UNIT", but the contents appear to be the same.

Config(local_data, platform_info, str] =, ...)	Top-level configuration for *message_ix_models* and message_data.
ConfigHelper()	Mix-in for dataclass-based configuration classes.
Context(*args, **kwargs)	Context and settings for *message_ix_models* code.
ScenarioInfo(scenario_obj, empty, ...)	Information about a Scenario object.
Spec(add, remove, require)	A specification for the structure of a model or variant.
broadcast(df[, labels])	Fill missing data in *df* by broadcasting.
cached(func)	Decorator to cache the return value of a function *func*.
check_support(context[, settings, desc])	Check whether a Context is compatible with certain *settings*.
convert_units(s, unit_info[, store])	Convert units of *s*, for use with apply().
copy_column(column_name)	For use with pandas.DataFrame.assign().
datetime_now_with_tz()	Current date and time with time zone information.
ffill(df, dim, values[, expr])	Forward-fill *df* on *dim* to cover *values*.
identify_nodes(scenario)	Return the ID of a node codelist given the contents of *scenario*.
iter_keys(base)	Return an iterator over a sequence of keys starting with *base_key*.
load_package_data(*parts[, suffix])	Load a *message_ix_models* package data file and return its contents.
load_private_data(*parts)	Load a private data file from message_data and return its contents.
local_data_path(*parts)	Construct a path for local data.
make_io(src, dest, efficiency[, on])	Return input and output data frames for a 1-to-1 technology.
make_matched_dfs(base, **par_value)	Return data frames derived from *base* for multiple parameters.
make_source_tech(info, common, **values)	Return parameter data for a 'source' technology.
maybe_query(series, query)	Apply pandas.DataFrame.query() if the *query* arg is not None.
merge_data(base, *others)	Merge dictionaries of DataFrames together into *base*.
minimum_version(expr)	Decorator for functions that require a minimum version of some upstream package.
nodes_ex_world(nodes)	Exclude "World" and anything containing "GLB" from *nodes*.
package_data_path(*parts)	Construct a path to a file under message_ix_models/data/.
private_data_path(*parts)	Construct a path to a file under data/ in message_data.
same_node(df[, from_col])	Fill 'node_{,dest,loc,origin,rel,share}' in *df* from *from_col*.
same_time(df)	Fill 'time_origin'/'time_dest' in *df* from 'time'.
series_of_pint_quantity(*args, **kwargs)	Suppress a spurious warning.
show_versions()	Output of ixmp.show_versions(), as a str.

class message_ix_models.util.**Adapter**

Adapt *data*.

Adapter is an abstract base class for tools that adapt data in any way, e.g. between different code lists for certain dimensions. An instance of an Adapter can be called with any of the following as *data*:

- genno.Quantity,
- pandas.DataFrame, or
- dict mapping str parameter names to values (either of the above types).

…and will return data of the same type.

Subclasses can implement different adapter logic by overriding the abstract `adapt()` method.

abstract adapt(*qty: AttrSeries*) → AttrSeries

Adapt data.

class message_ix_models.util.**MappingAdapter**(*maps: Mapping[str, Sequence[Tuple[str, str]]]*)

Adapt data using mappings for 1 or more dimension(s).

Parameters

maps (dict of Sequence of tuple) – Keys are names of dimensions. Values are sequences of 2-tuples; each tuple consists of an original label and a target label.

Examples

```
>>> a = MappingAdapter({"foo": [("a", "x"), ("a", "y"), ("b", "z")]})
>>> df = pd.DataFrame(
...     [["a", "m", 1], ["b", "n", 2]], columns=["foo", "bar", "value"]
... )
>>> a(df)
  foo  bar  value
0   x    m      1
1   y    m      1
2   z    n      2
```

adapt(*qty: AttrSeries*) → AttrSeries

Adapt data.

message_ix_models.util.**add_par_data**(*scenario: Scenario*, *data: Mapping[str, DataFrame]*, *dry_run: bool = False*)

Add *data* to *scenario*.

Parameters

- **data** – Dict with keys that are parameter names, and values are pd.DataFrame or other arguments
- **dry_run** (*optional*) – Only show what would be done.

See also:

strip_par_data

message_ix_models.util.**aggregate_codes**(*df: DataFrame*, *dim: str*, *codes*)

Aggregate *df* along dimension *dim* according to *codes*.

message_ix_models.util.**broadcast**(*df: DataFrame*, *labels: DataFrame | None = None*, ***kwargs*) → DataFrame

Fill missing data in *df* by broadcasting.

broadcast() is suitable for use with partly-filled data frames returned by message_ix.util.make_df(), with 1 column per dimension, plus a "value" column. It is also usable with pandas.DataFrame.pipe() for chained operations.

labels (if any) are handled first: one copy or duplicate of *df* is produced for each row (set of labels) in this argument. Then, *kwargs* are handled; *broadcast()* returns one copy for each element in the cartesian product of the dimension labels given by *kwargs*.

Parameters

- **labels** (pandas.DataFrame) – Each column (dimension) corresponds to one in *df*. Each row represents one matched set of labels for those dimensions.
- **kwargs** – Keys are dimensions. Values are labels along that dimension to fill.

Returns
The length is either 1 or an integer multiple of the length of *df*.

Return type
pandas.DataFrame

Raises
ValueError – if any of the columns in *labels* or *kwargs* are not present in *df*, or if those columns are present but not empty.

Examples

```
>>> from message_ix import make_df
>>> from message_ix_models.util import broadcast
# Create a base data frame with some empty columns
>>> base = make_df("input", technology="t", value=[1.1, 2.2])
# Broadcast (duplicate) the data across 2 dimensions
>>> df = base.pipe(broadcast, node_loc=["node A", "node B"], mode=["m0", "m1"])
# Show part of the result
>>> df.dropna(axis=1)
  mode node_loc technology  value
0   m0   node A          t    1.1
1   m0   node A          t    2.2
2   m0   node B          t    1.1
3   m0   node B          t    2.2
4   m1   node A          t    1.1
5   m1   node A          t    2.2
6   m1   node B          t    1.1
7   m1   node B          t    2.2
```

message_ix_models.util.**cached**(*func: Callable*) → Callable

Decorator to cache the return value of a function *func*.

On a first call, the data requested is returned and also cached under *Context.get_cache_path()*. On subsequent calls, if the cache exists, it is used instead of calling the (possibly slow) *func*.

When *SKIP_CACHE* is true, *func* is always called.

See also:

Caching in the genno documentation

message_ix_models.util.**check_support**(*context*, *settings={}*, *desc: str = ''*) → None

Check whether a Context is compatible with certain *settings*.

Raises

- **NotImplementedError** – if any *context* value for a key of *settings* is not among the values in *settings*.
- **KeyError** – if the key is not set on *context* at all.

See also:

Advertise and check compatibility

message_ix_models.util.**convert_units**(*s: Series*, *unit_info: Mapping[str, Tuple[float, str, str | None]]*, *store='magnitude'*) → Series

Convert units of *s*, for use with apply().

s.name is used to retrieve a tuple of (*factor*, *input_unit*, *output_unit*) from *unit_info*. The (float) values of *s* are converted to pint.Quantity with the *input_unit* and factor; then cast to *output_unit*, if provided.

Parameters

- **s** (pandas.Series) –

- **unit_info** (dict (str -> tuple)) – Mapping from quantity name (matched to `s.name`) to 3-tuples of (*factor*, *input_unit*, *output_unit*). *output_unit* may be `None`. For example, see `ikarus.UNITS`.
- **store** ("magnitude" or "quantity") – If "magnitude", the values of the returned series are the magnitudes of the results, with no output units. If "quantity", the values are scalar `Quantity` objects.

Returns
Same shape, index, and values as *s*, with output units.

Return type
`pandas.Series`

message_ix_models.util.**copy_column**(*column_name*)

For use with `pandas.DataFrame.assign()`.

Examples

Modify *df* by filling the column 'baz' with the value `3`, and copying the column 'bar' into column 'foo'.

```
>>> df.assign(foo=copy_column('bar'), baz=3)
```

Note that a similar assignment can be achieved with `eval()`:

```
>>> df.eval("foo = bar")
```

`copy_column()` is useful in the context of more complicated calls to `assign()`.

message_ix_models.util.**datetime_now_with_tz**() → datetime

Current date and time with time zone information.

message_ix_models.util.**ffill**(*df: DataFrame*, *dim: str*, *values: Sequence[str | Code]*, *expr: str | None = None*) → DataFrame

Forward-fill *df* on *dim* to cover *values*.

Parameters

- **df** (`pandas.DataFrame`) – Data to fill forwards.
- **dim** (`str`) – Dimension to fill along. Must be a column in *df*.
- **values** (`list` of `str`) – Labels along *dim* that must be present in the returned data frame.
- **expr** (`str`, *optional*) – If provided, `DataFrame.eval()` is called. This can be used to assign one column to another. For instance, if *dim* == "year_vtg" and *expr* is "year_act = year_vtg", then forward filling is performed along the "year_vtg" dimension/ column, and then the filled values are copied to the "year_act" column.

message_ix_models.util.**identify_nodes**(*scenario: Scenario*) → str

Return the ID of a node codelist given the contents of *scenario*.

Returns
The ID of the *Node code lists* containing the regions of *scenario*.

Return type
`str`

Raises
ValueError – if no codelist can be identified, or the nodes in the scenario do not match the children of the "World" node in the codelist.

message_ix_models.util.**iter_keys**(*base: genno.Key*) → KeyIterator

Return an iterator over a sequence of keys starting with *base_key*.

This can be used for shorthand when constructing sequences of `genno` computations.

Example

```
>>> base_key = genno.Key("foo:a-b-c")
>>> k = iter_keys(base_key)
>>> k()
<foo:a-b-c:0>
>>> k()
<foo:a-b-c:1>
>>> k()
<foo:a-b-c:2>
```

message_ix_models.util.**load_package_data**(**parts: str*, *suffix: str | None = '.yaml'*) → Any

Load a *message_ix_models* package data file and return its contents.

Data is re-used if already loaded.

Example

The single call:

```
>>> info = load_package_data("node", "R11")
```

1. loads the metadata file `data/node/R11.yaml`, parsing its contents,
2. stores those values at `PACKAGE_DATA["node R11"]` for use by other code, and
3. returns the loaded values.

Parameters

- **parts** (`Iterable` of `str`) – Used to construct a path under `message_ix_models/data/`.
- **suffix** (`str`, *optional*) – File name suffix, including, the ".", e.g. `.yaml`.

Returns

Configuration values that were loaded.

Return type

`dict`

message_ix_models.util.**load_private_data**(**parts: str*) → Mapping

Load a private data file from `message_data` and return its contents.

Analogous to *load_package_data()*, but for non-public data.

Parameters

parts (`Iterable` of `str`) – Used to construct a path under `data/` in the `message_data` repository.

Returns

Configuration values that were loaded.

Return type

`dict`

Raises

RuntimeError – if `message_data` is not installed.

message_ix_models.util.**local_data_path**(**parts*) → Path

Construct a path for local data.

The setting `message local data` in the user's ixmp configuration file is used as a base path. If this is not configured, the current working directory is used.

Parameters
parts (`Sequence` of `str` or `Path`) – Joined to the base path using `Path.joinpath()`.

See also:

Choose locations for data

message_ix_models.util.**make_io**(*src*, *dest*, *efficiency*, *on='input'*, ***kwargs*)

Return input and output data frames for a 1-to-1 technology.

Parameters

- **src** (`tuple` of `str`) – Input (commodity, level, unit)
- **dest** (`tuple` of `str`) – Output (commodity, level, unit)
- **efficiency** (`float`) – Conversion efficiency.
- **on** (`'input'` or `'output'`) – If 'input', *efficiency* applies to the input, and the output, thus the activity level of the technology, is in dest[2] units. If 'output', the opposite.
- **kwargs** – Passed to `make_df()`.

Returns
Keys are 'input' and 'output'; values are data frames.

Return type
`dict (str -> pd.DataFrame)`

message_ix_models.util.**make_matched_dfs**(*base: MutableMapping*, ***par_value: float | Quantity*) → Dict[str, DataFrame]

Return data frames derived from *base* for multiple parameters.

Creates one data frame per keyword argument.

Parameters

- **base** (`pandas.DataFrame`, `dict`, `etc.`) – Used to populate other columns of each data frame. Duplicates—which occur when the target parameter has fewer dimensions than *base*—are dropped.
- **par_values** – Argument names (e.g. 'fix_cost') are passed to `make_df()`. If the value is `float`, it overwrites the "value" column; if `pint.Quantity`, its magnitude overwrites "value" and its units the "units" column, as a formatted string.

Returns
one for each parameter in *par_values*.

Return type
`dict` of `pandas.DataFrame`

Examples

```
>>> input = make_df("input", ...)
>>> cf_tl = make_matched_dfs(
>>>     input,
>>>     capacity_factor=1,
>>>     technical_lifetime=pint.Quantity(8, "year"),
>>> )
```

message_ix_models.util.**make_source_tech**(*info: Scenario | ScenarioInfo*, *common*, ***values*) → Dict[str, DataFrame]

Return parameter data for a 'source' technology.

The technology has no inputs; its output commodity and/or level are determined by *common*; either single values, or `None` if the result will be `pipe()`'d through *`broadcast()`*.

Parameters

- **info** (`Scenario` or *`ScenarioInfo`*) –
- **common** (`dict`) – Passed to `make_df()`.
- ****values** – Values for 'capacity_factor' (optional; default 1.0), 'output', 'var_cost', and optionally 'technical_lifetime'.

Returns

Suitable for *`add_par_data()`*.

Return type

`dict`

message_ix_models.util.**mark_time**(*quiet: bool = False*) → None

Record and log (if *quiet* is `True`) a time mark.

message_ix_models.util.**maybe_query**(*series: Series*, *query: str | None*) → Series

Apply `pandas.DataFrame.query()` if the *query* arg is not `None`.

`query()` is not chainable (pandas-dev/pandas#37941). Use this function with `pandas.Series.pipe()`, passing an argument that may be `None`, to have a chainable query operation that can be a no-op.

message_ix_models.util.**merge_data**(*base: MutableMapping[str, DataFrame]*, **others: Mapping[str, DataFrame]*) → None

Merge dictionaries of DataFrames together into *base*.

message_ix_models.util.**minimum_version**(*expr: str*) → Callable

Decorator for functions that require a minimum version of some upstream package.

See *`prepare_reporter()`* / *`test_prepare_reporter()`* for a usage example.

Parameters

expr – Like "example 1.2.3.post0". The condition for the decorated function is that the installed version must be equal to or greater than this version.

message_ix_models.util.**package_data_path**(**parts*) → Path

Construct a path to a file under `message_ix_models/data/`.

Use this function to access data packaged and installed with *`message_ix_models`*.

Parameters

parts (`Sequence` of `str` or `Path`) – Joined to the base path using `joinpath()`.

See also:

Choose locations for data

message_ix_models.util.**path_fallback**(**parts: str | Path*, *where: str | List[str | Path] = ''*) → Path

Locate a path constructed from *parts* found in the first of several directories.

This allows to implement 'fallback' behaviour in which files or directories in certain locations are used preferentially.

Parameters

- **parts** – Path parts or fragments such as directory names and a final file name.
- **where** – Either:
  - `str` containing one or more of:
    - "cache": locate *parts* in the *message_ix_models* cache directory.
    - "package": locate *parts* in *message_ix_models* package data (same as *package_data_path()*).
    - "private": locate *parts* in the `message_data /data/` directory (same as *private_data_path()*).
    - "test": locate test data in `package_data_path("test", ...)`
  - `list` where each element is `str` (one of the above) or a `pathlib.Path`.

Returns

The first of the locations indicated by *where* in which the file or directory *parts* exists.

Return type

`pathlib.Path`

Raises

ValueError – If *where* is empty or *parts* are not found in any of the indicated locations.

message_ix_models.util.**preserve_log_level**()

Context manager to preserve the level of the `message_ix_models` logger.

message_ix_models.util.**private_data_path**(**parts*) → Path

Construct a path to a file under `data/` in `message_data`.

Use this function to access non-public (for instance, embargoed or proprietary) data stored in the `message_data` repository.

If the repository is not available, the function falls back to *Context.get_local_path()*, where users may put files obtained through other messages.

Parameters

parts (`Sequence` of `str` or `Path`) – Joined to the base path using `joinpath()`.

See also:

Choose locations for data

message_ix_models.util.**replace_par_data**(*scenario: Scenario*, *parameters: str | Sequence[str]*, *filters: Mapping[str, str | int | Collection[str] | Collection[int]]*, *to_replace: Mapping[str, Mapping[str, str] | Mapping[int, int]]*) → None

Replace data in *parameters* of *scenario*.

Parameters

- **scenario** – Scenario in which to replace data.
- **parameters** (`str` or `Sequence` of `str`) – Name(s) of parameters in which to replace data.
- **filters** – Passed to `Scenario.par()` argument of the same name.

- **to_replace** – Passed to `pandas.DataFrame.replace()` argument of the same name.

Examples

Replace data in the "relation_activity" parameter for a particular technology and relation: assign the same values as entries in a different relation name for the same technology.

```
>>> replace_par_data(
...     scenario,
...     "relation_activity",
...     dict(technology="hp_gas_i", relation="CO2_r_c"),
...     dict(relation={"CO2_r_c": "CO2_ind"}),
... )
```

message_ix_models.util.**same_node**(*df: DataFrame*, *from_col='node_loc'*) → DataFrame

Fill 'node_{,dest,loc,origin,rel,share}' in *df* from *from_col*.

message_ix_models.util.**same_time**(*df: DataFrame*) → DataFrame

Fill 'time_origin'/'time_dest' in *df* from 'time'.

message_ix_models.util.**series_of_pint_quantity**(**args*, ***kwargs*) → Series

Suppress a spurious warning.

Creating a `pandas.Series` with a list of `pint.Quantity` triggers a warning "The unit of the quantity is stripped when downcasting to ndarray," even though the entire object is being stored and the unit is **not** stripped. This function suppresses this warning.

message_ix_models.util.**show_versions**() → str

Output of `ixmp.show_versions()`, as a `str`.

message_ix_models.util.**silence_log**(*names: str | None = None*, *level: int = 40*)

Context manager to temporarily quiet 1 or more loggers.

Parameters

- **names** (`str`, *optional*) – Space-separated names of loggers to quiet.
- **level** (`int`, *optional*) – Minimum level of log messages to allow.

Examples

```
>>> with silence_log():
>>>     log.warning("This message is not recorded.")
```

message_ix_models.util.**strip_par_data**(*scenario: Scenario*, *set_name: str*, *element: str*, *dry_run: bool = False*, *dump: Dict[str, DataFrame] | None = None*) → int

Remove *element* from *set_name* in scenario, optionally dumping to *dump*.

Parameters

- **dry_run** (`bool`, *optional*) – If `True`, only show what would be done.
- **dump** (`dict`, *optional*) – If provided, stripped data are stored in this dictionary. Otherwise, they are discarded.

Returns

Total number of rows removed across all parameters.

Return type

`int`

See also:

add_par_data

message_ix_models.util.cache.**SKIP_CACHE = False**

Controls whether cached data is returned for functions decorated with *cached()*. Set to `True` to force reload.

8.11.1 util.click

Command-line utilities.

These are used for building CLIs using `click`.

PARAMS contains, among others:

- **--urls-from-file=...** Path to a file containing scenario URLs, one per line. These are parsed and stored on *Config.scenarios*.

class message_ix_models.util.click.**CliRunner**(*cli_cmd: ~click.core.Command*, *cli_module: str*, *env: ~typing.Mapping[str, str] = <factory>*, *charset: str = 'utf-8'*, *method: ~typing.Literal['click', 'subprocess'] = 'click'*)

Similar to `click.testing.CliRunner`, with extra features.

assert_exit_0(**args*, ***kwargs*) → Result

Assert a result has exit_code 0, or print its traceback.

If any *args* or *kwargs* are given, `invoke()` is first called. Otherwise, the result from the last call of `invoke()` is used.

Raises

AssertionError – if the result exit code is not 0.

cli_cmd: Command

CLI entry point

cli_module: str

CLI module

invoke_subprocess(**args*, ***kwargs*) → Result

Invoke the CLI in a subprocess.

method: Literal['click', 'subprocess'] = 'click'

Method for invoking the command

message_ix_models.util.click.**PARAMS = {'dest': <Option dest>, 'dry_run': <Option dry_run>, 'force': <Option force>, 'nodes': <Option nodes>, 'output_model': <Option output_model>, 'platform_dest': <Option platform_dest>, 'policy_path': <Option policy_path>, 'quiet': <Option quiet>, 'regions': <Option regions>, 'rep_out_path': <Option rep_out_path>, 'rep_template': <Option rep_template>, 'run_reporting_only': <Option run_reporting_only>, 'ssp': <Argument ssp>, 'urls_from_file': <Option urls_from_file>, 'verbose': <Option verbose>, 'years': <Option years>}**

Common command-line parameters (arguments and options). See *common_params()*.

message_ix_models.util.click.**common_params**(*param_names: str*)

Decorate a click.command with common parameters *param_names*.

param_names must be a space-separated string of names appearing in *PARAMS*, e.g. `"ssp force output_model"`. The decorated function receives keyword arguments with these names:

```
@click.command()
@common_params("ssp force output_model")
def mycmd(ssp, force, output_model)
    # ...
```

message_ix_models.util.click.**default_path_cb**(**default_parts*)

Return a callback function for click.Option handling.

If no option value is given, the callback uses *Context.get_local_path()* and *default_parts* to provide a path that is relative to local data directory, e.g. the current working directory (see *Data, metadata, and configuration*).

message_ix_models.util.click.**exec_cb**(*expression: str*) → Callable

Return a callback that exec()-utes an *expression*.

The *expression* is executed in a limited context that has only two names available:

- context: the *Context* instance.
- value: the value passed to the click.Parameter.

Example

```
>>> @click.command
... @click.option(
...     "--myopt", callback=exec_cb("context.my_mod.my_opt = value + 3")
... )
... def cmd(...):
...     ...
```

message_ix_models.util.click.**format_sys_argv**() → str

Format sys.argv in a readable manner.

message_ix_models.util.click.**store_context**(*context: Context* | Context, *param*, *value*)

Callback that simply stores a value on the *Context* object.

Use this for parameters that are not used directly in a @click.command() function, but need to be carried by the Context for later use.

message_ix_models.util.click.**temporary_command**(*group: Group*, *command: Command*)

Temporarily attach command *command* to *group*.

message_ix_models.util.click.**unique_id**() → str

Return a unique ID for a CLI invocation.

The return value resembles "mix-models-debug-3332d415ef65bf2a-2023-02-02T162931" and contains:

- The CLI name and (sub)commands.
- A hash of all the CLI parameters (options and arguments).
- The current date and time, in ISO format with Windows-incompatible ":" removed.

message_ix_models.util.click.**urls_from_file**(*context: Context* | Context, *param*, *value*) → List[*ScenarioInfo*]

Callback to parse scenario URLs from *value*.

8.11.2 util.config

class message_ix_models.util.config.**ConfigHelper**

Mix-in for `dataclass`-based configuration classes.

This provides 3 methods—*read_file()*, *replace()*, and *from_dict()*— that help to use `dataclass` classes for handling *message_ix_models* configuration.

All 3 methods take advantage of name manipulations: the characters "-" and " " are replaced with underscores ("_"). This allows to write the names of attributes in legible ways—e.g. "attribute name" or "attribute-name" instead of "attribute_name"— in configuration files and/or code.

classmethod from_dict(*data: Mapping*)

Construct an instance from *data* with name manipulation.

read_file(*path: Path*, *fail='raise'*) → None

Update configuration from file.

Parameters

- **path** – to a `.yaml` file containing a top-level mapping.
- **fail** (`str`) – if "raise" (the default), any names in *path* which do not match attributes of the dataclass raise a ValueError. Ottherwise, a message is logged.

replace(***kwargs*)

Like `dataclasses.replace()` with name manipulation.

update(***kwargs*)

Update attributes in-place.

Raises

AttributeError – Any of the *kwargs* are not fields in the data class.

8.11.3 util.context

Context and settings for *message_ix_models* code.

class message_ix_models.util.context.**Context**(**args*, ***kwargs*)

Context and settings for *message_ix_models* code.

Context is a subclass of `dict`, so common methods like `copy()` and `setdefault()` may be used to handle settings. To be forgiving, it also provides attribute access; `context.foo` is equivalent to `context["foo"]`.

A Context instance always has the following members:

- `core`: an instance of *message_ix_models.Config*.
- `model`: an instance of *message_ix_models.model.Config*.

Attributes of these classes may be accessed by shorthand, e.g. `context.regions` is shorthand for `context.model.regions`.

Context provides additional methods to do common tasks that depend on configurable settings:

clone_to_dest([create])	Return a scenario based on the `--dest` command-line option.
close_db()	
delete()	Hide the current Context from future *get_instance()* calls.
get_cache_path(*parts)	Return a path to a local cache file, i.e. within *Config.cache_path*.
get_local_path(*parts[, suffix])	Return a path under *Config.local_data*.
get_platform([reload])	Return a `Platform` from *Config.platform_info*.
get_scenario()	Return a `Scenario` from *scenario_info*.
handle_cli_args([url, platform, model_name, ...])	Handle command-line arguments.
only()	Return the only *Context* instance.
use_defaults(settings)	Update from *settings*.

clone_to_dest (*create=True*) → Scenario

Return a scenario based on the `--dest` command-line option.

Parameters

create (`bool`, *optional*) – If `True` (the default) and the base scenario does not exist, a bare RES scenario is created. Otherwise, an exception is raised.

Returns

To prevent the scenario from being garbage collected, keep a reference to its Platform:

Return type

`Scenario`

See also:

`create_res`

To use this method, either decorate a command with *common_params()*:

```
from message_data.tools.cli import common_params

@click.command()
@common_params("dest")
@click.pass_obj
def foo(context, dest):
    scenario, mp = context.clone_to_dest()
```

or, store the settings *Config.dest_scenario* and optionally *Config.dest_platform* on *context*:

```
c = Context.get_instance()

c.dest_scenario = dict(model="foo model", scenario="foo scenario")
scenario_mp = context.clone_to_dest()
```

The resulting scenario has the indicated model- and scenario names.

If `--url` (or `--platform`, `--model`, `--scenario`, and optionally `--version`) are given, the identified scenario is used as a 'base' scenario, and is cloned. If `--url`/`--platform` and `--dest` refer to different Platform instances, then this is a two-platform clone.

If no base scenario can be loaded, *bare.create_res()* is called to generate a base scenario.

delete ()

Hide the current Context from future *get_instance()* calls.

get_cache_path(**parts*) → Path

Return a path to a local cache file, i.e. within `Config.cache_path`.

The directory containing the resulting path is created if it does not already exist.

classmethod get_instance(*index=0*) → *Context*

Return a Context instance; by default, the first created.

Parameters

index (`int`, *optional*) – Index of the Context instance to return, e.g. `-1` for the most recently created.

get_local_path(**parts: str*, *suffix=None*) → Path

Return a path under `Config.local_data`.

Parameters

- **parts** – Path fragments, for instance directories, passed to `joinpath()`.
- **suffix** – File name suffix including a "."—for instance, ".csv"—passed to `with_suffix()`.

get_platform(*reload=False*) → Platform

Return a `Platform` from `Config.platform_info`.

When used through the CLI, `Config.platform_info` is a 'base' platform as indicated by the –url or –platform options.

If a Platform has previously been instantiated with `get_platform()`, the same object is returned unless *reload=True*.

get_scenario() → Scenario

Return a `Scenario` from `scenario_info`.

When used through the CLI, `scenario_info` is a 'base' scenario for an operation, indicated by the `--url` or `--platform/--model/--scenario` options.

handle_cli_args(*url=None*, *platform=None*, *model_name=None*, *scenario_name=None*, *version=None*, *local_data=None*, *verbose=False*, *_store_as=('platform_info', 'scenario_info')*)

Handle command-line arguments.

May update the `Config.local_data`, `platform_info`, `scenario_info`, and/or `url` settings.

classmethod only() → *Context*

Return the only `Context` instance.

Raises

IndexError – If there is more than one instance.

set_scenario(*scenario: Scenario*) → None

Update `Config.scenario_info` to match an existing *scenario*.

`Config.url` is also updated.

update([*E*], ***F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

use_defaults(*settings*)

Update from *settings*.

write_debug_archive() → None

Write an archive containing the files listed in *debug_paths*.

The archive file name is constructed using *unique_id()* and appears in a `debug` subdirectory under the *local data path*.

The archive also contains a file `command.txt` that gives the full command-line used to invoke **mix-models**.

8.11.4 util.importlib

Load model and project code from `message_data`.

class message_ix_models.util.importlib.**MessageDataFinder**

Load model and project code from `message_data`.

8.11.5 util._logging

Logging utilities.

class message_ix_models.util._logging.**Formatter**(*use_colour: bool = True*)

Formatter for log records.

Parameters
use_color (`bool`, *optional*) – If `True`, `colorama` is used to colour log messages.

format(*record*)

Format *record*.

Records are formatted like:

```
model.transport.data.add_par_data  220 rows in 'input'
...add_par_data  further messages
```

…with the calling function name (e.g. 'add_par_data') coloured for legibility on first occurrence, then dimmed when repeated.

class message_ix_models.util._logging.**QueueListener**(*queue*, **handlers*, *respect_handler_level=False*)

`logging.QueueListener` with a *flush()* method.

flush()

Flush the queue: join the listener/monitor thread and then restart.

class message_ix_models.util._logging.**SilenceFilter**(*names: str*, *level: int*)

Log filter that only allows records from *names* that are at or above *level*.

filter(*record*) → bool

Determine if the specified record is to be logged.

Returns True if the record should be logged, or False otherwise. If deemed appropriate, the record may be modified in-place.

class message_ix_models.util._logging.**StreamHandler**(*stream_name: str*)

Like `logging.StreamHandler`, but retrieve the stream on each access.

This avoids the case that `click`, `pytest`, or something else adjusts `sys.stdout` temporarily, but the handler's stored reference to the original is not updated.

stream_name: str

Name of the `sys` stream to use, as `str` rather than a direct reference.

message_ix_models.util._logging.**setup**(*level: str | int = 99*, *console: bool = True*, ***, *file: bool = False*) → None

Initialize logging.

Parameters

- **level** (str, *optional*) – Log level for the console log handler.
- **console** (bool, *optional*) – If False, do not print any messages to console.
- **file** (bool, *optional*) – If False, do not print any messages to file.

8.11.6 util.node

Utilities for nodes.

message_ix_models.util.node.**NODE_DIMS = ['n', 'node', 'node_loc', 'node_origin', 'node_dest', 'node_rel', 'node_share']**

Names of dimensions indexed by 'node'.

Todo: to be robust to changes in message_ix, read these names from that package.

message_ix_models.util.node.**R11_R12 = (('R11_AFR', 'R12_AFR'), ('R11_CPA', 'R12_CHN'), ('R11_EEU', 'R12_EEU'), ('R11_FSU', 'R12_FSU'), ('R11_LAM', 'R12_LAM'), ('R11_MEA', 'R12_MEA'), ('R11_NAM', 'R12_NAM'), ('R11_PAO', 'R12_PAO'), ('R11_PAS', 'R12_PAS'), ('R11_CPA', 'R12_RCPA'), ('R11_SAS', 'R12_SAS'), ('R11_WEU', 'R12_WEU'))**

Mapping from R11 to R12 node IDs.

message_ix_models.util.node.**R11_R14 = (('R11_AFR', 'R14_AFR'), ('R11_FSU', 'R14_CAS'), ('R11_CPA', 'R14_CPA'), ('R11_EEU', 'R14_EEU'), ('R11_LAM', 'R14_LAM'), ('R11_MEA', 'R14_MEA'), ('R11_NAM', 'R14_NAM'), ('R11_PAO', 'R14_PAO'), ('R11_PAS', 'R14_PAS'), ('R11_FSU', 'R14_RUS'), ('R11_SAS', 'R14_SAS'), ('R11_FSU', 'R14_SCS'), ('R11_FSU', 'R14_UBM'), ('R11_WEU', 'R14_WEU'))**

Mapping from R11 to R14 node IDs.

message_ix_models.util.node.**adapt_R11_R12 = <message_ix_models.util.common.MappingAdapter object>**

Adapt data from the R11 to the R14 node list.

The data is adapted using the mappings in *R11_R12* for each of the dimensions in *NODE_DIMS*.

message_ix_models.util.node.**adapt_R11_R14 = <message_ix_models.util.common.MappingAdapter object>**

Adapt data from the R11 to the R14 node list.

The data is adapted using the mappings in *R11_R14* for each of the dimensions in *NODE_DIMS*.

message_ix_models.util.node.**nodes_ex_world**(*nodes: Sequence[str | Code]*) → List[str | Code]

Exclude "World" and anything containing "GLB" from *nodes*.

May also be used as a genno (reporting) operator.

8.11.7 util.pooch

Utilities for using Pooch.

class message_ix_models.util.pooch.**Extract**(*members=None*, *extract_dir=None*)

Similar to pooch.Unzip, streamlined using pathlib.

This version supports:

- Absolute or relative paths for the *extract_dir* parameter.
- .zip or .tar.xz archives.

message_ix_models.util.pooch.**GH_MAIN = 'https://github.com/iiasa/message-ix-models/raw/main/message_ix_models/data'**

Base URL portion for files stored in the message-ix-models GitHub repository.

message_ix_models.util.pooch.**SOURCE: Mapping[str, Mapping[str, Any]] = {'MESSAGEix-Nexus': {'pooch_args': {'base_url': 'https://github.com/iiasa/message-ix-models/raw/main/message_ix_models/data/water/', 'registry': {'water.tar.xz': 'sha1:ec9e0655af90ca844c0158968bb03a194b8fa6c6'}}, 'processor': <message_ix_models.util.pooch.Extract object>}, 'PRIMAP': {'pooch_args': {'base_url': 'ftp://datapub.gfz-potsdam.de/download/10.5880.PIK.2019.001/', 'registry': {'PRIMAP-hist_v2.0_11-Dec-2018.zip': 'md5:f28d58abef4ecfc36fc8ce3e9eef2871'}}, 'processor': <message_ix_models.util.pooch.Extract object>}, 'SSP-Update-3.0': {'pooch_args': {'base_url': 'https://github.com/iiasa/message-ix-models/raw/main/message_ix_models/data/ssp/', 'registry': {'1706548837040-ssp_basic_drivers_release_3.0_full.csv.gz': 'sha1:e2af7a88aeed7d0e44ceaf2dff60f891cf551517'}}}, 'SSP-Update-3.0.1': {'pooch_args': {'base_url': 'https://github.com/iiasa/message-ix-models/raw/main/message_ix_models/data/ssp/', 'registry': {'1710759470883-ssp_basic_drivers_release_3.0.1_full.csv.gz': 'sha1:e5c24c27ee743e79dac5a578235b35a68cd64183'}}}, 'snapshot-0': {'pooch_args': {'base_url': 'doi:10.5281/zenodo.5793870', 'registry': {'MESSAGEix-GLOBIOM_1.1_R11_no-policy_baseline.xlsx': 'md5:222193405c25c3c29cc21cbae5e035f4'}}, 'processor': <message_ix_models.util.pooch.UnpackSnapshot object>}, 'snapshot-1': {'pooch_args': {'base_url': 'doi:10.5281/zenodo.10514052', 'registry': {'MESSAGEix-GLOBIOM_1.1_R11_no-policy_baseline.xlsx': 'md5:e7c0c562843e85c643ad9d84fecef979'}}}}**

Supported remote sources of data.

class message_ix_models.util.pooch.**UnpackSnapshot**

Pooch processor that calls *snapshot.unpack()*.

message_ix_models.util.pooch.**fetch**(*pooch_args: dict*, ***, *extra_cache_path: str | None = None*, ***fetch_kwargs*) → Tuple[Path, ...]

Create a Pooch instance and fetch a single file.

Files are stored under the directory identified by *Context.get_cache_path()*, unless *args* provides another location.

Parameters

- **args** – Passed to pooch.create().
- **kwargs** – Passed to pooch.Pooch.fetch().

Returns

Path to the fetched file.

Return type
Path

See also:

snapshot.load()

8.11.8 util.pycountry

message_ix_models.util.pycountry.**COUNTRY_NAME = {'Korea': 'Korea, Republic of', 'Republic of Korea': 'Korea, Republic of', 'Russia': 'Russian Federation', 'South Korea': 'Korea, Republic of', 'Turkey': 'Türkiye'}**

Mapping from common, non-standard country names to ISO 3166-1 names.

message_ix_models.util.pycountry.**iso_3166_alpha_3**(*name: str*) → str | None

Return an ISO 3166 alpha-3 code for a country *name*.

Parameters
name (str) – Country name. This is looked up in the pycountry 'name', 'official_name', or 'common_name' field. Values in *COUNTRY_NAME* are supported.

Return type
str or None

8.11.9 util.scenarioinfo

ScenarioInfo class.

class message_ix_models.util.scenarioinfo.**ScenarioInfo**(*scenario_obj: dataclasses.InitVar[typing.Optional[ForwardRef('Scenario')]] = None*, *empty: dataclasses.InitVar[bool] = False*, *platform_name: str | None = None*, *model: str | None = None*, *scenario: str | None = None*, *version: int | None = None*, *set: ~typing.Dict[str, ~typing.List] = <factory>*, *par: ~typing.Dict[str, ~pandas.core.frame.DataFrame] = <factory>*, *y0: int = -1*, *is_message_macro: bool = False*, *_yv_ya: ~pandas.core.frame.DataFrame | None = None*)

Information about a Scenario object.

Code that prepares data for a target Scenario can accept a ScenarioInfo instance. This avoids the need to create or load an actual Scenario, which can be slow under some conditions.

ScenarioInfo objects can also be used (for instance, by *apply_spec()*) to describe the contents of a Scenario *before* it is created.

ScenarioInfo objects have the following convenience attributes:

set	Elements of ixmp/message_ix sets.
io_units(technology, commodity[, level])	Return units for the MESSAGE input or output parameter.
is_message_macro	True if a MESSAGE-MACRO scenario.
N	Elements of the set 'node'.
units_for(set_name, id)	Return the units associated with code *id* in MESSAGE set *set_name*.
Y	Elements of the set 'year' that are >= the first model year.
y0	First model year, if set, else Y[0].
yv_ya	pandas.DataFrame with valid year_vtg, year_act pairs.

Parameters

scenario_obj (message_ix.Scenario) – If given, *set* is initialized from this existing scenario.

Examples

Iterating over an instance gives "model", "scenario", "version" and the values of the respective attributes: >>> si = ScenarioInfo.from_url("model name/scenario name#123") >>> dict(si) {'model': 'model name', 'scenario': 'scenario name', 'version': 123}

See also:

Spec

property N

Elements of the set 'node'.

See also:

nodes_ex_world

property Y: List[int]

Elements of the set 'year' that are >= the first model year.

classmethod from_url(*url: str*) → *ScenarioInfo*

Create an instance using only an *url*.

io_units(*technology: str*, *commodity: str*, *level: str | None = None*) → Unit

Return units for the MESSAGE input or output parameter.

These are implicitly determined as the ratio of:

- The units for the origin (for input) or destination *commodity*, per *units_for()*.
- The units of activity for the *technology*.

Parameters

level (str) – Placeholder for future functionality, i.e. to use different units per (commodity, level). Currently ignored. If given, a debug message is logged.

Raises

ValueError – if either *technology* or *commodity* lack defined units.

is_message_macro: bool = False

True if a MESSAGE-MACRO scenario.

model: str | None = None

Model name; equivalent to `TimeSeries.model`.

par: Dict[str, DataFrame]

Elements of `ixmp`/`message_ix` parameters.

property path: str

A valid file system path name similar to *`url`*.

Characters invalid in Windows paths are replaced with "_".

scenario: str | None = None

Scenario name; equivalent to `TimeSeries.scenario`.

set: Dict[str, List]

Elements of `ixmp`/`message_ix` sets.

units_for(*set_name: str*, *id: str*) → Unit

Return the units associated with code *id* in MESSAGE set *set_name*.

`ixmp` (or the sole `JDBCBackend`, as of v3.5.0) does not handle unit information for variables and equations (unlike parameter values), such as MESSAGE decision variables `ACT`, `CAP`, etc. In *`message_ix_models`* and `message_data`, the following conventions are (generally) followed:

- The units of `ACT` and others are consistent for each `technology`.
- The units of `COMMODITY_BALANCE`, `STOCK`, `commodity_stock`, etc. are consistent for each `commodity`.

Thus, codes for elements of these sets (e.g. *Commodities (commodity.yaml)*) can be used to carry the standard units for the corresponding quantities. *`units_for()`* retrieves these units, for use in model-building and reporting.

Todo: Expand this discussion and transfer to the `message_ix` docs.

See also:

`io_units`

update(*other:* ScenarioInfo)

Update with the set elements of *other*.

property url: str

Identical to `TimeSeries.url`.

version: int | None = None

Scenario version; equivalent to `TimeSeries.version`.

y0: int = -1

First model year, if set, else `Y[0]`.

year_from_codes(*codes: List[Code]*)

Update using a list of *codes*.

The following are updated:

- *`set`* `year`
- *`set`* `cat_year`, with the first model year.
- *`par`* `duration_period`

Any existing values are discarded.

After this, the attributes *`y0`* and *`Y`* give the first model year and model years, respectively.

Examples

Get a particular code list, create a ScenarioInfo instance, and update using the codes:

```
>>> years = get_codes("year/A")
>>> info = ScenarioInfo()
>>> info.year_from_codes(years)
```

Use populated values:

```
>>> info.y0
2020
>>> info.Y[:3]
[2020, 2030, 2040]
>>> info.Y[-3:]
[2090, 2100, 2110]
```

property yv_ya

pandas.DataFrame with valid year_vtg, year_act pairs.

class message_ix_models.util.scenarioinfo.**Spec**(*add: ~message_ix_models.util.scenarioinfo.ScenarioInfo = <factory>*, *remove: ~message_ix_models.util.scenarioinfo.ScenarioInfo = <factory>*, *require: ~message_ix_models.util.scenarioinfo.ScenarioInfo = <factory>*)

A specification for the structure of a model or variant.

A Spec collects 3 *ScenarioInfo* instances at the attributes *add*, *remove*, and *require*. This is the type that is accepted by *apply_spec()*; *Building models (model.build)* describes how a Spec is used to modify a Scenario. A Spec may also be used to express information about the target structure of data to be prepared; like ScenarioInfo, this can happen before the target Scenario exists.

Spec also provides:

- Dictionary-style access, e.g. s["add"] is equivalent to s.add.
- Error checking; setting keys other than add/remove/require results in an error.
- *merge()*, a helper method.

add: ***ScenarioInfo***

Structure to be added to a base scenario.

static merge(*a:* Spec, *b:* Spec) → *Spec*

Merge Specs *a* and *b* together.

Returns a new Spec where each member is a union of the respective members of *a* and *b*.

remove: ***ScenarioInfo***

Structure to be removed from a base scenario.

require: ***ScenarioInfo***

Structure that must be present in a base scenario.

8.11.10 util.sdmx

Utilities for handling objects from sdmx.

message_ix_models.util.sdmx.**as_codes**(*data: List[str] | Dict[str, str | Code]*) → List[Code]

Convert *data* to a list of Code objects.

Various inputs are accepted:

- list of str.
- dict, in which keys are id and values are further dict with keys matching other Code attributes.

message_ix_models.util.sdmx.**eval_anno**(*obj: AnnotableArtefact*, *id: str*)

Retrieve the annotation *id* from *obj*, run eval() on its contents.

Deprecated since version 2023.9.12: Use sdmx.model.common.AnnotableArtefact.eval_annotation(), which provides the same functionality.

message_ix_models.util.sdmx.**make_enum**(*urn*, *base=<enum 'Enum'>*)

Create an enum.Enum (or *base*) with members from codelist *urn*.

message_ix_models.util.sdmx.**read**(*urn: str*, *base_dir: PathLike | None = None*)

Read SDMX object from package data given its *urn*.

message_ix_models.util.sdmx.**register_agency**(*agency: Agency*) → AgencyScheme

Add *agency* to the AgencyScheme "IIASA_ECE:AGENCIES".

message_ix_models.util.sdmx.**write**(*obj*, *base_dir: PathLike | None = None*, *basename: str | None = None*)

Store an SDMX object as package data.

8.12 Test utilities and fixtures (testing)

Fixtures:

mix_models_cli(session_context, tmp_env)	A *CliRunner* object that invokes the **mix-models** CLI.
session_context(pytestconfig, tmp_env)	A *Context* connected to a temporary, in-memory database.
test_context(request, session_context)	A copy of *session_context()* scoped to one test function.
user_context(request)	Context which can access user's configuration, e.g.

Marks:

NIE	Shorthand for marking a parametrized test case that is expected to fail because it is not implemented.
not_ci([reason, action])	Mark a test as xfail or skipif if on CI infrastructure.

Others:

EXPORT_OMIT	Items with names that match (partially or fully) these names are omitted by *export_test_data()*.
bare_res(request, context[, solved])	Return or create a Scenario containing the bare RES for use in testing.
export_test_data(context)	Export a subset of data from a scenario, for use in tests.
pytest_addoption(parser)	Add two command-line options to pytest:

message_ix_models.testing.**EXPORT_OMIT = ['aeei', 'cost_MESSAGE', 'demand_MESSAGE', 'demand', 'depr', 'esub', 'gdp_calibrate', 'grow', 'historical_gdp', 'kgdp', 'kpvs', 'lakl', 'land', 'lotol', 'mapping_macro_sector', 'MERtoPPP', 'prfconst', 'price_MESSAGE', 'ref_', 'sector']**

Items with names that match (partially or fully) these names are omitted by *export_test_data()*.

message_ix_models.testing.**GHA = False**

True if tests occur on GitHub Actions.

message_ix_models.testing.**NIE = MarkDecorator(mark=Mark(name='xfail', args=(), kwargs={'raises': <class 'NotImplementedError'>}))**

Shorthand for marking a parametrized test case that is expected to fail because it is not implemented.

message_ix_models.testing.**bare_res**(*request*, *context:* Context, *solved: bool = False*) → Scenario

Return or create a Scenario containing the bare RES for use in testing.

The Scenario has a model name like "MESSAGEix-GLOBIOM [regions] Y[years]", for instance "MESSAGEix-GLOBIOM R14 YB" (see *bare.name()*) and a scenario name either from request.node.name or "baseline" plus a random string.

This function should:

- only be called from within test code, i.e. in message_data.tests.
- be called once for each test function, so that each test receives a fresh copy of the RES scenario.

Parameters

- **request** (FixtureRequest or None) – The pytest request fixture. If provided the pytest test node name is used for the scenario name of the returned Scenario.
- **context** (*Context*) – Passed to *testing.bare_res()*.
- **solved** (bool, *optional*) – Return a solved Scenario.

Returns

The scenario is a fresh clone, so can be modified freely without disturbing other tests.

Return type

Scenario

message_ix_models.testing.**export_test_data**(*context:* Context)

Export a subset of data from a scenario, for use in tests.

The context settings export_nodes (default: "R11_AFR" and "R11_CPA") and export_techs (default: "coal_ppl") are used to filter the data exported. In addition, any item (set, parameter, variable, or equation) with a name matching *EXPORT_OMIT* *or* the context setting export_exclude is discarded.

The output is stored at data/tests/*model name_scenario name_techs*.xlsx in message_data.

See also:

Prepare data for testing

message_ix_models.testing.**mix_models_cli**(*session_context*, *tmp_env*)

A *CliRunner* object that invokes the **mix-models** CLI.

NB this requires:

- The ixmp tmp_env() fixture. This sets IXMP_DATA to a temporary directory managed by pytest.
- The *session_context()* fixture. This (a) sets *Config.local_data* to a temporary directory within IXMP_DATA and (b) ensures changes to *Context* made by invoked commands do not reach other tests.

message_ix_models.testing.**not_ci**(*reason=None*, *action='skip'*)

Mark a test as xfail or skipif if on CI infrastructure.

Checks the GITHUB_ACTIONS environment variable; returns a pytest mark.

message_ix_models.testing.**pytest_addoption**(*parser*)

Add two command-line options to pytest:

--local-cache
Use existing, local cache files in tests. This option can speed up tests that *use* the results of slow data loading/parsing. However, if cached values are not up to date with the current code, unexpected failure may occur.

--jvmargs
Additional arguments to give for the Java Virtual Machine used by ixmp's JDBCBackend. Used by *session_context()*.

message_ix_models.testing.**session_context**(*pytestconfig*, *tmp_env*)

A *Context* connected to a temporary, in-memory database.

This Context is suitable for modifying and running test code that does not affect the user/developer's filesystem and configured ixmp databases.

Uses the tmp_env() fixture from ixmp. This fixture also sets:

- *Config.cache_path*, depending on whether the **--local-cache** CLI option was given:
  - If not given: pytest's standard cache directory.
  - If given: the /cache/ directory under the user's "message local data" directory.
- the "message local data" config key to a temporary directory /data/ under the pytest tmp_path directory.

message_ix_models.testing.**test_context**(*request*, *session_context*)

A copy of *session_context()* scoped to one test function.

message_ix_models.testing.**unpack_snapshot_data**(*context: Context*, *snapshot_id: int*)

Already-unpacked data for a snapshot.

This copies the .csv.gz files from message_ix_models/data/test/… to the directory where they *would* be unpacked by .model.snapshot._unpack. This causes the code to skip unpacking them, which can be very slow.

message_ix_models.testing.**user_context**(*request*)

Context which can access user's configuration, e.g. platform names.

8.13 Multi-scenario workflows (workflow)

8.13.1 Concept & design

Research with MESSAGEix models often involves multiple scenarios that are related to one another or derived from one another by certain modifications. Together, the solutions/reported information from these scenarios provide the output data used in research products, e.g. a plot comparing total emissions in a policy scenario to a reference scenario.

model.build provides tools to build models or scenarios based on (possibly empty) base scenarios; and *tools* provides tools for manipulating scenarios or model input data (parameters). The *Workflow* API provided in this module allows researchers to use these pieces and atomic, reusable functions to define arbitrarily complex workflows involving many, related scenarios; and then to solve, report, or otherwise operate on those scenarios.

The generic pattern for workflows is:

- Each scenario has zero or 1 (or more?) base/precursor/antecedent scenarios. These must exist before the target scenario can be created.
- A workflow 'step' includes:
  1. A precursor scenario is obtained.

     It may be returned by a prior workflow step, or loaded from a `Platform`.
  2. (Optional) The precursor scenario is cloned to a target model name and scenario name.
  3. A function is called to operate on the scenario. This function may do zero or more of:
     - Apply structure or data modifications, for example:
       * Set up a model variant, e.g. adding the MESSAGEix-Materials structure to a base MESSAGEix-GLOBIOM model.
       * Change policy variables via constraint parameters.
       * Any other possible modification.
     - Solve the target scenario.
     - Invoke reporting.
  4. The resulting function is passed to the next workflow step.
- A workflow can consist of any number of scenarios and steps.
- The same precursor scenario can be used as the basis for multiple target scenarios.
- A workflow is *Workflow.run()* starting with the earliest precursor scenario, ending with 1 or many target scenarios.

The implementation is based on the observation that these form a graph (specifically, a directed, acyclic graph, or DAG) of nodes (= scenarios) and edges (= steps), in the same way that `message_ix.report` calculations do; and so the `dask` DAG features (via `genno`) can be used to organize the workflow.

8.13.2 Usage

General

Define a workflow using ordinary Python functions, each handling the modifications/manipulations in an atomic workflow step. These functions **must**:

- Accept at least 2 arguments:
  1. A *Context* instance.
  2. The precursor scenario.
  3. Optionally additional, keyword-only arguments.
- Return either:

  - a `Scenario` object, that can be the same object provided as an argument, or a different scenario, e.g. a clone or a different scenario, even from a different platform.
  - `None`. In this case, any modifications implemented by the step should be reflected in the Scenario given as an argument.

The functions **may**:

- call any other code, and
- be as short (one line) or long (many lines) as desired;

and they **should**:

- respond in documented, simple ways to settings on the Context argument and/or their keyword argument(s), if any.

```
def changes_a(s: Scenario) -> None:
    """Change a scenario by modifying structure data, but not data."""
    with s.transact():
        s.add_set("technology", "test_tech")

    # Here, invoke other code to further modify `s`

def changes_b(s: Scenario, value=100.0) -> None:
    """Change a scenario by modifying parameter data, but not structure.

    This function takes an extra argument, `values`, so functools.partial()
    can be used to supply different values when it is used in different
    workflow steps. See below.
    """
    with s.transact():
        s.add_par(
            "technical_lifetime",
            make_df(
                "technical_lifetime",
                node_loc=s.set("node")[0],
                year_vtg=s.set("year")[0],
                technology="test_tech",
                value=100.0,
                unit="y",
            ),
        )

    # Here, invoke other code to further modify `s`
```

With the steps defined, the workflow is composed using a *Workflow* instance. Call *Workflow.add_step()* to define each target model with its precursor and the function that will create the former from the latter:

```
from message_ix_models import Context, Workflow

# Create the workflow
ctx = Context.get_instance()
wf = Workflow(ctx)

# "Model name/base" is loaded from an existing platform
wf.add_step(
    "base",
    None,
    target="ixmp://example-platform/Model name/base#123",
)

# "Model/A" is created from "Model/base" by calling changes_a()
wf.add_step("A", "base", changes_a, target="Model/A")
```

(continues on next page)

(continued from previous page)

```
# "Model/B1" is created from "Model/A" by calling changes_b() with the
# default value
wf.add_step("B1", "A", changes_b, target="Model/B1")

# "Model/B2" is similar, but uses a different value
wf.add_step("B2", "A", partial(changes_b, value=200.0), target="model/B2")
```

Finally, the workflow is triggered using *Workflow.run()*, giving either one step name or a list of names. The indicated scenarios are created (and solved, if the workflow steps involve solving); if this requires any precursor scenarios, those are first created and solved, etc. as required. Other, unrelated scenarios/steps are not created.

```
s1, s2 = wf.run(["B1", "B2"])
```

Usage examples

- `message_data.projects.navigate.workflow`

Todo: Expand with discussion of workflow patterns common in research projects using MESSAGEix, e.g.:

- Run the same scenario with multiple emissions budgets.

8.13.3 API reference

Tools for modeling workflows.

class message_ix_models.workflow.**Workflow**(*context: Context*)

Workflow for operations on multiple Scenarios.

Parameters

context (Context) – Context object with settings common to the entire workflow.

add_step(*name: str*, *base: str | None = None*, *action: Callable | None = None*, *replace=False*, ***kwargs*) → str

Add a *WorkflowStep* to the workflow.

Parameters

- **name** (str) – Name for the new step.
- **base** (str or None) – Previous step that produces the a pre-requisite scenario for this step.
- **action** (CallbackType) – Function to be executed to modify the base into the target Scenario.
- **replace** (bool) – True to replace an existing step.
- **kwargs** – Keyword arguments for *action*; passed to and stored on the *WorkflowStep* until used.

Returns

The same as *name*.

Return type

str

Raises

genno.KeyExistsError – if the step *name* already exists. Use *replace* to force overwriting an existing step.

guess_target(*step_name: str*, *kind: Literal['platform', 'scenario'] = 'scenario'*) → Tuple[Mapping, str]

Traverse the graph looking for non-empty platform_info/scenario_info.

Returns the info, and the step name containing it. Usually, this will identify the name of the platform, model, and/or scenario that is received and acted upon by *step_name*. This may not be the case if preceding workflow steps perform clone steps that are not recorded in the *target* parameter to *WorkflowStep*.

Parameters

- **step_name** (str) – Initial step from which to work backwards.
- **kind** (str, "platform" or "scenario") – Whether to look up *platform_info* or *scenario_info*.

run(*name_or_names: str | List[str]*)

Run all workflow steps necessary to produce *name_or_names*.

Parameters

name_or_names (str or list of str) – Identifier(s) of steps to run.

truncate(*name: str*)

Truncate the workflow at the step *name*.

The step *name* is replaced with a new *WorkflowStep* that simply loads the target Scenario that would be produced by the original step.

Raises

KeyError – if step *name* does not exist.

message_ix_models.workflow.**make_click_command**(*wf_callback: str*, *name: str*, *slug: str*, ***kwargs*) → Command

Generate a click CLI command to run a *Workflow*.

This command:

- when invoked, imports the module containing the *wf_callback*, retrieve and calls the function. This function receives the values for any click parameters (arguments and/or options) passed in *kwargs*. The module is not imported until/unless the command is run.
- …is automatically given the parameters:
  - **--go**: Actually run the workflow; otherwise the workflow is only displayed.
  - **--from**: Truncate the workflow at any step(s) whose names are a full match for this regular expression.
- uses the default_key (if any) of the *Workflow* returned by *wf_callback*, if the user does not provide **TARGET** on the command-line.

Parameters

- **wf_callback** (str) – Fully-resolved name (module and object name) for a function that generates the workflow; for instance "message_ix_models.project.foo.workflow.generate".
- **name** (str) – Descriptive workflow name used in the **--help** text.
- **slug** (str) – File name fragment for writing the workflow diagram; the path *slug*-workflow.svg is used.
- **kwargs** (*optional*) – Passed to click.command(), for instance to define additional parameters for the command.

class message_ix_models.workflow.**WorkflowStep**(*action: Callable | None*, *target=None*, *clone=False*, ***kwargs*)

Single step in a multi-scenario workflow.

Nothing occurs when the WorkflowStep is instantiated.

Parameters

- **name** (str) – "model name/scenario name" for the Scenario produced by the step.
- **action** (CallbackType, *optional*) – Function to be executed to modify the base into the target Scenario.
- **clone** (bool, *optional*) – True to clone the base scenario the target.
- **target** (str, *optional*) – URL for the scenario produced by the workflow step. Parsed to *scenario_info* and *platform_info*.
- **kwargs** (dict) – Keyword arguments for *action*.

__call__ (*context:* Context, *scenario:* *Scenario* | *None* = *None*) → Scenario

Execute the workflow step.

action: Callable | None = None

Function to be executed on the subject scenario. If None, the target scenario is loaded via Context.get_scenario().

clone: bool | dict = False

True or a dict with keyword arguments to clone before *action* is executed. Default: False, do not clone.

kwargs: dict

Keyword arguments passed to *action*.

platform_info: dict

Target platform name and additional options.

scenario_info: dict

Target model name, scenario name, and optional version.

8.14 MESSAGEix-GLOBIOM global model

These pages document the IIASA Integrated Assessment Modeling (IAM) framework, also referred to as **MESSAGEix-GLOBIOM**, owing to the fact that the energy model MESSAGE*ix* and the land use model GLOBIOM are its most important components. MESSAGE*ix*-GLOBIOM was developed for the quantification of the so-called Shared Socio-economic Pathways (SSPs) which are the first application of the IAM framework.

Note: The documentation in this section was originally available at https://docs.messageix.org/global/ and maintained in a separate repository at iiasa/message_doc. In the future, it will be maintained in iiasa/message-ix-models and appear at the current URL.

The overall *message_ix_models* documentation provides **technical** description of the Python package of the same name, associated data, and their usage for *all* models in the MESSAGE*ix*-GLOBIOM 'family'. This section provides a thorough **methodological** description of the particular, central, global-scope instance of MESSAGE*ix*-GLOBIOM developed by the IIASA ECE Program, from which most other instances derive.

This section is periodically updated and expanded with additional information to describe the current implementation and its changes over time.

When referring to MESSAGE*ix*-GLOBIOM as described in this section, please use the following citations:[1]

[1] Download these citations in RIS or BibTeX format (web only).

- V. Krey, P. Havlik, P. N. Kishimoto, O. Fricko, J. Zilliacus, M. Gidden, M. Strubegger, G. Kartasasmita, T. Ermolieva, N. Forsell, M. Gusti, N. Johnson, J. Kikstra, G. Kindermann, P. Kolp, F. Lovat, D. L. McCollum, J. Min, S. Pachauri, Parkinson S. C., S. Rao, J. Rogelj, H. and Ünlü, G. Valin, P. Wagner, B. Zakeri, M. Obersteiner, and K. Riahi. MESSAGEix-GLOBIOM Documentation – 2020 release. Technical Report, International Institute for Applied Systems Analysis (IIASA), Laxenburg, Austria, 2020. URL: https://pure.iiasa.ac.at/id/eprint/17115, doi:10.22022/iacc/03-2021.17115.
- O. Fricko, P. Havlik, J. Rogelj, Z. Klimont, M. Gusti, N. Johnson, P. Kolp, M. Strubegger, H. Valin, M. Amann, T. Ermolieva, N. Forsell, M. Herrero, C. Heyes, G. Kindermann, V. Krey, D. L. McCollum, M. Obersteiner, S. Pachauri, S. Rao, E. Schmid, W. Schoepp, and K. Riahi. The marker quantification of the Shared Socioeconomic Pathway 2: A middle-of-the-road scenario for the 21st century. *Global Environmental Change*, 42:251–267, 2017. doi:10.1016/j.gloenvcha.2016.06.004.

MESSAGE*ix*-GLOBIOM is based on the MESSAGEix model & framework, which provides a flexible, *generic* abstraction of energy systems optimization models that can be parametrized in many ways. `message_ix` includes the 'MACRO' computable general equilibrium (CGE) for implementing macro-economic feedback. To refer to the generic MESSAGE, MACRO, and combined models—rather than the particular MESSAGE*ix*-GLOBIOM IAM instance or its specific applications for various publications and assessments—please follow the "User guidelines and notice" section of the `message_ix` documentation.

We thank Edward Byers, Jessica Jewell, Ruslana Palatnik, Narasimha D. Rao, and Fabio Sferra for their valuable comments that helped improving the text.

8.14.1 Overview

The IIASA IAM framework consists of a combination of five different models or modules - the energy model MESSAGE*ix*, the land use model GLOBIOM, the air pollution and GHG model GAINS, the aggregated macro-economic model MACRO and the simple climate model MAGICC - which complement each other and are specialized in different areas. All models and modules together build the IIASA IAM framework, also referred to as MESSAGE*ix*-GLOBIOM owing to the fact that the energy model MESSAGE*ix* and the land use model GLOBIOM are its central components. The five models provide input to and iterate between each other during a typical scenario development cycle. Below is a brief overview of how the models interact with each other, specifically in the context of developing the SSP scenarios.

MESSAGE*ix* (Huppmann et al., 2019 [31]) represents the core of the IIASA IAM framework (Fig. 8.1) and its main task is to optimize the energy system so that it can satisfy specified energy demands at the lowest costs. MESSAGE carries out this optimization in an iterative setup with MACRO, a single sector macro-economic model, which provides estimates of the macro-economic demand response that results from energy system and services costs computed by MESSAGE*ix*. For the six commercial end-use demand categories depicted in MESSAGE (see *Energy demand*), based on demand prices MACRO will adjust useful energy demands, until the two models have reached equilibrium (see *Macro-economy (MACRO)*). This iteration reflects price-induced energy efficiency adjustments that can occur when energy prices change. MESSAGE can represent different energy- and climate-related *Policies*.

GLOBIOM provides MESSAGE*ix* with information on land use and its implications, including the availability and cost of bioenergy, and availability and cost of emission mitigation in the AFOLU (Agriculture, Forestry and Other Land Use) sector (see *Land-use (GLOBIOM)*). To reduce computational costs, MESSAGE iteratively queries a GLOBIOM emulator which provides an approximation of land-use outcomes during the optimization process instead of requiring the GLOBIOM model to be rerun iteratively. Only once the iteration between MESSAGE*ix* and MACRO has converged, the resulting bioenergy demands along with corresponding carbon prices are used for a concluding analysis with the full-fledged GLOBIOM model. This ensures full consistency of the results from MESSAGE and GLOBIOM, and also allows producing a more extensive set of land-use related indicators, including spatially explicit information on land use.

Air pollution implications of the energy system are accounted for in MESSAGE*ix* by applying technology-specific air pollution coefficients derived from the GAINS model (see *Air pollution*). This approach has been applied to the SSP process (Rao et al., 2017 [80]). Alternatively, GAINS can be run ex-post based on MESSAGE*ix*-GLOBIOM scenarios to estimate air pollution emissions, concentrations and the related health impacts. This approach allows analyzing different air pollution policy packages (e.g., current legislation, maximum feasible reduction), including the estimation of costs for air pollution control measures. Examples for applying this way of linking MESSAGE*ix*-GLOBIOM and GAINS can be found in McCollum et al. (2018 [53]) and Grubler et al. (2018 [21]).

In general, cumulative global carbon emissions from all sectors are constrained at different levels, with equivalent pricing applied to other GHGs, to reach the desired radiative forcing levels (cf. right-hand side Fig. 8.1). The climate constraints are thus taken up in the coupled MESSAGE*ix*-GLOBIOM optimization, and the resulting carbon price is fed back to the full-fledged GLOBIOM model for full consistency. Finally, the combined results for land use, energy, and industrial emissions from MESSAGE*ix* and GLOBIOM are merged and fed into MAGICC (see *Climate (MAGICC)*), a global carbon-cycle and climate model, which then provides estimates of the climate implications in terms of atmospheric concentrations, radiative forcing, and global-mean temperature increase. Importantly, climate impacts and impacts of the carbon cycle are – depending on the specific application – currently only partly accounted for in the IIASA IAM framework. The entire framework is linked to an online database infrastructure which allows straightforward visualisation, analysis, comparison and dissemination of results (Riahi et al., 2017 [89]).

The scientific software underlying the global MESSAGE-GLOBIOM model is called the MESSAGE*ix* framework, an open-source, versatile implementation of a linear optimization problem, with the option of coupling to the computable general equilibrium (CGE) model MACRO to incorporate the effect of price changes on economic activity and demand for commodities and resources. MESSAGE*ix* is integrated with the *ix modeling platform* (ixmp), a "data warehouse" for version control of reference timeseries, input data and model results. ixmp provides interfaces to the scientific programming languages Python and R for efficient, scripted workflows for data processing and visualisation of results (Huppmann et al., 2019 [31]).

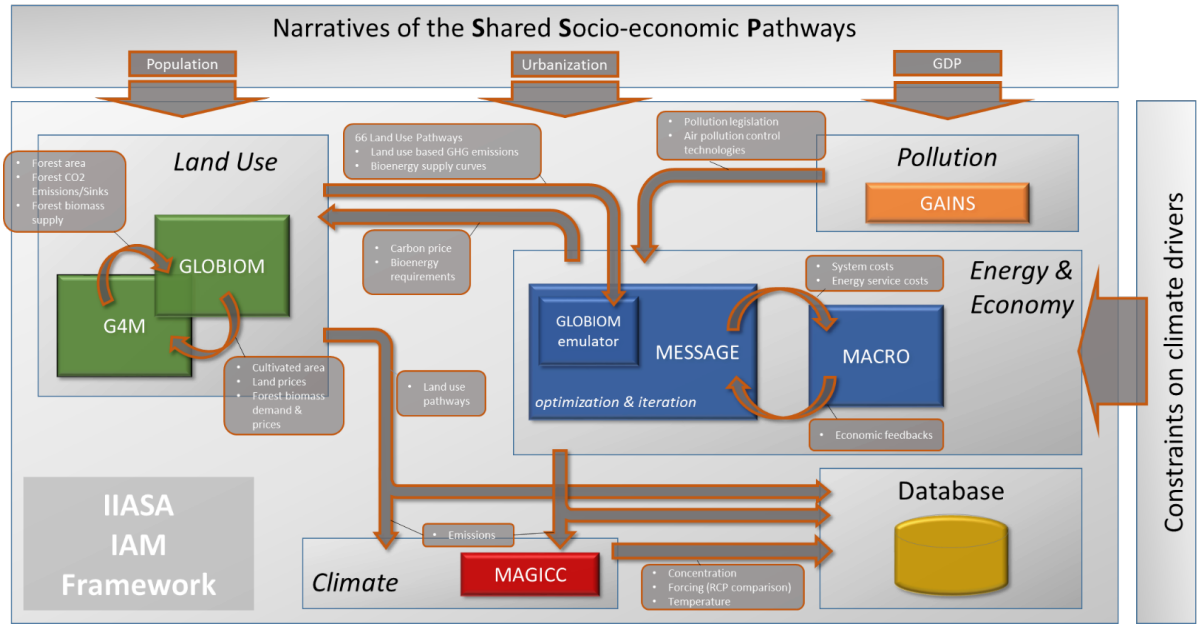

Fig. 8.1: Overview of the IIASA IAM framework. Coloured boxes represent respective specialized disciplinary models which are integrated for generating internally consistent scenarios (Fricko et al., 2017 [17]).

Regions

The combined MESSAGE*ix*-GLOBIOM framework has global coverage and divides the world into 11 regions which are also the native regions of the MESSAGE*ix* model (see Fig. 8.2 and Table 8.1 below). GLOBIOM natively operates at the level of 30 regions which in the linkage to MESSAGE*ix* are aggregated to the 11 regions as listed in Table 8.2.

The country definitions of the 11 MESSAGE*ix* regions are described in the table below (Table 8.1). In some scenarios, the MESSAGE*ix* region of FSU (Former Soviet Union) is disaggregated into four sub-regions resulting in a 14-region MESSAGE*ix* model.

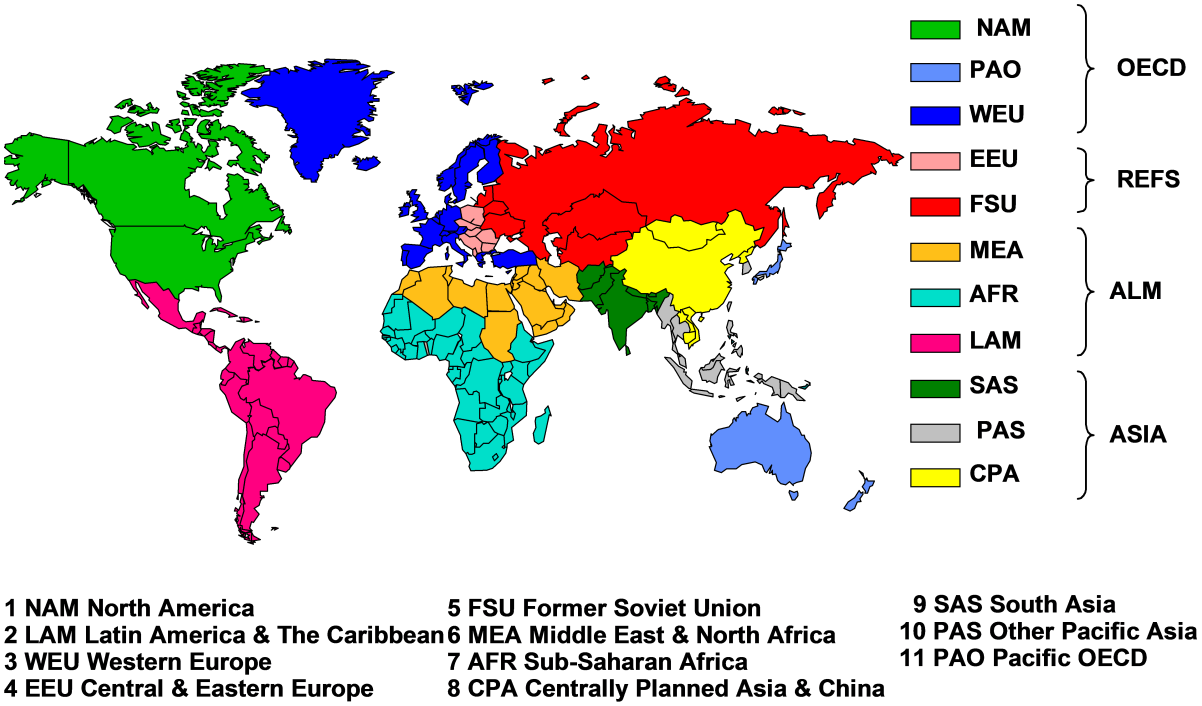

Fig. 8.2: Map of 11 MESSAGE*ix*-GLOBIOM regions including their aggregation to the four regions used in the Representative Concentration Pathways (RCPs).

Table 8.1: Listing of 11 regions used in MESSAGE*ix*-GLOBIOM, including their country definitions.

MESSAGE regions	Definition	List of countries
NAM	North America	Canada, Guam, Puerto Rico, United States of America, Virgin Islands
WEU	Western Europe	Andorra, Austria, Azores, Belgium, Canary Islands, Channel Islands, Cyprus, Denmark, Faeroe Islands, Finland, France, Germany, Gibraltar, Greece, Greenland, Iceland, Ireland, Isle of Man, Italy, Liechtenstein, Luxembourg, Madeira, Malta, Monaco, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, Turkey, United Kingdom
PAO	Pacific OECD	Australia, Japan, New Zealand
EEU	Central and Eastern Europe	Albania, Bosnia and Herzegovina, Bulgaria, Croatia, Czech Republic, The former Yugoslav Rep. of Macedonia, Hungary, Poland, Romania, Slovak Republic, Slovenia, Yugoslavia, Estonia, Latvia, Lithuania
FSU	Former Soviet Union	Armenia, Azerbaijan, Belarus, Georgia, Kazakhstan, Kyrgyzstan, Republic of Moldova, Russian Federation, Tajikistan, Turkmenistan, Ukraine, Uzbekistan
CPA	Centrally Planned Asia and China	Cambodia, China (incl. Hong Kong), Korea (DPR), Laos (PDR), Mongolia, Viet Nam
SAS	South Asia	Afghanistan, Bangladesh, Bhutan, India, Maldives, Nepal, Pakistan, Sri Lanka
PAS	Other Pacific Asia	American Samoa, Brunei Darussalam, Fiji, French Polynesia, Gilbert-Kiribati, Indonesia, Malaysia, Myanmar, New Caledonia, Papua, New Guinea, Philippines, Republic of Korea, Singapore, Solomon Islands, Taiwan (China), Thailand, Tonga, Vanuatu, Western Samoa
MEA	Middle East and North Africa	Algeria, Bahrain, Egypt (Arab Republic), Iraq, Iran (Islamic Republic), Israel, Jordan, Kuwait, Lebanon, Libya/SPLAJ, Morocco, Oman, Qatar, Saudi Arabia, Sudan, Syria (Arab Republic), Tunisia, United Arab Emirates, Yemen
LAM	Latin America and the Caribbean	Antigua and Barbuda, Argentina, Bahamas, Barbados, Belize, Bermuda, Bolivia, Brazil, Chile, Colombia, Costa Rica, Cuba, Dominica, Dominican Republic, Ecuador, El Salvador, French Guyana, Grenada, Guadeloupe, Guatemala, Guyana, Haiti, Honduras, Jamaica, Martinique, Mexico, Netherlands Antilles, Nicaragua, Panama, Paraguay, Peru, Saint Kitts and Nevis,

In addition to the 11 geographical regions, in the glboal MESSAGE*ix* model there is a global trade region where market clearing of global energy markets is happening and international shipping bunker fuel demand, uranium resource extraction and the nuclear fuel cycle are represented.

Table 8.2: Listing of 30 regions used in GLOBIOM, including their country definitions and the mapping to the 11 regions of the combined MESSAGE*ix*-GLOBIOM model.

MESSAGE regions	GLOBIOM regions	List of countries
NAM	Canada	Canada
	USA	United States of America
WEU	EU_MidWest	Austria, Belgium, Germany, France, Luxembourg, Netherlands
	EU_North	Denmark, Finland, Ireland, Sweden, United Kingdom
	EU_South	Cyprus, Greece, Italy, Malta, Portugal, Spain
	ROWE	Gibraltar, Iceland, Norway, Switzerland
	Turkey	Turkey
PAO	ANZ	Australia, New Zealand
	Japan	Japan
	Pacific_Islands	Fiji Islands, Kiribati, Papua New Guinea, Samoa, Solomon Islands, Tonga, Vanuatu
EEU	EU_Baltic	Estonia, Latvia, Lithuania
	EU_CentEast	Bulgaria, Czech Republic, Hungary, Poland, Romania, Slovakia, Slovenia
	RCEU	Albania, Bosnia and Herzegovina, Croatia, Macedonia, Serbia-Montenegro
FSU	Former_USSR	Armenia, Azerbaijan, Belarus, Georgia, Kazakhstan, Kyrgyzstan, Moldova, Russian Federation, Tajikistan, Turkmenistan, Ukraine, Uzbekistan
CPA	China	China
	RSEA_PAC	Cambodia, Korea DPR, Laos, Mongolia, Viet Nam
SAS	India	India
	RSAS	Afghanistan, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka
PAS	South_Korea	South Korea
	RSEA_OPA	Brunei Daressalaam, Indonesia, Singapore, Malaysia, Myanmar, Philippines, Thailand
MEA	MidEastNAfr	Algeria, Bahrain, Egypt, Iran, Iraq, Israel, Jordan, Kuwait, Lebanon, Libya, Morocco, Oman, Qatar, Saudi Arabia, Syria, Tunisia, United Arab Emirates, Yemen
LAM	Brazil	Brazil
	Mexico	Mexico
	RCAM	Bahamas, Barbados, Belize, Bermuda, Costa Rica, Cuba, Dominica, Dominican Republic, El Salvador, Grenada, Guatemala, Haiti, Honduras, Jamaica, Nicaragua, Netherland Antilles, Panama, St Lucia, St Vincent, Trinidad and Tobago
	RSAM	Argentina, Bolivia, Chile, Colombia, Ecuador, Guyana, Paraguay, Peru, Suriname, Uruguay, Venezuela
AFR	Congo_Basin	Cameroon, Central African Republic, Congo Republic, Democratic Republic of Congo, Equatorial, Guinea, Gabon
	EasternAf	Burundi, Ethiopia, Kenya, Rwanda, Tanzania, Uganda
	SouthAf	South Africa
	RoSAfr	Angola, Botswana, Comoros, Lesotho, Madagascar, Malawi, Mauritius, Mozambique, Namibia, Swaziland, Zambia, Zimbabwe
	WestCentAfr	Benin, Burkina Faso, Cape Verde, Chad, Cote d'Ivoire, Djibouti, Eritrea, Gambia, Ghana, Guinea, Guinea Bissau, Liberia, Mali, Mauritania, Niger, Nigeria, Senegal, Sierra Leone, Somalia, Sudan, Togo

Time steps

In global MESSAGE*ix* models the time horizon of 2010 to 2110 is generally subdivided into 5 or 10-year periods, using 2010 or 2015 as the base year. The 2020 period is partly calibrated so far, some recent trends are included in this time period, but some flexibility remains. The reporting years are the final years of periods which implies that investments that lead to the capacities in the reporting year are the average annual investments over the entire period the reporting year belongs to. In recent model versions, the model has been calibrated to 2015 running with 5-year modeling periods until the middle of the century (2020, 2025, 2030, 2035, 2040, 2045, 2050, 2055, 2060) and 10-year periods between 2060 and 2110.

MESSAGE*ix* can both operate perfect foresight over the entire time horizon, limited foresight (e.g., two or three periods into the future) or myopically, optimizing one period at a time (Keppo and Strubegger, 2010 [41]) (see Mathematical Specification for more details). Most frequently MESSAGE*ix* is run with perfect foresight, but for specific applications such as delayed participation in a global climate regime without anticipation (Krey and Riahi, 2009 [46]; O'Neill et al., 2010 [67]) limited foresight is used.

GLOBIOM models the time horizon 2000 to 2100 in 10 year time steps (2000, 2010, 2020, 2030, 2040, 2050, 2060, 2070, 2080, 2090, 2100) with the year 2000 being the base year of the model. The model is recursive-dynamic, i.e. it is solved for each period individually and then passes on results to the subsequent periods. The linkage between MESSAGE*ix* and GLOBIOM relies on the model results of the periods 2020 to 2100.

Policies

A number of different energy- and climate-related policies are, depending on the scenario setup and the research question addressed, explicitly represented in MESSAGE*ix*. This includes the following list of policies:

- GHG emission pricing
- GHG emission caps and trading emission allowances
- Renewable energy portfolio standards (e.g., share of renewable energy in electricity generation)
- Renewable energy and other technology capacity targets
- Energy import tariffs
- Fuel subsidies and micro-financing for achieving universal access to modern energy services in developing countries (via linkage to the MESSAGE-Access model)
- Air pollution legislation packages (fixed legislation, current and planned legislation, stringent legislation, maximum feasible reduction via linkage to the GAINS model)

In general, these policies are implemented via constraints or cost coefficients (negative and positive) in the optimization problem (see Section *Modeling policies* for more details). In the case of air pollution policies, the different legislation packages are implemented via a set of emission coefficients and associated costs derived from the GAINS model. The cost coefficients are, however, not part of the optimization procedure, but instead allow an ex-post quantification of air pollution policy costs for a specific energy scenario.

8.14.2 Socio-economic development

Behavioural change

With increasing affluence, consumers of final energy are more likely to demand technologies that are more convenient in their use, even if they cost more than less convenient energy forms. Examples of this empirically observed phenomenon are room heating with gas, electricity or district heat, which are more convenient than heating with coal. The affluent end-user does not like to fill up the coal furnace manually and is willing to pay more for a convenient technology. If MESSAGEix is to correctly reflect this phenomenon, the model's cost-minimizing behavior mustbe modified accordingly. As a model feature to accomplish this task, the concept of inconvenience factors has been introduced in the definition of end-use technologies. The inconvenience factors are specified for each end-use technology, time period and world region. The cost entry in the objective function is calculated as the monetary costs, multiplied by the inconvenience factor. The inconvenience factors for a given world region increase with the level of affluence

(GDP per capita) in this region. Flexible and grid-dependent energy technologies, such as electricity, gas and district heating have low inconvenience factors. A second mechanism for taking into account non-monetary decision criteria in the end-use sectors is the application of implicit discount rates which change perceived upfront investment costs by consumers. These two concepts are predominantly applied in the consumer dominated energy end-use sectors transportation (see *Transport sector*) and residential and commercial (see *Residential and commercial sectors*). Below, this is described in more detail for the MESSAGEix-Access model, an extension of MESSAGEix that focuses on residential energy services in developing countries which are characterized by high reliance on traditional fuels.

Behavioral change in MESSAGEix-Access

MESSAGEix-Access is a variant of the MESSAGEix model that provides a detailed representation of energy use for the residential sector in developing country regions. It is fully integrated with the MESSAGEix supply side model, but not in call scenarios is the the detailed demand-side representation used, but instead a more aggregated formulation with just seven demand categories is used (see *Energy demand*) which is parametrized off the detailed MESSAGEix-Access formulation. The objective function maximizes household utility by choosing an energy-equipment combination for an individual household group that meets a particular energy service demand at lowest cost. The model is calibrated with data on existing household energy use patterns, derived from national household surveys and energy statistics and balances for the base year 2005. Assumptions regarding urbanization, income growth and changes in income distributions over time drive the model outcomes in the future. In its current version the model is implemented only for 3 of the 11 MESSAGEix regions (see *Regions*), SAS, PAS and AFR, that are developing regions where access to modern energy remains the most limited.

The model distinguishes between two primary energy end-uses in the residential sector – (1) thermal, largely cooking demand and (2) electricity demand for lighting and appliance use. Several alternative fuel and technology options can be specified in the model to meet each of these respective service demands. To reflect heterogeneity among consumers, the household or residential sector is further disaggregated into several sub-groups that distinguish among rural and urban households and five or more expenditure classes within the rural and urban sub-sectors (Fig. 8.3).

The methodology for modeling energy choices in the residential sector of this model is described in detail in Ekholm et al. (2010) [11] and in the Supplementary Materials section of Pachauri et al. (2013) [69]. In addition to energy prices, technology costs and performance parameters, and income level of a household determining the least-cost energy-equipment combination that meets a specific energy need, two additional parameters determine choices in the model. The first is referred to as the "inconvenience cost". An inconvenience cost is a cost related to the inconveniences associated with obtaining and using certain types of fuels. For example, gathering firewood involves an opportunity cost for the time spent in collecting it and a dis-utility to users from exposure to the smoke they inhale when it is combusted. This non-monetary cost is captured by estimating an inconvenience cost (see Ekholm et al. (2010) [11] for further details regarding the methodology) for each household group and fuel. This is considered an additional cost that must be taken into account by the household in making a decision regarding the choice of fuels. The second parameter that also determines energy choices for households is income dependent implicit discount rates that determine the annualized capital costs of equipment depending on their individual lifetimes.

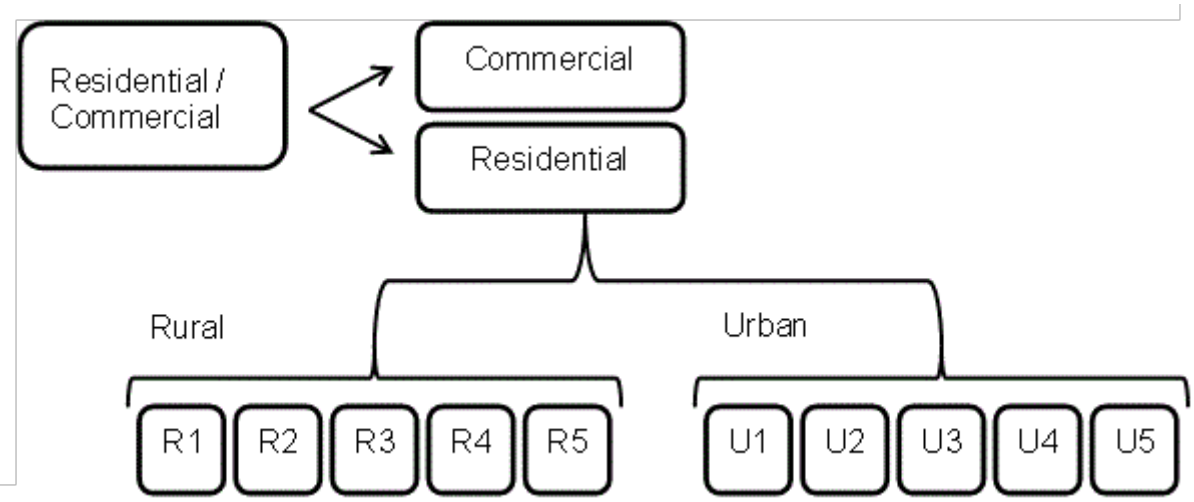

Fig. 8.3: Split of residential energy demand into different spatial (urban/rural) and income (1-5) categories.

SSP narratives

Narratives have been developed for the Shared Socioeconomic Pathways (SSPs) (O'Neill et al., 2015 [65]). These descriptions of alternative futures of societal development span a range of possible worlds that stretch along two climate-change-related dimensions: mitigation and adaptation challenges. The SSPs reflect five different developments of the world that are characterized by varying levels of global challenges (see Riahi et al., 2017 [89] for an overview). In the following, the three narratives that have been translated into quantitative scenarios with MESSAGE-GLOBIOM are presented (Fricko et al., 2017 [17]):

SSP1 Narrative: Sustainability — Taking the green road

"The world shifts gradually, but pervasively, toward a more sustainable path, emphasizing more inclusive development that respects perceived environmental boundaries. Increasing evidence of and accounting for the social, cultural, and economic costs of environmental degradation and inequality drive this shift. Management of the global commons slowly improves, facilitated by increasingly effective and persistent cooperation and collaboration of local, national, and international organizations and institutions, the private sector, and civil society. Educational and health investments accelerate the demographic transition, leading to a relatively low population. Beginning with current high-income countries, the emphasis on economic growth shifts toward a broader emphasis on human well-being, even at the expense of somewhat slower economic growth over the longer term. Driven by an increasing commitment to achieving development goals, inequality is reduced both across and within countries. Investment in environmental technology and changes in tax structures lead to improved resource efficiency, reducing overall energy and resource use and improving environmental conditions over the longer term. Increased investment, financial incentives and changing perceptions make renewable energy more attractive. Consumption is oriented toward low material growth and lower resource and energy intensity. The combination of directed development of environmentally friendly technologies, a favorable outlook for renewable energy, institutions that can facilitate international cooperation, and relatively low energy demand results in relatively low challenges to mitigation. At the same time, the improvements in human well-being, along with strong and flexible global, regional, and national institutions imply low challenges to adaptation." (O'Neill et al., 2015 [65])

SSP2 Narrative: Middle of the Road

"The world follows a path in which social, economic, and technological trends do not shift markedly from historical patterns. Development and income growth proceed unevenly, with some countries making relatively good progress while others fall short of expectations. Most economies are politically stable. Globally connected markets function imperfectly. Global and national institutions work toward but make slow progress in achieving sustainable development goals, including improved living conditions and access to education, safe water, and health care. Technological development proceeds apace, but without fundamental breakthroughs. Environmental systems experience degradation, although there are some improvements and overall the intensity of resource and energy use declines. Even though fossil fuel dependency decreases slowly, there is no reluctance to use unconventional fossil resources. Global population growth is moderate and levels off in the second half of the century as a consequence of completion of the demographic transition. However, education investments are not high enough to accelerate the transition to low fertility rates in low-income countries and to rapidly slow population growth. This growth, along with income inequality that persists or improves only slowly, continuing societal stratification, and limited social cohesion, maintain challenges to reducing vulnerability to societal and environmental changes and constrain significant advances in sustainable development. These moderate

SSP3 Narrative: Regional rivalry — A rocky road

"A resurgent nationalism, concerns about competitiveness and security, and regional conflicts push countries to increasingly focus on domestic or, at most, regional issues. This trend is reinforced by the limited number of comparatively weak global institutions, with uneven coordination and cooperation for addressing environmental and other global concerns. Policies shift over time to become increasingly oriented toward national and regional security issues, including barriers to trade, particularly in the energy resource and agricultural markets. Countries focus on achieving energy and food security goals within their own regions at the expense of broader-based development, and in several regions move toward more authoritarian forms of government with highly regulated economies. Investments in education and technological development decline. Economic development is slow, consumption is material-intensive, and inequalities persist or worsen over time, especially in developing countries. There are pockets of extreme poverty alongside pockets of moderate wealth, with many countries struggling to maintain living standards and provide access to safe water, improved sanitation, and health care for disadvantaged populations. A low international priority for addressing environmental concerns leads to strong environmental degradation in some regions. The combination of impeded development and limited environmental concern results in poor progress toward sustainability. Population growth is low in industrialized and high in developing countries. Growing resource intensity and fossil fuel dependency along with difficulty in achieving international cooperation and slow technological change imply high challenges to mitigation. The limited progress on human development, slow income growth, and lack of effective institutions, especially those that can act across regions, implies high challenges to adaptation for many groups in all regions." (O'Neill et al., 2015 [65])

Population and GDP

Population and economic developments have strong implications for the anticipated mitigation and adaptation challenges. For example, a larger, poorer and less educated population will have more difficulties to adapt to the detrimental effects of climate change (O'Neill et al., 2014 [66]). The primary drivers of future energy demand in MESSAGEix are projections of total population and GDP at purchasing power parity exchange rates, denoted as GDP (PPP). In addition to total population, the urban/rural split of population is relevant for the MESSAGEix-Access version of the model which distinguishes rural and urban population with different household incomes in developing country regions.

Understanding how population and economic growth develops in the SSPs gives a first layer of understanding of the multiple mitigation and adaptation challenges. Population growth evolves in response to how fertility, mortality, migration, and education of various social strata are assumed to change over time. In SSP2, global population peaks at 9.4 billion people around 2070, and slowly declines thereafter (KC and Lutz, 2015 [40]). Gross Domestic Product (GDP) follows regional historical trends (Dellink et al., 2015 [9]). In SSP2, average income grows by a factor of six and reaches about 60,000 USD/capita by the end of the century (all GDP/capita figures use USD2005 and purchasing-power-parity – PPP). The SSP2 GDP projection is situated in-between the estimates for SSP1 and SSP3, which reach global average income levels of 82,000 USD2005 and 22,000 USD2005, respectively, by the end of the century. SSP2 depicts a future of global progress where developing countries achieve significant economic growth. Today, average per capita income in the global North is about five times higher than in the global South. In SSP2, developing countries reach today's average income levels of the OECD between 2060 and 2090, depending on the region. However, modest improvements of educational attainment levels result in declines in education-specific fertility rates, leading to incomplete economic convergence across different world regions. This is particularly an issue for Africa. Overall, both the population and GDP developments in SSP2 (Fricko et al., 2016 []) are designed to be situated in the middle of the road between SSP1 and SSP3, see KC and Lutz (2015) [40] and Dellink et al (2015) [9] for details.

The full quantitative data set of demographic and economic projections for the SSPs can be found in an online database (SSP database).

8.14.3 Energy (MESSAGE*ix*)

The |MESSAGEix| modeling framework, briefly known also as MESSAGE (Model for Energy Supply Strategy Alternatives and their General Environmental Impact), is a linear programming (LP) energy engineering model with global coverage. As a systems engineering optimization model, MESSAGE*ix* is primarily used for medium- to long-term energy system planning, energy policy analysis, and scenario development (Huppmann et al., 2019 [31]; Messner and Strubegger, 1995 [62]). The model provides a framework for representing an energy system with all its interdependencies from resource extraction, imports and exports, conversion, transport, and distribution, to the provision of energy end-use services such as light, space conditioning, industrial production processes, and transportation. In addition, MESSAGE*ix* links to GLOBIOM (GLObal BIOsphere Model, cf. Section *Land-use (GLOBIOM)*) to consistently assess the implications of utilizing bioenergy of different types and to integrate the GHG emissions from energy and land use and to the aggregated macro-economic model MACRO (cf. Section *Macro-economy (MACRO)*) to assess economic implications and to capture economic feedbacks.

MESSAGE*ix* covers all greenhouse gas (GHG)-emitting sectors, including energy, industrial processes as well as - through its linkage to GLOBIOM - agriculture and forestry. The emissions of the full basket of greenhouse gases including CO2, CH4, N2O and F-gases (CF4, C2F6, HFC125, HFC134a, HFC143a, HFC227ea, HFC245ca and SF6) as well as other radiatively active gases, such as NOx, volatile organic compounds (VOCs), CO, SO2, and BC/OC is represented in hte model. MESSAGE is used in conjunction with MAGICC (Model for Greenhouse gas Induced Climate Change) version 6.8 (cf. Section *Climate (MAGICC)*) for calculating atmospheric concentrations, radiative forcing, and annual-mean global surface air temperature increase.

The model is designed to formulate and evaluate alternative energy supply strategies consonant with the user-defined constraints such as limits on new investment, fuel availability and trade, environmental regulations and policies as well as diffusion rates of new technologies. Environmental aspects can be analysed by accounting, and if necessary limiting, the amounts of pollutants emitted by various technologies at various steps in energy supplies. This helps to evaluate the impact of environmental regulations on energy system development.

Its principal results comprise, among others, estimates of technology-specific multi-sector response strategies for specific climate stabilization targets. By doing so, the model identifies the least-cost portfolio of mitigation technologies. The choice of the individual mitigation options across gases and sectors is driven by the relative economics of the abatement measures, assuming full temporal and spatial flexibility (i.e., emissions-reduction measures are assumed to occur when and where they are cheapest to implement).

The Reference Energy System (RES) defines the full set of available energy conversion technologies. In MESSAGE*ix* terms, energy conversion technology refers to all types of energy technologies from resource extraction to transformation, transport, distribution of energy carriers, and end-use technologies.

Because few conversion technologies convert resources directly into useful energy, the energy system in MESSAGE*ix* is divided into 5 energy levels:

- Resources: raw resources (e.g., coal, oil, natural gas in the ground or biomass on the field)
- Primary energy: raw product at a generation site (e.g., crude oil input to the refinery)
- Secondary energy: finalized product at a generation site (e.g., gasoline or diesel fuel output from the refinery)
- Final energy: finalized product at its consumption point (e.g., gasoline in the tank of a car or electricity leaving a socket)
- Useful energy: finalized product satisfying demand for services (e.g., heating, lighting or moving people)

Technologies can take in energy commodities from one level and put out at another level (e.g., refineries produce refined oil products at secondary level from crude oil at the primary level) or at the same level (e.g., hydrogen electrolyzers produce hydrogen at the secondary energy level from electricity at the secondary level). The energy forms defined in each level can be envisioned as a transfer hub, that the various technologies feed into or pump away from. The useful energy demand is given as a time series. Technology characteristics generally vary over time period.

The mathematical formulation of MESSAGE*ix* ensures that the flows are consistent: demand is met, inflows equal outflows and constraints are not exceeded. In other words, MESSAGE*ix* itself is a partial equilibrium model. However, through its linkage to MACRO general equilibrium effects are taken into account (cf. Section *Macro-economy (MACRO)*).

Energy resource endowments

Fossil Fuel Reserves and Resources

The availability and costs of fossil fuels influences the future development of the energy system, and therewith future mitigation challenges. Understanding the variations in fossil fuel availability and the underlying extraction cost assumptions across the SSPs is hence important. Our fossil energy resource assumptions in MESSAGE are derived from various sources, including global databases such as The Federal Institute for Geosciences and Natural Resources (BGR) and The U.S. Geological Survey (USGS), as well as market reports and outlooks provided by different energy institutes and agencies. The availability of fossil energy resources in different regions under different socio-economic assumptions are then aligned with the storylines of the individual SSPs (Rogner, 1997 [96]; Riahi et al., 2012 [85]). While the physical resource base is identical across the SSPs, considerable differences are assumed regarding the technical and economic availability of overall resources, for example, of unconventional oil and gas.

What ultimately determines the attractiveness of a particular type of resource is not just the cost at which it can be brought to the surface, but the cost at which it can be used to provide energy services. Assumptions on fossil energy resources should thus be considered together with those on related conversion technologies. In line with the narratives, technological change in fossil fuel extraction and conversion technologies is assumed to be slowest in SSP1, while comparatively faster technological change occurs in SSP3 thereby considerably enlarging the economic potentials of coal and unconventional hydrocarbons (Table 8.3, Fig. 8.4). However, driven by the tendency toward regional fragmentation, the focus in SSP3 is assumed to be on developing coal technologies which in the longer term leads to a replacement of oil products by synthetic fuels based on coal-to-liquids technologies. In contrast, for SSP2 we assume a continuation of recent trends, focusing more on developing extraction technologies for unconventional hydrocarbon resources, thereby leading to higher potential cumulative oil extraction than in the other SSPs (Fig. 8.4, the middle panel).

Table 8.3 shows the assumed total quantities of fossil fuel resources in the MESSAGE model for 2005. Fig. 8.4 gives these resource estimates as cumulative resource supply curves. In addition, the assumptions are compared with estimates from the Global Energy Assessment (Rogner et al., 2012 [95]) and the databases mentioned earlier. Estimating fossil fuel reserves is built on both economic and technological assumptions. With an improvement in technology or a change in purchasing power, the amount that may be considered a “reserve” vs. a “resource” (generically referred to here as resources) can actually vary quite widely.

‘Reserves’ are generally defined as being those quantities for which geological and engineering information indicate with reasonable certainty that they can be recovered in the future from known reservoirs under existing economic and operating conditions. ‘Resources’ are detected quantities that cannot be profitably recovered with the current technology, but might be recoverable in the future, as well as those quantities that are geologically possible, but yet to be found. The remainder are ‘Undiscovered resources’ and, by definition, one can only speculate on their existence. Definitions are based on Rogner et al. (2012) [95].

Table 8.3: Assumed global fossil fuel reserves and resources in the MESSAGE model. Estimates from the Global Energy Assessment (Rogner et al., 2012 [95]) also added for comparison.

Source	MESSAGE (Rogner et al., 1997 [96]) Reserves+Resources [ZJ]	Rogner et al., 2012 [95] Reserves [ZJ]	Rogner et al., 2012 [95] Resources [ZJ]
Coal	259	17.3 – 21.0	291 – 435
Conventional Oil	9.8	4.0 – 7.6	4.2 – 6.2
Unconventional Oil	23.0	3.8 – 5.6	11.3 – 14.9
Conventional Gas	16.8	5.0 – 7.1	7.2 – 8.9
Unconventional Gas	23.0	20.1 – 67.1	40.2 – 122

The following table (Table 8.4) presents the ultimate fossil resource availability for coal, oil and gas, for SSP1, SSP2 and SSP3, respectively.

Table 8.4: Fossil resource availability for SSP1, SSP2, and SSP3 (Fricko et al., 2017 [17]).

Type	SSP1 [ZJ]	SSP2 [ZJ]	SSP3 [ZJ]
Coal	93	92	243
Oil	17	40	17
Gas	39	37	24

Coal is the largest resource among fossil fuels; it accounts for more than 50% of total fossil reserve plus resource estimates even at the higher end of the assumptions, which includes considerable amounts of unconventional hydrocarbons. Oil is the fastest depleting fossil fuel with less than 10 ZJ of conventional oil and possibly less than 10 ZJ of unconventional oil. Natural gas is more abundant in both the conventional and unconventional categories.

Fig. 8.4 presents the cumulative global resource supply curves for coal, oil and gas in the IIASA IAM framework. Green shaded resources are technically and economically extractable in all SSPs, purple shaded resources are additionally available in SSP1 and SSP2 and blue shaded resources are additionally available in SSP2. Coloured vertical lines represent the cumulative use of each resource between 2010 and 2100 in the SSP baselines (see the top panel for colour coding), and are thus the result of the combined effect of the assumptions on fossil resource availability and conversion technologies in the SSP baseline scenarios.

Conventional oil and gas are distributed unevenly throughout the world, with only a few regions dominating the reserves. Nearly half of the reserves of conventional oil is found in Middle East and North Africa, and close to 40% of conventional gas is found in Russia and the Former Soviet Union states. The situation is somewhat different for unconventional oil of which North and Latin America potentially possess significantly higher global shares. Unconventional gas in turn is distributed quite evenly throughout the world, with North America holding most (roughly 25% of global resources). The distribution of coal reserves shows the highest geographical diversity which in the more fragmented SSP3 world contributes to increased overall reliance on this resource. Russia and the former Soviet Union states, Pacific OECD, North America, and Centrally Planned Asia and China all possess more than 10 ZJ of reserves.

Nuclear Resources

Estimates of available uranium resources in the literature vary considerably, which could become relevant if advanced nuclear fuel cycles (e.g., the plutonium cycle including fast breeder reactors, the thorium cycle) are not available. In MESSAGE advanced nuclear cycles such as the plutonium cycle and nuclear fuel reprocessing are in principle represented, but their availability varies following the scenario narrative. Fig. 8.5 below shows the levels of uranium resources assumed available in the MESSAGE SSP scenarios, building upon earlier work developed in the Global Energy Assessment (see Riahi et al., 2012 [85]). These span a considerable range of the estimates in the literature, but at the same time none of them fall at the extreme ends of the spectrum (see Rogner et al., 2012 [95], Section 7.5.2 for a more detailed discussion of uranium resources). Nuclear resources and fuel cycle are modeled at the global level.

Non-Biomass Renewable Resources

Table 8.5 shows the assumed total potentials of non-biomass renewable energy deployment (by resource type) in the MESSAGE model. In addition, the technical potential estimates are based on different sources, such as the U.S. National Renewable Energy Laboratory database as described in the Global Energy Assessment (Rogner et al., 2012 [95]). In this context, it is important to note that typical MESSAGE scenarios do not consider the full technical potential of renewable energy resources, but rather only a subset of those potentials, owing to additional constraints (e.g., sustainability criteria, technology diffusion and systems integration issues, and other economic considerations). These constraints may lead to a significant reduction of the technical potential.

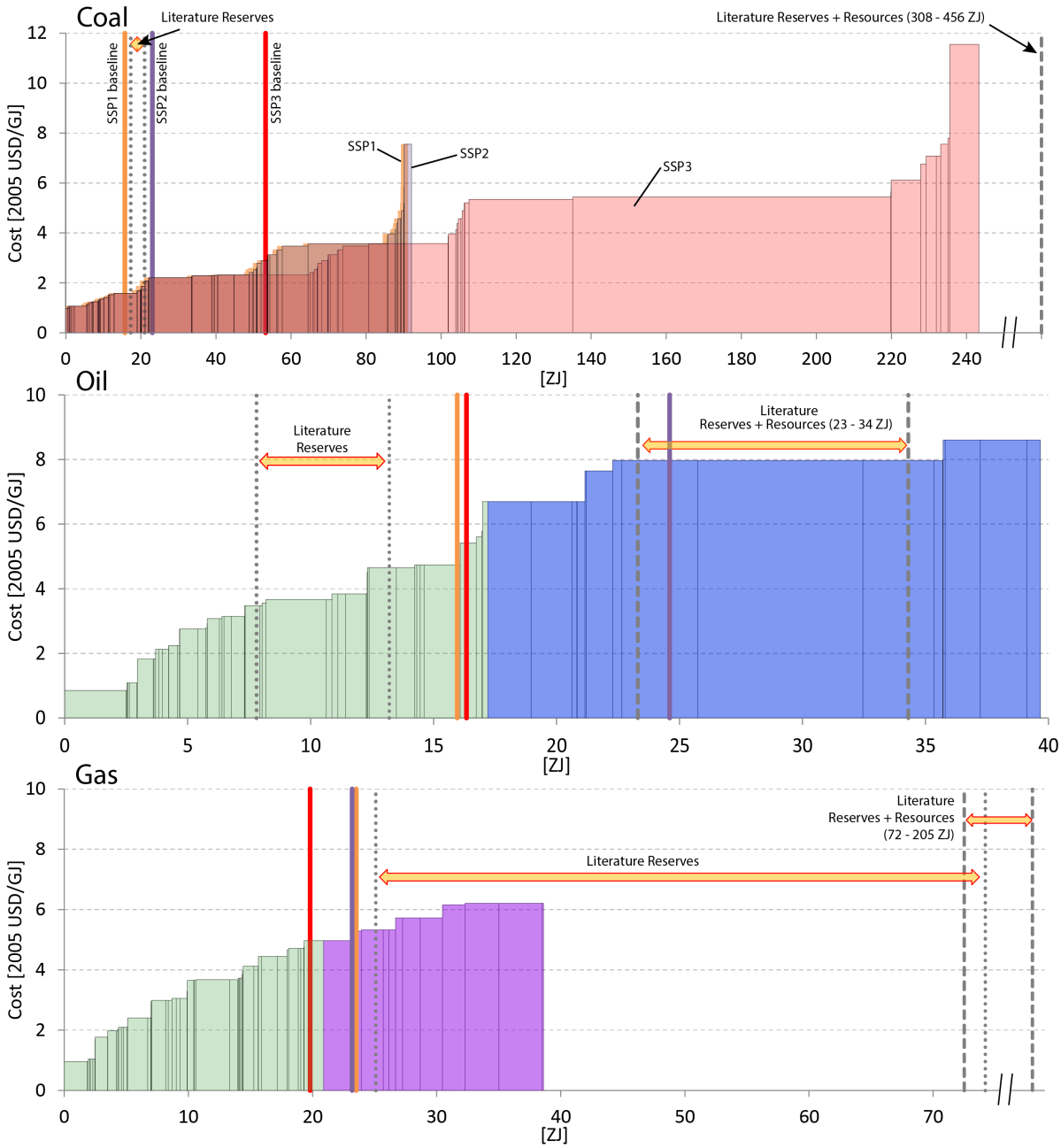

Fig. 8.4: Cumulative global resource supply curves for coal (top), oil (middle), and gas (bottom) in the IIASA IAM framework (Fricko et al., 2017 [17]).

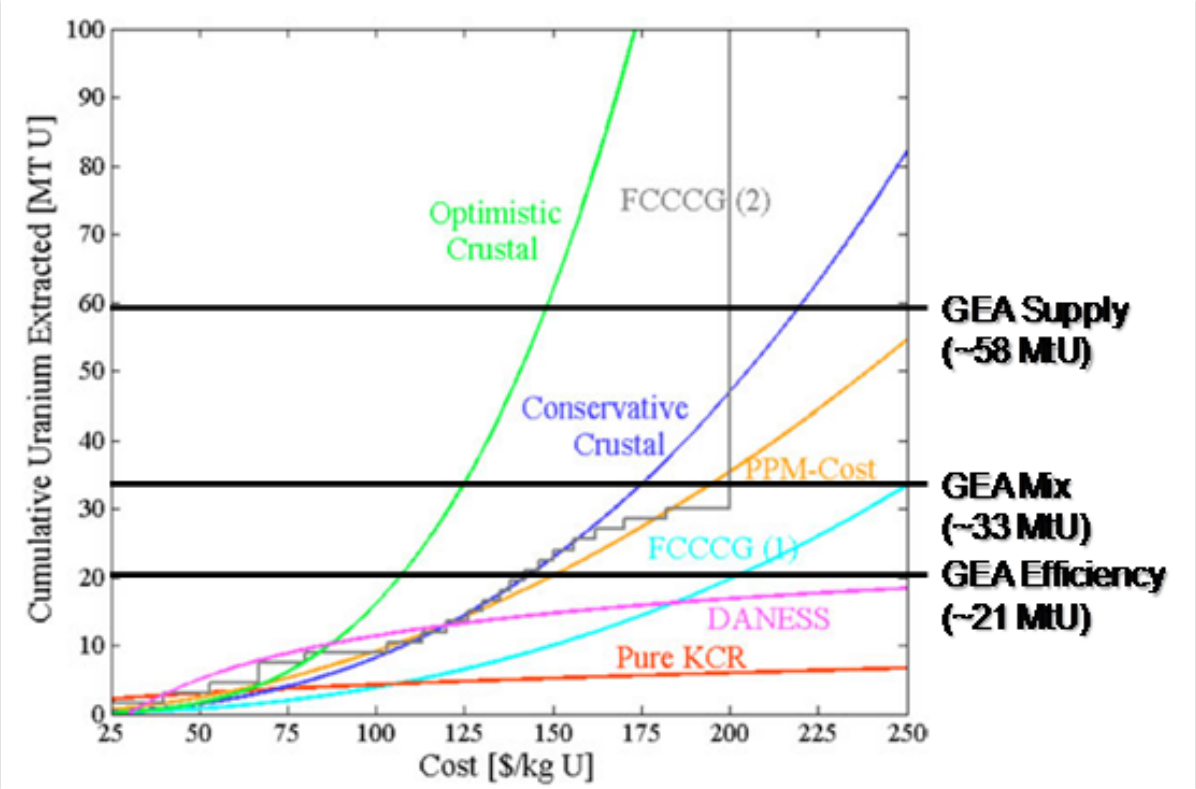

Fig. 8.5: Global uranium resources in the MESSAGE interpretation of the SSPs compared to seven supply curves from a literature review (Schneider and Sailor, 2008 [101]). Conservative Crustal and Optimistic Crustal refer to simple crustal models of uranium distribution in the crust and the of extraction costs on the concentration. Pure-KCR refers to a fit of a simple crustal model to known conventional resources (KCR) as estimated by the Red Book 2003 (OECD/NEA, 2004 [117]). PPM-Cost over the simple crustal models include a relationship between uranium grade and extraction costs. FCCCG(1) and (2) as well as DANESS refer to estimats from more complicated models of the dependency of extraction costs on uranium concentration (and therefore resource grade).

Table 8.5: Assumed global non-biomass renewable energy deployment potentials in the MESSAGE model. Estimates from the Global Energy Assessment (Rogner et al., 2012 [95]) also added for comparison.

Source	MESSAGE Deployment Potential [EJ/yr]	Rogner et al., 2012 [95] Technical Potential [EJ/yr]
Hydro	38	50 - 60
Wind (on-/offshore)	689/287	1250 - 2250
Solar PV	6064	62,000 - 280,000
CSP	2132	same as Solar PV above
Geothermal	23	810 - 1400

Notes: MESSAGE renewable energy potentials are estimated based on the methods explained in Pietzcker et al., 2014 [74], Eurek et al., 2017 [13], Christiansson, 1995 [6], *and* Rogner et al., 2012 [95]. *The potentials for non-combustible renewable energy sources are specified in terms of the electricity or heat that can be produced by specific technologies (i.e., from a secondary energy perspective). By contrast, the technical potentials from* [95] *refer to the flows of energy that could become available as inputs for technology conversion. So for example, the technical potential for wind is given as the kinetic energy available for wind power generation, whereas the deployment potential would only be the electricity that could be generated by the wind turbines.*

Regional resource potentials for solar and wind are classified according to resource quality (annual capacity factor) based on Pietzcker et al. (2014, [74]) and Eurek et al. (in review, []). Regional resource potentials as implemented into MESSAGE are provided by region and capacity factor for solar PV, concentrating solar power (CSP), and onshore/offshore wind in Johnson et al. (2016, [38]). The physical potential of these sources is assumed to be the same across all SSPs. Table 8.6, Table 8.7, Table 8.8, Table 8.10 show the resource potential for solar PV, CSP (solar multiples (SM) of 1 & 3), on- and offshore wind respectivey. For wind, Table 8.9 and Table 8.11 list the capacity factors corresponding to the wind classes used in the resource tables. It is important to note that part of the resource that is useable at economically competitive costs is assumed to differ widely (see Section *Electricity*).

Table 8.6: Resource potential (EJ) by region and capacity factor for solar photovoltaic (PV) technology (Johnson et al., 2016 [38]). For a description of each of the regions represented in the table, see Regions.

		By grade							
Capacity Factor (fraction of year)		**0.28**	**0.21**	**0.20**	**0.19**	**0.18**	**0.17**	**0.15**	**0.14**
Resource Potential (EJ)	AFR	0.0	1.1	46.5	176.6	233.4	218.2	169.9	61.9
	CPA	0.0	0.0	0.0	10.3	194.3	315.5	159.4	41.9
	EEU	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.0
	FSU	0.0	0.0	0.0	0.2	2.8	23.6	94.9	116.6
	LAM	0.1	4.9	49.4	165.6	157.5	167.4	81.4	48.5
	MEA	0.2	3.1	100.8	533.6	621.8	310.1	75.3	14.5
	NAM	0.0	0.3	24.3	140.4	131.0	116.3	155.7	106.4
	PAO	0.0	0.0	0.1	2.2	53.1	226.4	311.2	158.9
	PAS	0.0	0.0	0.0	0.2	0.8	17.0	31.2	12.8
	SAS	0.0	0.0	6.1	42.7	67.2	82.3	23.7	4.1
	WEU	0.0	0.1	0.2	3.0	12.8	39.4	58.3	33.3
	Global	0.3	9.6	227.4	1074.7	1474.6	1516.3	1160.9	600.0

Table 8.7: Resource potential (EJ) by region and capacity factor for concentrating solar power (CSP) technologies with solar multiples (SM) of 1 and 3 (Johnson et al., 2016 [38]).

		By grade							
Capacity Factor (fraction of year)	**SM1**	**0.27**	**0.25**	**0.23**	**0.22**	**0.20**	**0.18**	**0.17**	**0.15**
	SM3	**0.75**	**0.68**	**0.64**	**0.59**	**0.55**	**0.50**	**0.46**	**0.41**
Resource Potential (EJ)	AFR	0.0	3.6	19.0	81.6	106.7	62.8	59.6	37.8
	CPA	0.0	0.0	0.0	0.0	0.0	0.3	11.5	53.0
	EEU	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	FSU	0.0	0.0	0.0	0.0	0.0	0.1	0.4	6.1
	LAM	0.0	2.0	7.0	11.8	29.3	57.1	56.8	53.5
	MEA	0.1	3.7	24.8	122.4	155.3	144.5	68.4	34.0
	NAM	0.0	0.0	0.0	6.3	19.7	20.2	29.6	43.2
	PAO	0.0	3.0	75.1	326.9	158.3	140.4	40.2	10.2
	PAS	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.6
	SAS	0.0	0.0	0.0	0.1	3.9	8.7	16.1	9.8
	WEU	0.0	0.0	0.0	0.0	0.2	0.7	2.4	3.0
	Global	0.1	12.3	126.0	549.2	473.3	434.8	285.0	251.3

Table 8.8: Resource potential (EJ) by region and wind class for onshore wind (Johnson et al., 2016 [38]).

	Wind Class					
	3	**4**	**5**	**6**	**7**	**8+**
AFR	38.2	21.3	13.4	6.8	2.6	2.1
CPA	24.7	11.4	5.4	2.6	0.3	0.0
EEU	6.1	5.7	0.3	0.0	0.0	0.0
FSU	52.3	83.8	5.8	0.8	0.0	0.0
LAM	33.5	15.9	9.6	5.7	3.9	3.7
MEA	56.1	22.2	6.0	2.1	0.9	0.3
NAM	28.6	66.4	23.7	1.5	0.4	0.0
PAO	18.9	18.8	3.6	1.4	1.8	0.5
PAS	5.2	2.9	0.8	0.2	0.0	0.0
SAS	12.3	7.9	2.4	1.6	0.9	0.3
WEU	16.1	10.5	6.6	8.2	3.7	0.6
World	292.1	266.8	77.5	30.9	14.3	7.5

Table 8.9: Capacity factor by region and wind class for onshore wind (Johnson et al., 2016 [38]).

	Wind Class					
	3	**4**	**5**	**6**	**7**	**8+**
AFR	0.24	0.28	0.32	0.36	0.40	0.45
CPA	0.24	0.28	0.32	0.36	0.38	0.45
EEU	0.24	0.27	0.31	0.36	0.38	0.45
FSU	0.24	0.28	0.31	0.35	0.38	0.45
LAM	0.24	0.28	0.32	0.36	0.39	0.46
MEA	0.24	0.27	0.32	0.35	0.39	0.45
NAM	0.24	0.28	0.31	0.36	0.39	0.45
PAO	0.24	0.28	0.32	0.36	0.40	0.43
PAS	0.24	0.27	0.32	0.35	0.40	0.45
SAS	0.24	0.27	0.32	0.36	0.39	0.42
WEU	0.24	0.28	0.32	0.36	0.39	0.43

Table 8.10: Resource potential (EJ) by region and wind class for offshore wind (Johnson et al., 2016 [38]).

	Wind Class					
	3	**4**	**5**	**6**	**7**	**8+**
AFR	3.1	2.4	2.0	2.0	1.1	1.7
CPA	3.5	4.3	2.6	0.9	1.3	0.1
EEU	0.7	0.6	1.0	0.0	0.0	0.0
FSU	1.8	4.6	14.2	13.3	4.3	0.7
LAM	7.1	7.3	5.3	2.7	2.6	5.9
MEA	3.2	0.9	0.8	0.9	0.6	0.9
NAM	4.5	18.2	24.0	16.0	7.3	2.1
PAO	5.8	11.2	15.3	9.8	2.6	2.5
PAS	5.3	6.6	4.7	1.5	0.1	0.0
SAS	1.9	0.9	0.6	0.5	0.0	0.0
WEU	3.5	4.7	8.8	12.9	10.3	0.9
World	40.4	61.5	79.4	60.5	30.3	14.8

Table 8.11: Capacity factor by region and wind class for offshore wind (Johnson et al., 2016 [38]).

	Wind class					
	3	**4**	**5**	**6**	**7**	**8+**
AFR	0.24	0.28	0.32	0.36	0.41	0.47
CPA	0.24	0.28	0.32	0.36	0.40	0.42
EEU	0.24	0.29	0.32	0.34	0.40	0.42
FSU	0.25	0.28	0.32	0.35	0.39	0.43
LAM	0.24	0.28	0.32	0.36	0.40	0.49
MEA	0.24	0.28	0.32	0.36	0.40	0.45
NAM	0.25	0.28	0.32	0.36	0.40	0.43
PAO	0.24	0.28	0.32	0.36	0.40	0.47
PAS	0.24	0.28	0.32	0.35	0.39	0.42
SAS	0.24	0.27	0.32	0.36	0.40	0.42
WEU	0.24	0.28	0.32	0.36	0.40	0.42

Biomass Resources

Biomass energy is another potentially important renewable energy resource in the MESSAGE model. This includes both commercial and non-commercial use. Commercial refers to the use of bioenergy in, for example, power plants or biofuel refineries, while non-commercial refers to the use of bioenergy for residential heating and cooking, primarily in rural households of today's developing countries. Bioenergy potentials are derived from the GLOBIOM model and differ across SSPs as a result of different levels of competition over land for food and fibre, but ultimately only vary to a limited degree (Fig. 8.6). The drivers underlying this competition are different land-use developments in the SSPs, which are determined by agricultural productivity and global demand for food consumption. (Fricko et al., 2017 [17])

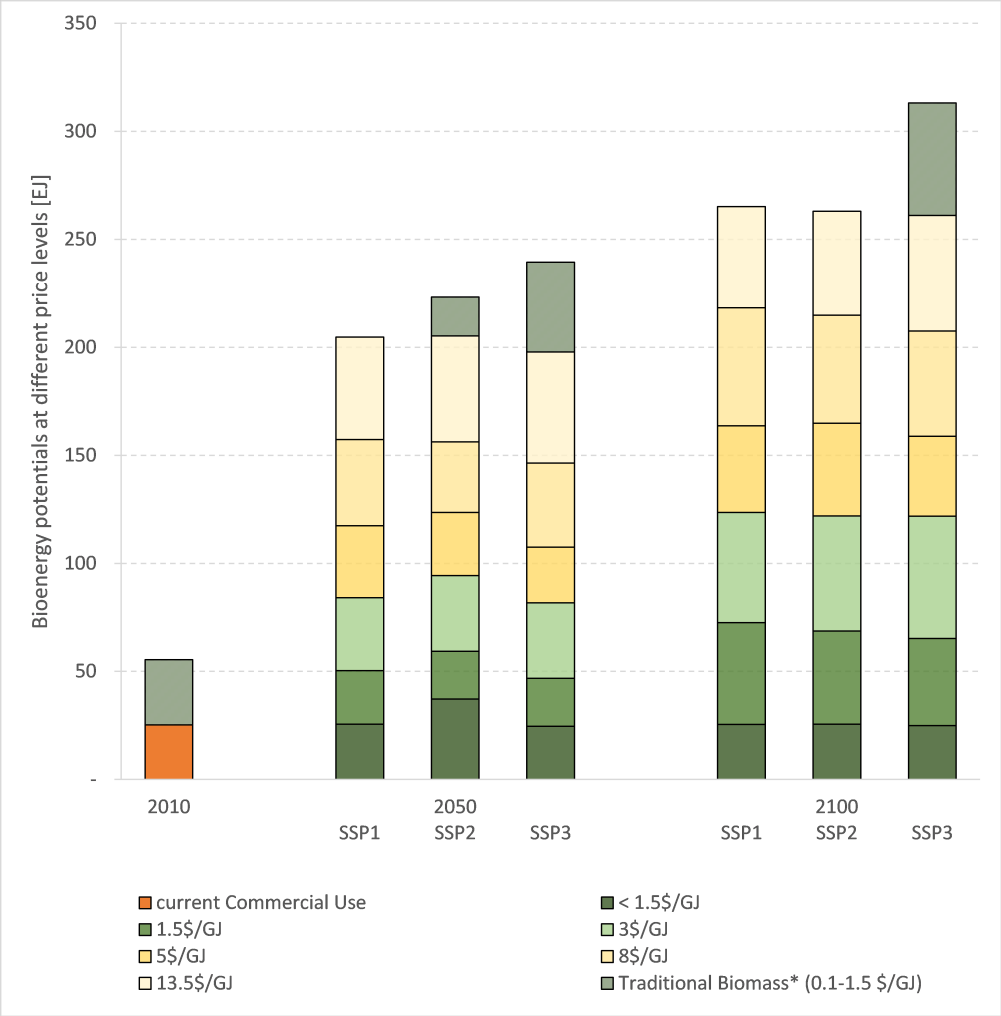

Fig. 8.6: Global bioenergy potential. Availability of bioenergy at different price levels in the MESSAGE-GLOBIOM model for the three SSPs (Fricko et al., 2017 [17]). Typically non-commercial biomass is not traded or sold, however in some cases there is a market – prices range from 0.1-1.5$/GJ (Pachauri et al., 2013 [69]) ($ equals 2005 USD).

Energy conversion

Energy technologies are characterized by numerical model inputs describing their economic (e.g., investment costs, fixed and variable operation and maintenance costs), technical (e.g., conversion efficiencies), ecological (e.g., GHG and air pollutant emissions), and sociopolitical characteristics. An example for the sociopolitical situation in a world region would be the decision by a country or world region to ban certain types of technologies (e.g., nuclear power plants). Model input data reflecting this situation would be constraining the use of these technologies or, equivalently, their omission from the data set for this region altogether.

Each energy conversion technology is characterized in MESSAGE by the following data:

- Energy inputs and outputs together with the respective conversion efficiencies. Most energy conversion technologies have one energy input and one output and thereby one associated efficiency. But technologies may also use different fuels (either jointly or alternatively), may have different operation modes and different outputs, which also may have varying shares. An example of different operation modes would be a passout turbine, which can generate electricity and heat at the same time when operated in co-generation mode or which can produce electricity only. For each technology, one output and one input are defined as main output and main input respectively. The activity variables of technologies are given in the units of the main input consumed by the technology or, if there is no explicit input (as for solar-energy conversion technologies), in units of the main output.
- Specific investment costs (e.g., per kilowatt, kW) and time of construction as well as distribution of capital costs over construction time.
- Fixed operating and maintenance costs (per unit of capacity, e.g., per kW).
- Variable operating costs (per unit of output, e.g. per kilowatt-hour, kWh, excluding fuel costs).
- Plant availability or maximum utilization time per year. This parameter also reflects maintenance periods and other technological limitations that prevent the continuous operation of the technology.
- Technical lifetime of the conversion technology in years.
- Year of first commercial availability and last year of commercial availability of the technology.
- Consumption or production of certain materials (e.g. emissions of kg of CO2 or SO2 per produced kWh).
- Limitations on the (annual) activity and on the installed capacity of a technology.
- Constraints on the rate of growth or decrease of the annually new installed capacity and on the growth or decrease of the activity of a technology.
- Technical application constraints, e.g., maximum possible shares of wind or solar power in an electricity network without storage capabilities.
- Inventory upon startup and shutdown, e.g., initial nuclear core needed at the startup of a nuclear power plant.
- Lag time between input and output of the technology.
- Minimum unit size, e.g. for nuclear power plants it does not make sense to build plants with a capacity of a few kilowatt power (optional, not used in current model version).
- Sociopolitical constraints, e.g., ban of nuclear power plants.
- Inconvenience costs which are specified only for end-use technologies (e.g. cook stoves)

The specific technologies represented in various parts of the energy conversion sector are discussed in the following sections on *Electricity*, *Heat*, *Other conversion* and *Grid, Infrastructure and System Reliability*.

Electricity

MESSAGE covers a large number of electricity generation options utilizing a wide range of primary energy sources. For fossil-based electricity generation technologies, typically a number of different technology variants with different efficiencies, environmental characteristics and costs are represented. For example, in the case of coal, MESSAGE distinguishes subcritical and supercritical pulverized coal (PC) power plants where the subcritical variant is available with and without flue gas desulpherization/denox and one internal gasification combined cycle (IGCC) power plant. The supercritical PC and IGCC plants are also available with carbon capture and storage (CCS) which also can be retrofitted to some of the existing PC power plants (see Fig. 8.7). Table 8.12 below shows the different power plant types represented in MESSAGE.

Four different nuclear power plant types are represented in MESSAGE, i.e. two light water reactor types, a fast breeder reactor and a high temperature reactor, but only the two light water types are included in the majority of scenarios being developed with MESSAGE in the recent past. In addition, MESSAGE includes a representation of the nuclear fuel cycle, including reprocessing and the plutonium fuel cycle, and keeps track of the amounts of nuclear waste being produced.

The conversion of five renewable energy sources to electricity is represented in MESSAGE (see Fig. 8.8). For wind power, both on- and offshore electricity generation are covered and for solar energy, photovoltaics (PV) and solar thermal (concentrating solar power, CSP) electricity generation are included in MESSAGE (see also sections on *Non-Biomass Renewable Resources* and *Systems Integration and Reliability*). Two CSP technologies are modeled: (1) a flexible plant with a solar multiple of one (SM1) and 6 h of thermal storage and (2) a baseload plant with a solar multiple of three (SM3) and 12 h of storage (Johnson et al. 2016, [38]).

Most thermal power plants offer the option of coupled heat production (CHP, see Table 8.12). This option is modeled as a passout turbine via a penalty on the electricity generation efficiency. In addition to the main electricity generation technologies described in this section, also the co-generation of electricity in conversion technologies primarily devoted to producing non-electric energy carriers (e.g., synthetic liquid fuels) is included in MESSAGE (see section on *Other conversion*).

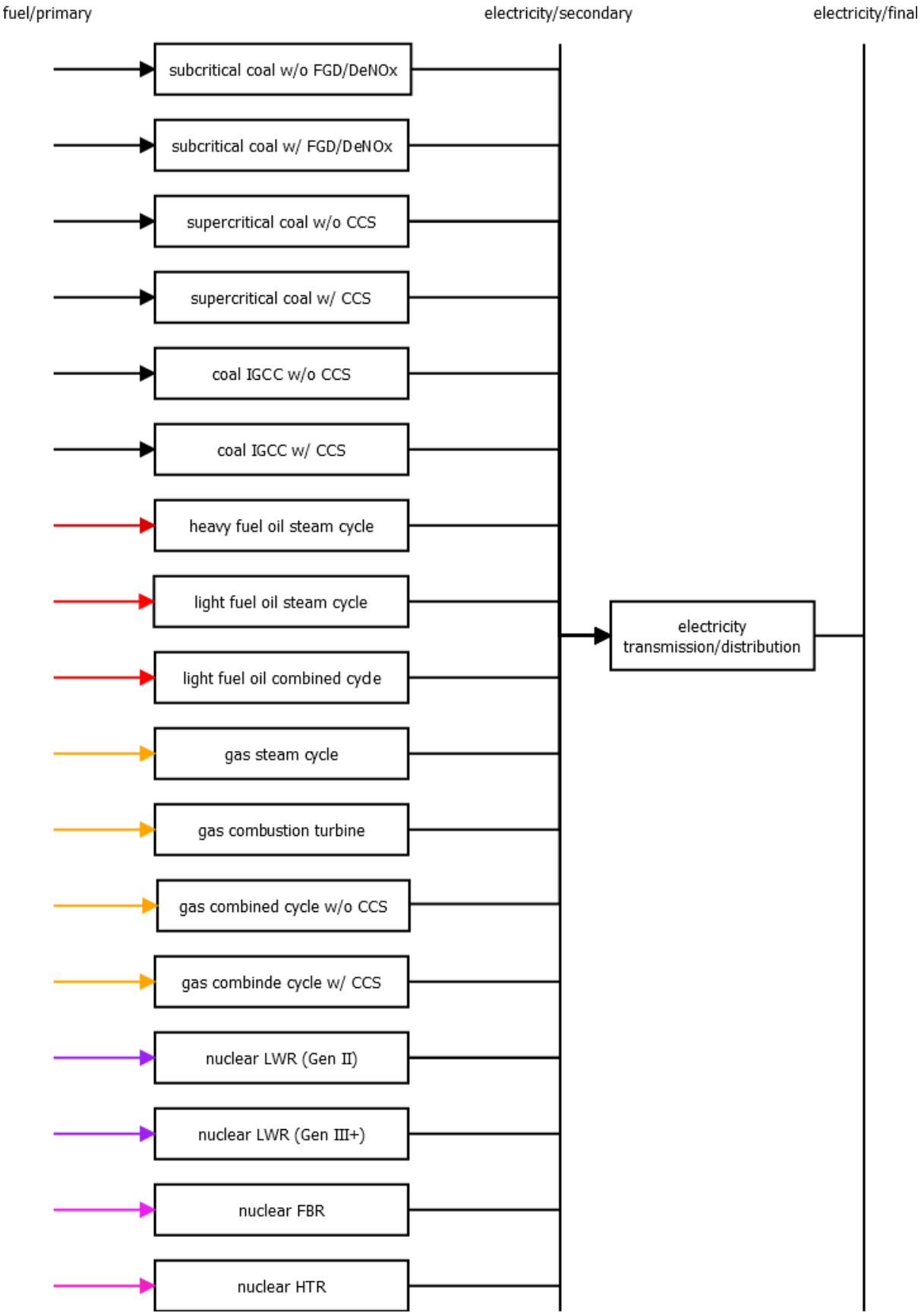

Fig. 8.7: Schematic diagram of the fossil and nuclear power plants represented in MESSAGEix.

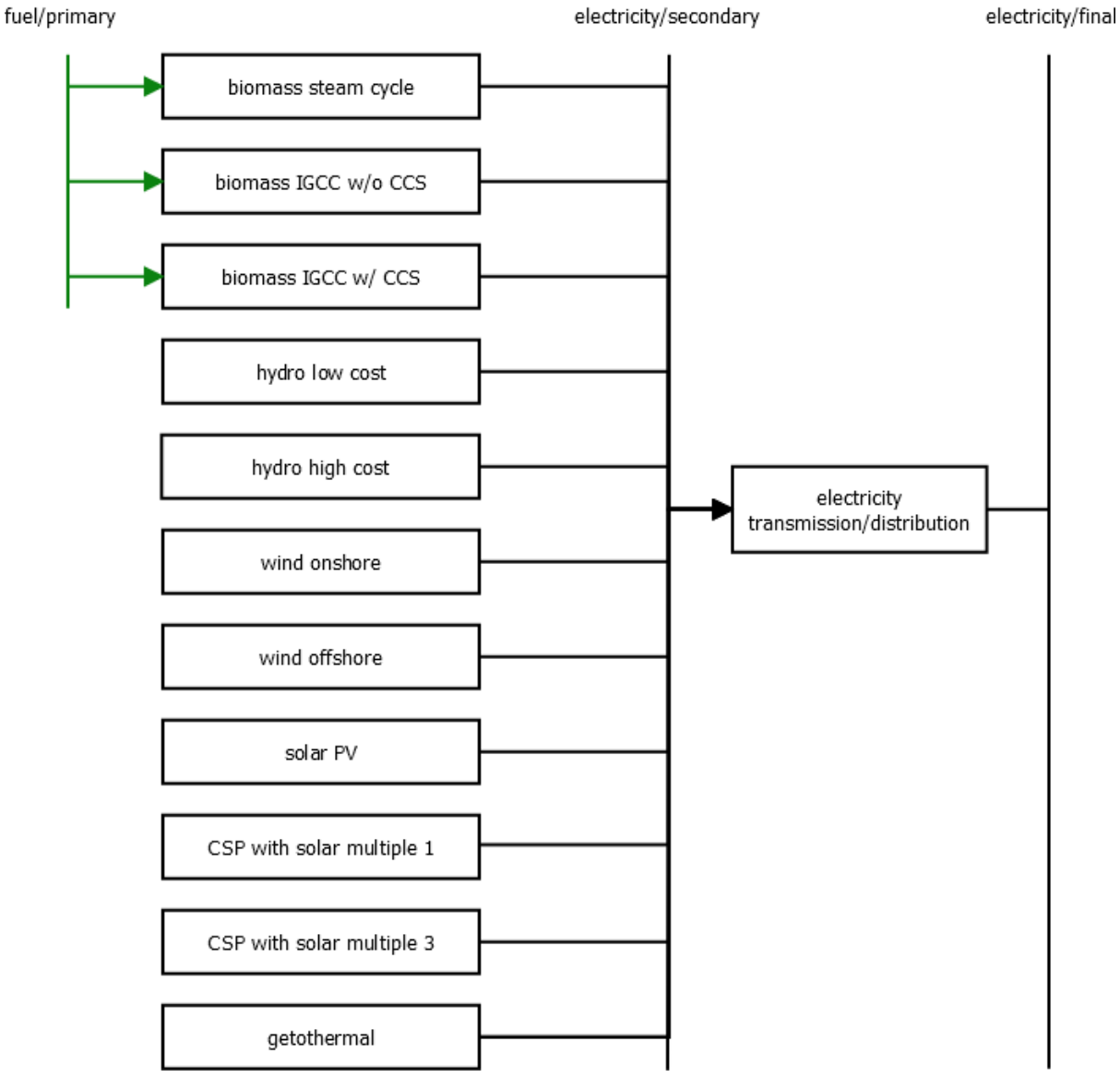

Fig. 8.8: Schematic diagram of the renewable power generation options represented in MESSAGEix.

Table 8.12: List of electricity generation technologies represented in MESSAGE-GLOBIOM by energy source.

Energy source	Technology	CHP option
coal	subcritical PC power plant without desulphurization/denox	yes
	subcritical PC power plant with desulphurization/denox	yes
	supercritical PC power plant with desulphurization/denox	yes
	supercritical PC power plant with desulphurization/denox and CCS	yes
	IGCC power plant	yes
	IGCC power plant with CCS	yes
oil	heavy fuel oil steam power plant	yes
	light fuel oil steam power plant	yes
	light fuel oil combined cycle power plant	yes
gas	gas steam power plant	yes
	gas combustion turbine gas	yes
	combined cycle power plant	yes
	combined cycle power plant with CCS	yes
nuclear	nuclear light water reactor (Gen II)	yes
	nuclear light water reactor (Gen III+)	yes
	fast breeder reactor	
	high temperature reactor	
biomass	biomass steam power plant	yes
	biomass IGCC power plant	yes
	biomass IGCC power plant with CCS	yes
hydro	hydro power plant (2 cost categories)	no
wind	onshore wind turbine	no
	offshore wind turbine	no
solar	solar photovoltaics (PV)	no
	concentrating solar power (CSP) with a solar multiple of 1 (SM1)	no
	concentrating solar power (CSP) with a solar multiple of 3 (SM3)	no
geothermal	geothermal power plant	yes

In Fig. 8.9, the black ranges show historical cost ranges for 2005. Green, blue, and red ranges show cost ranges in 2100 for SSP1, SSP2, and SSP3, respectively (see description of the *SSP narratives*). Global values are represented by solid ranges. Values in the global South are represented by dashed ranges. The diamonds show the costs in the "North America" region (Fricko et al., 2017 [17]).

In Fig. 8.10, the black ranges show historical cost ranges for 2005. Green, blue, and red ranges show cost ranges in 2100 for SSP1, SSP2, and SSP3, respectively. Global values are represented by solid ranges. Values in the global South are represented by dashed ranges. The diamonds show the costs in the "North America" region. PV – Photovoltaic (Fricko et al., 2017 [17]).

Heat

A number centralized district heating technologies based on fossil and renewable energy sources are represented in MESSAGE (see Table 8.13). Similar to coupled heat and power (CHP) technologies that are described in the *Electricity* sector, these heating plants feed low temperature heat into the district heating system that is then used in the end-use sectors. In addition, there are (decentralized) heat generation options in the *Industrial sector* and *Residential and commercial sectors*.

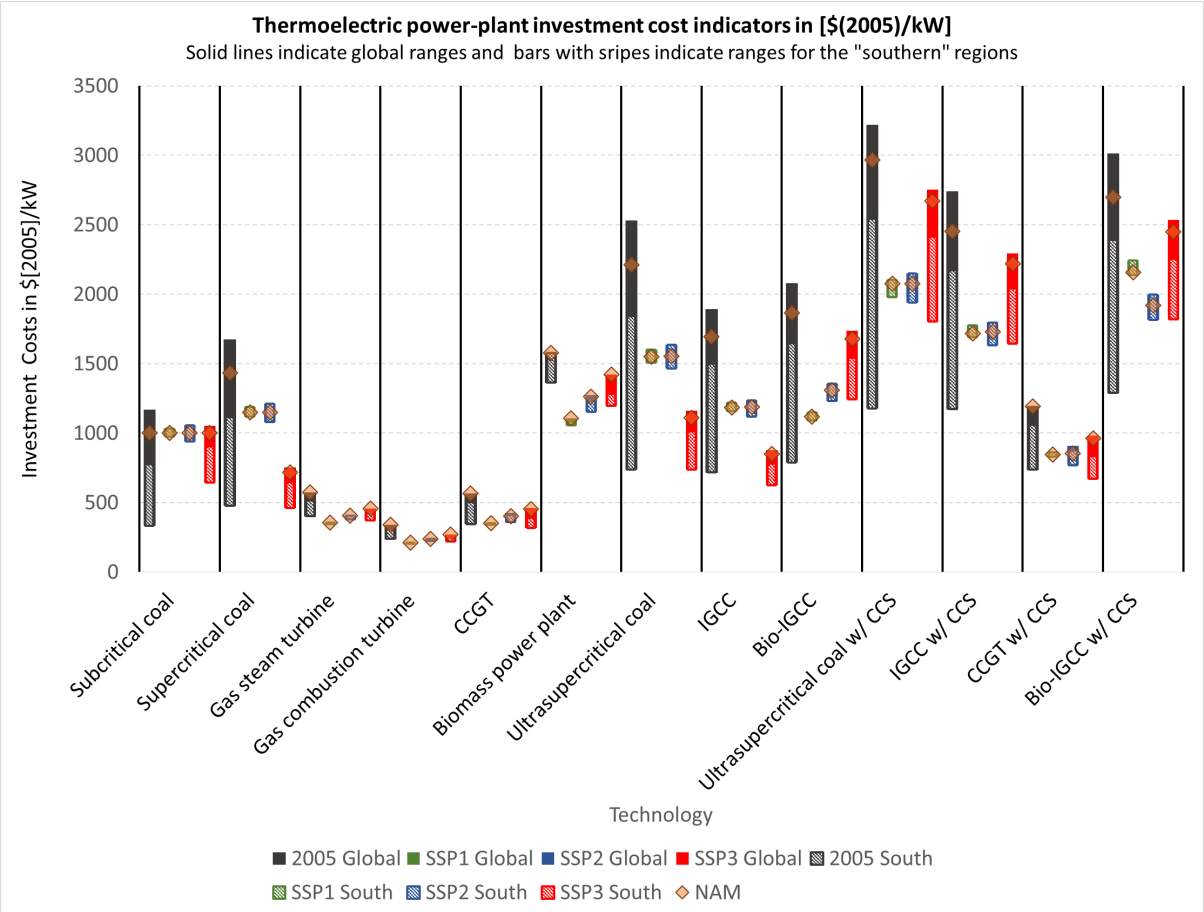

Fig. 8.9: Cost indicators for thermoelectric power-plant investment (Fricko et al., 2017 [17]).

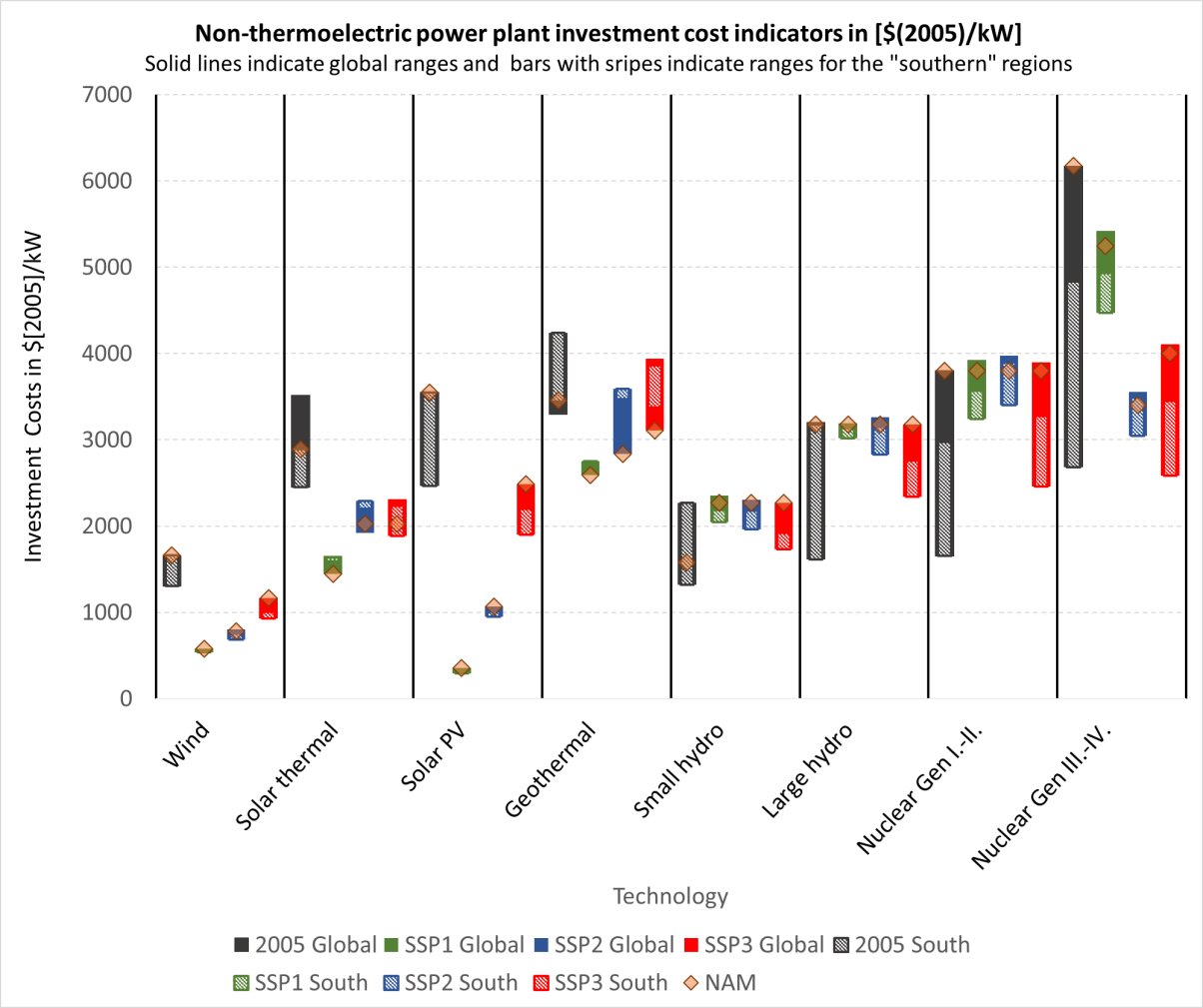

Fig. 8.10: Cost indicators for non-thermoelectric power-plant investment (Fricko et al., 2017 [17]). Abbreviations: CCS – Carbon Capture and Storage; IGCC – Integrated gasification combined cycles; ST – Steam turbine; CT – Combustion turbine; CCGT – Combined cycle gas turbine

Table 8.13: List of centralized heat generation technologies represented in MESSAGE by energy source.

Energy Source	Technology
coal	coal district heating plant
oil	light fuel oil district heating plant
gas	gas district heating plant
biomass	solid biomass district heating plant
geothermal	geothermal district heating plant

Other conversion

Beyond electricity and centralized heat generation there are three further subsectors of the conversion sector represented in MESSAGE, liquid fuel production, gaseous fuel production and hydrogen production. Fig. 8.11 provides an overview of the investment cost ranges for these conversion technologies. The black bars show historical cost ranges for 2005. Green, blue, and red bars show cost ranges in 2100 for SSP1, SSP2, and SSP3, respectively. Global values are represented by solid ranges. Values in the global South are represented by dashed ranges. The diamonds show the costs in the "North America" region.

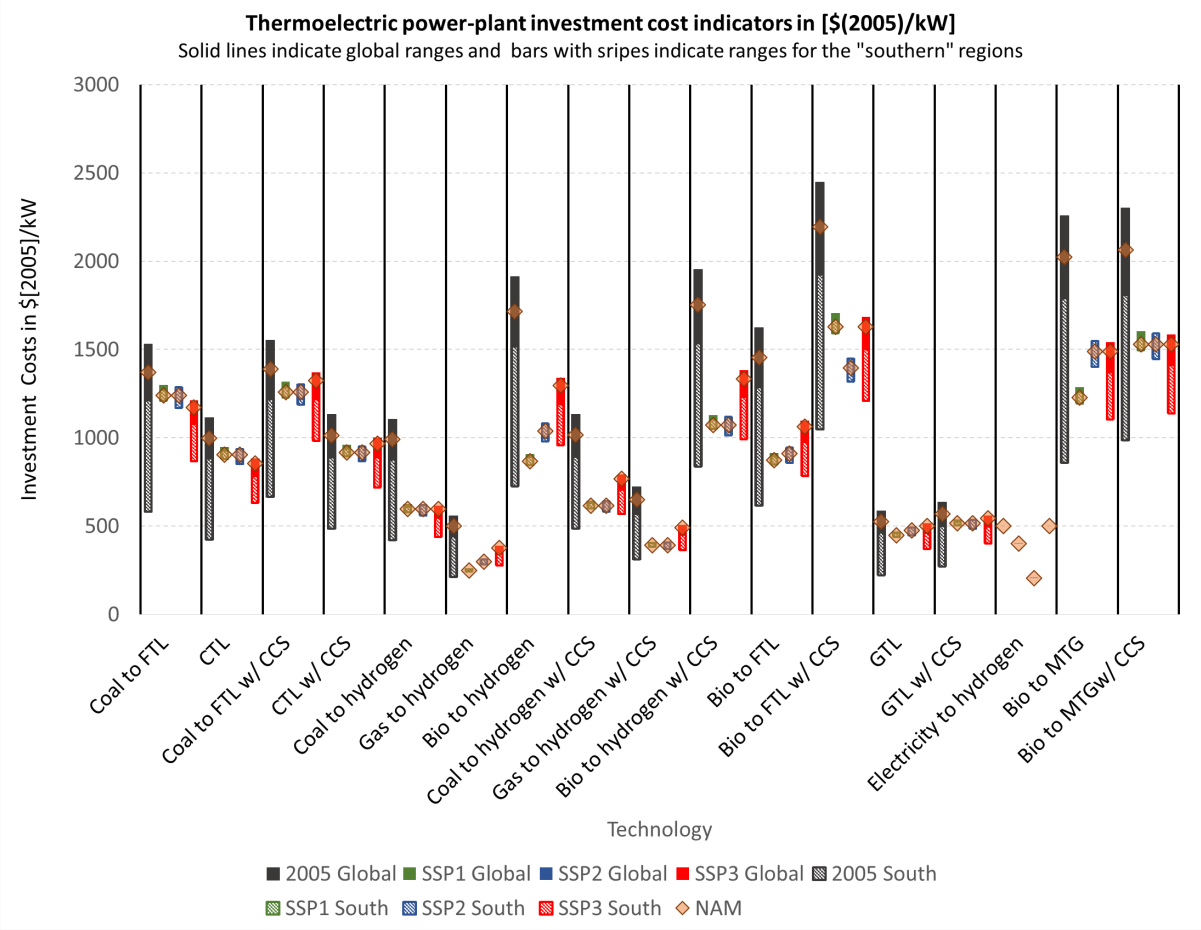

Fig. 8.11: Cost indicators for other conversion technology investment (Fricko et al., 2017 [17]) Abbreviations: CCS – Carbon capture and storage; CTL – Coal to liquids; GTL – Gas to liquids; BTL – Biomass to liquids.

Liquid Fuel Production

Apart from oil refining as predominant supply technology for liquid fuels at present a number of alternative liquid fuel production routes from different feedstocks are represented in MESSAGE (see Table 8.14). Different processes for coal liquefaction, gas-to-liquids technologiesand biomass-to-liquids technologies both with and without CCS are covered. Some of these technologies include co-generation of electricity, for example, by burning unconverted syngas from a Fischer-Tropsch synthesis in a gas turbine (c.f. Larson et al., 2012 [47]). Technology costs for the synthetic liquid fuel production options are based on Larson et al. (2012) [47].

Table 8.14: Liquid fuel production technologies in MESSAGE by energy source.

Energy Source	Technology	Electricity cogeneration
biomass	Fischer-Tropsch biomass-to-liquids	yes
	Fischer-Tropsch biomass-to-liquids with CCS	yes
	Gasoline via the Methanol-to-Gasoline (MTG) Process	yes
	Gasoline via the Methanol-to-Gasoline (MTG) Process with CCS	yes
coal	Fischer-Tropsch coal-to-liquids	yes
	Fischer-Tropsch coal-to-liquids with CCS	yes
	coal methanol-to-gasoline	yes
	coal methanol-to-gasoline with CCS	yes
gas	Fischer-Tropsch gas-to-liquids	no
	Fischer-Tropsch gas-to-liquids with CCS	no
oil	simple refinery	no
	complex refinery	no

Gaseous Fuel Production

Gaseous fuel production technologies represented in MESSAGE are gasification of solids including coal and biomass. In both cases carbon capture and storage (CCS) can be combined with the gasification process to capture to a good part the carbon that is not included in the synthetically produced methane. Table 8.15 provides a listing of all gaseous fuel production technologies.

Table 8.15: Gaseous fuel production technologies in MESSAGE by energy source.

Energy Source	Technology
biomass	biomass gasification
coal	coal gasification

Hydrogen Production

A number of hydrogen production options are represented in MESSAGE. These include gasification processes for coal and biomass, steam methane reforming from natural gas and hydrogen electrolysis. The fossil fuel and biomass based options can be combined with CCS to reduce carbon emissions. Table 8.16 provides a full list of hydrogen production technologies.

Table 8.16: Hydrogen production technologies in MESSAGE by energy source.

Energy source	Technology	Electricity cogeneration
coal	coal gasification	yes
	coal gasification with CCS	yes
biomass	biomass gasification	yes
	biomass gasification with CCS	yes
gas	steam methane reforming	yes
	steam methane reforming with CCS	no
electricity	electrolysis	no

Grid, Infrastructure and System Reliability

Energy Transmission and Distribution Infrastructure

Energy transport and distribution infrastructure is included in MESSAGE at a level relevant to represent the associated costs as well as transmission and distribution losses. Within individual model regions the capital stock of transmission and distribution infrastructure and its turnover is modeled for the following set of energy carriers:

- electricity
- district heat
- natural gas
- hydrogen

For all solid (coal, biomass) and liquid energy carriers (oil products, biofuels, fossil synfuels) a simpler approach is taken and only transmission and distribution losses and costs are taken into account.

Inter-regional energy transmission infrastructure, such as natural gas pipelines and high voltage electricity grids, are also represented between geographically adjacent regions. Solid and liquid fuel trade is, similar to the transmission and distribution within regions, modeled by taking into account distribution losses and costs. A special case are gases that can be traded in liquified form, i.e. liquified natural gas (LNG) and liquid hydrogen, where liquefaction and re-gasification infrastructure is explicitly represented in addition to the actual transport process.

Systems Integration and Reliability

The global MESSAGE model includes a single annual time period within each modeling year characterized by average annual load and 11 geographic regions. Seasonal and diurnal load curves and spatial issues such as transmission constraints or renewable resource heterogeneity are treated in a stylized way in the model. The mechanism to represent power system reliability in MESSAGE is based on (Sullivan et al., 2013 [108]). This method elevates the stylization of temporal resolution by introducing two concepts, peak reserve capacity and general-timescale flexibility (for mathematical representation see this Section). To represent capacity reserves in MESSAGE, a requirement is defined that each region build sufficient firm generating capacity to maintain reliability through reasonable load and contingency events. As a proxy for complex system reliability metrics, a reserve margin-based metric was used, setting the capacity requirement at a multiple of average load, based on electric-system parameters. While many of the same issues apply to both electricity from wind and solar energy, the description below focuses on wind.

Toward meeting the firm capacity requirement, conventional generating technologies contribute their nameplate generation capacity while variable renewables contribute a capacity value that declines as the market share of the technology increases. This reflects the fact that wind and solar generators do not always generate when needed, and that their output is generally self-correlated. In order to adjust wind capacity values for different levels of penetration, it was necessary to introduce a stepwise-linear supply curve for wind power (shown in the Fig. 8.12 below). Each bin covers a range of wind penetration levels as fraction of load and has discrete coefficients for the two constraints. The bins are predefined, and therefore are not able to allow, for example, resource diversification to increase capacity value at a given level of wind penetration.

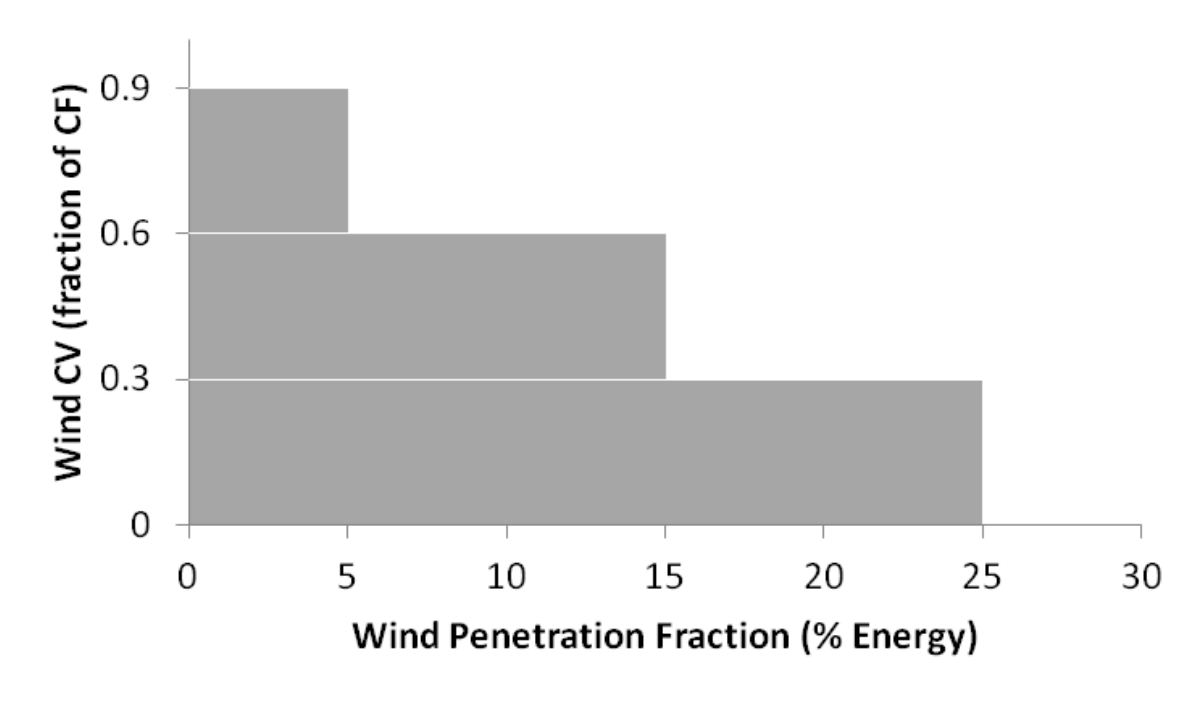

Fig. 8.12: Parameterization of Wind Capacity Value.

The capacity value bins are independent of the wind supply curve bins that already existed in MESSAGE, which are based on quality of the wind resource. That supply curve is defined by absolute wind capacity built, not fraction of load; and the bins differ based on their annual average capacity factor, not capacity value. Solar PV is treated in a similar way as wind with the parameters obviously being different ones. In contrast, concentrating solar power (CSP) is modeled very much like dispatchable power plants in MESSAGE, because it is assumed to come with several hours of thermal storage, making it almost capable of running in baseload mode.

In order to ensure adequate reserve dispatch, dynamic shadow prices are placed on capacity investments of intermittent technologies (e.g., wind and solar). The prices are a function of the cumulative installed capacity of the intermittent technologies, the ability for the convential power supply to act as reserve dispatch, and the demand-side reliability requirements. For instance, a large amount of storage capacity should, all else being equal, lower the shadow price for additional wind. Conversely, an inflexible, coal- or nuclear-heavy generating base should increase the cost of investment in wind by demanding additional expenditures in the form of natural gas combustion turbines or storage or improved demand-side management to maintain system reliability.

Starting from the energy metric used in MESSAGE (electricity is considered as annual average load; there are no time-slices or load-curves), the flexibility requirement uses MWh of generation as its unit of note. The metric is inherently limited because operating reserves are often characterized by energy not-generated: a natural gas combustion turbine (gas CT) that is standing by, ready to start-up at a moment's notice; a combined-cycle plant operating below its peak output to enable ramping in the event of a surge in demand. Nevertheless, because there is generally a portion of generation associated with providing operating reserves (e.g. that on-call gas CT plant will be called some fraction of the time), it is posited that using generated energy to gauge flexibility is a reasonable metric considering the simplifications that need to be made. Furthermore, ancillary services associated with ramping and peaking often do involve real energy generation, and variable renewable technologies generally increase the need for ramping.

Electric-sector flexibility in MESSAGE is represented as follows: each generating technology is assigned a coefficient between -1 and 1 representing (if positive) the fraction of generation from that technology that is considered to be flexible or (if negative) the additional flexible generation required for each unit of generation from that technology. Load also has a parameter (a negative one) representing the amount of flexible energy the system requires solely to meet changes and uncertainty in load. Table 8.17 below displays the parameters that were estimated using a unit-commitment model that commits and dispatches a fixed generation system at hourly resolution to meet load an ancilliary service requirements while hewing to generator and transmission operation limitations (Sullivan et al., 2013 [108]). Technologies that were not included in the unit-commitment model (nuclear, hydrogen electrolysis, solar PV) have estimated coefficients.

Table 8.17: Flexibility Coefficients by Technology (Sullivan et al., 2013 [108]).

Technology	Flexibility Parameter
Load	-0.1
Wind	-0.08
Solar PV	-0.05
Geothermal	0
Nuclear	0
Coal	0.15
Biopower	0.3
Gas CC	0.5
Hydropower	0.5
H2 Electrolysis	0.5
Oil/Gas Steam	1
Gas CT	1
Electricity Storage	1

Thus, a technology like a natural gas combustion turbine, used almost exclusively for ancillary services, has a flexibility coefficient of 1, while a coal plant, which provides mostly bulk power but can supply some ancillary services, has a small, positive coefficient. Electric storage systems (e.g., pumped hydropower, compressed air storage, flow batteries) and flexible demand-side technologies like hydrogen-production contribute as well. Meanwhile, wind power and solar PV, which require additional system flexibility to smooth out fluctuations, have negative flexibility coefficients.

Energy end-use

MESSAGEix distinguishes three energy end-use sectors, i.e. transport, residential/commercial (also referred to as the buildings sector) and industry. Given the long-term nature of the scenarios, the model version used for the SSPs, represents these end-use sectors in a stylized way. For more detailed short-term analysis, a model version with a more detailed transport sector module that distinguishes different transport modes, vehicle classes and consumer types exists (McCollum et al., 2016 [54]).

Transport sector

The most commonly applied MESSAGEix transport sector representation is stylized and essentially includes fuel switching and price-elastic demands (via MACRO linkage) as the main responses to energy and climate policy (see Fig. 8.13).

In this stylized transport sector representation fuel switching is a key option to reduce emissions, i.e., different final energy forms that provide energy for transportation can be chosen from. In addition to the alternative energy carriers that serve as input to these stylized transportation options, their relative efficiencies are also different. The useful energy demand in the transportation sector is specified as internal combustion engine (ICE) equivalent demands which therefore by definition has a conversion efficiency of final to useful energy of 1. Relative to that the conversion efficiency of alternative fuels is higher, for example, electricity in 2010 has about a factor of three, higher final to useful efficiency than the regular oil-product based ICE. The overall efficiency improvements of the ICE in the transportation sector and modal switching over time is implicitly included in the demand specifications, coming from the scenario generator (see section on demand). Additional demand reduction in response to price increases in policy scenarios then occurs via the fuel switching option (due to the fuel-specific relative efficiencies) as well as via the linkage with the macro-economic model MACRO as illustrated in Fig. 8.13 below.

Limitations of switching to alternative fuels may occur for example as a result of restricted infrastructure availability (e.g., rail network) or some energy carriers being unsuitable for certain transport modes (e.g., electrification of aviation). To reflect these limitations, share constraints of energy carriers (e.g., electricity) and energy carrier groups (e.g., liquid fuels) are used in the transport sector. In addition, the diffusion speed of alternative fuels is limited to mimic bottlenecks in the supply chains, not explicitly represented in MESSAGEix (e.g., non-energy related infras-

tructure). Both the share as well as the diffusion constraints are usually parametrized based on transport sector studies that analyze such developments and their feasibility in much greater detail.

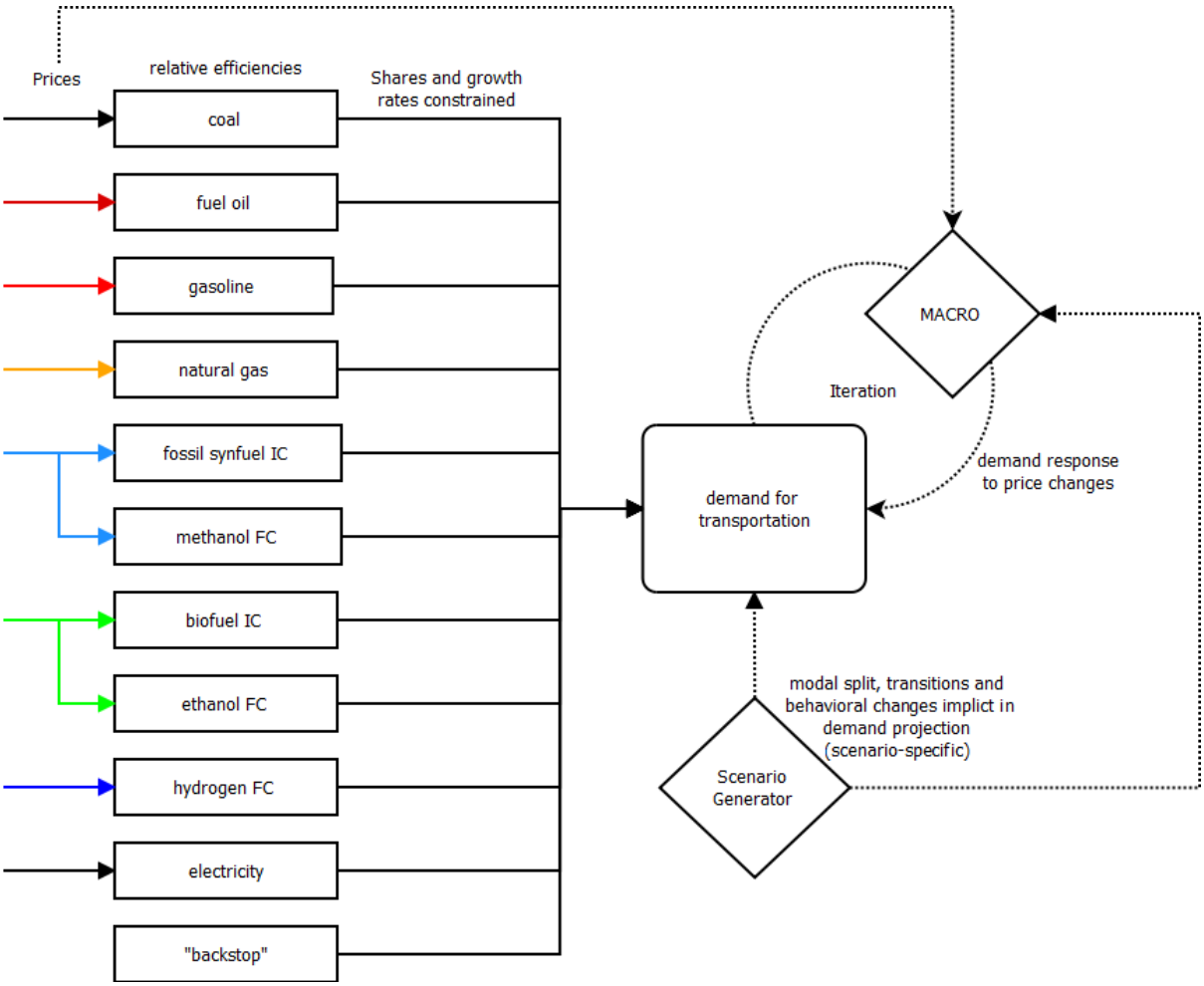

Fig. 8.13: Schematic diagram of the stylized transport sector representation in MESSAGEix.

The demand for international shipping is modeled in a simplified way with a number of different energy carrier options (light and heavy fuel oil, biofuels, natural gas, and hydrogen). The demand for international shipping is coupled to global GDP development with an income elasticity, but to date no demand response in mitigation scenarios is implemented.

Table 8.18 presents the quantitative translation of the the storyline elements of SSP1, SSP2 and SSP3 in terms of electrification rate for transport (Fricko et al., 2017 [17]).

Table 8.18: Electrification rate within transport for SSP1, SSP2 and SSP3 (Fricko et al., 2017 [17]). The indicators apply to 2010-2100; Intensity improvements are presented in Final Energy (FE)/GDP annually.

	SSP1	SSP2	SSP3
Transport	High electrification (max. 75% of total transport possible)	Medium electrification (max. 50% of total transport possible)	Low electrification (max. 10% of total transport possible)

Residential and commercial sectors

The residential and commercial sector in MESSAGEix distinguishes two demand categories, thermal and specific. Thermal demand, i.e., low temperature heat, can be supplied by a variety of different energy carriers while specific demand requires electricity (or a decentralized technology to convert other energy carriers to electricity).

The residential and commercial thermal energy demand includes fuel switching as the main option, i.e., different choices about final energy forms to provide thermal energy. In addition to the alternative energy carriers that serve as input to these thermal energy supply options, their relative efficiencies also vary. For example, solid fuels such as coal have lower conversion efficiencies than natural gas, direct electric heating or electric heat pumps. Additional demand reduction in response to price increases in policy scenarios is included via the fuel switching option (due to the fuel-specific relative efficiencies) as well as via the linkage with the macro-economic model MACRO (see Fig. 8.14 below). The specific residential and commercial demand can be satisfied either by electricity from the grid or with decentralized electricity generation options such as fuel cells and on-site CHP.

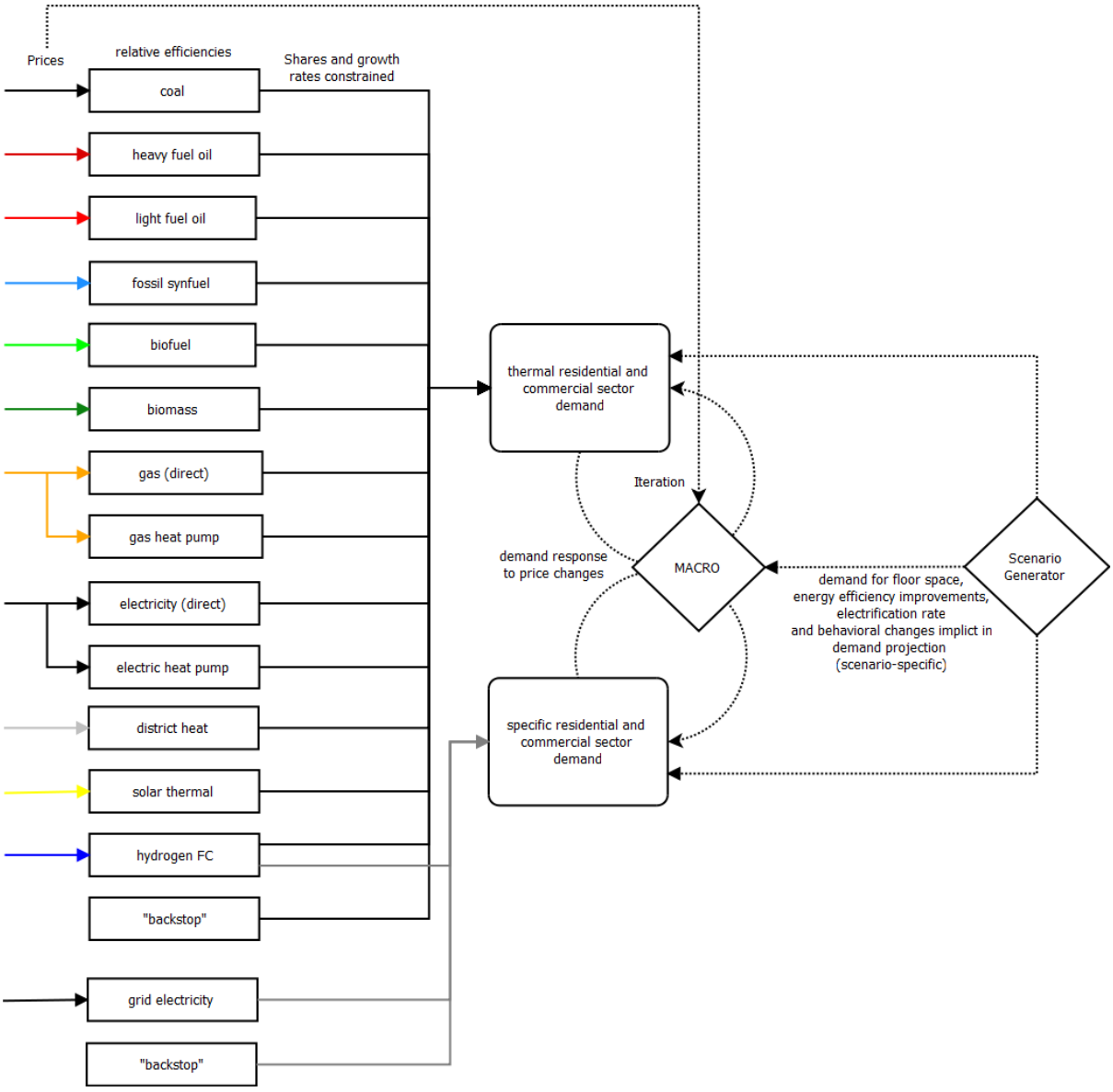

Fig. 8.14: Schematic diagram of the residential and commercial sector representation in MESSAGEix.

To reflect limitations of switching to alternative fuels, for example as a result of limited infrastructure availability (e.g., district heating network) or some energy carriers being unsuitable for certain applications, share constraints of energy carriers (e.g., electricity) and energy carrier groups (e.g., liquid fuels) are used in the residential and commercial sector. In addition, as in the transport sector, the diffusion speed of alternative fuels is limited to mimic bottlenecks

in the supply chains, not explicitly represented in MESSAGEix (e.g., non-energy related infrastructure).

Table 8.19 presents the quantitative translation of the the storyline elements of SSP1, SSP2 and SSP3 in terms of electrification rate for the residential and commercial sectors. These indicators apply to 2010-2100; Intensity improvements are in FE/GDP annually (Fricko et al., 2017 [17]).

Table 8.19: Electrification rate within the residential and commercial sectors for SSP1, SSP2 and SSP3 (Fricko et al., 2017 [17])

	SSP1	SSP2	SSP3
Residential & Commercial	High electrification rate:1.44% (Regional range from 0.35% to 4%)	Medium electrification rate: 1.07% (Regional range from 0.23% to 3%)	Low electrification rate: 0.87% (Regional range from 0.37% to 2%)

Industrial sector

Similar to the residential and commercial sectors, the industrial sector in MESSAGEix distinguishes two demand categories, thermal and specific. Thermal demand, i.e., heat at different temperature levels, can be supplied by a variety of different energy carriers while specific demand requires electricity (or a decentralized technology to convert other energy carriers to electricity).

This stylized industrial thermal energy demand includes fuel switching as the main option, i.e., different final energy forms that provide energy for thermal energy can be chosen from. In addition to the alternative energy carriers that serve as input to these thermal energy supply options, their relative efficiencies also vary. For example, solid fuels such as coal have lower conversion efficiencies than natural gas, direct electric heating or electric heat pumps. To account for the fact that some technologies cannot supply temperature at high temperature levels (e.g., electric heat pumps, district heat), the share of these technologies in the provision of industrial thermal demand is constrained. Additional demand reduction in response to price increases in policy scenarios is included via the fuel switching option (due to the fuel-specific relative efficiencies) as well as via the linkage with the macro-economic model MACRO (see Fig. 8.15 below). The specific industrial demand can be satisfied either by electricity from the grid or with decentralized electricity generation options such as fuel cells and on-site CHP.

While cement production is not explicitly modeled at the process level in MESSAGEix, the amount of cement production is linked to industrial activity (more specifically the industrial thermal demand in MESSAGEix) and the associated CO2 emissions from the calcination process are accounted for explicitly. In addition, adding carbon capture and storage to mitigate these process-based CO2 emission is available.

Table 8.20 presents the quantitative translation of the the storyline elements of SSP1, SSP2 and SSP3 in terms of electrification rate for industry and feedstocks. These indicators apply to 2010-2100; Intensity improvements are in FE/GDP annually (Fricko et al., 2017 [17]).

Table 8.20: Electrification rate within industry and feedstocks for SSP1, SSP2 and SSP3 (Fricko et al., 2017 [17])

	SSP1	SSP2	SSP3
Industry	High electrification rate: 0.56% (Regional range from 0.2% to 1.2%)	Medium electrification rate: 0.47% (Regional range from 0.07% to 1.08%)	Low electrification rate: 0.12% (Regional range from -0.03% to 0.71%)
Feedstock (non-energy use)	High feedstock reduction rate: -0.33% (Regional range from -0.51 to 0.59%)	Medium feedstock reduction rate: -0.27% (Regional range from -0.45% to 0.64%)	Low feedstock reduction rate: -0.24% (Regional range from -0.38% to 0.51%)

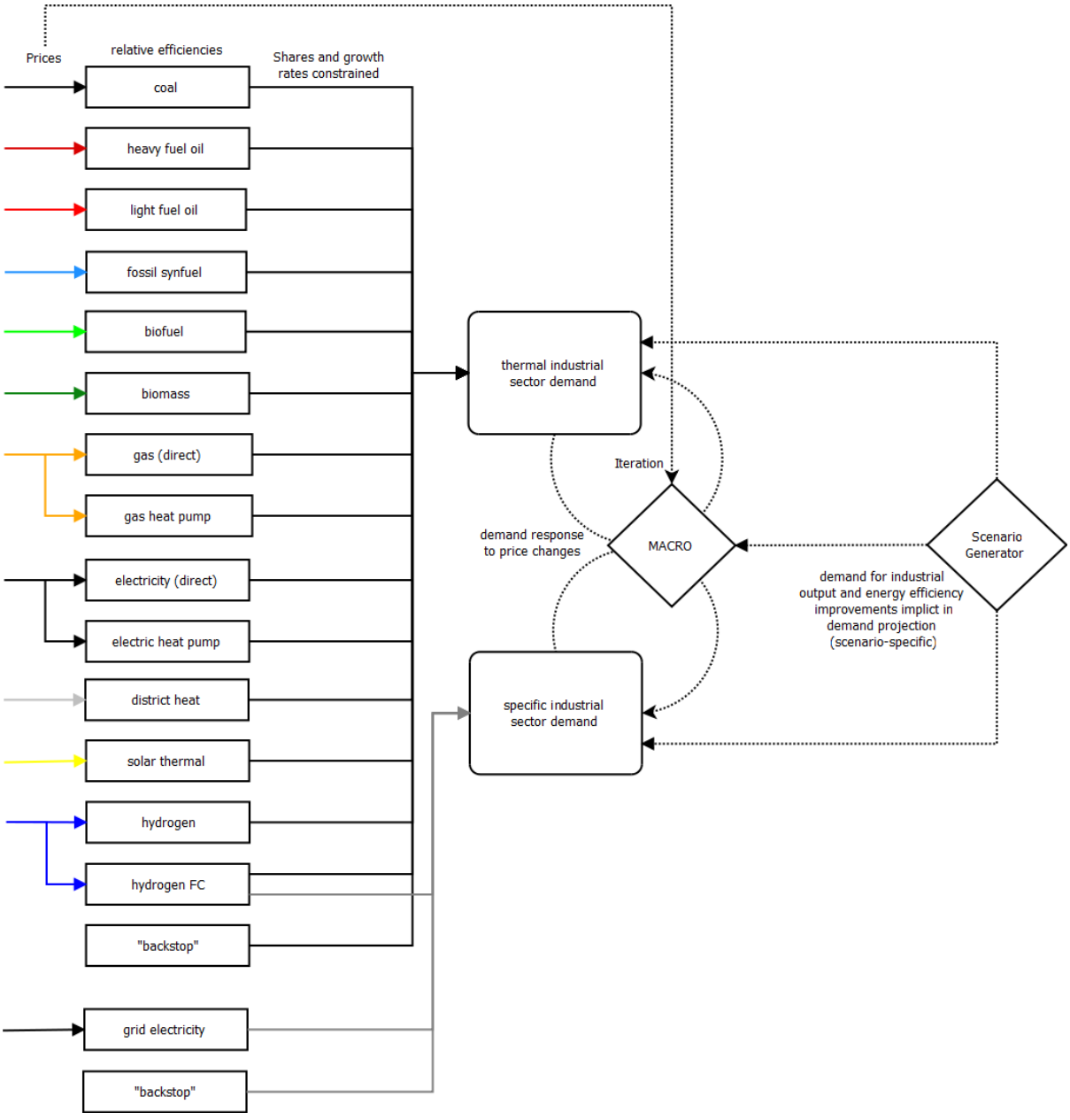

Fig. 8.15: Schematic diagram of the industrial sector representation in MESSAGEix.

Technological change

Technological change in MESSAGE*ix* is generally treated exogenously, although pioneering works on the endogenization of technological change via learning curves in energy-engineering type models (Messner, 1997 [60]) and the dependence of technology costs on market structure have been done with MESSAGE*ix* (Leibowicz, 2015 [49]). The current cost and performance parameters, including conversion efficiencies and emission coefficients are generally derived from the relevant engineering literature. For the future, alternative cost and performance projections are developed to cover a relatively wide range of uncertainties that influence model results to a good extent.

Technology cost

The quantitative assumptions about technology cost development are derived from the overarching qualitative SSP narratives (cf. section *SSP narratives*). In SSP1, for instance, whose "green-growth" storyline is more consistent with a sustainable development paradigm, higher rates of technological progress and learning are assumed for renewable energy technologies and other advanced technologies that may replace fossil fuels (e.g., the potential for electric mobility is assumed to be higher in SSP1 compared to SSP2 or SSP3). In contrast, SSP3 assumes limited progress across a host of advanced technologies, particularly for renewables and hydrogen; more optimistic assumptions are instead made for coal-based technologies, not only for power generation but also for liquid fuels production (e.g., coal-to-liquids). Meanwhile, the middle-of-the-road SSP2 narrative is characterized by a fairly balanced view of progress for both conventional fossil and non-fossil technologies. In this sense, technological development in SSP2 is not biased toward any particular technology group.

Technological costs vary regionally in all SSPs, reflecting marked differences in engineering and construction costs across countries observed in the real world. The regional differentiation of technology costs for the initial modeling periods are based on IEA data (IEA, 2014 [32]) with convergence of costs assumed over time driven by economic development (GDP/cap). Generally, costs start out lower in the developing world and are assumed to converge to those of present-day industrialized countries as the former becomes richer throughout the century (thus, the cost projections consider both labour and capital components). This catch-up in costs is assumed to be fastest in SSP1 and slowest in SSP3 (where differences remain, even in 2100); SSP2 is in between. Estimates for present-day and fully learned-out technology costs are from the Global Energy Assessment (Riahi et al., 2012 [85]) and World Energy Outlook (IEA, 2014 [2]). A summary of these cost assumptions can be found in sections *Electricity* and *Other conversion*.

Technology diffusion

MESSAGE tracks investments by vintage, an important feature to represent the inertia in the energy system due to its long-lived capital stock. In case of shocks (e.g., introduction of stringent climate policy), it is however possible to prematurely retire existing capital stock such as power plants or other energy conversion technologies and switch to more suitable alternatives.

An important factor in this context that influences technology adoption in MESSAGE*ix* are technology diffusion constraints. Technology diffusion in MESSAGE*ix* is determined by dynamic constraints that relate the construction of a technology added or the activity (level of production) of a technology in a period *t* to construction or the activity in the previous period *t-1* (Messner and Strubegger, 1995 [62], cf. section Dynamic constraints).

While limiting the possibility of flip-flop behavior as is frequently observed in unconstrained Linear Programming (LP) models such as MESSAGE*ix*, a drawback of such hard growth constraints is that the relative advantage of some technology over another technology is not taken into account and therefore even for very competitive technologies, no rapid acceleration of technology diffusion is possible. In response to this limitation, so called flexible or soft dynamic constraints have been introduced into MESSAGE (Keppo and Strubegger, 2010 [41]). These allow faster technology diffusion at additional costs and therefore generate additional model flexibility while still reducing the flip-flop behavior and sudden penetration of technologies.

Fig. 8.16 below illustrates the maximum technology growth starting at a level of 1 in year *t* =0 for a set of five diffusion constraints which jointly lead to a soft constraint.

For a more detailed description of the implementation of technology diffusion constraints, see the Section Dynamic constraints of the |MESSAGEix| documentation.

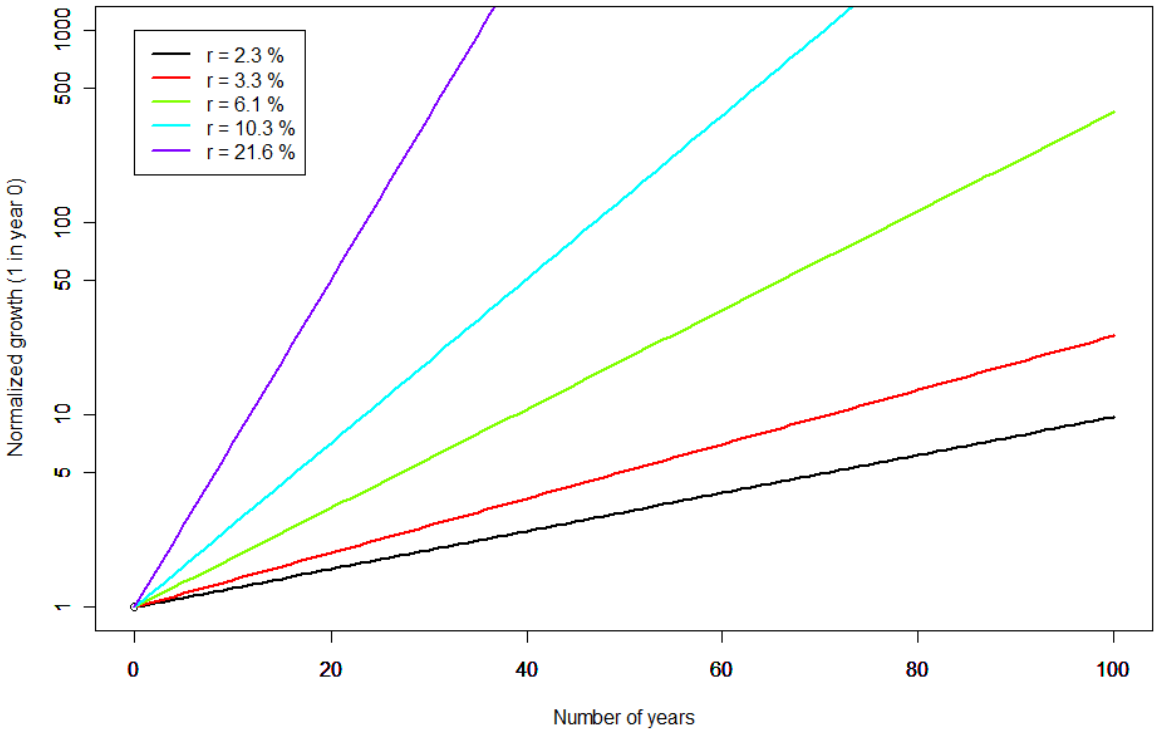

Fig. 8.16: Illustration of maximum technology growth starting at a level of 1 in year t=0 for a set of soft diffusion constraints with effective growth rates r as shown in the legend.

Add-on technologies

Add-on technologies in the global model refer to a distinct formulation in MESSAGE*ix*. The formulation is used to represent two main types of technical extensions/options for technologies. Add-on technologies provide additional modes of operation for a single or multiple technologies. They can also be used to depict emission mitigation options.

General description of add-on technologies

Add-on technologies can be defined using all the same parameters as any other technology. What makes a technology an *add-on technology*, is the fact that their activity is bound to the activity of one or more other technologies, henceforth referred to as the parent technology. The mathematical formulation can be found here. One of the main benefits of the add-on technology formulation, over specifying an alternative *mode*, is that it allows a single add-on technology to be coupled to the activity of multiple parent technologies. Furthermore, multiple add-on technologies can be linked to the activity of a single parent technology.

Modelling Combined Heat Powerplants (CHPs)

In the global model, there are numerous electricity generation technologies (cf. Section *Electricity*). A separate technology, known as a *pass out turbine*, is represented in the model to provide select electricity generation technologies the option to reduce their electricity output in favor of generating electricity and heat. The pass out turbine, which is a steam turbine in which a certain amount of the pressurized steam is passed out of the turbine for the purpose of heat production, is restricted to a share of the activity of the selected electricity generation technologies. Technically, this means that the electricity output of the electricity generation technologies remains unaltered, yet each unit of heat generated by the pass out turbine, requires a certain electricity input. The figure below is an excerpt of the Reference Energy System (RES), showing how the pass-out turbine is modelled.

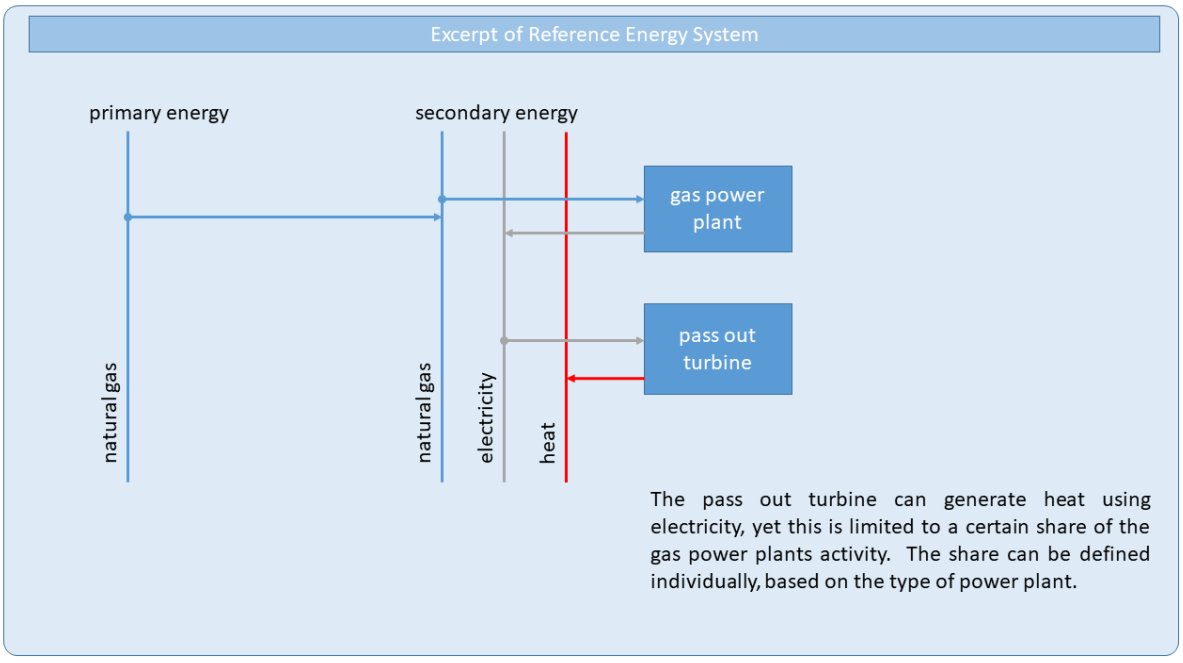

Fig. 8.17: Reference Energy System excerpt depicting the modelling of CHPs.

Modelling emission mitigation options using add-on technologies

CO2 emission mitigation options for electricity and synthetic fuel generation can be modelled either as green-field power plants with carbon, capture and storage capabilities (CCS), but there is also the possibility to retrofit existing fossil fuel based energy generation technologies with CCS units. The latter, which is less efficient than the greenfield option, but an effective transition option to minimize stranded assets in deep mitigation scenarios, is modelled using the add-on technology formulation. Analogue to the way in which CHPs are modelled, a separate CCS-retrofit unit is depicted in the model, which is constrained by the activity of the respective parent technologies. The CCS-retrofit option requires electricity as an input, therefore mimicking the efficiency reduction associated with the operation of the CCS-retrofit unit. Per unit of activity of the CCS-retrofit, CO2 emissions are reduced, which differ depending on the assumed capture rates. CCS-retrofits are available for: coal power plants including internal gasification combined cycle plants (IGCC), select gas power plants, biomass power plants, gas and coal fuel cells as well as for hydrogen and cement production.

In the global model, emission mitigation options are modelled using add-on technologies for several other emission sources. N2O emissions from nitirc and adipic acid are driven by industrial GDP and CH4 landfill emissions are driven by population (Rao and Riahi, 2006 [82]). As both GDP and population are model inputs, the developments for these specific sources are therefore not endogenous to the model. Similarly, HFC and SF6 emissions are linked to specific useful energy demands, which are again a model input, and electricity transmission, respectively. In order to provide mitigation options for these emissions sources, depending on the source, one or several mitigation options are modelled. For each source, the combined activity and therefore the mitigation is coupled to the activity of the parent technology. The share of the total emissions which can be reduced is limited to the technical feasibility and the combination of which mitigation technologies are employed are economically driven.

Fuel Blending

Fuel blending in the energy system is a common practice, which allows the shared use of infrastructure by fuels with similar chemical attributes and thus use at the secondary and final energy level, without requiring the consumer to adapt the power plant or enduse devices. Fuel blending in the global energy model is modelled for two distinct blending processes. The first relates to the blending of natural gas with other synthetic gases. The second is related to the blending of light oil with coal derived synthetic liquids. In order to ensure that emissions and energy flows are correctly accounted for, blended fuels types are nevertheless explicitly modelled.

Natural gas and synthetic gas

Natural gas can be blended with hydrogen or with synthetic gas derived from the gasification of biomass or coal (cf. Section *Other conversion*). Despite the fact that in the real world, hydrogen or other synthetic gases are physically injected into a natural gas network, it is important to be able to track the use of blended fuels in the energy model for two reasons. Not all blended fuels can be used equally within all natural gas applications. For example, hydrogen mixed into the natural gas network is restricted to use in non-CCS applications only. Secondly, it is essential to keep track of where which of the blended fuels is being used in order to correctly report emissions and also to potentially restrict the degree to which fuels can be blended for individual applications. For example, natural gas end-use appliances may only be able to cope with a certain share of hydrogen while still guaranteeing their safety and longevity. Similarly, for policy analysis, it could be required that a certain minimum share of a synthetic gas is used sector specifically.

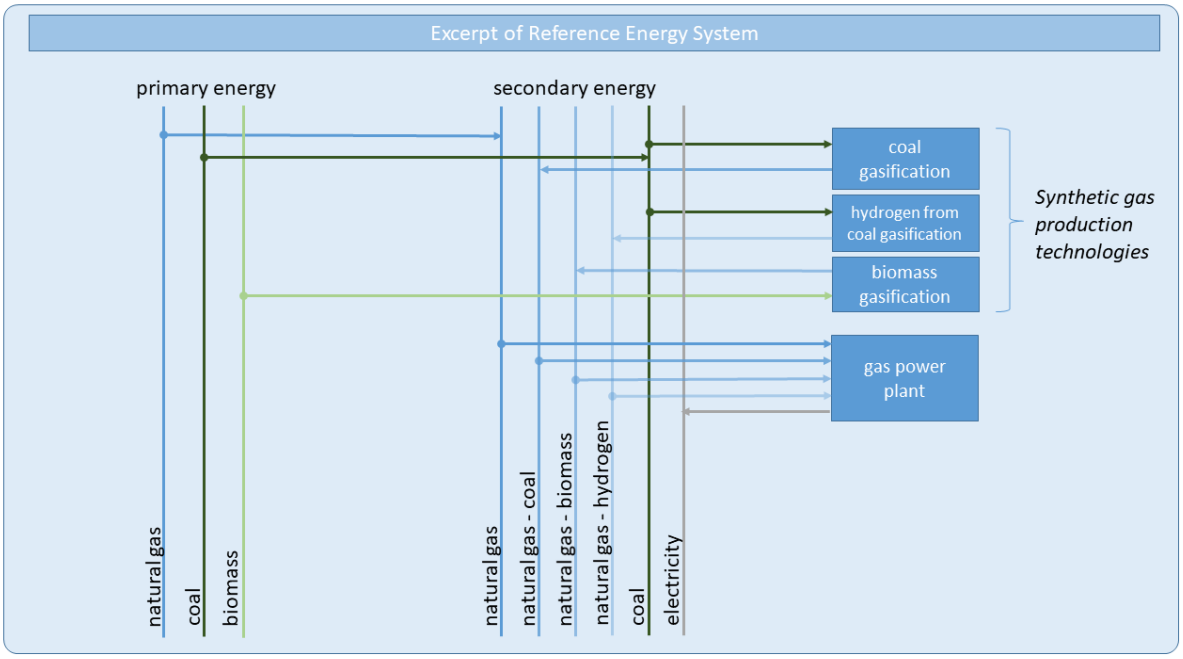

Fig. 8.18: Reference Energy System excerpt depicting the modelling of fuel blending.

Synthetic liquids and lightoil

Synthetic fueloil via coal liquefaction is blended into the lightoil stream at the secondary energy level.

Energy demand

Baseline energy service demands are provided exogenously to MESSAGE*ix*, though they can be adjusted endogenously based on energy prices using the MESSAGE*ix*-MACRO link. There are seven energy service demands that are provided to MESSAGE*ix*, including:

1. Residential/commercial thermal
2. Residential/commercial specific
3. Industrial thermal
4. Industrial specific
5. Industrial feedstock (non-energy)
6. Transportation
7. Non-commercial biomass.

These demands are generated using a so-called scenario generator which is implemented in the script language R. The scenario generator relates historical country-level GDP per capita (PPP) to final energy and, using projections of GDP (PPP) and population, extrapolate the seven energy service demands into the future. The sources for the historical and projected datasets are the following:

1. Historical GDP (PPP) – World Bank (World Development Indicators, 2012 [118])
2. Historical Population – UN Population Division (World Population Projection, 2010 [10])
3. Historical Final Energy – International Energy Agency Energy Balances (IEA, 2012 [1])
4. Projected GDP (PPP) – Dellink et al. (2015) [9], also see Shared Socio-Economic Pathways database (SSP scenarios)

5. Projected Population – KC and Lutz (2014) [40], also see Shared Socio-Economic Pathways database (SSP scenarios)

The scenario generator runs regressions on the historical datasets to establish the relationship for each of the eleven MESSAGE*ix* regions between the independent variable (GDP (PPP) per capita) and the following dependent variables:

1. Total final energy intensity (MJ/2005USD)
2. Shares of final energy among several energy end-use sectors (transport, residential/commercial and industry)
3. Shares of electricity use between the industrial and residential/commercial sectors.

In the case of final energy intensity, the relationship is best modeled by a power function so both variables are log-transformed. In the case of most sectoral shares, only the independent variable is log-transformed. The exception is the industrial share of final energy, which uses a hump-shaped function inspired by Schafer (2005) [99].

In parallel, the same historical data are used, now globally, in quantile regressions to develop global trend lines that represent each percentile of the cumulative distribution function (CDF) of each dependent variable. Given the regional regressions and global trend lines, final energy intensity and sectoral shares can be extrapolated based on projected GDP per capita, or average income.

A basic assumption here is that the regional trends derived above will converge to certain quantiles of the global trend when each region reaches a certain income level. Hence, two key user-defined inputs allow users to tailor the extrapolations to individual socio-economic scenarios: convergence quantile and the corresponding income. In the case of final energy intensity (FEI), the extrapolation is produced for each region by defining the quantile at which FEI converges (e.g., the 20th percentile within the global trend) and the income at which the convergence occurs. For example, while final energy intensity converges quickly to the lowest quantile (0.001) in SSP1, it converges more slowly to a larger quantile (0.5 to 0.7 depending on the region) in SSP3. Convergence quantiles and incomes are provided for each SSP and region in Table 8.21, Table 8.22, Table 8.23. The convergence quantile allows one to identify the magnitude of FEI while the convergence income establishes the rate at which the quantile is approached. For the sectoral shares, users can specify the global quantile at which the extrapolation should converge, the income at which the extrapolation diverges from the regional regression line and turns parallel to the specified convergence quantile (i.e., how long the sectoral share follows the historical trajectory), and the income at which the extrapolation converges to the quantile. Given these input parameters, users can extrapolate both FEI and sectoral shares.

The total final energy in each region is then calculated by multiplying the extrapolated final energy intensity by the projected GDP (PPP) in each time period. Next, the extrapolated shares are multiplied by the total final energy to identify final energy demand for each of the seven energy service demands used in MESSAGE. Finally, final energy is converted to useful energy in each region by using the average final-to-useful energy efficiencies used in the MESSAGE model for each model region (*Regions*).

Table 8.21: Convergence quantile and income for each quantity and region for SSP1 (for region descriptions, see: Regions)

SSP1	AFR	CPA	EEU	FSU	LAM	MEA	NAM	PAO	PAS	SAS	WEU
Convergence Quantile											
Final Energy Intensity (FEI)	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Share NC Biomass	0.01	0.25	0.01	0.75	0.01	0.3	0.01	0.01	0.01	0.01	0.01
Share Transport	0.05	0.02	0.2	0.05	0.2	0.05	0.2	0.2	0.04	0.03	0.2
Share Res/Com	0.25	0.25	0.2	0.2	0.28	0.3	0.25	0.2	0.28	0.3	0.2
Share Industry	0.1	0.2	0.1	0.5	0.28	0.2	0.3	0.3	0.28	0.2	0.3
Elec Share Res/Com	0.45	0.45	0.45	0.45	0.63	0.62	0.4	0.63	0.62	0.64	0.43
Feedstock Share Industry	0.18	0.2	0.24	0.24	0.2	0.26	0.26	0.23	0.26	0.22	0.24
Elec Share Industry	0.4	0.4	0.42	0.36	0.4	0.33	0.36	0.36	0.4	0.4	0.4
Convergence Income											
Final Energy Intensity (FEI)	112295	98603	299177	112307	100188	113404	112350	112263	106323	112300	107636
Share NC Biomass	5981	46015	34405	40951	20038	34894	112350	112263	16357	11105	48153
Share Transport	99676	32868	112343	71664	112310	113404	123018	94337	112293	97169	141627
Share Res/Com	119613	112270	179500	153565	112310	112270	123018	157229	112293	112300	141627
Share Industry	39870	105177	164547	92139	40075	112270	123018	112263	126769	83288	127464
Elec Share Res/Com	112295	112270	112343	112307	112310	87234	131219	132077	112293	112300	112168
Feedstock Share Industry	112295	112270	112343	112307	112310	112270	123018	125783	112293	112300	112168
Elec Share Industry	112295	98603	299177	112307	100188	113404	112350	112263	106323	112300	107636

Table 8.22: Convergence quantile and income for each quantity and region for SSP2 (for region descriptions, see: Regions)

SSP2	AFR	CPA	EEU	FSU	LAM	MEA	NAM	PAO	PAS	SAS	WEU
Convergence Quantile											
Final Energy Intensity (FEI)	0.03	0.03	0.03	0.04	0.04	0.04	0.05	0.02	0.03	0.03	0.02
Share NC Biomass	0.6	0.6	0.75	0.75	0.25	0.75	0.75	0.75	0.6	0.6	0.75
Share Transport	0.05	0.04	0.15	0.1	0.5	0.3	0.5	0.14	0.2	0.05	0.15
Share Res/Com	0.15	0.28	0.5	0.5	0.3	0.5	0.3	0.35	0.3	0.28	0.33
Share Industry	0.25	0.4	0.15	0.25	0.15	0.25	0.25	0.25	0.25	0.6	0.25
Elec Share Res/Com	0.42	0.4	0.35	0.22	0.58	0.6	0.14	0.57	0.6	0.51	0.18
Feedstock Share Industry	0.15	0.22	0.26	0.26	0.18	0.27	0.32	0.27	0.3	0.22	0.27
Elec Share Industry	0.39	0.38	0.4	0.45	0.35	0.4	0.4	0.4	0.4	0.43	0.35
Convergence Income											
Final Energy Intensity (FEI)	200009	200031	299177	266179	199975	139574	246036	141506	199968	200002	199977
Share NC Biomass	19935	26294	77786	40951	20038	94649	94724	132072	12268	18046	48153
Share Transport	49838	105177	94540	94596	80150	94649	94724	94652	81787	27763	99139
Share Res/Com	119611	65735	89753	71664	94577	69787	94724	110066	81787	83288	113301
Share Industry	31896	105177	44877	102377	100188	78511	94724	141506	98144	13881	94607
Elec Share Res/Com	69773	94593	94540	102377	94577	87234	123018	141506	94627	55525	113301
Feedstock Share Industry	19935	94593	94540	94596	94577	94649	94724	94652	94627	94615	94607
Elec Share Industry	200009	200031	299177	266179	199975	139574	246036	141506	199968	200002	199977

Table 8.23: Convergence quantile and income for each quantity and region for SSP3 (for region descriptions, see: Regions)

SSP3	AFR	CPA	EEU	FSU	LAM	MEA	NAM	PAO	PAS	SAS	WEU
Convergence Quantile											
Final Energy Intensity (FEI)	0.6	0.55	0.5	0.7	0.7	0.5	0.7	0.5	0.5	0.7	0.6
Share NC Biomass	0.9	0.6	0.75	0.75	0.25	0.75	0.75	0.75	0.6	0.9	0.75
Share Transport	0.1	0.05	0.7	0.2	0.45	0.5	0.7	0.25	0.5	0.1	0.7
Share Res/Com	0.25	0.25	0.55	0.55	0.3	0.5	0.35	0.6	0.25	0.2	0.5
Share Industry	0.1	0.6	0.2	0.1	0.2	0.2	0.1	0.1	0.6	0.2	0.1
Elec Share Res/Com	0.4	0.6	0.45	0.4	0.9	0.9	0.25	0.65	0.9	0.6	0.33
Feedstock Share Industry	0.2	0.22	0.26	0.24	0.2	0.3	0.32	0.29	0.3	0.22	0.27
Elec Share Industry	0.3	0.43	0.37	0.45	0.3	0.4	0.35	0.45	0.4	0.35	0.4
Convergence Income											
Final Energy Intensity (FEI)	200009	200031	200006	200044	199975	200027	200109	199995	199968	200002	199977
Share NC Biomass	13955	26294	80927	40951	12023	80953	80782	132077	12268	12771	48153
Share Transport	13955	46015	59835	51188	70131	69787	80782	132077	32715	55525	81010
Share Res/Com	23922	65735	59835	61426	80952	52340	80782	80816	199968	80512	81010
Share Industry	5981	52588	200006	122857	18034	43617	200109	199995	81787	30539	198277
Elec Share Res/Com	80976	80986	80927	61426	80952	69787	80782	80816	80969	80956	81010
Feedstock Share Industry	19935	26294	80927	80980	80952	80953	80782	80816	80969	80956	81010
Elec Share Industry	200009	200031	200006	200044	199975	200027	200109	199995	199968	200002	199977

Modeling policies

The global energy model distinguishes between eleven global regions (cf. Section *Regions*). It is nevertheless important to represent current and planned national policies - such as the nationally determined contributions (NDCs) as agreed upon in the Paris Agreement - at a lower geographical resolution, in order to be able to adequately account for future changes in the scenario development processes.

Representation of single country Nationally Determined Contributions (NDCs)

The targets formulated in the NDCs come in many different flavors. This applies to the sectors and gases covered by these policies, but it also applies to how these are expressed and quantified. In the global energy model, four broad categories of policy types related to the NDCs are represented, each of which is translated into a set of constraints.

1. Emission targets
2. Energy shares
3. Capacity or generation targets
4. Macro-economic targets

A detailed description of the methodological implementation of the NDCs in the global energy model, along with an extensive list of the energy-related targets considered can be found in Rogelj et al. (2017) [91]. Additional policies implemented in the model can also be found in Roelfsema et al. (2020) [90].

Emission targets

Country-specific emission reduction targets are specified either in relation to historical emissions (e.g. x% reduction compared to 1990) or in relation to a reference emission trajectory (in the form of a baseline or business as usual scenario (BAU); e.g. x% reduction compared to 2030 emission levels in the baseline). The targets themselves are expressed as either (1) absolute reduction, (2) a percentage reduction or (3) intensity reductions e.g. emissions per GDP or per capita. In order to account for these different reduction targets in the global energy model, the targets are translated so that a regionally specific upper bound on emissions can be formulated. If not further specified, emission constraints are assumed to apply to all sectors and all gases, i.e. total GHGs.

Energy shares

Energy share targets refer to any target which aims to provide a specific energy level (e.g. primary, secondary or final energy) through a specific sub-set of energy forms. The five different forms in which these are formulated in the NDCs are: (1) renewable energy as share of total primary energy, (2) non-fossil energy forms as share of total primary energy, (3) renewable energy as a share of total electricity generation, (4) non-fossil energy as a share of total electricity generation, (5) renewable energy as a form of final energy. All of these share constraint variants can be implemented in the model using the following mathematical formulation. In order to be able to implement these for aggregate regions, it is necessary to harmonize these to single type of share constraint, so that their effects are considered cumulatively within a region. All variants are therefore harmonized to either the share type specified by the largest country, in terms of share of energy within a region, or the most frequently specified type within a region. Separately biofuel shares are implemented specifically for the transport sector.

Capacity and generation targets

Some NDCs specify capacity installation targets, e.g. for planned power plants which will be operational by a certain year. Others specify that a given energy commodity will come from a specific source, for example a certain amount of electricity will stem form a specific intermittent renewable source or nuclear. These targets types are implemented in the model as lower bounds on generation.

Macro-economic targets

Representation of taxes and subsidies

Another set of policies addressed as part of climate change analysis, are energy-related taxes and subsidies. Removing fossil fuel subsidies could help reduce emissions by discouraging the use of inefficient energy forms. In the global energy model, fossil fuel prices are endogenously derived based on underlying supply curves representing the technical costs associated with the extraction of the resources (cf. Section *Fossil Fuel Reserves and Resources*). Refining and processing as well as transmission and distribution costs will be added to the total fuel cost. In order to account for taxes, price adjustment factors are applied, based on the underlying data set as described in Jewell et al. (2018) [36].

8.14.4 Macro-economy (MACRO)

The detailed energy supply model (MESSAGE) is soft-linked to an aggregated, single-sector macro-economic model (MACRO) which has been derived from the so-called Global 2100 or ETA-MACRO model (Manne and Richels, 1992 [51]), a predecessor of the MERGE model. The reason for linking the two models is to consistently reflect the influence of energy supply costs, as calculated by MESSAGE, the mix of production factors considered in MACRO, and the effect of changes in energy prices on energy service demands. The combined MESSAGE-MACRO model (Messner and Schrattenholzer, 2000 [61]) can generate a consistent economic response to changes in energy prices and estimate overall economic consequences (e.g., changes in GDP or household consumption) of energy or climate policies.

MACRO is a macroeconomic model maximizing the intertemporal utility function of a single representative producer-consumer in each world region. The optimization result is a sequence of optimal savings, investment, and

consumption decisions. The main variables of the model are capital stock, available labor, and energy inputs, which together determine the total output of an economy according to a nested CES (constant elasticity of substitution) production function. End-use service demands in the (commercial) demand categories of MESSAGE (see *Energy demand*) is determined within the MACRO model, and is consistent with energy supply from MESSAGE, which is an input to the MACRO. The model's most important driving input variables are the projected growth rates of total labor, i.e., the combined effect of labor force and labor productivity growth, and the annual rates of reference energy intensity reduction, i.e. the so-called autonomous energy efficiency improvement (AEEI) coefficients. The latter are calibrated to the developments in a MESSAGE baseline scenario to ensure consistency between the two models. Labor supply growth is also referred to as reference or potential GDP growth. In the absence of price changes, energy demands grow at rates that are the approximate result of potential GDP growth rates, reduced by the rates of overall energy intensity reduction. Price changes of the six demand categories, for example induced by energy or climate policies, can alter this path significantly.

MACRO's production function includes six commercial energy demand categories represented in MESSAGE. To optimize, MACRO requires cost information for each demand category. The exact definitions of these costs as a function over all positive quantities of energy cannot be given in closed form because each point of the function would be a result of a full MESSAGE run. However, the optimality conditions implicit in the formulation of MACRO only require the functional values and its derivatives at the optimal point to be consistent between the two models. Since these requirements are therefore only local, most functions with this feature will simulate the combined energy-economic system in the neighborhood of the optimal point. The regional costs (of energy use and imports) and revenues (from energy exports) of providing energy in MACRO are approximated by a Taylor expansion to first order of the energy system costs as calculated by MESSAGE. From an initial MESSAGE model run, the total energy system cost (including costs/revenues from energy trade) and additional abatement costs (e.g., abatement costs from non-energy sources) as well as the shadow prices of the six commercial demand categories by region are passed to MACRO. In addition to the economic implications of energy trade, the data exchange from MESSAGE to MACRO may also include the revenues or costs of trade in GHG permits.

Consult the MACRO section of the MESSAGEix documentation for a description of the MACRO system of equations, its implementation in `message_ix`, parameterization, and calibration procedure.

8.14.5 Land-use (GLOBIOM)

Land-use dynamics are modelled with the GLOBIOM (GLobal BIOsphere Management) model, which is a partial-equilibrium model (Havlik et al., 2011 [25]; Havlik et al., 2014 [26]). GLOBIOM represents the competition between different land-use based activities. It includes a detailed representation of the agricultural, forestry and bio-energy sector, which allows for the inclusion of detailed grid-cell information on biophysical constraints and technological costs, as well as a rich set of environmental parameters, incl. comprehensive AFOLU (agriculture, forestry and other land use) GHG emission accounts and irrigation water use. For spatially explicit projections of the change in afforestation, deforestation, forest management, and their related CO2 emissions, GLOBIOM is coupled with the G4M (Global FORest Model) model (Kindermann et al., 2006 [45]; Kindermann et al., 2008 [43]; Gusti, 2010 [22]). The spatially explicit G4M model compares the income of forest (difference of wood price and harvesting costs, income by storing carbon in forests) with income by alternative land use on the same place, and decides on afforestation, deforestation or alternative management options. As outputs, G4M provides estimates of forest area change, carbon uptake and release by forests, and supply of biomass for bioenergy and timber.

As a partial equilibrium model representing land-use based activities, including agriculture, forestry and bioenergy sectors (see Fig. 8.19), production adjusts to meet the demand at the level of 30 economic regions (see list of the regions in Section *Regions*). International trade representation is based on the spatial equilibrium modelling approach, where individual regions trade with each other based purely on cost competitiveness because goods are assumed to be homogenous (Takayama and Judge, 1971 [109]; Schneider, McCarl et al., 2007 [102]). Market equilibrium is determined through mathematical optimization which allocates land and other resources to maximize the sum of consumer and producer surplus (McCarl and Spreen, 1980 [52]). As in other partial equilibrium models, prices are endogenous. The model is run recursively dynamic with a 10 year time step, going from 2000 to 2100. The model is solved using a linear programming solver and can be run on a personal computer with the GAMS software.

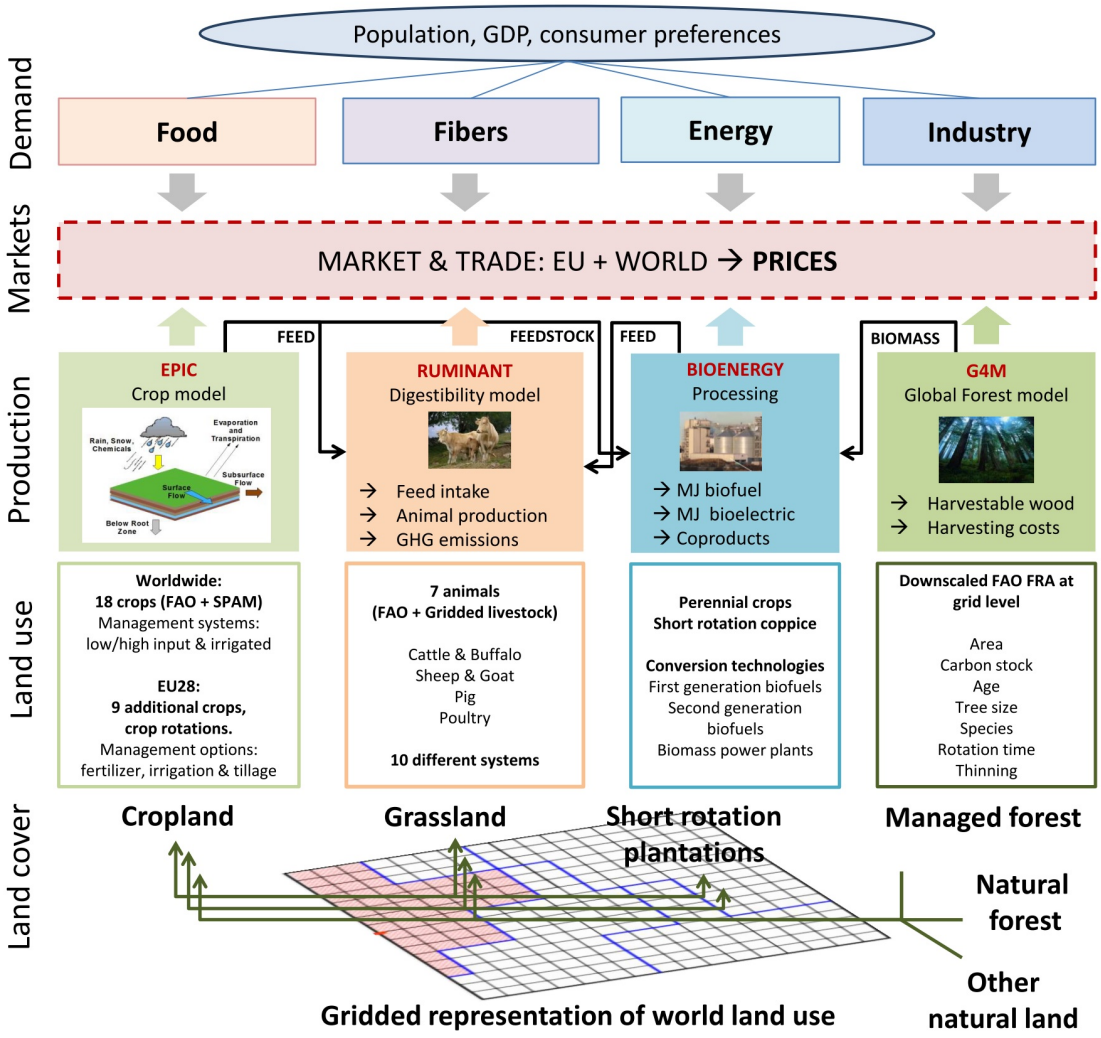

Fig. 8.19: GLOBIOM land use and product structure.

Spatial resolution

Land resources and their characteristics are the fundamental elements of the GLOBIOM modelling approach. In order to enable global bio-physical process modelling of agricultural and forest production, a comprehensive database has been built (Skalsky et al., 2008 [105]), which contains geo-spatial data on soil, climate/weather, topography, land cover/use, and crop management (e.g. fertilization, irrigation). The data were compiled from various sources (FAO, ISRIC, USGS, NASA, CRU UEA, JRC, IFRPI, IFA, WISE, etc.) and significantly vary with respect to spatial, temporal, and attribute resolutions, thematic relevance, accuracy, and reliability. Therefore, data were harmonized into several common spatial resolution layers including 5 and 30 Arcmin as well as country layers. Subsequently, Homogeneous Response Units (HRU) have been delineated by geographically clustering according to only those parameters of the landscape, which are generally not changing over time and are thus invariant with respect to land use and management or climate change. At the global scale, five altitude classes, seven slope classes, and five soil classes have been included.

In a second step, the HRU layer is intersected with a 0.5 x 0.5 degree grid and country boundaries to delineate Simulation Units (SimUs) which contain other relevant information such as global climate data, land category/use data, irrigation data, etc. In total, 212,707 SimUs are delineated by clustering 5 x 5 minutes of arc pixels according to five criteria: altitude, slope, and soil class, 0.5 x 0.5 degrees grid, and the country boundaries. The SimUs are the basis for estimation of land use/management parameters in all other supporting models as well. For each SimU a number of land management options are simulated using the bio-physical process model EPIC (Environmental Policy Integrated Climate) (Izaurralde et al., 2006 [35]; Williams and Singh, 1995 [112]). For the SSP application of GLOBIOM, in order to ease computation time, the input data sets and the model resolution were agregated to 2 x 2 degree cells disaggregated only by country boundaries and by three agro-ecological zones used in the livestock production system classification: arid, humid, temperate and tropical highlands. This led to a total of 10,894 different Supply Units.

Crop production

GLOBIOM directly represents production from three major land cover types: cropland, managed forest, and areas suitable for short rotation tree plantations. Crop production accounts for more than 30 of the globally most important crops. The average yield level for each crop in each country is taken from FAOSTAT. Management related yield coefficients according to fertilizer and irrigation rates are explicitly simulated with the EPIC model (Williams and Singh, 1995 [112]) for 17 crops (barley, dry beans, cassava, chickpea, corn, cotton, ground nuts, millet, potatoes, rapeseed, rice, soybeans, sorghum, sugarcane, sunflower, sweet potatoes, and wheat). These 17 crops together represent nearly 80 % of the 2007 harvested area and 85% of the vegetal calorie supply as reported by FAOSTAT. Four management systems are considered (irrigated, high input - rainfed, low input - rainfed and subsistence management systems) corresponding to the International Food and Policy Research Institute (IFPRI) crop distribution data classification (You and Wood, 2006 [114]). Within each management system, input structure is fixed following a Leontieff production function. But crop yields can change in reaction to external socio-economic drivers through switch to another management system or reallocation of the production to a more or less productive Supply Unit.

Besides the endogennous mechanisms, an exogenous component representing long-term technological change is also considered. Only two management systems are differentiated for the remaining crops (bananas, other dry beans, coconuts, coffee, lentils, mustard seed, olives, oil palm, plantains, peas, other pulses, sesame seed, sugar beet, and yams) – rainfed and irrigated. Rainfed and irrigated crop yield coefficients, and crop specific irrigation water requirements for crops not simulated with EPIC, and costs for four irrigation systems for all crops, are derived from a variety of sources as described in Sauer et al. (2008) [98]. Crop supply can enter one of three processing/demand channels: consumption, livestock production and biofuel production (see Fig. 8.19).

Livestock

Livestock population

The principal variable characterizing the livestock production in GLOBIOM is the number of animals by species, production system and production type in each Simulation Unit. GLOBIOM differentiates four species aggregates: cattle and buffaloes (bovines), sheep and goats (small ruminants), pigs, and poultry. Eight production systems are specified for ruminants: grazing systems in arid (LGA), humid (LGH) and temperate/highland areas (LGT); mixed systems in arid (MXA), humid (MXH) and temperate/highland areas (MXT); urban systems (URB); and other systems (OTH). Mixed systems are an aggregate of the more detailed original Sere and Steinfeld's classes (Sere and Steinfeld, 1996 [104]) – mixed rainfed and mixed irrigated. Two production systems are specified for monogastrics: smallholders (SMH) and industrial systems (IND). In terms of production type, dairy and meat herds are modeled separately for ruminants: dairy herd includes adult females and replacement heifers, whose diets are distinguished. Poultry in smallholder systems is considered as mixed producer of meat and eggs, and poultry in industrial systems is split into laying hens and broilers, with differentiated diet regimes. Overall livestock numbers at the country level are, where possible while respecting minimum herd dynamics rules, harmonized with FAOSTAT.

The spatial distribution of ruminants and their allocation between production systems follows an updated version of Wint and Robinson (Wint and Robinson, 2007 [113]). Since better information is not available, it is assumed that the share of dairy and meat herds within one region is the same in all production systems. The share is obtained from the FAO country level data about milk producing animals and total herd size. Monogastrics are not treated in a spatially explicit way since no reliable maps are currently available, and because monogastrics are not linked in the model to specific spatial features, like grasslands. The split between smallholder and industrial systems follows Herrero et al. (2013) [27].

Livestock products

Each livestock category is characterized by product yield, feed requirements, and a set of direct GHG emission coefficients. On the output side, seven products are defined: bovine meat and milk, small ruminant meat and milk, pig meat, poultry meat, and eggs. For each region, production type and production system, individual productivities are determined.

Bovine and small ruminant productivities are estimated through the RUMINANT model (Herrero et al., 2008 [28]; Herrero et al., 2013 [27]), in a three steps process which consists of first, specifying a plausible feed ration; second, calculating in RUMINANT the corresponding yield; and finally confronting at the region level with FAOSTAT (Supply Utilization Accounts) data on production. These three steps were repeated in a loop until a match with the statistical data was obtained. Monogastrics productivities were disaggregated from FAOSTAT based on assumptions about potential productivities and the relative differences in productivities between smallholder and industrial systems. The full detail of this procedure is provided in Herrero et al. (2013) [27].

Final livestock products are expressed in primary commodity equivalents. Each product is considered as a differentiated good with a specific market except for bovine and small ruminant milk that are merged in a single milk market. The two milk types are therefore treated as perfect substitutes.

Livestock feed

Feed requirements for ruminants are computed simultaneously with the yields (Herrero et al., 2013 [27]). Specific diets are defined for the adult dairy females, and for the other animals. The feed requirements are first calculated at the level of four aggregates – grains (concentrates), stover, grass, and other. When estimating the feed-yield couples, the RUMINANT model takes into account different qualities of these aggregates across regions and systems. Feed requirements for monogastrics are at this level determined through literature review presented in Herrero et al. (2013) [27]. In general, it is assumed that in industrial systems pigs and poultry consume 10 and 12 kg dry matter of concentrates per TLU and day, respectively, and concentrates are the only feed sources. Smallholder animals get only one quarter of the amount of grains fed in industrial systems, the rest is supposed to come from other sources, like household waste, not explicitly represented in GLOBIOM.

The aggregate GRAINS input group is harmonized with feed quantities as reported at the country level in Commodity Balances of FAOSTAT. The harmonization proceeds in two steps, where first, GRAINS in the feed rations are

adjusted so that total feed requirements at the country level match with total feed quantity in Commodity Balances, and second, "Grains" is disaggregated into 11 feed groups: Barley, Corn, Pulses, Rice, Sorghum & Millet, Soybeans, Wheat, Cereal Other, Oilseed Other, Crops Other, Animal Products. The adjustment of total GRAINS quantities is first done through shifts between the GRAINS and OTHER categories in ruminant systems. Hence, if total GRAINS are lower than the statistics, a part or total feed from the OTHER category is moved to GRAINS. If this is not enough, all GRAINS requirements of ruminants are shifted up in the same proportions. If total GRAINS are higher than the statistics, then firstly a part of them must be reallocated to the OTHER category. If this is not enough, values are to be kept, which then results in higher GRAINS demand than reported in FAOSTAT. This inconsistency is overcome in GLOBIOM, by creating a "reserve" of the missing GRAINS. This reserve is in simulations kept constant, thus it enables to reproduce the base year activity levels mostly consistent with FAOSTAT, but requires that all additional GRAINS demand arising over the simulation horizon is satisfied from real production. The decomposition of GRAINS into the 11 subcategories has to follow predefined minima and maxima of the shares of feedstuffs in a ration differentiated by species and region. At the same time, the shares of the feedstuffs corresponding to country level statistics need to be respected. This problem is solved as minimization of the square deviations from the prescribed minimum and maximum limits. In GLOBIOM, the balance between demand and supply of the crop products entering the GRAINS subcategories needs to be satisfied at regional level. Substitution ratios are defined for the byproducts of biofuel industry so that they can also enter the feed supply.

STOVER is supposed less mobile than GRAINS, therefore stover demand in GLOBIOM is forced to match supply at grid level. The demand is mostly far below the stover availability. In the cells where this is not the case, the same system of reserve is implemented as for the grains. No adjustments are done to the feed rations as such.

There are unfortunately no worldwide statistics available on either consumption or production of grass. Hence grass requirements were entirely based on the values calculated with RUMINANT, and were used to estimate the grassland extent and productivity. (This procedure is described in the next section.)

Finally, the feed aggregate OTHER is represented in a simplified way, where it is assumed that it is satisfied entirely from a reserve in the base year, and all additional demand needs to be satisfied by forage production on grasslands.

Grazing forage availability

The demand and supply of grass need to match at the level of Simulation Unit in GLOBIOM. But reliable information about grass forage supply is not available even at the country level. The forage supply is a product of the utilized grassland area and of forage productivity. However, at global scale, Ramankutty et al. (2008) [78] estimated that the extent of pastures spans in the 90% confidence interval between 2.36 and 3.00 billion hectares. The FAOSTAT estimate of 3.44 billion hectares itself falls outside of this interval which illustrates the level of uncertainty in the grassland extent. Similarly, with respect to forage productivity, different grassland production models perform better for different forage production systems and all are confronted with considerable uncertainty due to limited information about vegetation types, management practices, etc. (Conant and Paustian, 2004 [8]). These limitations precluded reliance on any single source of information or output from a single model. Therefore three different grass productivity sources were considered: CENTURY on native grasslands, CENTURY on native and managed grasslands, and EPIC on managed grasslands.

A systematic process was developed for selecting the suitable productivity source for each of GLOBIOM's 30 regions. This process allowed reliance on sound productivity estimates that are consistent with other GLOBIOM datasets like spatial livestock distribution and feed requirements. Within this selection process, the area of utilized grasslands corresponding to the base year 2000 was determined simultaneously with the suitable forage productivity layer. Two selection criteria were used: livestock requirements for forage and area of permanent meadows and pastures from FAOSTAT. The selection process was based on simultaneous minimization of i) the difference between livestock demand for forage and the model-estimates of forage supply and ii) the difference between the utilized grassland area and FAOSTAT statistics on permanent meadows and pastures. Regional differentiation in grassland management intensity, ranging from dry grasslands with minimal inputs to mesic, planted pastures that are intensively managed with large external inputs – further informed the model selection by enabling constraints in the number of models for dry grasslands.

To calculate the utilized grassland area, the potential grassland area was first defined as the area belonging to one of the following GLC2000 land cover classes: 13 (Herbaceous Cover, closed-open), 16-18 (Cultivated and managed areas, Mosaic: Cropland / Tree Cover / Other natural vegetation, Mosaic: Cropland / Shrub and/or grass cover), excluding area identified as cropland according to the IFPRI crop distribution map (You and Wood, 2006 [114]), and 11, 12, 14 (Shrub Cover, closed-open, evergreen, Shrub Cover, closed-open, deciduous, Sparse herbaceous or

sparse shrub cover). In each Simulation Unit the utilized area was calculated by dividing total forage requirements by forage productivity. In Simulation Units where utilized area was smaller than the potential grassland area, the difference would be allocated to either "Other Natural Land" or "Other Agricultural Land" depending on the underlying GLC2000 class. In Simulation Units where the grassland area necessary to produce the forage required in the base year was larger than the potential grassland area, a "reserve" was created to ensure base year feasibility, but all the additional grass demand arising through future livestock production increases needed to be satisfied from grasslands.

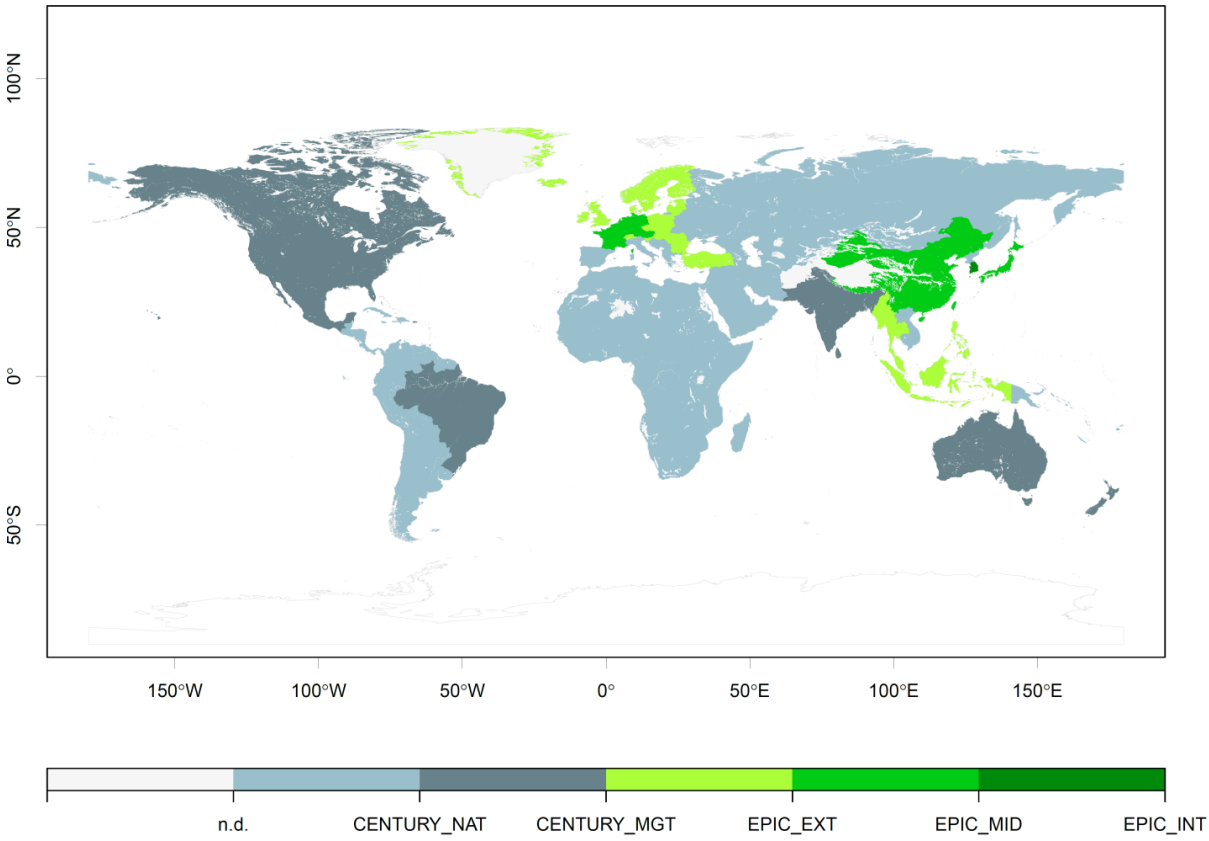

Fig. 8.20: Data sources used to parameterize forage availability in different world regions. CENTURY_NAT – CENTURY model for native grasslands; CENTURY_MGT – CENTURY model for productive grasslands; EPIC_EXT – EPIC model for grasslands under extensive management; EPIC_MID – EPIC model for grasslands under semi-intensive management; EPIC_INT – EPIC model for grasslands under intensive management.

Forage productivity was estimated using the CENTURY (Parton et al., 1987 [73]; Parton et al., 1993 [72]) and EPIC (Williams and Singh, 1995 [112]) models. The CENTURY model was run globally at 0.5 degree resolution to estimate native forage and browse and planted pastures productivity. It was initiated with 2000 year spin-ups using mean monthly climate from the Climate Research Unit (CRU) of the University of East Anglia with native vegetation for each grid cell, except cells dominated by rock, ice, and water, which were excluded. Information about native vegetation was derived from the Potsdam intermodal comparison study (Schloss et al., 1999 [100]). Plant community and land management (grazing) was based on growing-season grazing and 50 per cent forage removal. Areas under native vegetation that were grazed were identified using the map of native biomes subject to grazing and subtracting estimated crop area within those biomes in 2006 (Ramankutty et al., 2008 [78]). It is assumed 50 per cent grazing efficiency for grass, and 25 per cent for browse for native grasslands. These CENTURY-based estimates of native grassland forage production (CENTURY_NAT) were used for most regions with low-productivity grasslands (Fig. 8.20).

Both the CENTURY and EPIC models were used to estimate forage production in mesic, more productive regions. For the CENTURY model, forage yield was simulated using a highly-productive, warm-season grass parameterization. Production was modeled in all cells and applied to areas of planted pasture, which were estimated based on biomes that were not native rangelands, but were under pasture in 2006 according to Ramankutty (Ramankutty et al., 2008 [78]). Pastures were replanted in the late winter every ten years, with grazing starting in the second year. Observed monthly precipitation and minimum and maximum temperatures between 1901 and 2006 were from the

CRU Time Series data, CRU TS30 (Mitchell and Jones, 2005 [63]) Soils data were derived from the FAO Soil Map of the World, as modified by Reynolds et al. (2000) [84]. CENTURY model output for productive pastures (CENTURY_MGT) were the best-match for area/forage demand in much of the world with a mixture of mesic and drier pastures.

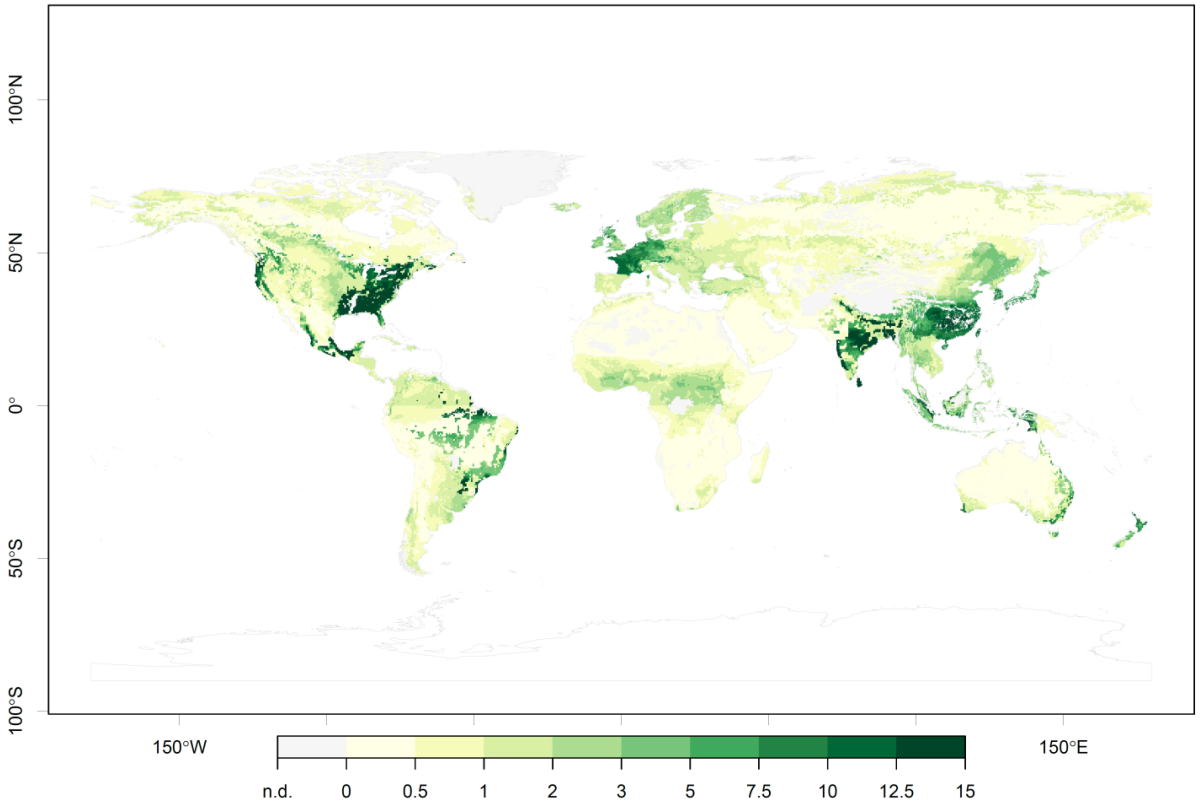

Fig. 8.21: Forage available for livestock in tonnes of dry matter per hectare as the result of combination of outputs from the CENTURY and EPIC models.

The EPIC model was the best fit for much of Europe and Eastern Asia, where most of the forage production is in intensively-managed grasslands. The EPIC simulations used the same soil and climatic drivers as the CENTURY runs plus topography data (high-resolution global Shuttle Radar Topography Mission digital elevation model (SRTM) and the Global 30 Arc Second Elevation Data (GTOPO30). Warm and cold seasonal grasses were simulated in EPIC, and the simulations included a range of management intensities represented by different levels of nitrogen fertilizer inputs and off-take rates. The most intensive management minimizing nitrogen stress and applying 80% off-take rates (EPIC_INT) was found to be the best match for South Korea. Highly fertilized grasslands but with an off-take rate of 50% only were identified in Western Europe, China and Japan (EPIC_MID), and finally extensive management, only partially satisfying the nitrogen requirements and considering 20% off-take rates corresponded best to Central and Northern Europe and South-East Asia (EPIC_EXT). The resulting hybrid forage availability map is represented in Fig. 8.21.

Livestock dynamics

In general, the number of animals of a given species and production type in a particular production system and Supply Unit is an endogenous variable. This means that it will decrease or increase in relation to changes in demand and the relative profitability with respect to competing activities.

Herd dynamics constraints need however to be respected. First, dairy herds are constituted of adult females and followers, and expansion therefore occurs in predefined proportions in the two groups. Moreover, for regions where the specialized meat herds are insignificant (no suckler cows), expansion of meat animals (surplus heifers and males) is also assumed proportional in size to the dairy herd. The ruminants in urban systems are not allowed to expand because this category is not well known and because it is fairly constrained by available space in growing cities. Finally, the decrease of animals per system and production type higher than 15 per cent per 10 years period are not

considered, and no increase by more than 100 per cent on the same period. At the level of individual systems, the decrease can however be as deep as 50 per cent per system on a single period.

For monogastrics, the assumption is made that all additional supply will come from industrial systems and hence the number of animals in other systems is kept constant (Keyzer et al., 2005 [42]).

Forestry

The forestry sector is represented in GLOBIOM with five categories of primary products (pulp logs, saw logs, biomass for energy, traditional fuel wood, and other industrial logs) which are consumed by industrial energy, cooking fuel demand, or processed and sold on the market as final products (wood pulp and sawnwood). These products are supplied from managed forests and short rotation plantations. Harvesting cost and mean annual increments are informed by the G4M global forestry model (Kindermann et al., 2006 [45]) which in turn calculates them based on thinning strategies and length of the rotation period.

Primary forest production from traditional managed forests is characterized also at the level of SimUs. The most important parameters for the model are mean annual increment, maximum share of saw logs in the mean annual increment, and harvesting cost. These parameters are shared with the G4M model – a successor of the model described by Kindermann et al. (2006) [45]. More specifically, mean annual increment for the current management, is obtained by downscaling biomass stock data from the Global Forest Resources Assessment (FAO, 2006 [14]) from the country level to a 0.5 x 0.5 degree grid using the method described in Kindermann et al. (2008) [44]. The downscaled biomass stock data is subsequently used to parameterize increment curves. Finally, the saw logs share is estimated by the tree size, which in turn depends on yield and rotation time. Harvesting costs are adjusted for slope and tree size as well. Among the five primary forest products, saw logs, pulp logs and biomass for energy are further processed. Sawn wood and wood pulp production and demand parameters rely on the 4DSM model described in Rametsteiner et al. (2007) [79]. FAO data and other secondary sources have been used for quantities and prices of sawn wood and wood pulp. For processing cost estimates of these products an internal IIASA database and proprietary data (e.g. RISI database for locations of individual pulp and paper mills, with additional economic and technical information, http://www.risiinfo.com) were used. Biomass for energy can be converted in several processes: combined heat and power production, fermentation for ethanol, heat, power and gas production, and gasification for methanol and heat production. Processing cost and conversion coefficients are obtained from various sources (Biomass Technology Group, 2005 [20]; Hamelinck and Faaij, 2001 [23]; Leduc et al., 2008 [48]; Sorensen, 2005 [106]). Demand for woody bioenergy production is implemented through minimum quantity constraints, similar to demand for other industrial logs and for firewood. Woody biomass for bioenergy can also be produced on short rotation tree plantations. To parameterize this land use type in terms of yields, an evaluation of the land availability and suitability was carried out. Calculated plantation costs involve the establishment cost and the harvesting cost. The establishment related capital cost includes only sapling cost for manual planting (Carpentieri et al., 1993 [5]; Herzogbaum GmbH, 2008 [29]). Labour requirements for plantation establishment are based on Jurvelius (1997) [39], and consider land preparation, saplings transport, planting and fertilization. These labour requirements are adjusted for temperate and boreal regions to take into account the different site conditions. The average wages for planting are obtained from ILO (2007) [33]. Harvesting cost includes logging and timber extraction. The unit cost of harvesting equipment and labour is derived from various datasets for Europe and North America (e.g. FPP, 1999 [16]; Jiroušek et al., 2007 [37] ; Stokes et al., 1986 [107] ; Wang et al., 2004 [111]). Because the productivity of harvesting equipment depends on terrain conditions, a slope factor (Hartsough et al., 2001 [24]) was integrated to estimate total harvesting cost. The labour cost, as well as the cost of saplings, is regionally adjusted by the ratio of mean PPP (purchasing power parity over GDP), (Heston et al., 2006 [30]).

Land use change

The model optimizes over six land cover types: cropland, grassland, short rotation plantations, managed forests, unmanaged forests and other natural land. Economic activities are associated with the first four land cover types. There are other three land cover types represented in the model: other agricultural land, wetlands, and not relevant (bare areas, water bodies, snow and ice, and artificial surfaces). These three categories are currently kept constant. Each Simulation Unit can contain the nine land cover types. The base year spatial distribution of land cover is based on the Global Land Cover 2000 (GLC2000). However, as any other global dataset of this type, GLC2000 suffers from large uncertainty (Fritz et al., 2011 [19]). Therefore auxiliary datasets and procedures are used to transform this "raw" data into a consistent dataset corresponding to the model needs.

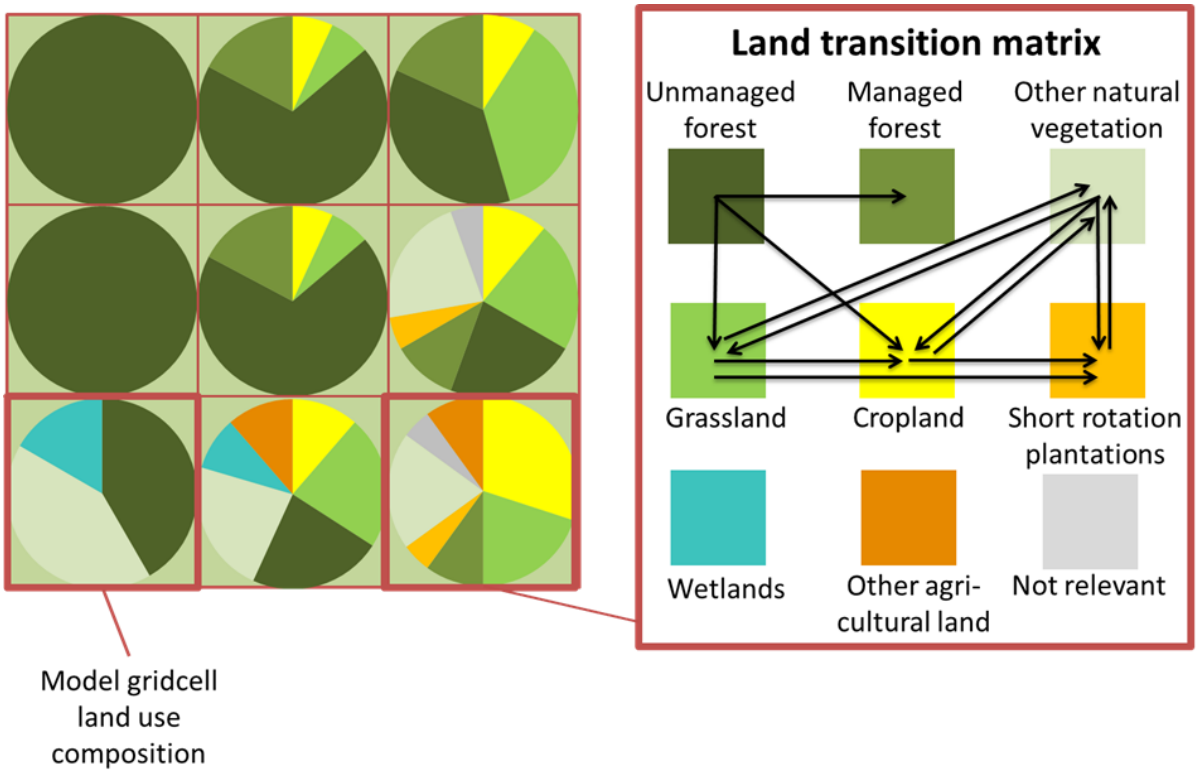

Fig. 8.22: Land cover representation in GLOBIOM and the matrix of endogenous land cover change possibilities (Havlik et al., 2014 [26]).

Land conversion over the simulation period is endogenously determined for each Supply Unit within the available land resources. Such conversion implies a conversion cost – increasing with the area of land converted - that is taken into account in the producer optimization behavior. Land conversion possibilities are further restricted through biophysical land suitability and production potentials, and through a matrix of potential land cover transitions (Fig. 8.22).

Food demand

Food demand is in GLOBIOM endogenous and depends on population, gross domestic product (GDP) and own produt price. Population and GDP are exogenous variables while prices are endogenous. The simple demand system is presented in Eq. Eq.8.1. First, for each product i in region r and period t, the prior demand quantity Q is calculated as a function of population POP, GDP per capita GDP^{cap} adjusted by the income elasticity ε^{GDP}, and the base year consumption level as reported in the Food Balance Sheets of FAOSTAT. If the prior demand quantity could be satisfied at the base year price P, this would be also the optimal demand quantity Q. However, usually the optimal quantity will be different from the prior quantity, and will depend on the optimal price P and the price elasticity ε^{price}, the latter calculated from USDA (Seale et al., 2003 [103]), updated in Muhammad et al. (2011) [64] for the base year 2000. Because food demand in developed countries is more inelastic than in developing ones, the value of this elasticity is assumed to decrease with the level of GDP per capita. The rule applied is that the price elasticity of

developing countries converges to the price elasticity of the USA in 2000 at the same pace as their GDP per capita reach the USA GDP per capita value of 2000. This allows capturing the effect of change in relative prices on food consumption taking into account heterogeneity of responses across regions, products and over time.

$$\frac{Q_{i,r,t}}{\overline{Q}_{i,r,t}} = \left(\frac{P_{i,r,t}}{\overline{P}_{i,r,2000}} \right)^{\varepsilon_{i,r,t}^{price}} \tag{8.1}$$

where

$\overline{Q}_{i,r,t} = \frac{POP_{r,t}}{POP_{r,2000}} \times \left(\frac{GDP_{r,t}^{cap}}{GDP_{r,2000}^{cap}} \right)^{\varepsilon_{i,r,t}^{price}} \times \overline{Q}_{i,r,2000}$

This demand function has the virtue of being easy to linearize as GLOBIOM is solved as a linear program. This is currently necessary because of the size of the model and the performance of non-linear solvers. However, this demand function has although some limitations which need to be kept in mind when considering the results obtained with respect to climate change mitigation and food availability. One of them is that it does not consider direct substitution effects on the consumer side which could be captured through cross price demand elasticities. Such a demand representation could lead to increased consumption of some products like legumes or cereals when prices of GHG intensive products like rice or beef would go up as a consequence of a carbon price targeting emissions for the agricultural sector. Neglecting the direct substitution effects may lead to an overestimation of the negative impact of such mitigation policies on total food consumption. However, the effect on emissions would be only of second order, because consumption would increase for commodities the least affected by the carbon price, and hence the least emission intensive. Although direct substitution effects on the demand side are not represented, substitution can still occur due to changes in prices on the supply side and can in some cases lead to a partial compensation of the decreased demand for commodities affected the most by a mitigation policy.

Land-Use Emulator

The land-use emulator integrates a set of land-use scenarios into MESSAGEix energy system model. These land-use scenarios are developed by an economic land-use model GLOBIOM, which can assess competition for land-use between agriculture, bioenergy, and forestry. The land-use scenarios represent a two dimensional scenario matrix (so called Lookup-Table) combining different carbon and biomass price trajectories which allows to represent biomass supply curves conditional on different carbon prices as well as marginal abatement cost curves conditional on different biomass prices for the land-use sector in MESSAGEix. This linkage between an energy model, here MESSAGEix, and a land-use model is important to explore the potential of bioenergy and the implications of using biomass for energy generation on emissions, the cost of the system, and related land-use implications. In MESSAGEix formulation, there is a dedicated set of land use equations, to establish this linkage as follows. Each land-use scenario represents a distinct land-use development pathway for a given biomass potential and carbon price. The biomass potentials for use in the energy sector are determined by the biomass price. At lower biomass prices, biomass mainly stems from forest residues, for example from sawmills or logging residues. With increasing prices, land-use will be shifted to make room for fast-rotation tree plantations, purposely grown for use in energy production which may cause indirectly through increased competition with agricultural land deforestation of today's forest. At very high prices, roundwood will be harvested for energy production (for further details see *Forestry*) competing with material uses. In addition, for each level of biomass potential, different carbon prices reflect the cost of mitigation for land-use related greenhouse gas (GHG) emissions. For example, the matrix depicted below (Fig. 8.23) illustrates the combination of biomass and carbon prices for each of which a distinct land-use scenario has been provided by GLOBIOM.

In their entirety, the combination of these distinct land-use pathways provide MESSAGEix with a range of biomass potentials available for energy generation at different costs, so called BIO-categories, along with the associated land-use related emissions (CO2, CH4 and N2O). The different carbon prices provide MESSAGEix with options for mitigating land-use related GHG emissions, referred to as GHG-categories. The combination of land-use pathways can therefore be depicted as a trade-off surface, illustrated for SSP2 (Fricko et al., 2017 [17]) in the figure below (Fig. 8.24). The figure depicts global biomass potentials and respective GHG emissions at different carbon prices cumulated from 2010 to 2100.

From the trade-off surface it possible to deduct that for a MESSAGEix scenario without climate policy, land-use pathways of the lower BIO-categories and lowest GHG-categories will be used. The energy system will therefore only use biomass for energy production to the extent that it is economically viable without mitigating emissions. When climate policy scenarios are run in MESSAGEix, the land-use pathways will be chosen such that the optimal balance between the land-use related emission and biomass use in the energy system is obtained. In addition to

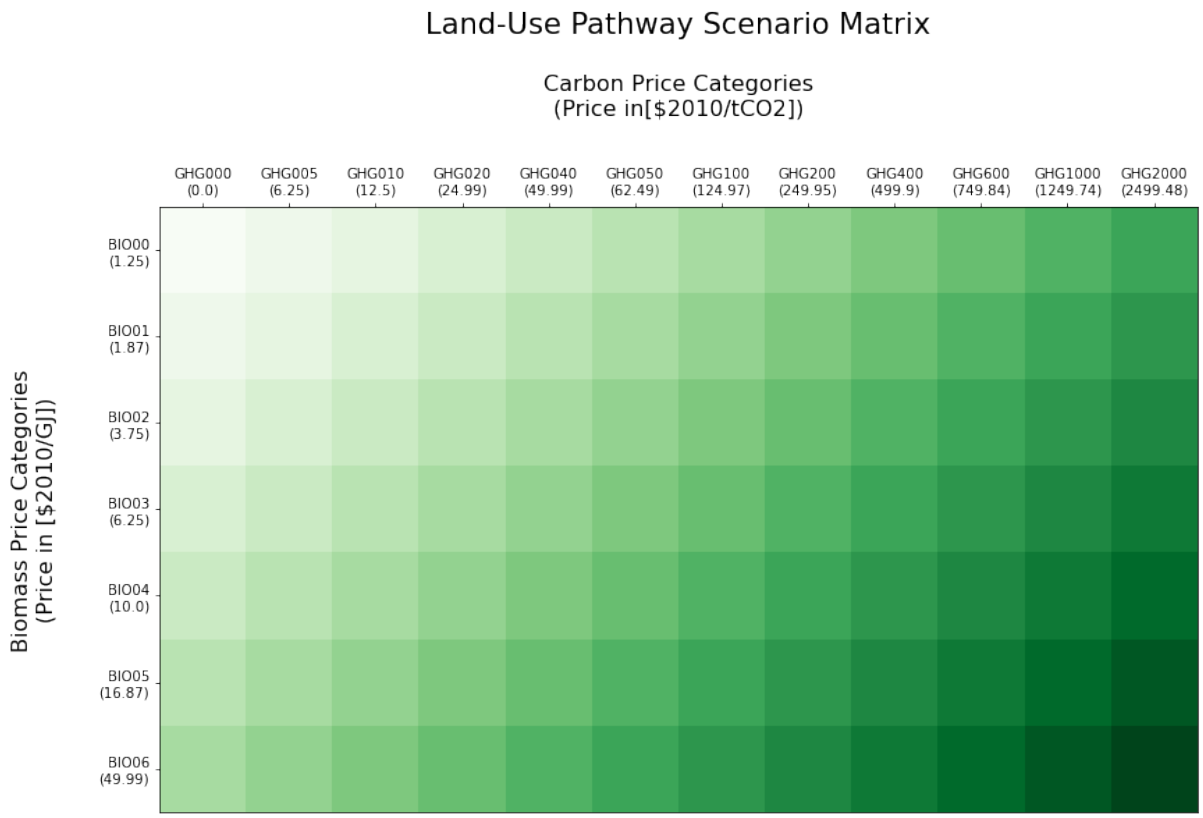

Fig. 8.23: Land-Use Scenario Matrix.

serving as a commodity from which energy can be generated, biomass can also be used to obtain negative emissions via BECCS.

Adaptation of the Reference-Energy-System (RES)

Prior to the use of the land-use emulator, biomass supply-curves were used to inform the energy system of the biomass availability. The emulator replaces supply-curves, by incorporating all the land-use scenarios in MESSAGEix, therefore the choice of which land-use pathway(s) becomes part of the entire optimization problem. Conceptually, each land-use scenario is incorporated similarly to any other technology in MESSAGEix, each providing biomass at a given price and corresponding GHG-emissions. The incorporation of the land-use emulator requires two changes to the RES to be undertaken. On the on hand, an additional level/commodity has been introduced to link the land-use pathways with the energy system, while land-use emissions are accounted for in the emissions equation (emissions equations in MESSAGEix).

Biomass, independent of the type of feedstock, is treated as a single commodity in the energy system. Bioenergy can therefore be used for use in power generation or liquefaction or gasification process alike (see *Other conversion* for further details). The only exception is made for non-commercial biomass (fuel wood). Non-commercial biomass supply and demand have been aligned between the two models. These are derived based on population and GDP projections for each of the SSP storyline projections (Riahi et al., 2017 [85], Pachauri et al., 2017 [68]). In MESSAGEix, non-commercial biomass is explicitly modeled as a demand category (see *Energy demand* for further details). The reduction of non-commercial biomass demand therefore is not possible in the global energy model, without the use of an additional addon module specifically developed to address this issue (Poblete et al., 2018 [75], Poblete et al., [76]). The reason for this is the fact that non-commercial biomass is not a traded commodity and therefore its use us not determined as a function of cost.

Note, that because each of the land-use pathways has been calculated accounting for mitigation of all GHGs, MESSAGEix scenarios aiming to only reduce a single green-house-gas for example, will either need to account for the fact that a price on CH4 for example will equally result in reductions of CO2 and N2O in the land-use sector. Equally, other land-use policies, such as the limitation of deforestation, can be implemented, but will most likely include other

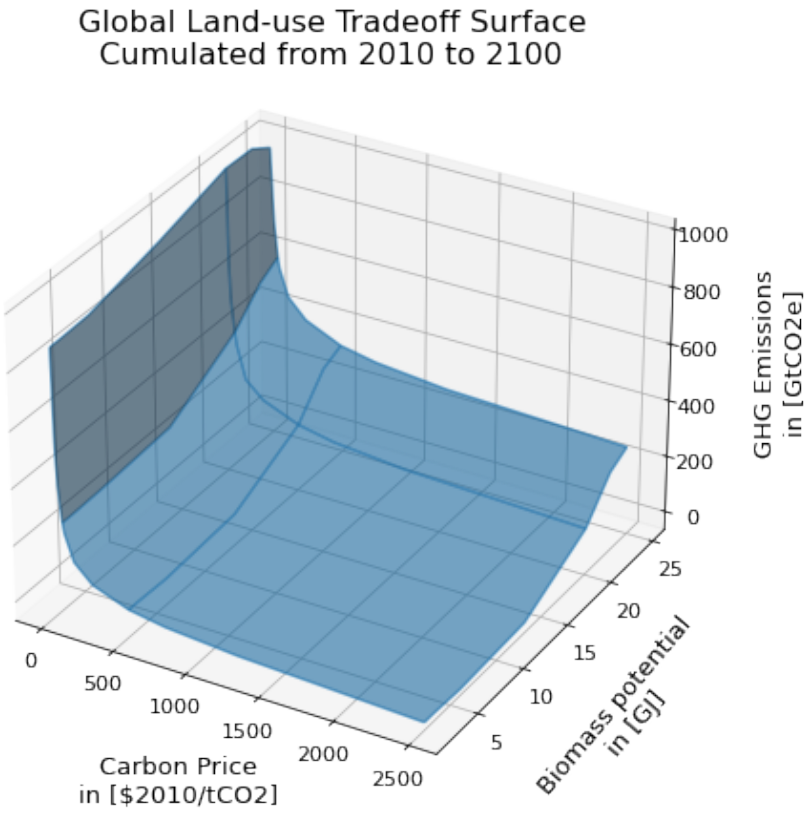

Fig. 8.24: Land-Use Pathway Trade-Off Surface for SSP2.

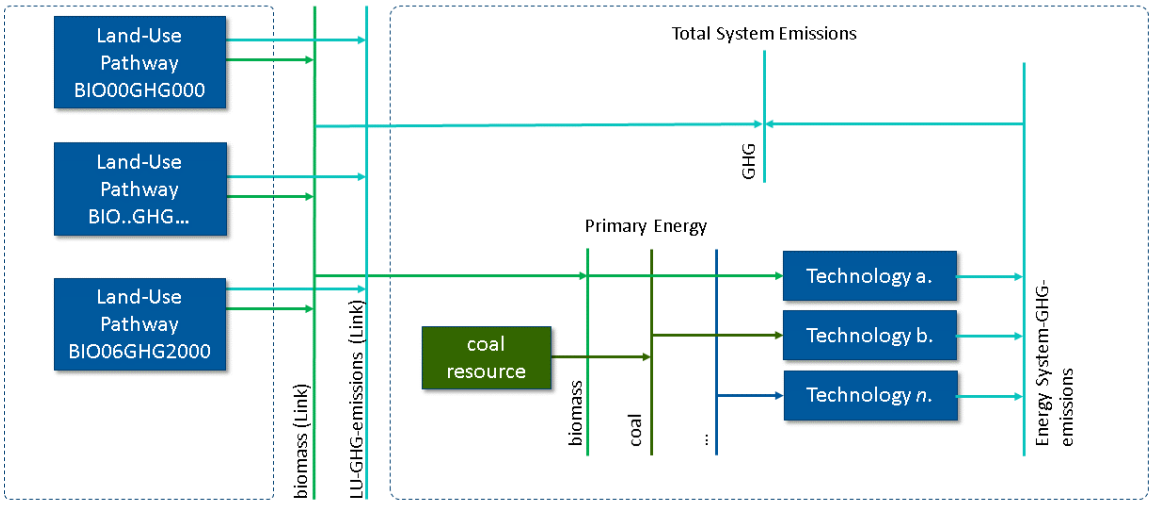

Fig. 8.25: Adaptations of a simplified RES for inclusion of the land-use emulator.

land-use related trends, which are artifacts as opposed to results of the policy, due to the limitations of using an emulator, and therefore a limited solution space. The land-use pathways are meant to represent the broad, as opposed to a specific policy landscape, consistent with SSP storylines (Popp et al., 2017 [77]). For some larger projects or studies, matrixes, i.e. input data sets from GLOBIOM, can be tailored to allow the analysis of specific policies in MESSAGE.

Equations and constraints

The land use equations in MESSAGEix state that the linear combination of land-use pathways must be equal to 1 (Eq.8.2). Therefore, separately for each region, either a single discrete land-use scenario can be used, or shares of multiple scenarios can be combined linearly to obtain, for example, biomass quantities which are not explicitly represented as part of the land-use matrix. This also applies to the mitigation dimension, i.e., to the GHG categories.

$$\sum_{s \in S} LAND_{n,s,y} = 1 \tag{8.2}$$

In order to correctly represent the transitional dynamics between land-use pathways, such as the rate at which changes in land-use can occur, e.g. the conversion from land-type A to land-type B, additional constraints are required as the underlying dependencies between these land-use pathways are only represented in the full fletched GLOBIOM model. Based on rates derived from GLOBIOM, for each of MESSAGEix model regions, the upscaling of plantation forest area is limited using dynamic constraints on land-use. The total area of plantation forest in a given region and time-period is determined, by summing up the shares of area (Mha) for other land types (crop-, grass- and other natural land) in the previous time-period in that region (Eq.8.3). Therefore, the bigger area for the three land types is available, the bigger plantation forest area can be expanded in the following time-period. This growth constraint is applied for each land-use pathway individually.

$$plantation\_forest_{n,s,y} <= crop\_land_{n,s,y-1} * X_n + grass\_land_{n,s,y-1} * Y_n + other\_natural\_land_{n,s,y-1} * Z_n \tag{8.3}$$

The table below shows the shares of each land type for each region, X_n, Y_n, Z_n. (for further details see *Land use change*).

Table 8.24: Shares of land-type by region used to derive the growth rate of plantation forest.

Region	Crop land [%], X_n	Grass land [%], Y_n	Other natural land [%], Z_n
Sub-Saharan Africa	0.05	0.05	0.05
Centrally Planned Asia and China	0.05	0.05	0.02
Central and Eastern Europe	0.05	0.02	0.02
Former Soviet Union	0.05	0.05	0.02
Latin America and the Caribbean	0.05	0.05	0.05
Middle East and North Africa	0.05	0.05	0.05
North America	0.05	0.05	0.02
Pacific OECD	0.05	0.05	0.05
Other Pacific Asia	0.05	0.05	0.05
South Asia	0.05	0.05	0.05
Western Europe	0.05	0.02	0.02

The growth constraint on plantation forest upscaling therefore implies that, should high quantities of biomass be required in the energy system, either a combination of land-use pathways needs to be used over time that will allow enough plantation forest area to be available under this specific constraint or alternatively land-use pathways corresponding to the highest BIO-category could be used from the very beginning of the century. The latter would require the energy system to transition quickly enough to allow the use of such high biomass-quantities.

In addition to constraining the growth of plantation forest (for further details see *Forestry*), the increase of the current forest area, representing the area of land currently covered by forests, is prohibited (Eq.8.4. The existing forest area can only be de-forested, and afforestation is depicted as another land-use type.

$$old\_forest_{n,s,y} <= old\_forest_{n,s,y-1} \tag{8.4}$$

The third and last set of constraints required for the land-use emulator enforce gradual transitions between land-use pathways. Too rapid switches between land-use pathways, i.e. full transitioning between land-use pathways in adjacent timesteps, can occur for several reasons. Slight numerical *non-convexities* in input data, i.e. numerical inconsistencies can occur for individual time-steps. Land-use pathways, cumulatively (across time) depict consistent behavior i.e. as carbon prices increase, the cumulative emissions decrease within a single biomass potential category (see Fig. 8.24). Yet for the same carbon price across multiple biomass potential categories, inconsistencies may occur, for example as a result of data scaling or aggregation. Without a transitional constraint between pathways, the optimal least-cost solution could be to switch between two land-use pathways for only a single timestep, introducing artifacts in the model result (e.g. unreasonable price inconsistencies). The carbon price categories have been chosen to span a broad range of mitigation options (see Fig. 8.23), with stepped carbon price growth that best reflect increases in global mitigation efforts, while at the same time ensuring that inclusion of the land-use emulator in MESSAGEix, does not result in too long solving times. The transitional constraints between pathways further contribute to smoothing the step wise increases between the carbon price categories. The transition rate has been set, so that land-use pathways can be phased out at a rate of 5% annually. This value was derived based on a sensitivity analysis, showing that this factor best matched the transition results of the full fletched GLOBIOM model.

Land-use Price

In the figure depicting the land-use scenario matrix (Fig. 8.23), various biomass and carbon price categories are depicted. This information, together with the quantities of biomass and respective emission reductions are used to determine the land-use scenario price (objective function in MESSAGEix), which the model effectively interprets as the biomass price. Based on the first biomass potential category, *BIO00*, the price (P) for a distinct land-use scenario, in the example below without a carbon price (Eq.8.5), is a result of the biomass quantity (BQ) times the biomass price (BPr).

$$P_{n,s_{BIO00,GHG000},y} = BQ_{n,s_{BIO00,GHG000},y} * BPr_{n,s_{BIO00},y} \\ Landusepriceequation for BIO00GHG000 \tag{8.5}$$

Following on from the above example, therefore staying within the lowest biomass potential category, as the carbon price increases, the costs of emission mitigation must be accounted for as part of the price (Eq.8.6). Hence, in addition to the quantity of biomass, the emissions savings must be calculated and multiplied with the carbon price (EPr). Below, we look at this example for the first carbon price of 5$, *GHG005*.

$$P_{n,s_{BIO00,GHG005},y} = BQ_{n,s_{BIO00,GHG005},y} * BPr_{n,s_{BIO05},y} + (E_{n,s_{BIO00,GHG000},y} - E_{n,s_{BIO00,GHG005},y}) * EPr_{n,s_{BIO05},y} \tag{8.6}$$

where E are the GHG-Emissions.

This can be generalized as follows:

$$P_{n,s_{b,g},y} = BQ_{n,s_{b,g},y} * BPr_{n,s_b,y} + (E_{n,s_{b,g-1},y} - E_{n,s_{b,g},y}) * EPr_{n,s_g,y} \tag{8.7}$$

where b represents the biomass-potential category, and g represents the carbon-price category.

The fact that biomass is the only land-use related commodity which MESSAGEix accounts for when optimizing, also means that all the costs associated with the mitigation of land-use related emissions are therefore perceived as being part of the biomass-price. This is a drawback of the approach, but nevertheless provides a full representation of the land-use scenario specific costs.

Results and validation

The first step in validating the emulator implementation, looks at how scenarios navigate throughout the land-use pathways over the course of a scenario. The figure below (see Fig. 8.26), shows the global mean temperature (panel a.) as well as the carbon price development for the various scenarios (panel b.). These include 1.) "Baseline", a SSP2 based no-policy scenario, 2.) "NPi 1600", a SSP2 based policy scenario with a cumulative CO2 budget of 1600 GtCO2 (limiting global temperature increase compared to pre-industrial times to approximately 1.9 °C), 3.) "NPi 1000", a SSP2 based policy scenario with a cumulative CO2 budget of 1000 GtCO2 (limiting global temperature increase compared to pre-industrial times to approximately 1.6 °C), 4.) "NPi 400", a SSP2 based policy scenario with a cumulative CO2 budget of 400 GtCO2 (limiting global temperature increase compared to pre-industrial times to approximately 1.3 °C). More details on these scenarios can be found here.

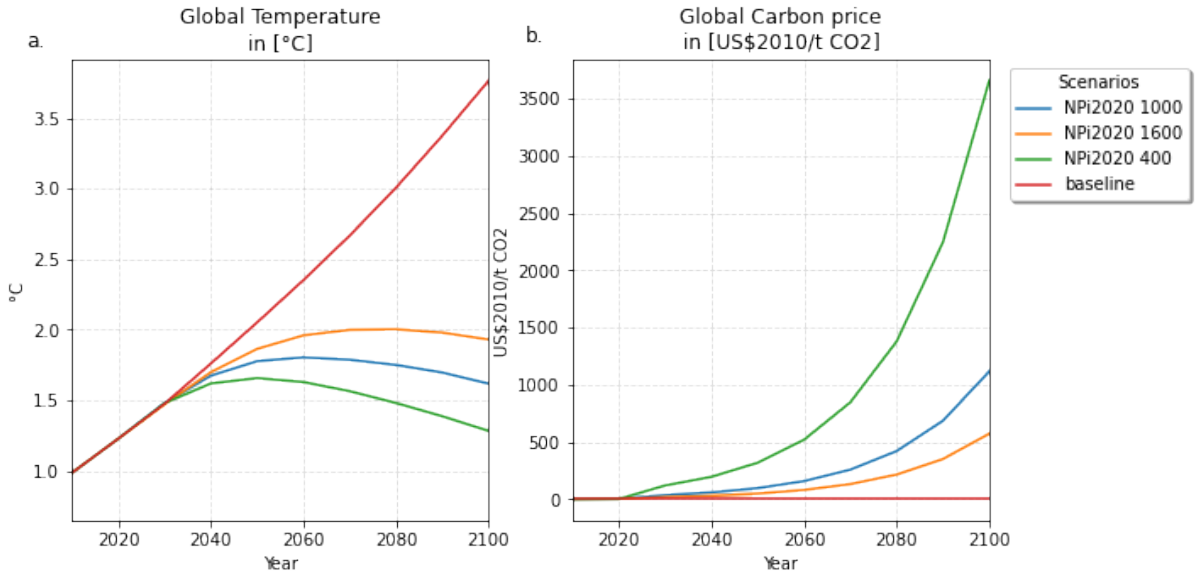

Fig. 8.26: Temperature and carbon-price development across CD-Links scenario set.

For each of the four scenarios, the land-use surface trade-off areas have been plotted (see Fig. 8.27). The orange shaded areas represent the choice of land-use pathways combined over time for all regions. In the "Baseline" scenario (see Fig. 8.27, panel a), only land-use pathways without a carbon price are used. In the least stringent mitigation scenario, "NPi 1600", the carbon price reaches approximately 570 $2010/tCO2 in 2100. In 2090, the carbon price is approximately 350 $2010/tCO2, hence it is to be expected that by the end of the century land-use pathway categories no higher than GHG400 are used, (see Fig. 8.27, panel b). For the "NPi 1000" and the "NPi 400" scenarios, the land-use pathways with the highest carbon price, GHG2000 (which corresponds to approximately 2500 $2010/tCO2 are employed. Not visible from the figure is the timing at which the highest carbon price pathways are used. While in the "NPi 1000" scenario, the carbon price reaches approximately 1100 $2010/tCO2 and 1800 $2010/tCO2 in 2100 and 2110 respectively, the highest price land-use pathways are only partially used in some regions towards the end of the century. The categories which are mostly used are the GHG1000 categories, which correspond to ~1250 $2010/tCO2, (see Fig. 8.27, panel c). For the "NPi 400" scenario, where the carbon price rises above 2000 $2010/tCO2 already in 2090, the GHG2000 categories are used most commonly across all regions (see Fig. 8.27, panel d).

Further validation of the land-use emulator implementation, is performed by setting the carbon price in MESSAGEix such that a specific GHG-category is predominantly used e.g. by setting the global carbon price in MESSAGEix slightly above the price for a specific GHG-category. If the carbon price is therefore set slightly above 500 $2010/tCO2 in MESSAGE, it is to be expected that the land-use emulator would use land-use pathways which fall into the GHG400 category. Fig. 8.28 depicts the results of four such validation scenarios. The carbon price in MESSAGEix is set so that the GHG-categories, GHG005, GHG100, GHG400 and GHG1000, (depicted in panel a., b., c. and d. respectively) are predominantly used cumulatively across all regions and the entire optimization time-horizon.

In addition to informing MESSAGEix of the biomass potential and land-use related emission quantities and prices, the land-use input matrix includes information related to land-use by type, production and demand of other non-bioenergy related land produces as well as information on crop-yields, irrigation water-use, amongst others. Region specific

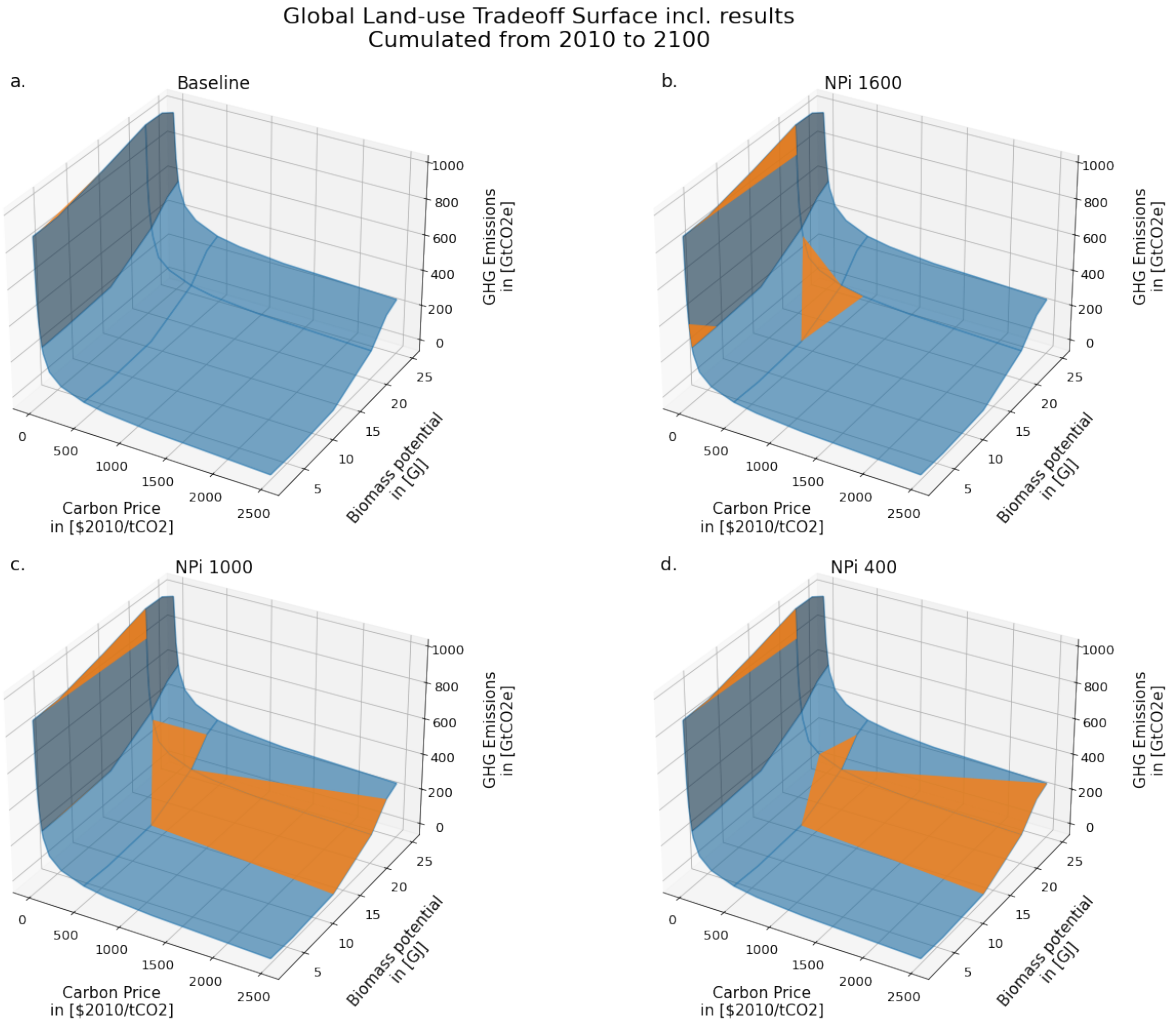

Fig. 8.27: Global land-use pathway choice across CD-Links scenario set.

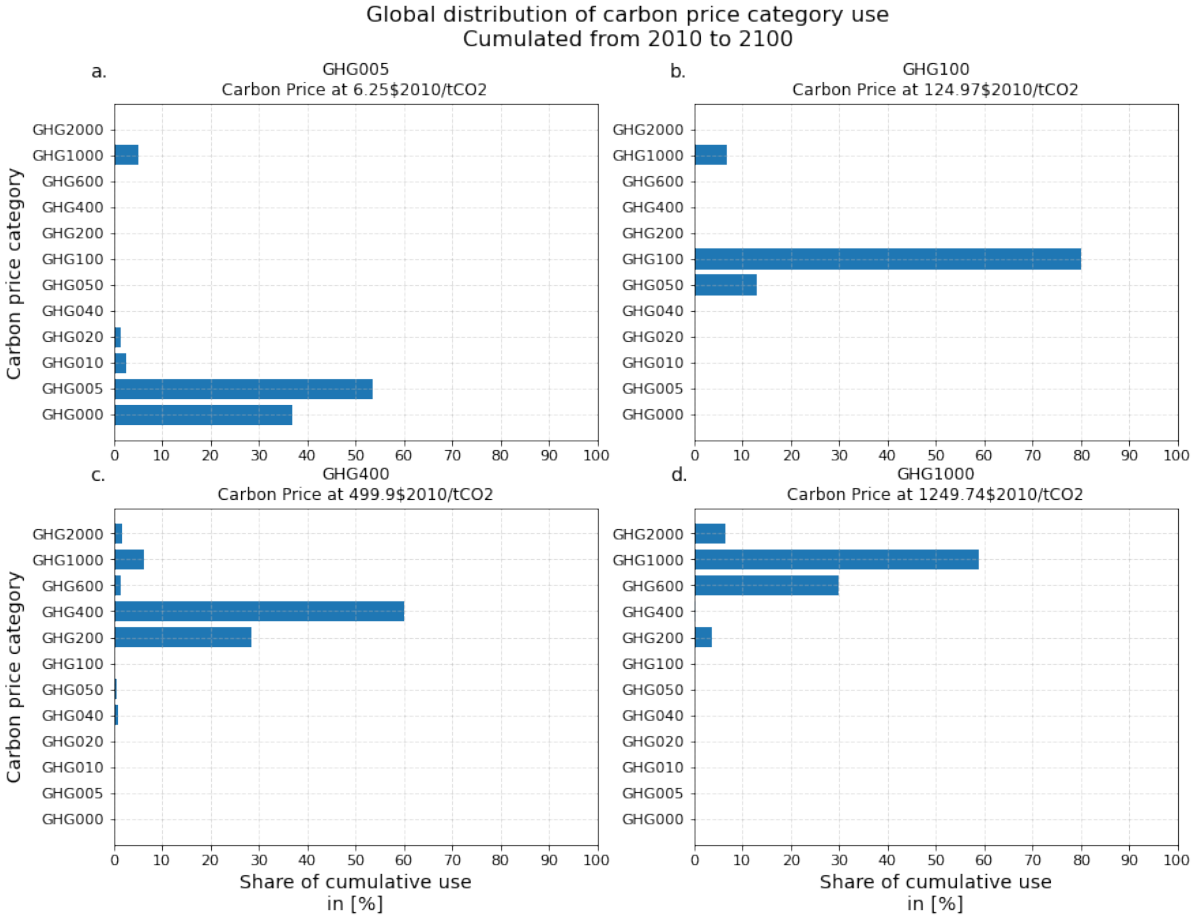

Fig. 8.28: Distribution of land-use related carbon price category use for different carbon price levels.

quantities of biomass from different feedstocks, the carbon price trajectory as well as GDP developments can be *plugged* back into the full fletched GLOBIOM land-use model. Thus, despite the slightly adjusted results, allows the land-use impacts to be analyzed in greater detail. Such validation or *feedback-runs* were conducted for the Shared Socioeconomic Pathways (Riahi et al., 2017 [89]). Fig. 8.29 compares how the emulated results (full lines) for GHG- (panel a.) and CH4 emissions (panel b.) across various scenarios compare with the results of the full fletched GLOBIOM model. The differences in emissions are updated in the original MESSAGEix scenario in order to correctly account for changes in atmospheric concentrations.

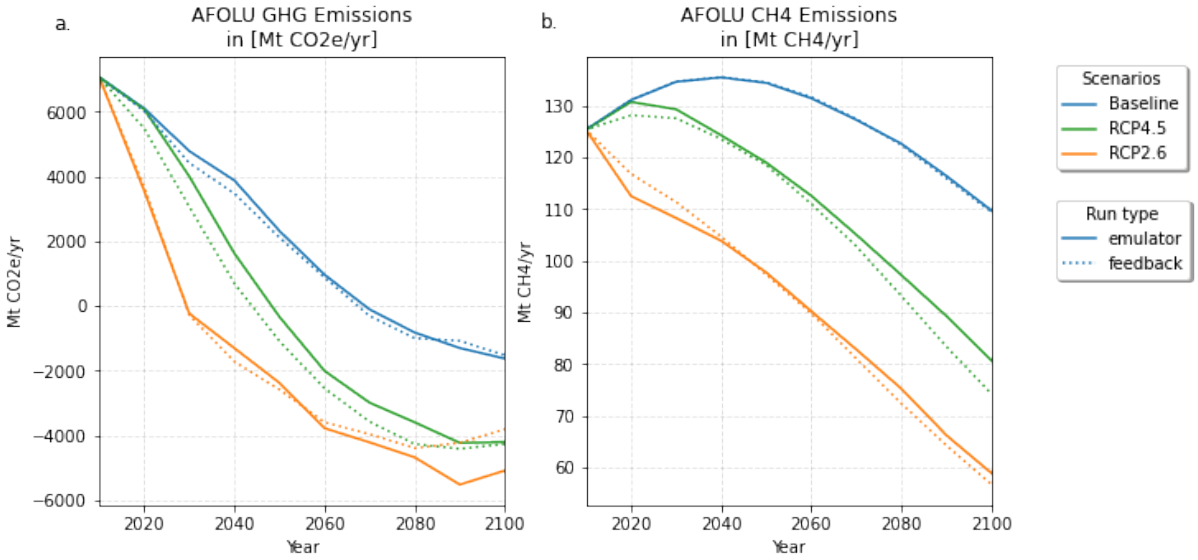

Fig. 8.29: SSP1 Emulated land-use results vs. GLOBIOM feedback.

8.14.6 Water

The water withdrawal and return flows from energy technologies are calculated in MESSAGE following the approach described in Fricko et al., (2016) [18]. Each technology is prescribed a water withdrawal and consumption intensity (e.g., m3 per kWh) that translates technology outputs optimized in MESSAGE into water requirements and return flows.

For power plant cooling technologies, the amount of water required and energy dissipated to water bodies as heat is linked to the parameterized power plant fuel conversion efficiency (heat rate). Looking at a simple thermal energy balance at the power plant (Fig. 8.30), total combustion energy (E_{comb}) is conveterted into electricity (E_{elec}), emissions (E_{emis}) and additional thermal energy that must be absorbed by the cooling system (E_{cool}):

$$E_{comb} = E_{elec} + E_{emis} + E_{cool}$$

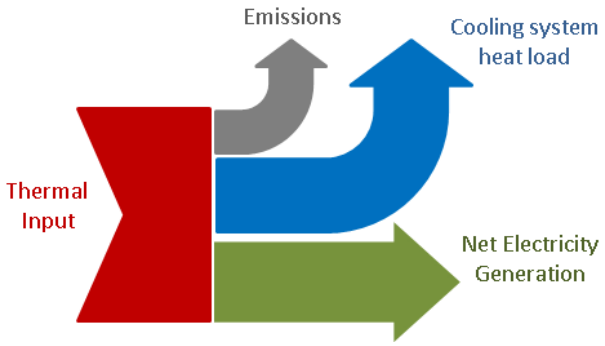

Fig. 8.30: Simplified power plant energy balance.

Converting to per unit electricity, we can estimate the cooling required per unit of electricity generation (ϕ_{cool}) based on average heat-rate (ϕ_{comb}) and heat lost to emissions (ϕ_{emis}), and this data is identified from the literature [18].

$$\phi_{cool} = \phi_{comb} - \phi_{emis} - 1$$

With time-varying heat-rates (i.e., $t = 0, 1, 2, ...$) and a constant share of energy to emissions and electricity:

$$\phi_{cool}[t] = \phi_{comb}[t] \cdot \left(1 - \frac{\phi_{emis}}{\phi_{comb}[0]}\right) - 1$$

Increased fuel efficiency (lower heat-rate) reduces the cooling requirement per unit of electricity generated. This enables heat rate improvements for power plants represented in MESSAGE to be translated into improvements in water intensity. Water withdrawal and consumption intensities for power plant cooling technologies are calibrated to the range reported in Meldrum et al., (2013) [59]. Additional parasitic electricity demands from recirculating and dry cooling technologies are accounted for explicitly in the electricity balance calculation. All other technologies follow the data reported in Fricko et al. (2016) [18].

A key feature of the implementation is the representation of power plant cooling technology options for individual power plant types (Fig. 8.31). Each power plant type that requires cooling in MESSAGE is connected to a corresponding cooling technology option (once-through, recirculating or air cooling), with the investment into and operation of the cooling technologies included in the optimization decision variables [70]. This enables MESSAGE to choose the type of cooling technology for each power plant type and track how the operation of the cooling technologies impact water withdrawals, return flows, thermal pollution and parasitic electricity use.

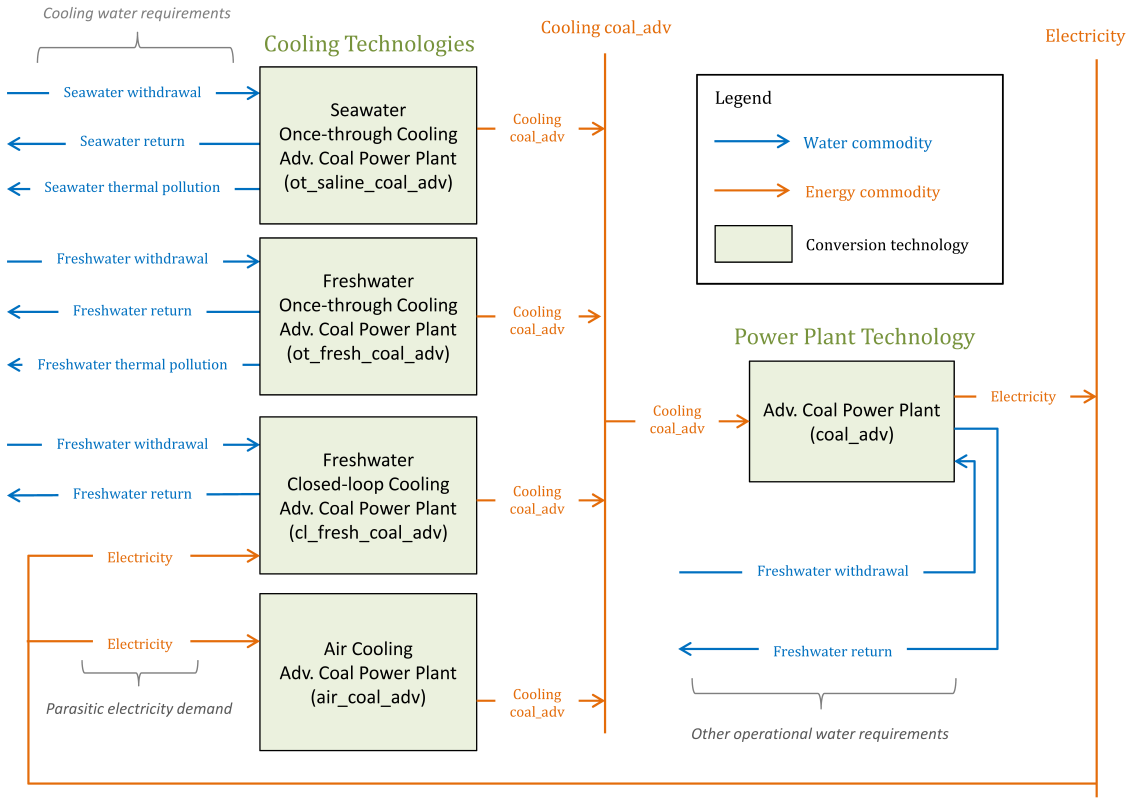

Fig. 8.31: Implementation of cooling technologies in the MESSAGE IAM.

Costs and efficiency for cooling technologies are estimated following previous technology assessments [50, 115, 116]. The initial distribution of cooling technologies in each region and for each technology is estimated with the dataset described in Raptis and Pfister (2016) [83]. The shares estimated at the river basin-scale are depcited in Fig. 8.32 .

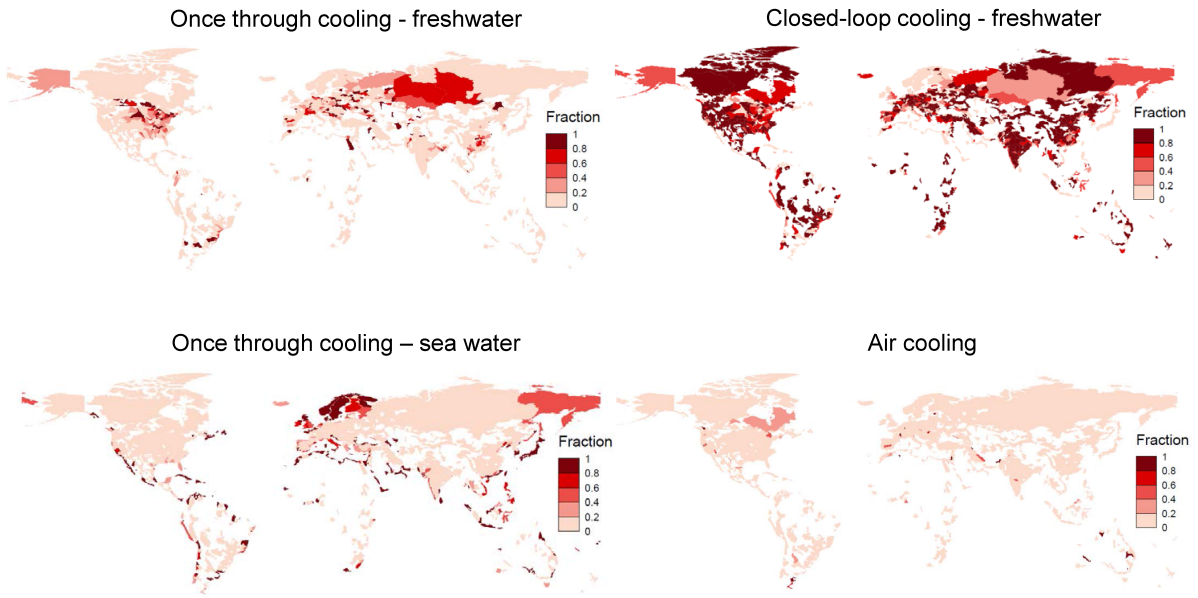

Fig. 8.32: Average cooling technology shares across all power plant types at the river basin-scale.

8.14.7 Emissions

Emission from energy (MESSAGE)

Carbon-dioxide (CO2)

The MESSAGE model includes a detailed representation of energy-related and - via the link to GLOBIOM - land-use CO2 emissions (Riahi and Roehrl, 2000 [87]; Riahi, Rubin et al., 2004 [88]; Rao and Riahi, 2006 [82]; Riahi et al., 2011 [86]). CO2 emission factors of fossil fuels and biomass are based on the 1996 version of the IPCC guidelines for national greenhouse gas inventories [34] (see Table 8.25). It is important to note that biomass is generally treated as being carbon neutral in the energy system, because the effects on the terrestrial carbon stocks are accounted for on the land use side, i.e. in GLOBIOM (see section *Land-use (GLOBIOM)*). The CO2 emission factor of biomass is, however, relevant in the application of carbon capture and storage (CCS) where the carbon content of the fuel and the capture efficiency of the applied process determine the amount of carbon captured per unit of energy.

Table 8.25: Carbon emission factors used in MESSAGE based on IPCC (1996, Table 1-2 [34]). For convenience, emission factors are shown in three different units.

Fuel	Emission factor [tC/TJ]	Emission factor [tCO2/TJ]	Emission factor [tC/kWyr]
Hard coal	25.8	94.6	0.814
Lignite	27.6	101.2	0.870
Crude oil	20.0	73.3	0.631
Light fuel oil	20.0	73.3	0.631
Heavy fuel oil	21.1	77.4	0.665
Methanol	17.4	63.8	0.549
Natural gas	15.3	56.1	0.482
Solid biomass	29.9	109.6	0.942

CO2 emissions of fossil fuels for the entire energy system are accounted for at the resource extraction level by applying the CO2 emission factors listed in Table 8.25 to the extracted fossil fuel quantities. In this economy-wide accounting, carbon emissions captured in CCS processes remove carbon from the balance equation, i.e. they contribute with a negative emission coefficient. In parallel, a sectoral acounting of CO2 emissions is performed which applies the

same emission factors to fossil fuels used in individual conversion processes. In addition to conversion processes, also CO2 emissions from energy use in fossil fuel resource extraction are explicitly accounted for. A relevant feature of MESSAGE in this context is that CO2 emissions from the extraction process increase when moving from conventional to unconventional fossil fuel resources (McJeon et al., 2014 [55]).

CO2 mitigation options in the energy system include technology and fuel shifts; efficiency improvements; and CCS. A large number of specific mitigation technologies are modeled bottom-up in MESSAGE with a dynamic representation of costs and efficiencies. As mentioend above, MESSAGE also includes a detailed representation of carbon capture and sequestration from both fossil fuel and biomass combustion (see Table 8.26).

Table 8.26: Carbon capture rates in [%]

Conversion Process	Plant type	Capture rate
Electricity generation	supercritical PC power plant with desulphurization/denox and CCS	90%
Electricity generation	IGCC power plant with CCS	90%
Electricity generation	biomass IGCC power plant with CCS	86%
Liquid fuel production	Fischer-Tropsch coal-to-liquids with CCS	85%
Liquid fuel production	coal methanol-to-gasoline with CCS	85%
Liquid fuel production	Fischer-Tropsch gas-to-liquids with CCS	90%
Liquid fuel production	Fischer-Tropsch biomass-to-liquids with CCS	65%
Liquid fuel production	Biomass to Gasoline via the Methanol-to-Gasoline (MTG) Process with CCS	67%
Hydrogen production	coal gasification with CCS	92%
Hydrogen production	biomass gasification with CCS	85%
Hydrogen production	steam methane reforming with CCS	90%

Non-CO2 GHGs

MESSAGE includes a representation of non-CO2 GHGs (CH4, N2O, HFCs, SF6, PFCs) mandated by the Kyoto Protocol (Rao and Riahi, 2006 [82]) with the exception of NF3. Included is a representation of emissions and mitigation options from both energy related processes as well as non-energy sources like municipal solid waste disposal and wastewater. CH4 and N2O emissions from land are taken care of by the link to GLOBIOM (see Section *Emissions from land (GLOBIOM)*).

Air pollution

Air pollution implications are derived with the help of the GAINS (Greenhouse gas-Air pollution INteractions and Synergies) model. GAINS allows for the development of cost-effective emission control strategies to meet environmental objectives on climate, human health and ecosystem impacts until 2030 (Amann et al., 2011 [3]). These impacts are considered in a multi-pollutant context, quantifying the contributions of sulfur dioxide (SO2), nitrogen oxides (NOx), ammonia (NH3), non-methane volatile organic compounds (VOC), and primary emissions of particulate matter (PM), including fine and coarse PM as well as carbonaceous particles (BC, OC). As a stand-alone model, it also tracks emissions of six greenhouse gases of the Kyoto basket with exception of NF3. The GAINS model has global coverage and holds essential information about key sources of emissions, environmental policies, and further mitigation opportunities for about 170 country-regions. The model relies on exogenous projections of energy use, industrial production, and agricultural activity for which it distinguishes all key emission sources and several hundred control measures. GAINS can develop finely resolved mid-term air pollutant emission trajectories with different levels of mitigation ambition (Cofala et al., 2007 [7]; Amann et al., 2013 [4]). The results of such scenarios are used as input to global IAM frameworks to characterize air pollution trajectories associated with various long-term energy developments (see further for example Riahi et al., 2012 [85]; Rao et al., 2013 [81]; Fricko et al., 2017 [17]).

Emissions from land (GLOBIOM)

Crop sector emissions

Crop emissions sources accounted in GLOBIOM are N2O fertilization emissions, from synthetic fertilizer and from organic fertilizers, as well as CH4 methane emissions from rice cultivation. Synthetic fertilizers are calculated on a Tier 1 approach, using the information provided by EPIC on the fertilizer use for each management system at the Simulation Unit level and applying the emission factor from IPCC AFOLU guidelines. Synthetic fertilizer use is therefore built in a bottom up approach, but upscaled to the International Fertilizer Association statics on total fertilizer use per crop at the national level for the case where calculated fertilizers are found too low at the aggregated level. This correction ensures a full consistency with observed fertilizer purchases. In the case of rice, only a Tier 1 approach was applied, with a simple formula where emissions are proportional to the area of rice cultivated. Emission factor is taken from EPA (2012) [12].

Livestock emissions

In GLOBIOM, the following emission accounts were assigned to livestock directly: CH4 from enteric fermentation, CH4 and N2O from manure management, and N2O from excreta on pasture (N2O from manure applied on cropland is reported in a separate account linked to crop production). In brief, CH4 from enteric fermentation is a simultaneous output of the feed-yield calculations done with the RUMINANT model, as well as nitrogen content of excreta and the amount of volatile solids. The assumptions about proportions of different manure management systems, manure uses, and emission coefficients are based on detailed literature review. A detailed description of how these coefficients have been determined including the literature review is provided in (Herrero et al., 2013 [27]).

Land use change emissions

Land use change emissions are computed based on the difference between initial and final land cover equilibrium carbon stock. For forest, above and below-ground living biomass carbon data are sourced from Kindermann et al. (2008) [43], where geographically explicit allocation of the carbon stocks is provided. The carbon stocks are consistent with the 2010 Forest Assessment Report (FAO, 2010 [15]). Therefore, the emission factors for deforestation are in line with those of FAO. Additionally, carbon stock from grasslands and other natural vegetation is also taken into account using the above and below ground carbon from the biomass map from (Ruesch and Gibbs, 2008 [97]). When forest or natural vegetation is converted into agricultural use, it is considered in this approach that all below and above ground biomass is released in the atmosphere. However, the following are not accounted for: litter, dead wood and soil organic carbon.

Comparison with other literature

In order to put the numbers in perspective with other sources they were compared with FAO (Tubiello et al., 2013 [110]) where a simple but transparent approach is used, largely relying on FAOSTAT activity numbers and IPCC Tier 1 emission coefficients (see Table 8.27).

The 2000 data for crops are overall about 11% higher than Tubiello et al., mainly because of rice where the data are closer to EPA (EPA 2012 [12]) which is higher than Tubiello et al. For livestock, it is by some 18% lower than Tubiello et al. So in total there is about 10% GHG emissions less in 2000 than the values reported. The year 2010 is already the result of simulations and hence may be interesting to compare with the data. In order to facilitate the comparison, the columns e), f) and g) in Table 1 are3 included. Columns e) and f) compare GLOBIOM data for 2000 and projections for 2010 respectively, with numbers reported by Tubiello et al. Column g) compares the relative change in emissions between 2000 and 2010 from these two sources (1.00 would indicate the same relative change in GLOBIOM and in Tubiello et al.). It is apparent that the relative change in total agricultural emissions in GLOBIOM is the same as the development reported by Tubiello et al. – an increase by 11%. The behavior of GLOBIOM is over this period very close to the reported trends also at the level of individual accounts. The only exception is emissions from manure management where the relative change projected in GLOBIOM is by 13% higher than the relative change observed in Tubiello's numbers.

Table 8.27: Comparison of agricultural GHG emissions from GLOBIOM and from FAO for the years 2000 and 2010

	GLOBIOM		Tubiello et al.				
	(a)	(b)	(c)	(d)	(e)	(f)	(g)
	2000	2010	2000	2010	2000	2010	2010/2000
Crops	1,239	1,365	1,114	1,298	1.11	1.05	0.95
Synthetic fertilizer	522	640	521	683	1.00	0.94	0.93
Manure applied	83	96	103	116	0.81	0.83	1.03
Rice	633	629	490	499	1.29	1.26	0.98
Livestock	2,362	2,625	2,893	3,135	0.82	0.84	1.03
Enteric fermentation	1,502	1,661	1,863	2,018	0.81	0.82	1.02
Manure on pastures	403	441	682	764	0.59	0.58	0.98
Manure management	457	524	348	353	1.31	1.48	1.13
Total Agriculture	3,601	3,991	4,007	4,433	0.90	0.90	1.00

8.14.8 Climate (MAGICC)

The response of the carbon-cycle and climate to anthropogenic climate drivers is modelled with the MAGICC model (Model for the Assessment of Greenhouse-gas Induced Climate Change). MAGICC is a reduced-complexity coupled global climate and carbon cycle model which calculates projections for atmospheric concentrations of GHGs and other atmospheric climate drivers like air pollutants, together with consistent projections of radiative forcing, global annual-mean surface air temperature, and ocean-heat uptake (Meinshausen et al., 2011a [57]). MAGICC is an upwelling-diffusion, energy-balance model, which produces outputs for global- and hemispheric-mean temperature. MAGICC is most commonly used in a deterministic setup (Meinshausen et al., 2011b [58]), but also a probabilistic setup (Meinshausen et al., 2009 [56]) is available which allows to estimate the probabilities of limiting warming to below specific temperature levels given a specified emissions path (Rogelj et al., 2013a [92]; Rogelj et al., 2013b [93]; Rogelj et al., 2015 [94]). Climate feedbacks on the global carbon cycle are accounted for through the interactive coupling of the climate model and a range of gas-cycle models. (Fricko et al., 2017 [17])

For more information about the model, see www.magicc.org.

8.14.9 Annex: mathematical formulation

This mathematical formulation of MESSAGE-GLOBIOM relies on the generalized MESSAGEix energy model framework. See the MESSAGEix documentation for a complete description of its formulation. The equation system of the older MESSAGE V implementation can be found in the 2017 release of this documentation.

8.14.10 Further reading

1. Environmental Protection Agency (EPA). Global Mitigation of Non-CO2 Greenhouse Gases: 2010-2030. 2013. URL: https://www3.epa.gov/climatechange/Downloads/EPAactivities/MAC_Report_2013.pdf.
2. Nikos Alexandratos and Jelle Bruinsma. World agriculture towards 2030/2050: the 2012 revision. Report 12-03, FAO, June 2012.
3. Markus Amann, Rafal Cabala, Janusz Cofala, Chris Heyes, Zbigniew Klimont, Wolfgang Schopp, Leonor Tarrason, David Simpson, Peter Wind, and Jan-Eiof Jonson. "Current Legislation" and the "Maximum Technically Feasible Reduction" cases for the CAFE baseline emission projections. *IIASA, Vienna*, 2004. URL: https://www.researchgate.net/profile/Zbigniew_Klimont/publication/230709494_The_Current_Legislation_and_the_Maximum_Technically_Feasible_Reduction_cases_for_the_CAFE_baseline_emission_projections._CAFE_Report__2/links/0deec53cd2d778aafb000000.pdf (visited on 2016-03-24).
4. Goran Berndes, Monique Hoogwijk, and Richard van den Broek. The contribution of biomass in the future global energy supply: a review of 17 studies. *Biomass and Bioenergy*, 25(1):1–28, 7 2003. doi:10.1016/S0961-9534(02)00185-X.
5. A.F. Bouwman, K.W. Van der Hoek, B. Eickhout, and I. Soenario. Exploring changes in world ruminant production systems. *Agricultural Systems*, 84(2):121 – 153, 2005. doi:10.1016/j.agsy.2004.05.006.
6. Stefan Bringezu, Helmut Schutz, Meghan O'Brien, Lea Kauppi, Robert W Howarth, and Jeff McNeely. *Assessing biofuels: towards sustainable production and use of resources*. United Nations Environment Programme, 2009. ISBN 92-807-3052-5.
7. Veronika Dornburg, APC Faaij, PA Verweij, Martin Banse, Kees van Diepen, Herman van Keulen, Hans Langeveld, Marieke Meeusen, Gerrie van de Ven, and Flip Wester. Biomass assessment: assessment of global biomass potentials and their links to food, water, biodiversity, energy demand and economy: inventory and analysis of existing studies: supporting document. *Report/WAB*, 2008.
8. Bas Eickhout, Gert Jan van den Born, Jos Notenboom, M van Oorschot, JPM Ros, DP Van Vuuren, and HJ Westhoek. Local and global consequences of the EU renewable directive for biofuels: Testing the sustainability criteria. *Local and global consequences of the EU renewable directive for biofuels: testing the sustainability criteria*, 2008.
9. RA Fischer, Derek Byerlee, and Gregory O Edmeades. Can technology deliver on the yield challenge to 2050? 2009. URL: http://www.fao.org/3/a-ak542e/ak542e12a.pdf.
10. Chris E Forest, Peter H Stone, Andrei P Sokolov, Myles R Allen, and Mort D Webster. Quantifying uncertainties in climate system properties with the use of recent climate observations. *Science*, 295(5552):113–117, 2002.
11. Claire Granier, Bertrand Bessagnet, Tami Bond, Ariela D'Angiola, Hugo Denier van Der Gon, Gregory J Frost, Angelika Heil, Johannes W Kaiser, Stefan Kinne, and Zbigniew Klimont. Evolution of anthropogenic and biomass burning emissions of air pollutants at global and regional scales during the 1980–2010 period. *Climatic Change*, 109(1-2):163–190, 2011.
12. Monique Hoogwijk and Wina Graus. Global potential of renewable energy sources: a literature assessment. *Background report prepared by order of REN21. Ecofys, PECSNL072975*, 2008.
13. Monique Maria Hoogwijk. *On the global and regional potential of renewable energy sources*. PhD, Department of Science, Technology and Society. Utrecht University, 2004.
14. IPCC. *Climate Change 2007: Synthesis Report. Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, Geneva, Switzerland, 2007. URL: http://www.ipcc.ch/pdf/assessment-report/ar4/syr/ar4_syr_full_report.pdf.

15. Global Emissions Joint Research Centre. Emission Database for Global Atmospheric Research EDGAR v4.2. 11 2011. URL: http://edgar.jrc.ec.europa.eu/overview.php?v=42.

16. Ilkka Keppo, Brian C O'Neill, and Keywan Riahi. Probabilistic temperature change projections and energy system implications of greenhouse gas emission scenarios. *Technological Forecasting and Social Change*, 74(7):936–961, 2007.

17. Richard Loulou, Gary Goldstein, and Ken Noble. *Documentation for the MARKAL Family of Models - Part II: MARKAL-MACRO*. IEA Energy Technology Systems Analysis Programme (ETSAP), October 2004. URL: https://www.iea-etsap.org/MrklDoc-II_MARKALMACRO.pdf.

18. Malte Meinshausen. What does a 2 C target mean for greenhouse gas concentrations? A brief analysis based on multi-gas emission pathways and several climate sensitivity uncertainty estimates. *Avoiding dangerous climate change*, 2006.

19. Sanderine Nonhebel. Energy from agricultural residues and consequences for land requirements for food production. *Agricultural Systems*, 94(2):586–592, 2007.

20. B.C. O'Neill, T.R. Carter, K.L. Ebi, J. Edmonds, S. Hallegatte, E. Kemp-Benedict, E. Kriegler, L. Mearns, R. Moss, K. Riahi, B. van Ruijven, and D. van Vuuren. Meeting report of the workshop on the nature and use of new socioeconomic pathways for climate change research. Report, NCAR, November 2-4, 2011 2012. URL: http://www.isp.ucar.edu/socio-economic-pathways.

21. Andrew J Plantinga, Thomas Mauldin, and Douglas J Miller. An econometric analysis of the costs of sequestering carbon in forests. *American Journal of Agricultural Economics*, 81(4):812–824, 1999.

22. Shilpa Rao, Vadim Chirkov, Frank Dentener, Rita Van Dingenen, Shonali Pachauri, Pallav Purohit, Markus Amann, Chris Heyes, Patrick Kinney, and Peter Kolp. Environmental modeling and methods for estimation of the global health impacts of air pollution. *Environmental Modeling & Assessment*, 17(6):613–622, 2012.

23. Keywan Riahi, Arnulf Grubler, and Nebojsa Nakicenovic. Scenarios of long-term socio-economic and environmental development under climate stabilization. *Technological Forecasting and Social Change*, 74(7):887–935, 2007.

24. Dmitry Rokityanskiy, Pablo C Benitez, Florian Kraxner, Ian McCallum, Michael Obersteiner, Ewald Rametsteiner, and Yoshiki Yamagata. Geographically explicit global modeling of land-use change, carbon sequestration, and biomass supply. *Technological Forecasting and Social Change*, 74(7):1057–1082, 2007.

25. P. Russ, T. Wiesenthal, D. van Regemorter, and J.C. Ciscar. Global climate policy scenarios for 2030 and beyond: analysis of greenhouse gas emission reduction pathway scenarios with the poles and geme3 models. *Institute for Prospective technological Studies, October*, 2007.

26. Jayant Sathaye, Peter Chan, Larry Dale, Willy Makundi, and Ken Andrasko. A summary note estimating global forestry GHG mitigation potential and costs: A dynamic partial equilibrium approach. *working draft, August*, 10:448–457, 2003.

27. Jayant Sathaye, Willy Makundi, Larry Dale, Peter Chan, and Kenneth Andrasko. GHG mitigation potential, costs and benefits in global forests: a dynamic partial equilibrium approach. *The Energy Journal*, pages 127–162, 2006.

28. Timothy Searchinger, Ralph Heimlich, Richard A Houghton, Fengxia Dong, Amani Elobeid, Jacinto Fabiosa, Simla Tokgoz, Dermot Hayes, and Tun-Hsiang Yu. Use of US croplands for biofuels increases greenhouse gases through emissions from land-use change. *Science*, 319(5867):1238–1240, 2008.

29. Edward MW Smeets, Andre PC Faaij, Iris M Lewandowski, and Wim C Turkenburg. A bottom-up assessment and review of global bio-energy potentials to 2050. *Progress in Energy and combustion science*, 33(1):56–106, 2007.

30. Pete Smith, Peter J Gregory, Detlef Van Vuuren, Michael Obersteiner, Petr Havlik, Mark Rounsevell, Jeremy Woods, Elke Stehfest, and Jessica Bellarby. Competition for land. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 365(1554):2941–2957, 2010.

31. Robert N Stavins. The costs of carbon sequestration: a revealed-preference approach. *The American Economic Review*, 89(4):994–1009, 1999.

32. Elke Stehfest, Lex Bouwman, Detlef P Van Vuuren, Michel GJ Den Elzen, Bas Eickhout, and Pavel Kabat. Climate benefits of changing diet. *Climatic change*, 95(1-2):83–102, 2009.

33. Francesco N Tubiello and Gunther Fischer. Reducing climate change impacts on agriculture: Global and regional effects of mitigation, 2000–2080. *Technological Forecasting and Social Change*, 74(7):1030–1056, 2007.

34. Jasper van Vliet, Maarten van den Berg, Michiel Schaeffer, Detlef P van Vuuren, Michel Den Elzen, Andries F Hof, Angelica Mendoza Beltran, and Malte Meinshausen. Copenhagen accord pledges imply higher costs for staying below 2 C warming. *Climatic Change*, 113(2):551–561, 2012.

35. Detlef van Vuuren, Washington Ochola, Susan Riha, Mario Giampietro, Hector Ginzo, Thomas Henrichs, Sajidin Hussain Hussain, Kaspar Kok, Moraka Makhura Makhura, and Monirul Mirza. Outlook on agricultural changes and its drivers. In *Agriculture at a Crossroads-the Global Report of the International Assessment of Agricultural Knowledge, Science, and Technology*. Island Press, 2009.

36. Detlef P Van Vuuren, Elie Bellevrat, Alban Kitous, and Morna Isaac. Bio-energy use and low stabilization scenarios. *The Energy Journal*, pages 193–221, 2010.

37. Detlef P Van Vuuren, Jasper van Vliet, and Elke Stehfest. Future bio-energy potential under various natural constraints. *Energy Policy*, 37(11):4220–4230, 2009.

38. Tom ML Wigley. MAGICC/SCENGEN 5.3: User manual (version 2). *NCAR, Boulder, CO*, 2008.

8.15 MESSAGEix-Nexus (`model.water`)

`message_ix_models.model.water` adds water usage and demand related representation to the MESSAGEix-GLOBIOM global model. The resulting model is referred to as **"MESSAGEix-Nexus"**. This work extends the water sector linkage described by Parkinson et al. (2019) [71].

8.15.1 CLI usage

Use the *CLI* command `mix-data water` to invoke the commands defined in *`water.cli`*. Example: `mix-models --url=ixmp://ixmp_dev/ENGAGE_SSP2_v4.1.7/baseline_clone_test water cooling` model and scenario specifications can be either set manually in `cli.py` or specified in the `--url` option

```
Usage: mix-models water [OPTIONS] COMMAND [ARGS]...

Options:
```

(continues on next page)

(continued from previous page)

```
--regions [ISR|R11|R12|R14|R32|RCP|ZMB]
                                  Code list to use for 'node' dimension.
--help                           Show this message and exit.

Commands:
cooling  Build and solve model with new cooling technologies.
nexus    Add basin structure connected to the energy sector and water...
report   function to run the water report_full from cli to the scenario...
```

Country vs Global implementation

The *message_ix_models.model.water* is designed to being able to add water components to either a global R11 (or R12) model or any country model designed with the MESSAGEix single country model prototype. For any of the region configuration a shapefile is needed to run the pre-processing part, while, once the data is prepared, only a .csv file similar to those in *message_ix_models.data.water.delineation* is needed.

To work with a country model please ensure that:

1. country model and scenario are specified either in `--url` or in the `cli.py` script
2. the option `--regions` is used with the ISO3 code of the country (e.g. for Israel `--regions=ISR`)
3. Following the Israel example add a 'country'.yaml file in *message_ix_models.data.node* for the specific country
4. Following the Israel example add the country ISO3 code in the 'regions' options in *message_ix_models.utils.click*

Annual vs sub-annual implementation

if a sub-annual timestep is defined (e.g. seasons or months), the water module will automatically generate the water system with seasonality, using the *time* set components that is not *year* (assuming that the data has been prepared accordingly).

In the case you want to add seasonality in the water sector to a model with only annual timesteps, the best way is to pre-define the required sets (e.g. *time* and *map_time*). Then, running the water nexus module will automatically build the sub-annual water module.

In the case there are already multiple sub-annual time-steps levels already defined and not all are relevant to the water module, the components of the set *time* that are of interest for the water module should be manually added as a cli option (e.g. *–time=year* or *–time=[1,2]*)

8.15.2 Code reference

`message_ix_models.model.water.`**`read_config`**(*context=None*)

Read the water model configuration / metadata from file.

Numerical values are converted to computation-ready data structures.

Returns
The current Context, with the loaded configuration.

Return type
Context

Build and run

message_ix_models.model.water.build.**get_spec**(*context*) → Mapping[str, *ScenarioInfo*]

Return the specification for nexus implementation

Parameters
context (*Context*) – The key `regions` determines the regional aggregation used.

message_ix_models.model.water.build.**main**(*context*, *scenario*, ***options*)

Set up MESSAGEix-Nexus on *scenario*.

See also:

`add_data`, `apply_spec`, *get_spec*

message_ix_models.model.water.build.**map_basin**(*context*) → Mapping[str, *ScenarioInfo*]

Return specification for mapping basins to regions

The basins are spatially consolidated from HydroSHEDS basins delineation database.This delineation is then intersected with MESSAGE regions to form new water sector regions for the nexus module. The nomenclature for basin names is <basin_id>|<MESSAGEregion> such as R1|AFR

Data preparation

Generate input data.

message_ix_models.model.water.data.**add_data**(*scenario*, *context*, *dry_run=False*)

Populate *scenario* with MESSAGEix-Nexus data.

Prepare data for water use for cooling & energy technologies.

message_ix_models.model.water.data.water_for_ppl.**cool_tech**(*context*)

Process cooling technology data for a scenario instance. The input values of parent technologies are read in from a scenario instance and then cooling fractions are calculated by using the data from `tech_water_performance_ssp_msg.csv`. It adds cooling technologies as addons to the parent technologies.The nomenclature for cooling technology is <parenttechnologyname>__<coolingtype>. E.g: *coal_ppl__ot_fresh* :param context: :type context: *Context*

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["water build info"]`, plus the additional year 2010.

Return type
`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.water_for_ppl.**non_cooling_tec**(*context*)

Process data for water usage of power plants (non-cooling technology related). Water withdrawal values for power plants are read in from `tech_water_performance_ssp_msg.csv` :param context: :type context: *Context*

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["transport build info"]`, plus the additional year 2010.

Return type
`dict` of `(str -> pandas.DataFrame)`

Prepare data for adding demands

message_ix_models.model.water.data.demands.**add_irrigation_demand**(*context*)

Adds endogenous irrigation water demands from GLOBIOM emulator :param context: :type context: *Context*

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`.

Return type
`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.demands.**add_sectoral_demands**(*context*)

Adds water sectoral demands :param context: :type context: *Context*

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`.

Return type
`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.demands.**add_water_availability**(*context*)

Adds water supply constraints :param context: :type context: *Context*

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`.

Return type
`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.demands.**get_basin_sizes**(*basin*, *node*)

Returns the sizes of developing and developed basins for a given node

message_ix_models.model.water.data.demands.**read_water_availability**(*context*)

Reads water availability data and bias correct it for the historical years and no climate scenario assumptions.

Parameters
context (*Context*) –

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`.

Return type
`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.demands.**set_target_rate**(*df*, *node*, *year*, *target*)

Sets the target value for a given node and year

message_ix_models.model.water.data.demands.**set_target_rate_developed**(*df*, *node*, *target*)

Sets target rate for a developed basin

message_ix_models.model.water.data.demands.**set_target_rate_developing**(*df*, *node*, *target*)

Sets target rate for a developing basin

message_ix_models.model.water.data.demands.**set_target_rates**(*df*, *basin*, *val*)

Sets target rates for all nodes in a given basin

message_ix_models.model.water.data.demands.**target_rate**(*df*, *basin*, *val*)

Sets target connection and sanitation rates for SDG scenario. The function filters out the basins as developing and developed based on the countries overlapping basins. If the number of developing countries in the basins are more than basin is categorized as developing and vice versa. If the number of developing and developed countries are equal in a basin, then the basin is assumed developing. For developed basins, target is set at 2030. For developing basins, the access target is set at 2040 and 2035 target is the average of 2030 original rate and 2040 target. :returns: Data frame with updated value column. :rtype: df (pandas.DataFrame)

message_ix_models.model.water.data.demands.**target_rate_trt**(*df*, *basin*)

Sets target treatment rates for SDG scenario. The target value for developed and developing region is making sure that the amount of untreated wastewater is halved beyond 2030 & 2040 respectively. :returns: **data** :rtype: `dict` of `(str -> pandas.DataFrame)`

Prepare data for adding techs related to water distribution, treatment in urban & rural

message_ix_models.model.water.data.infrastructure.**add_desalination**(*context*)

Add desalination infrastructure Two types of desalination are considered; 1. Membrane 2. Distillation :param context: :type context: *Context*

Returns

data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["water build info"]`, plus the additional year 2010.

Return type

`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.infrastructure.**add_infrastructure_techs**(*context*)

Process water distribution data for a scenario instance. :param context: :type context: *Context*

Returns

data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["water build info"]`, plus the additional year 2010.

Return type

`dict` of `(str -> pandas.DataFrame)`

Prepare data for water use for cooling & energy technologies.

message_ix_models.model.water.data.water_supply.**add_e_flow**(*context*)

Add environmental flows This function bounds the available water and allocates the environmental flows.Environmental flow bounds are calculated using Variable Monthly Flow (VMF) method. The VMF method is applied to wet and dry seasonal runoff values. These wet and dry seasonal values are then aggregated to annual values.Environmental flows in the model will be incorporated as bounds on 'return_flow' technology. The lower bound on this technology will ensure that certain amount of water remain :param context: :type context: *Context*

Returns

data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["water build info"]`, plus the additional year 2010.

Return type

`dict` of `(str -> pandas.DataFrame)`

message_ix_models.model.water.data.water_supply.**add_water_supply**(*context*)

Add Water supply infrastructure This function links the water supply based on different settings and options. It defines the supply linkages for freshwater, groundwater and salinewater. :param context: :type context: *Context*

Returns

data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data

frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["water build info"]`, plus the additional year 2010.

Return type
`dict` of `(str -> pandas.DataFrame)`

`message_ix_models.model.water.data.water_supply.`**`map_basin_region_wat`**(*context*)

Calculate share of water avaialbility of basins per each parent region.

The parent region could be global message regions or country

Parameters
`context` (*Context*) –

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`.

Return type
`dict` of `(str -> pandas.DataFrame)`

Prepare data for water use for cooling & energy technologies.

`message_ix_models.model.water.data.irrigation.`**`add_irr_structure`**(*context*)

Add irrigation withdrawal infrastructure The irrigation demands are added in :param context: :type context: *Context*

Returns
data – Keys are MESSAGE parameter names such as 'input', 'fix_cost'. Values are data frames ready for `add_par()`. Years in the data include the model horizon indicated by `context["water build info"]`, plus the additional year 2010.

Return type
`dict` of `(str -> pandas.DataFrame)`

Utilities and CLI

`message_ix_models.model.water.utils.`**`map_add_on`**(*rtype=<class 'sdmx.model.common.Code'>*)

Map addon & type_addon in `sets.yaml`.

`message_ix_models.model.water.utils.`**`map_yv_ya_lt`**(*periods: Tuple[int, ...]*, *lt: int*, *ya: int | None = None*) → DataFrame

All meaningful combinations of (vintage year, active year) given *periods*.

Parameters

- **`labels`** (`pandas.DataFrame`) – Each column (dimension) corresponds to one in *df*. Each row represents one matched set of labels for those dimensions.
- **`lt`** (`int`, `lifetime`) –

`message_ix_models.model.water.cli.`**`cooling`**(*context*, *regions*, *rcps*, *rels*)

Build and solve model with new cooling technologies.

Use the –url option to specify the base scenario.

Parameters

- **`context`** (*class*:message.Context) – Information about target Scenario.
- **`regions`** (`str (if not defined already in context.regions)`) – Specifies what region definition is used ['R11','R12','ISO3']
- **`RCP`** (`str`) – Specifies the climate scenario used ['no_climate','6p0','2p6']

message_ix_models.model.water.cli.**nexus**(*context*, *regions*, *rcps*, *sdgs*, *rels*, *macro=False*)

Add basin structure connected to the energy sector and water balance linking different water demands to supply.

Use the –url option to specify the base scenario.

Parameters

- **context** (*class*:message.Context) – Information about target Scenario.
- **regions** (str (if not defined already in context.regions)) – Specifies what region definition is used ['R11','R12','ISO3']
- **RCP** (str) – Specifies the climate scenario used ['no_climate','6p0','2p6']
- **SDG** (Str) – Defines if and what water SDG measures are activated
- **REL** (str) – Specifies the reliability of hydrological data ['low','mid','high']

message_ix_models.model.water.cli.**water_ini**(*context*, *regions*, *time*)

Add components of the MESSAGEix-Nexus module

This function modifies model name & scenario name and verifies the region setup :param context: Information about target Scenario. :type context: *class*:message.Context :param regions: Specifies what region definition is used ['R11','R12','ISO3'] :type regions: str (if not defined already in context.regions)

Reporting

Warning: The current reporting features only work for the global model.

message_ix_models.model.water.reporting.**multiply_electricity_output_of_hydro**(*elec_hydro_var*, *report_iam*)

Function to multiply electricity output of hydro to get withdrawals :param elec_hydro_var: List of variables with electricity output of hydro :type elec_hydro_var: list :param report_iam: Report in pyam format :type report_iam: pyam.IamDataFrame

Returns
report_iam – Report in pyam format

Return type
pyam.IamDataFrame

message_ix_models.model.water.reporting.**report**(*sc=False*, *reg=''*, *sdgs=False*)

Report nexus module results

message_ix_models.model.water.reporting.**report_full**(*sc=False*, *reg=''*, *sdgs=False*)

Combine old and new reporting workflows

message_ix_models.model.water.reporting.**report_iam_definition**(*sc*, *rep*, *df_dmd*, *rep_dm*, *report_df*, *suban*)

Function to define the report iam dataframe :param sc: Scenario to report :type sc: ixmp.Scenario :param rep: Reporter object :type rep: Reporter :param suban: True if subannual, False if annual :type suban: bool :param df_dmd: Dataframe with demands :type df_dmd: pd.DataFrame :param rep_dm: Reporter object for demands :type rep_dm: Reporter :param report_df: Dataframe with report :type report_df: pd.DataFrame

Returns
report_iam – Report in pyam format

Return type
pyam.IamDataFrame

8.15.3 Data, metadata, and config files

See also *Data and configuration files*.

- data/water/: contains input data used for building the Nexus module
  - delineation/: contains geospatial files for basin mapping and MESSAGE regions. These spatial files are created through intersecting HydroSHEDS basin and the MESSAGE region shapefile. The scripts and processing data at 'P:ene.modelNESTdelineation'
  - ppl_cooling_tech/: contains cooling technology shares, costs and water intensities for different regional definitions
  - water_demands/: contains water sectoral demands, connection rates for basins
  - water_dist/: contains water infrastructure (distribution, treatment mapping) and historical and projected capacities of desalination technologies
  - technology.yaml: metadata for the 'technology' dimension.
  - set.yaml: metadata for other sets.

8.15.4 Pre-processing

- data/water/: contains scripts used in pre_processing source data for the water sector implementation
  - calculate_ppl_cooling_technology_shares.r: contains script for processing cooling technology shares at global level for different regional specifications.
  - groundwater_harmonize.r: contains workflow to calculate historical capacity of renewable groundwater, table depth and energy consumption
  - generate_water_constraints.r: contains function to calculate municipal, manufacturing, rural water demands, water access and sanitation rates
  - desalination.r: contains script for assessing the historical and possible future desalination capacity of a region or country
  - hydro_agg_raster.py: contains workflow for processing the hydrological data in NC4 and adjust the unit conversions, daily to monthly aggregation.
  - hydro_agg_spatial.R: contains workflow for spatially aggregating monthly hydrological data onto basin using appropriate raster masking onto shapefiles
  - hydro_agg_basin.py: contains workflow for aggregating monthly data to 5 yearly averages using appropriate statistical methods (quantiles, averages etc.). It also calculates e flows based on Variable MF method.

8.15.5 Deprecated R Code

- data/water/deprecated: contains *R* scripts from the older water sector implementation
  - Figures.R: R script for producing figures
  - cooling_tech_av.R: contains similar code as in the above-mentioned scripts, but this was originated from another workstream.
  - add_water_infrastructure.R: contains spatially-explicit analysis of gridded demands and socioeconomic indicators to develop pathways for sectoral water withdrawals, return flows and infrastructure penetration rates in each MESSAGE region. The pathways feature branching points reflecting a specific water sector development narrative (e.g., convergence towards achieving specific SDG targets).

8.15.6 Reference

Data and configuration files

- *Technology (`water/technology.yaml`)*
- *Other sets (`water/set.yaml`)*
- *Configuration (`water/config.yaml`)*

Technology (`water/technology.yaml`)

```yaml
#  Water technologies
nexus:
  extract_surfacewater:
    description: >-
      Freshwater extraction technology
    input: {commodity: freshwater_supply}

  extract_salinewater:
    description: >-
      Sea water extraction used for cooling technologies requiring seawater
    input: {commodity: saline_supply}

  extract_groundwater:
    description: >-
      Groundwater extraction technology
    input: {commodity: freshwater_supply}

  extract_gw_fossil:
    description: >-
      Groundwater extraction from fossil reservoirs (unlimited used as slack)

  extract_salinewater_basin:
    description: >-
      Sea water extraction used for desalination and providing freshwater
    input: {commodity: saline_supply_basin}

  bio_hpl__ot_fresh:
    description: >-
      Biomass heating power plant cooling by once through cooling technology
      using freshwater
    input: {commodity: freshwater_supply}

  bio_hpl__cl_fresh:
    description: >-
      Biomass heating power plant cooling by closed loop cooling technology
      using freshwater  & freshwater supply
    input: {commodity: freshwater_supply, electr}

  bio_hpl__ot_saline:
    description: >-
      Biomass heating power plant cooling by once through cooling technology
      using saline
    input: {commodity: saline_supply_ppl}

  bio_hpl__air:
    description: >-
```

(continues on next page)

(continued from previous page)

```
    Biomass heating power plant cooling by air cooled technology
    using freshwater supply
  input: {commodity: electr}

bio_istig__ot_fresh:
  description: >-
    # TODO fill in description of bio_istig cooling by once through cooling
    # technology using freshwater
  input: {commodity: freshwater_supply}

bio_istig__cl_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by closed loop  cooling
  # technology using freshwater
  input: {commodity: freshwater_supply,electr}

bio_istig__ot_saline:
  description: >-
  #   TODO fill in description of bio_istig through once through cooling
  #   technology using saline water
  input: {commodity: saline_supply_ppl}

bio_istig__air:
  description: >-
  #   TODO fill in description of bio_istig through air cooled
  #   technology using freshwater supply
  input: {commodity: electr}


bio_istig_ccs__ot_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by once through cooling
  # technology using freshwater
  input: {commodity: freshwater_supply}

bio_istig_ccs__cl_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by closed loop  cooling
  # technology using freshwater
  input: {commodity: freshwater_supply,electr}

bio_istig_ccs__ot_saline:
  description: >-
  #   TODO fill in description of bio_istig through once through cooling
  #   technology using saline water
  input: {commodity: saline_supply_ppl}

bio_istig_ccs__air:
  description: >-
  #   TODO fill in description of bio_istig through air cooled
  #   technology using freshwater supply
  input: {commodity: electr}

bio_ppl__ot_fresh:
  description: >-
    Bio power plant cooling by once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

bio_ppl__cl_fresh:
  description: >-
    Bio power plant cooling by closed loop cooling technology using
```

(continues on next page)

(continued from previous page)

```
    freshwater & freshwater supply
  input: {commodity: freshwater_supply, electr}

bio_ppl__ot_saline:
  description: >-
    Bio power plant cooling by once through cooling technology
    using saline water
  input: {commodity: saline_supply_ppl}

bio_ppl__air:
  description: >-
    Biopower plant cooling by air cooled technology
    using freshwater supply
  input: {commodity: electr}

coal_adv__ot_fresh:
  description: >-
    Advanced coal power plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

coal_adv__cl_fresh:
  description: >-
    Advanced coal power plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

coal_adv__ot_saline:
  description: >-
    Advanced coal power plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

coal_adv__air:
  description: >-
    Advanced coal power plant cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

coal_adv_ccs__ot_fresh:
  description: >-
    Advanced coal power plant with carbon capture and storage cooling by once
    through cooling technology using freshwater
  input: {commodity: freshwater_supply}

coal_adv_ccs__cl_fresh:
  description: >-
    Advanced coal power plant with carbon capture and storage by closed loop
    cooling technology using freshwater
  input: {commodity: freshwater_supply, electr}

coal_adv_ccs__ot_saline:
  description: >-
    Advanced coal power plant with carbon capture and storage by once through
    cooling technology using freshwater
  input: {commodity: saline_supply_ppl}

coal_adv_ccs__air:
  description: >-
    Advanced coal power plant with carbon capture and storage by air cooled
    cooling technology using freshwater supply
```

(continues on next page)

(continued from previous page)

```
  input: {commodity: electr}

coal_ppl__ot_fresh:
  description: >-
    Coal power plant cooling by once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

coal_ppl__cl_fresh:
  description: >-
    Coal power plant cooling by closed loop cooling technology using freshwater
    & freshwater supply
  input: {commodity: freshwater_supply,electr}

coal_ppl__ot_saline:
  description: >-
    Coal power plant cooling by once through cooling technology using freshwater
  input: {commodity: saline_supply_ppl}

coal_ppl__air:
  description: >-
    Coal power plant cooling by air cooled cooling technology using parasitic
    electricity
  input: {commodity: electr}

coal_ppl_u__ot_fresh:
  description: >-
    Coal power plant without abatement measures cooling by once through
    cooling technology using freshwater
  input: {commodity: freshwater_supply}

coal_ppl_u__cl_fresh:
  description: >-
    Coal power plant without abatement measures cooling by closed loop
    cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

coal_ppl_u__ot_saline:
  description: >-
    Coal power plant without abatement measures cooling by once through
    cooling technology using saline water
  input: {commodity: saline_supply_ppl}

coal_ppl_u__air:
  description: >-
    Coal power plant without abatement measures cooling by air cooled
    cooling technology using freshwater supply
  input: {commodity: electr}

foil_ppl__ot_fresh:
  description: >-
    New standard oil power plant (Rankine cycle), cooling by once through
    cooling technology using freshwater
    using freshwater
  input: {commodity: freshwater_supply}

foil_ppl__cl_fresh:
  description: >-
    New standard oil power plant (Rankine cycle),cooling by closed loop
    cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply, electr}
```

(continues on next page)

(continued from previous page)

```
foil_ppl__ot_saline:
  description: >-
    New standard oil power plant (Rankine cycle),cooling by once through
    cooling technology using saline water
  input: {commodity: saline_supply_ppl}

foil_ppl__air:
  description: >-
    New standard oil power plant (Rankine cycle),cooling by air cooled
    cooling technology using parasitic electricity
  input: {commodity: electr}

foil_hpl__ot_fresh:
  description: >-
    Fuel oil heating plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

foil_hpl__cl_fresh:
  description: >-
    Fuel oil heating plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

foil_hpl__ot_saline:
  description: >-
    Fuel oil heating plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

foil_hpl__air:
  description: >-
    Fuel oil heating plant cooling by cooling by air cooled cooling technology
    using parasitic electricity
  input: {commodity: electr}

gas_cc__ot_fresh:
  description: >-
    Gas combined cycle power plant cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}

gas_cc__cl_fresh:
  description: >-
    Gas combined cycle power plant cooling by closed loop cooling technology
    using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_cc__ot_saline:
  description: >-
    Gas combined cycle power plant cooling by once through cooling technology
    using saline water
  input: {commodity: saline_supply_ppl}

gas_cc__air:
  description: >-
    Gas combined cycle power plant cooling by air cooled cooling technology
    using parasitic electricity
  input: {commodity: electr}

gas_cc_ccs__ot_fresh:
```

(continues on next page)

(continued from previous page)

```
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

gas_cc_ccs__cl_fresh:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    closed loop cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_cc_ccs__ot_saline:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    once through cooling technology using saline water
  input: {commodity: saline_supply_ppl}

gas_cc_ccs__air:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    air cooled cooling technology using parasitic electricity
  input: {commodity: electr}

gas_hpl__ot_fresh:
  description: >-
    Natural gas heating plant plant cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}

gas_hpl__cl_fresh:
  description: >-
    Natural gas heating plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_hpl__ot_saline:
  description: >-
    Natural gas heating plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

gas_hpl__air:
  description: >-
    Natural gas heating plant cooling by once through cooling technology by
    air cooled cooling technology using parasitic electricity
  input: {commodity: electr}

gas_ppl__ot_fresh:
  description: >-
    Gas power plant (Rankine cycle) cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}

gas_ppl__cl_fresh:
  description: >-
    Gas power plant (Rankine cycle) cooling by closed loop cooling technology
    using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_ppl__ot_saline:
  description: >-
```

(continues on next page)

(continued from previous page)

```
    Biomass heating power plant cooling by once through cooling technology using
    freshwater
  input: {commodity: saline_supply_ppl}

gas_ppl__air:
  description: >-
    Biomass heating power plant cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

geo_hpl__ot_fresh:
  description: >-
    Geothermal heat plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

geo_hpl__cl_fresh:
  description: >-
    Geothermal heat plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

geo_hpl__ot_saline:
  description: >-
    Geothermal heat plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

geo_hpl__air:
  description: >-
    Geothermal heat plant cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

geo_ppl__ot_fresh:
  description: >-
    Geothermal power plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

geo_ppl__cl_fresh:
  description: >-
    Geothermal power plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

geo_ppl__ot_saline:
  description: >-
    Geothermal power plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

geo_ppl__air:
  description: >-
    Geothermal power plant cooling by closed loop cooling technology using
    freshwater supply
  input: {commodity: electr}

igcc__ot_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by once
```

(continues on next page)

(continued from previous page)

```
    through cooling technology using freshwater
  input: {commodity: freshwater_supply}

igcc__cl_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by closed
    loop cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

igcc__ot_saline:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by once
    through cooling technology using saline water
  input: {commodity: saline_supply_ppl}

igcc__air:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by air␣
↪cooled
    cooling technology using freshwater supply
  input: {commodity: electr}

igcc_ccs__ot_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

igcc_ccs__cl_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by closed loop cooling technology using freshwater &
    freshwater supply
  input: {commodity: freshwater_supply,electr}

igcc_ccs__ot_saline:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by once through cooling technology using saline water
  input: {commodity: saline_supply_ppl}

igcc_ccs__air:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by air cooled cooling technology using freshwater supply
  input: {commodity: electr}

loil_cc__ot_fresh:
  description: >-
    Light oil combined cycle cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

loil_cc__cl_fresh:
  description: >-
    Light oil combined cycle cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

loil_cc__ot_saline:
  description: >-
```

(continues on next page)

(continued from previous page)

```
    Light oil combined cycle cooling by once through cooling technology using
    saline supply
  input: {commodity: saline_supply_ppl}

loil_cc__air:
  description: >-
    Light oil combined cycle cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

loil_ppl__ot_fresh:
  description: >-
    Existing light oil power plant cooling by once through cooling technology
    using extract_freshwater_supply
  input: {commodity: freshwater_supply}

loil_ppl__cl_fresh:
  description: >-
    Existing light oil power plant cooling by once through cooling technology␣
↪using
    freshwater supply
  input: {commodity: freshwater_supply,electr}

loil_ppl__ot_saline:
  description: >-
    Existing light oil power plant cooling by once through cooling technology
    using freshwater supply
  input: {commodity: electr}

loil_ppl__air:
  description: >-
    Existing light oil power plant cooling by air cooled cooling technology
    using parasitic electricity
  input: {commodity: electr}

nuc_hc__ot_fresh:
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by once through cooling
    technology using freshwater supply
  input: {commodity: freshwater_supply}

nuc_hc__cl_fresh:
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by closed loop cooling
    technology using freshwater supply & parasitic
  input: {commodity: freshwater_supply,electr}

nuc_hc__ot_saline:
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by once through cooling
    technology using saline water supply
  input: {commodity: saline_supply_ppl}

nuc_hc__air:
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by once through cooling
    technology using parasitic electricity
  input: {commodity: electr}

nuc_lc__ot_fresh:
  description: >-
```

(continues on next page)

(continued from previous page)

```
    Nuclear power plant (~GEN III+) (low cost) cooling by once through cooling
    technology using freshwater supply
  input: {commodity: freshwater_supply}

nuc_lc__cl_fresh:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by closed loop cooling
    technology using freshwater supply & parasitic
  input: {commodity: freshwater_supply,electr}

nuc_lc__ot_saline:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by once through cooling
    technology using saline water supply
  input: {commodity: saline_supply_ppl}

nuc_lc__air:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by once through cooling
    technology using parasitic electricity
  input: {commodity: electr}

solar_th_ppl__ot_fresh:
  description: >-
    Solar thermal power plant with storage cooling by once through cooling
    technology using freshwater supply
  input: {commodity: freshwater_supply}

solar_th_ppl__cl_fresh:
  description: >-
    Solar thermal power plant with storage cooling by closed loop cooling
    technology using freshwater supply & parasitic electricity
  input: {commodity: freshwater_supply,electr}

solar_th_ppl__ot_saline:
  description: >-
    Solar thermal power plant with storage cooling by once through cooling
    technology using freshwater supply
  input: {commodity: saline_supply_ppl}

solar_th_ppl__air:
  description: >-
    Solar thermal power plant with storage cooling by once through cooling
    technology using parasitic electricity
  input: {commodity: electr}

urban_t_d:
  description: >-
    Urban water transmission & distribution
  input: {commodity: freshwater_supply}

rural_t_d:
  description: >-
    Rural water transmission & distribution
  input: {commodity: freshwater_supply}

industry_unconnected:
  description: >-
    unconnected industry water not connected to system
  input: {commodity: freshwater_supply}
```

(continues on next page)

(continued from previous page)

```
industry_untreated:
  description: >-
    untreated industry water return flows
  input: {commodity: urban_uncollected_wst}

urban_unconnected:
  description: >-
    untreated urban water not connected to system
  input: {commodity: freshwater_supply}

rural_unconnected:
  description: >-
    untreated rural water not connected to system
  input: {commodity: freshwater_supply}

urban_sewerage:
  description: >-
    urban wastewater
  input: {commodity: freshwater_supply}

urban_untreated:
  description: >-
    untreated urban water return flows
  input: {commodity: urban_uncollected_wst}

urban_discharge:
  description: >-
    treated urban water which is discharged
  input: {commodity: electr, urban_collected_wst}

urban_recycle:
  description: >-
    treated wastewater used as available water
  input: {commodity: electr, urban_collected_wst}

rural_discharge:
  description: >-
    treated rural water which is discharged
  input: {commodity: rural_collected_wst}

rural_untreated:
  description: >-
    untreated rural water return flows
  input: {commodity: rural_collected_wst}

membrane:
  description: >-
    desalination technology using membrane i.e water and salts is separated
    through a semipermeable membrane
  input: {commodity: electr,saline_supply}

distillation:
  description: >-
    traditional desalination technology i.e boiling and recondensation of
    seawater to leave salt  and impurities behin
  input: {commodity: electr,saline_supply,d_heat}

desal_t_d:
  description: >-
    Desalinated water transmission and distribution
  input: {commodity: desalinated_water}
```

(continues on next page)

(continued from previous page)

```
  saline_ppl_t_d:
    description: >-
      Saline power plant transmission & distribution
      # TODO verify description
    input: {commodity: saline_supply}

  basin_to_reg:
    description: >-
      dummy technology to map basin technologies water supply to energy␣
↪technologies

    input: {commodity: freshwater_supply}

  ueff1:
    description: >-
      low urban efficiency

    input: {commodity: urban_mw, urban_collected_wst}

  ueff2:
    description: >-
      low urban efficiency

    input: {commodity: urban_mw, urban_collected_wst}

  ueff3:
    description: >-
      low urban efficiency

    input: {commodity: urban_mw, urban_collected_wst}

  reff1:
    description: >-
      low rural efficiency

    input: {commodity: rural_mw, rural_collected_wst}

  reff2:
    description: >-
      mid rural efficiency

    input: {commodity: rural_mw, rural_collected_wst}

  reff3:
    description: >-
      high rural efficiency

    input: {commodity: rural_mw, rural_collected_wst}

  ieff1:
    description: >-
      low irrigation efficiency

    input: {commodity: freshwater_supply}

  ieff2:
    description: >-
      mid irrigation efficiency

    input: {commodity: freshwater_supply}
```

(continues on next page)

(continued from previous page)

```
  ieff3:
    description: >-
      high irrigation efficiency

    input: {commodity: freshwater_supply}

  return_flow:
    description: >-
      remaining surface water left after consumption

  gw_recharge:
    description: >-
      remaining groundwater left after consumption

  rural_recycle:
    description: >-
      recycled water after rural treatment

  rural_sewerage:
    description: >-
      treated rural water

  irrigation_oilcrops:
    description: >-
      Irrigation technology connecting irrigation withdrawals with the demands
    input: {commodity: freshwater}

  irrigation_sugarcrops:
    description: >-
      Irrigation technology connecting irrigation withdrawals with the demands
    input: {commodity: freshwater}

  irrigation_cereal:
    description: >-
      Irrigation technology connecting irrigation withdrawals with the demands
    input: {commodity: freshwater}

  salinewater_return:
    description: >-
      remaining salinewater left after consumption


cooling:
  #  Water technologies
  extract_surfacewater:
    description: >-
      Freshwater extraction technology
    input: {commodity: freshwater_supply}

  extract_salinewater:
    description: >-
      Sea water extraction used for cooling technologies requiring seawater
    input: {commodity: saline_supply_ppl}

  extract_groundwater:
    description: >-
      Groundwater extraction technology
    input: {commodity: freshwater_supply}

  bio_hpl__ot_fresh:
```

(continues on next page)

(continued from previous page)

```
  description: >-
    Biomass heating power plant cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}

bio_hpl__cl_fresh:
  description: >-
    Biomass heating power plant cooling by closed loop cooling technology
    using freshwater  & freshwater supply
  input: {commodity: freshwater_supply, electr}

bio_hpl__ot_saline:
  description: >-
    Biomass heating power plant cooling by once through cooling technology
    using saline
  input: {commodity: saline_supply_ppl}

bio_hpl__air:
  description: >-
    Biomass heating power plant cooling by air cooled technology
    using freshwater supply
  input: {commodity: electr}

bio_istig__ot_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by once through cooling
  # technology using freshwater
  input: {commodity: freshwater_supply}

bio_istig__cl_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by closed loop  cooling
  # technology using freshwater
  input: {commodity: freshwater_supply,electr}

bio_istig__ot_saline:
  description: >-
  #   TODO fill in description of bio_istig through once through cooling
  #   technology using saline water
  input: {commodity: saline_supply_ppl}

bio_istig__air:
  description: >-
  #   TODO fill in description of bio_istig through air cooled
  #   technology using freshwater supply
  input: {commodity: electr}


bio_istig_ccs__ot_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by once through cooling
  # technology using freshwater
  input: {commodity: freshwater_supply}

bio_istig_ccs__cl_fresh:
  description: >-
  # TODO fill in description of bio_istig cooling by closed loop  cooling
  # technology using freshwater
  input: {commodity: freshwater_supply,electr}

bio_istig_ccs__ot_saline:
```

(continues on next page)

(continued from previous page)

```
  description: >-
  #   TODO fill in description of bio_istig through once through cooling
  #   technology using saline water
  input: {commodity: saline_supply_ppl}

bio_istig_ccs__air:
  description: >-
  #   TODO fill in description of bio_istig through air cooled
  #   technology using freshwater supply
  input: {commodity: electr}

bio_ppl__ot_fresh:
  description: >-
    Bio power plant cooling by once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

bio_ppl__cl_fresh:
  description: >-
    Bio power plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply, electr}

bio_ppl__ot_saline:
  description: >-
    Bio power plant cooling by once through cooling technology
    using saline water
  input: {commodity: saline_supply_ppl}

bio_ppl__air:
  description: >-
    Biopower plant cooling by air cooled technology
    using freshwater supply
  input: {commodity: electr}

coal_adv__ot_fresh:
  description: >-
    Advanced coal power plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

coal_adv__cl_fresh:
  description: >-
    Advanced coal power plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

coal_adv__ot_saline:
  description: >-
    Advanced coal power plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

coal_adv__air:
  description: >-
    Advanced coal power plant cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

coal_adv_ccs__ot_fresh:
  description: >-
    Advanced coal power plant with carbon capture and storage cooling by once
```

(continues on next page)

(continued from previous page)

```
    through cooling technology using freshwater
  input: {commodity: freshwater_supply}

coal_adv_ccs__cl_fresh:
  description: >-
    Advanced coal power plant with carbon capture and storage by closed loop
    cooling technology using freshwater
  input: {commodity: freshwater_supply, electr}

coal_adv_ccs__ot_saline:
  description: >-
    Advanced coal power plant with carbon capture and storage by once through
    cooling technology using freshwater
  input: {commodity: saline_supply_ppl}

coal_adv_ccs__air:
  description: >-
    Advanced coal power plant with carbon capture and storage by air cooled
    cooling technology using freshwater supply
  input: {commodity: electr}

coal_ppl__ot_fresh:
  description: >-
    Coal power plant cooling by once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

coal_ppl__cl_fresh:
  description: >-
    Coal power plant cooling by closed loop cooling technology using freshwater
    & freshwater supply
  input: {commodity: freshwater_supply,electr}

coal_ppl__ot_saline:
  description: >-
    Coal power plant cooling by once through cooling technology using freshwater
  input: {commodity: saline_supply_ppl}

coal_ppl__air:
  description: >-
    Coal power plant cooling by air cooled cooling technology using parasitic
    electricity
  input: {commodity: electr}

coal_ppl_u__ot_fresh:
  description: >-
    Coal power plant without abatement measures cooling by once through
    cooling technology using freshwater
  input: {commodity: freshwater_supply}

coal_ppl_u__cl_fresh:
  description: >-
    Coal power plant without abatement measures cooling by closed loop
    cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

coal_ppl_u__ot_saline:
  description: >-
    Coal power plant without abatement measures cooling by once through
    cooling technology using saline water
  input: {commodity: saline_supply_ppl}
```

(continues on next page)

(continued from previous page)

```
coal_ppl_u__air:
  description: >-
    Coal power plant without abatement measures cooling by air cooled
    cooling technology using freshwater supply
  input: {commodity: electr}

foil_ppl__ot_fresh:
  description: >-
    New standard oil power plant (Rankine cycle), cooling by once through
    cooling technology using freshwater
    using freshwater
  input: {commodity: freshwater_supply}

foil_ppl__cl_fresh:
  description: >-
    New standard oil power plant (Rankine cycle),cooling by closed loop
    cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply, electr}

foil_ppl__ot_saline:
  description: >-
    New standard oil power plant (Rankine cycle),cooling by once through
    cooling technology using saline water
  input: {commodity: saline_supply_ppl}

foil_ppl__air:
  description: >-
    New standard oil power plant (Rankine cycle),cooling by air cooled
    cooling technology using parasitic electricity
  input: {commodity: electr}

foil_hpl__ot_fresh:
  description: >-
    Fuel oil heating plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

foil_hpl__cl_fresh:
  description: >-
    Fuel oil heating plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

foil_hpl__ot_saline:
  description: >-
    Fuel oil heating plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

foil_hpl__air:
  description: >-
    Fuel oil heating plant cooling by cooling by air cooled cooling technology
    using parasitic electricity
  input: {commodity: electr}

gas_cc__ot_fresh:
  description: >-
    Gas combined cycle power plant cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}
```

(continues on next page)

(continued from previous page)

```
gas_cc__cl_fresh:
  description: >-
    Gas combined cycle power plant cooling by closed loop cooling technology
    using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_cc__ot_saline:
  description: >-
    Gas combined cycle power plant cooling by once through cooling technology
    using saline water
  input: {commodity: saline_supply_ppl}

gas_cc__air:
  description: >-
    Gas combined cycle power plant cooling by air cooled cooling technology
    using parasitic electricity
  input: {commodity: electr}

gas_cc_ccs__ot_fresh:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

gas_cc_ccs__cl_fresh:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    closed loop cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_cc_ccs__ot_saline:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    once through cooling technology using saline water
  input: {commodity: saline_supply_ppl}

gas_cc_ccs__air:
  description: >-
    Gas combined cycle power-plant with carbon capture and storage cooling by
    air cooled cooling technology using parasitic electricity
  input: {commodity: electr}

gas_hpl__ot_fresh:
  description: >-
    Natural gas heating plant plant cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}

gas_hpl__cl_fresh:
  description: >-
    Natural gas heating plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_hpl__ot_saline:
  description: >-
    Natural gas heating plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

gas_hpl__air:
```

(continues on next page)

(continued from previous page)

```
  description: >-
    Natural gas heating plant cooling by once through cooling technology by
    air cooled cooling technology using parasitic electricity
  input: {commodity: electr}

gas_ppl__ot_fresh:
  description: >-
    Gas power plant (Rankine cycle) cooling by once through cooling technology
    using freshwater
  input: {commodity: freshwater_supply}

gas_ppl__cl_fresh:
  description: >-
    Gas power plant (Rankine cycle) cooling by closed loop cooling technology
    using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

gas_ppl__ot_saline:
  description: >-
    Biomass heating power plant cooling by once through cooling technology using
    freshwater
  input: {commodity: saline_supply_ppl}

gas_ppl__air:
  description: >-
    Biomass heating power plant cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

geo_hpl__ot_fresh:
  description: >-
    Geothermal heat plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

geo_hpl__cl_fresh:
  description: >-
    Geothermal heat plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

geo_hpl__ot_saline:
  description: >-
    Geothermal heat plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

geo_hpl__air:
  description: >-
    Geothermal heat plant cooling by air cooled cooling technology using
    freshwater supply
  input: {commodity: electr}

geo_ppl__ot_fresh:
  description: >-
    Geothermal power plant cooling by once through cooling technology using
    freshwater
  input: {commodity: freshwater_supply}

geo_ppl__cl_fresh:
  description: >-
```

(continues on next page)

(continued from previous page)

```
    Geothermal power plant cooling by closed loop cooling technology using
    freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

geo_ppl__ot_saline:
  description: >-
    Geothermal power plant cooling by once through cooling technology using
    saline water
  input: {commodity: saline_supply_ppl}

geo_ppl__air:
  description: >-
    Geothermal power plant cooling by closed loop cooling technology using
    freshwater supply
  input: {commodity: electr}

igcc__ot_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by once
    through cooling technology using freshwater
  input: {commodity: freshwater_supply}

igcc__cl_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by closed
    loop cooling technology using freshwater & freshwater supply
  input: {commodity: freshwater_supply,electr}

igcc__ot_saline:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by once
    through cooling technology using saline water
  input: {commodity: saline_supply_ppl}

igcc__air:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant cooling by air␣
↪cooled
    cooling technology using freshwater supply
  input: {commodity: electr}

igcc_ccs__ot_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by once through cooling technology using freshwater
  input: {commodity: freshwater_supply}

igcc_ccs__cl_fresh:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by closed loop cooling technology using freshwater &
    freshwater supply
  input: {commodity: freshwater_supply,electr}

igcc_ccs__ot_saline:
  description: >-
    Integrated gasification combined cycle (IGCC) power plant with carbon capture
    and storage cooling by once through cooling technology using saline water
  input: {commodity: saline_supply_ppl}

igcc_ccs__air:
```

(continues on next page)

(continued from previous page)

```
    description: >-
      Integrated gasification combined cycle (IGCC) power plant with carbon capture
      and storage cooling by air cooled cooling technology using freshwater supply
    input: {commodity: electr}

  loil_cc__ot_fresh:
    description: >-
      Light oil combined cycle cooling by once through cooling technology using
      freshwater
    input: {commodity: freshwater_supply}

  loil_cc__cl_fresh:
    description: >-
      Light oil combined cycle cooling by closed loop cooling technology using
      freshwater & freshwater supply
    input: {commodity: freshwater_supply,electr}

  loil_cc__ot_saline:
    description: >-
      Light oil combined cycle cooling by once through cooling technology using
      saline supply
    input: {commodity: saline_supply_ppl}

  loil_cc__air:
    description: >-
      Light oil combined cycle cooling by air cooled cooling technology using
      freshwater supply
    input: {commodity: electr}

  loil_ppl__ot_fresh:
    description: >-
      Existing light oil power plant cooling by once through cooling technology
      using extract_freshwater_supply
    input: {commodity: freshwater_supply}

  loil_ppl__cl_fresh:
    description: >-
      Existing light oil power plant cooling by once through cooling technology␣
↪using
      freshwater supply
    input: {commodity: freshwater_supply,electr}

  loil_ppl__ot_saline:
    description: >-
      Existing light oil power plant cooling by once through cooling technology
      using freshwater supply
    input: {commodity: electr}

  loil_ppl__air:
    description: >-
      Existing light oil power plant cooling by air cooled cooling technology
      using parasitic electricity
    input: {commodity: electr}

  nuc_hc__ot_fresh:
    description: >-
      Nuclear power plant (~GEN III+) (high cost) cooling by once through cooling
      technology using freshwater supply
    input: {commodity: freshwater_supply}

  nuc_hc__cl_fresh:
```

(continues on next page)

(continued from previous page)

```
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by closed loop cooling
    technology using freshwater supply & parasitic
  input: {commodity: freshwater_supply,electr}

nuc_hc__ot_saline:
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by once through cooling
    technology using saline water supply
  input: {commodity: saline_supply_ppl}

nuc_hc__air:
  description: >-
    Nuclear power plant (~GEN III+) (high cost) cooling by once through cooling
    technology using parasitic electricity
  input: {commodity: electr}

nuc_lc__ot_fresh:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by once through cooling
    technology using freshwater supply
  input: {commodity: freshwater_supply}

nuc_lc__cl_fresh:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by closed loop cooling
    technology using freshwater supply & parasitic
  input: {commodity: freshwater_supply,electr}

nuc_lc__ot_saline:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by once through cooling
    technology using saline water supply
  input: {commodity: saline_supply_ppl}

nuc_lc__air:
  description: >-
    Nuclear power plant (~GEN III+) (low cost) cooling by once through cooling
    technology using parasitic electricity
  input: {commodity: electr}

solar_th_ppl__ot_fresh:
  description: >-
    Solar thermal power plant with storage cooling by once through cooling
    technology using freshwater supply
  input: {commodity: freshwater_supply}

solar_th_ppl__cl_fresh:
  description: >-
    Solar thermal power plant with storage cooling by closed loop cooling
    technology using freshwater supply & parasitic electricity
  input: {commodity: freshwater_supply,electr}

solar_th_ppl__ot_saline:
  description: >-
    Solar thermal power plant with storage cooling by once through cooling
    technology using freshwater supply
  input: {commodity: saline_supply_ppl}

solar_th_ppl__air:
  description: >-
```

(continues on next page)

(continued from previous page)

```
      Solar thermal power plant with storage cooling by once through cooling
      technology using parasitic electricity
    input: {commodity: electr}
```

Other sets (`water/set.yaml`)

```
#Set configuration for the MESSAGE-Water model

# For each set in the MESSAGEix framework, the group contains:
# - 'require': elements that must be present for the model to be set up.
# - 'remove': elements to remove.
# - 'add': elements to add. This is a mapping from element short names to a
#   longer description.

nexus:
  commodity:
    require:
    - electr

    remove:
    # removing all technologies which are related to water in the previous model
    # Re added in the newer implementation with lesser commodities
    - saline_supply
    #- freshwater_instream
    - freshwater_supply
    - desalinated_water
    - saline_supply_ppl
    - urban_collected_wst
    - urban_uncollected_wst
    - yield_freshwater_suply
    - rural_collected_wst
    - rural_uncollected_wst
    - urban_mw
    - urban_dis
    - rural_mw
    - rural_dis
    - cooling__bio_hpl
    - cooling__bio_ppl
    - cooling__bio_istig
    - cooling__bio_ppl
    - cooling__coal_adv
    - cooling__coal_adv_ccs
    - cooling__coal_ppl
    - cooling__coal_ppl_u
    - cooling__foil_hpl
    - cooling__foil_ppl
    - cooling__gas_cc
    - cooling__gas_cc_ccs
    - cooling__gas_hpl
    - cooling__gas_ppl
    - cooling__geo_hpl
    - cooling__geo_ppl
    - cooling__igcc
    - cooling__igcc_ccs
    - cooling__loil_cc
    - cooling__loil_ppl
    - cooling__nuc_hc
    - cooling__solar_th_ppl
```

(continues on next page)

(continued from previous page)

```
  add:
  - electr
  - salinewater
  - freshwater_instream
  - freshwater
  - freshwater_basin
  - surfacewater_basin
  - groundwater_basin
  - salinewater_basin
  - desalinated_water
  - saline_ppl
  - urban_collected_wst
  - urban_uncollected_wst
  - yield_freshwater_suply
  - rural_collected_wst
  - rural_uncollected_wst
  - urban_mw
  - industry_mw
  - industry_uncollected_wst
  - urban_disconnected
  - rural_mw
  - rural_disconnected


level:
  require:
  - secondary

  remove:
  - water_supply
  - cooling
  - desalination_supply
  - water_treat
  add:
  - water_supply
  - water_supply_basin
  - saline_supply_basin
  - water_treat
  - water_avail_basin
  - saline_supply
  - irr_cereal
  - irr_sugarcrops
  - irr_oilcrops

mode:
  require:
  - all
  add:
  - Mf # efficient mode

technology:
  remove:
  - rural_treatment
  - urban_treatment
  - extract__saline_supply
  - bio_hpl__ot_fresh
  - bio_hpl__cl_fresh
  - bio_hpl__cl_fresh
  - bio_hpl__ot_saline
  - bio_hpl__air
  - bio_istig__ot_fresh
```

(continues on next page)

(continued from previous page)

```
    - bio_istig__cl_fresh
    - bio_istig__cl_fresh
    - bio_istig__ot_saline
    - bio_istig__air
    - bio_istig_ccs__ot_fresh
    - bio_istig_ccs__cl_fresh
    - bio_istig_ccs__cl_fresh
    - bio_istig_ccs__ot_saline
    - bio_istig_ccs__air
    - bio_ppl__ot_fresh
    - bio_ppl__cl_fresh
    - bio_ppl__cl_fresh
    - bio_ppl__ot_saline
    - bio_ppl__air
    - coal_adv__ot_fresh
    - coal_adv__cl_fresh
    - coal_adv__cl_fresh
    - coal_adv__ot_saline
    - coal_adv__air
    - coal_adv_ccs__ot_fresh
    - coal_adv_ccs__cl_fresh
    - coal_adv_ccs__cl_fresh
    - coal_adv_ccs__ot_saline
    - coal_ppl__ot_fresh
    - coal_ppl__cl_fresh
    - coal_ppl__cl_fresh
    - coal_ppl__ot_saline
    - coal_ppl__air
    - coal_ppl_u__ot_fresh
    - coal_ppl_u__cl_fresh
    - coal_ppl_u__cl_fresh
    - coal_ppl_u__ot_saline
    - coal_ppl_u__air
    - foil_hpl__ot_fresh
    - foil_hpl__cl_fresh
    - foil_hpl__cl_fresh
    - foil_hpl__ot_saline
    - foil_hpl__air
    - foil_ppl__ot_fresh
    - foil_ppl__cl_fresh
    - foil_ppl__cl_fresh
    - foil_ppl__ot_saline
    - foil_ppl__air
    - gas_cc__ot_fresh
    - gas_cc__cl_fresh
    - gas_cc__cl_fresh
    - gas_cc__ot_saline
    - gas_cc__air
    - gas_cc_ccs__ot_fresh
    - gas_cc_ccs__cl_fresh
    - gas_cc_ccs__cl_fresh
    - gas_cc_ccs__ot_saline
    - gas_hpl__ot_fresh
    - gas_hpl__cl_fresh
    - gas_hpl__cl_fresh
    - gas_hpl__ot_saline
    - gas_hpl__air
    - gas_ppl__ot_fresh
    - gas_ppl__cl_fresh
    - gas_ppl__cl_fresh
    - gas_ppl__ot_saline
```

(continues on next page)

(continued from previous page)

```
    - gas_ppl__air
    - geo_hpl__ot_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__ot_saline
    - geo_hpl__air
    - geo_ppl__ot_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__ot_saline
    - geo_ppl__air
    - igcc__ot_fresh
    - igcc__cl_fresh
    - igcc__cl_fresh
    - igcc__ot_saline
    - igcc__air
    - igcc_ccs__ot_fresh
    - igcc_ccs__cl_fresh
    - igcc_ccs__cl_fresh
    - igcc_ccs__ot_saline
    - loil_cc__ot_fresh
    - loil_cc__cl_fresh
    - loil_cc__cl_fresh
    - loil_cc__ot_saline
    - loil_cc__air
    - loil_ppl__ot_fresh
    - loil_ppl__cl_fresh
    - loil_ppl__cl_fresh
    - loil_ppl__ot_saline
    - loil_ppl__air
    - nuc_hc__ot_fresh
    - nuc_hc__cl_fresh
    - nuc_hc__cl_fresh
    - nuc_hc__ot_saline
    - solar_th_ppl__ot_fresh
    - solar_th_ppl__cl_fresh
    - solar_th_ppl__cl_fresh
    - solar_th_ppl__ot_saline
    - solar_th_ppl__air
    - extract__freshwater_supply
    - extract__freshwater_instream
    - extract__saline_supply
    #add:
    # Adds all technologies from technologies.yaml file again to reconfigure

  time:
    # NB cannot handle models with sub-annual time resolution
    require:
    - year

  # Addon techs include cooling technologies for now

  addon:
    add:
    - bio_hpl__ot_fresh
    - bio_hpl__cl_fresh
    - bio_hpl__ot_saline
    - bio_hpl__air
    - bio_istig__ot_fresh
    - bio_istig__cl_fresh
    - bio_istig__ot_saline
```

(continues on next page)

(continued from previous page)

```
    - bio_istig__air
    - bio_istig_ccs__ot_fresh
    - bio_istig_ccs__cl_fresh
    - bio_istig_ccs__ot_saline
    - bio_istig_ccs__air
    - bio_ppl__ot_fresh
    - bio_ppl__cl_fresh
    - bio_ppl__ot_saline
    - bio_ppl__air
    - coal_adv__ot_fresh
    - coal_adv__cl_fresh
    - coal_adv__ot_saline
    - coal_adv__air
    - coal_adv_ccs__ot_fresh
    - coal_adv_ccs__cl_fresh
    - coal_adv_ccs__ot_saline
    - coal_adv_ccs__air
    - coal_ppl__ot_fresh
    - coal_ppl__cl_fresh
    - coal_ppl__ot_saline
    - coal_ppl__air
    - coal_ppl_u__ot_fresh
    - coal_ppl_u__cl_fresh
    - coal_ppl_u__ot_saline
    - coal_ppl_u__air
    - foil_ppl__ot_fresh
    - foil_ppl__cl_fresh
    - foil_ppl__ot_saline
    - foil_ppl__air
    - foil_hpl__ot_fresh
    - foil_hpl__cl_fresh
    - foil_hpl__ot_saline
    - foil_hpl__air
    - gas_cc__ot_fresh
    - gas_cc__cl_fresh
    - gas_cc__ot_saline
    - gas_cc__air
    - gas_cc_ccs__ot_fresh
    - gas_cc_ccs__cl_fresh
    - gas_cc_ccs__ot_saline
    - gas_cc_ccs__air
    - gas_hpl__ot_fresh
    - gas_hpl__cl_fresh
    - gas_hpl__ot_saline
    - gas_hpl__air
    - gas_ppl__ot_fresh
    - gas_ppl__cl_fresh
    - gas_ppl__ot_saline
    - gas_ppl__air
    - geo_hpl__ot_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__ot_saline
    - geo_hpl__air
    - geo_ppl__ot_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__ot_saline
    - geo_ppl__air
    - igcc__ot_fresh
    - igcc__cl_fresh
```

(continues on next page)

(continued from previous page)

```
  - igcc__ot_saline
  - igcc__air
  - igcc_ccs__ot_fresh
  - igcc_ccs__cl_fresh
  - igcc_ccs__ot_saline
  - igcc_ccs__air
  - loil_cc__ot_fresh
  - loil_cc__cl_fresh
  - loil_cc__ot_saline
  - loil_cc__air
  - loil_ppl__ot_fresh
  - loil_ppl__cl_fresh
  - loil_ppl__ot_saline
  - loil_ppl__air
  - nuc_hc__ot_fresh
  - nuc_hc__cl_fresh
  - nuc_hc__ot_saline
  - nuc_hc__air
  - solar_th_ppl__ot_fresh
  - solar_th_ppl__cl_fresh
  - solar_th_ppl__ot_saline
  - solar_th_ppl__air

# cat_addon includes cooling technology addons for parent technologies
type_addon:
  add:
  - cooling__bio_hpl
  - cooling__bio_istig
  - cooling__bio_istig_ccs
  - cooling__bio_ppl
  - cooling__coal_adv
  - cooling__coal_adv_ccs
  - cooling__coal_ppl
  - cooling__coal_ppl_u
  - cooling__foil_hpl
  - cooling__foil_ppl
  - cooling__gas_cc
  - cooling__gas_cc_ccs
  - cooling__gas_hpl
  - cooling__gas_ppl
  - cooling__geo_hpl
  - cooling__geo_ppl
  - cooling__igcc
  - cooling__igcc_ccs
  - cooling__loil_cc
  - cooling__loil_ppl
  - cooling__nuc_hc
  - cooling__nuc_lc
  - cooling__solar_th_ppl

map_tec_addon:
  add:
  - [bio_hpl,cooling__bio_hpl]
  - [bio_istig, cooling__bio_istig]
  - [bio_istig_ccs, cooling__bio_istig_ccs]
  - [bio_ppl,cooling__bio_ppl]
  - [coal_adv, cooling__coal_adv]
  - [coal_adv_ccs, cooling__coal_adv_ccs]
  - [coal_ppl, cooling__coal_ppl]
  - [coal_ppl_u, cooling__coal_ppl_u]
  - [foil_hpl, cooling__foil_hpl]
```

(continues on next page)

(continued from previous page)

```
    - [foil_ppl, cooling__foil_ppl]
    - [gas_cc, cooling__gas_cc]
    - [gas_cc_ccs, cooling__gas_cc_ccs]
    - [gas_hpl, cooling__gas_hpl]
    - [gas_ppl, cooling__gas_ppl]
    - [geo_hpl, cooling__geo_hpl]
    - [geo_ppl, cooling__geo_ppl]
    - [igcc, cooling__igcc]
    - [igcc_ccs, cooling__igcc_ccs]
    - [loil_cc, cooling__loil_cc]
    - [loil_ppl, cooling__loil_ppl]
    - [nuc_hc, cooling__nuc_hc]
    - [nuc_lc, cooling__nuc_lc]
    - [solar_th_ppl, cooling__solar_th_ppl]


  cat_addon:
  #   #TODO check whether these needs to be added ?
    add:
    - [cooling__bio_hpl, bio_hpl__ot_fresh]
    - [cooling__bio_hpl, bio_hpl__cl_fresh]
    - [cooling__bio_hpl, bio_hpl__ot_saline]
    - [cooling__bio_hpl, bio_hpl__air]
    - [cooling__bio_istig, bio_istig__ot_fresh]
    - [cooling__bio_istig, bio_istig__cl_fresh]
    - [cooling__bio_istig, bio_istig__ot_saline]
    - [cooling__bio_istig, bio_istig__air]
    - [cooling__bio_istig_ccs, bio_istig_ccs__ot_fresh]
    - [cooling__bio_istig_ccs, bio_istig_ccs__cl_fresh]
    - [cooling__bio_istig_ccs, bio_istig_ccs__ot_saline]
    - [cooling__bio_istig_ccs, bio_istig_ccs__air]
    - [cooling__coal_adv, coal_adv__ot_fresh]
    - [cooling__coal_adv, coal_adv__cl_fresh]
    - [cooling__coal_adv, coal_adv__ot_saline]
    - [cooling__coal_adv, coal_adv__air]
    - [cooling__coal_adv_ccs, coal_adv_ccs__ot_fresh]
    - [cooling__coal_adv_ccs, coal_adv_ccs__cl_fresh]
    - [cooling__coal_adv_ccs, coal_adv_ccs__ot_saline]
    - [cooling__coal_adv_ccs, coal_adv_ccs__air]
    - [cooling__bio_ppl, bio_ppl__ot_fresh]
    - [cooling__bio_ppl, bio_ppl__cl_fresh]
    - [cooling__bio_ppl, bio_ppl__ot_saline]
    - [cooling__bio_ppl, bio_ppl__air]
    - [cooling__coal_ppl, coal_ppl__ot_fresh]
    - [cooling__coal_ppl, coal_ppl__cl_fresh]
    - [cooling__coal_ppl, coal_ppl__ot_saline]
    - [cooling__coal_ppl, coal_ppl__air]
    - [cooling__coal_ppl_u, coal_ppl_u__ot_fresh]
    - [cooling__coal_ppl_u, coal_ppl_u__cl_fresh]
    - [cooling__coal_ppl_u, coal_ppl_u__ot_saline]
    - [cooling__coal_ppl_u, coal_ppl_u__air]
    - [cooling__foil_hpl, foil_hpl__ot_fresh]
    - [cooling__foil_hpl, foil_hpl__cl_fresh]
    - [cooling__foil_hpl, foil_hpl__ot_saline]
    - [cooling__foil_hpl, foil_hpl__air]
    - [cooling__foil_ppl, foil_ppl__ot_fresh]
    - [cooling__foil_ppl, foil_ppl__cl_fresh]
    - [cooling__foil_ppl, foil_ppl__ot_saline]
    - [cooling__foil_ppl, foil_ppl__air]
    - [cooling__gas_cc, gas_cc__ot_fresh]
    - [cooling__gas_cc, gas_cc__cl_fresh]
```

(continues on next page)

(continued from previous page)

```
    - [cooling__gas_cc, gas_cc__ot_saline]
    - [cooling__gas_cc, gas_cc__air]
    - [cooling__gas_cc_ccs, gas_cc_ccs__ot_fresh]
    - [cooling__gas_cc_ccs, gas_cc_ccs__cl_fresh]
    - [cooling__gas_cc_ccs, gas_cc_ccs__ot_saline]
    - [cooling__gas_cc_ccs, gas_cc_ccs__air]
    - [cooling__gas_hpl, gas_hpl__ot_fresh]
    - [cooling__gas_hpl, gas_hpl__cl_fresh]
    - [cooling__gas_hpl, gas_hpl__ot_saline]
    - [cooling__gas_hpl, gas_hpl__air]
    - [cooling__gas_ppl, gas_ppl__ot_fresh]
    - [cooling__gas_ppl, gas_ppl__cl_fresh]
    - [cooling__gas_ppl, gas_ppl__ot_saline]
    - [cooling__gas_ppl, gas_ppl__air]
    - [cooling__geo_hpl, gas_hpl__ot_fresh]
    - [cooling__geo_hpl, gas_hpl__cl_fresh]
    - [cooling__geo_hpl, gas_hpl__ot_saline]
    - [cooling__geo_hpl, gas_hpl__air]
    - [cooling__geo_ppl, geo_ppl__ot_fresh]
    - [cooling__geo_ppl, geo_ppl__cl_fresh]
    - [cooling__geo_ppl, geo_ppl__ot_saline]
    - [cooling__geo_ppl, geo_ppl__air]
    - [cooling__igcc, igcc__ot_fresh]
    - [cooling__igcc, igcc__cl_fresh]
    - [cooling__igcc, igcc__ot_saline]
    - [cooling__igcc, igcc__air]
    - [cooling__igcc_ccs, igcc_ccs__ot_fresh]
    - [cooling__igcc_ccs, igcc_ccs__cl_fresh]
    - [cooling__igcc_ccs, igcc_ccs__ot_saline]
    - [cooling__igcc_ccs, igcc_ccs__air]
    - [cooling__loil_cc, loil_cc__ot_fresh]
    - [cooling__loil_cc, loil_cc__cl_fresh]
    - [cooling__loil_cc, loil_cc__ot_saline]
    - [cooling__loil_cc, loil_cc__air]
    - [cooling__loil_ppl, loil_ppl__ot_fresh]
    - [cooling__loil_ppl, loil_ppl__cl_fresh]
    - [cooling__loil_ppl, loil_ppl__ot_saline]
    - [cooling__loil_ppl, loil_ppl__air]
    - [cooling__nuc_hc, nuc_hc__ot_fresh]
    - [cooling__nuc_hc, nuc_hc__cl_fresh]
    - [cooling__nuc_hc, nuc_hc__ot_saline]
    - [cooling__nuc_hc, nuc_hc__air]
    - [cooling__nuc_lc, nuc_lc__ot_fresh]
    - [cooling__nuc_lc, nuc_lc__cl_fresh]
    - [cooling__nuc_lc, nuc_lc__ot_saline]
    - [cooling__nuc_lc, nuc_lc__air]
    - [cooling__solar_th_ppl, solar_th_ppl__ot_fresh]
    - [cooling__solar_th_ppl, solar_th_ppl__cl_fresh]
    - [cooling__solar_th_ppl, solar_th_ppl__ot_saline]
    - [cooling__solar_th_ppl, solar_th_ppl__air]

  type_tec:
    add:
    - water_distribution
    - water_efficiency
    - wastewater_treatment
    - desalination
    - share_low_lim_GWat_total
    - share_low_lim_GWat_share
    - share_wat_recycle_total
    - share_wat_recycle_share
```

(continues on next page)

(continued from previous page)

```
cat_tec:
  add:
  - [share_low_lim_GWat_total, extract_surfacewater]
  - [share_low_lim_GWat_total, extract_groundwater]
  - [share_low_lim_GWat_share, extract_groundwater]
  - [share_wat_recycle_total,urban_recycle]
  - [share_wat_recycle_total,urban_discharge]
  - [share_wat_recycle_share,urban_recycle]
  - [water_distribution, urban_t_d]
  - [water_distribution, rural_t_d]
  - [water_distribution, urban_unconnected]
  - [water_distribution, rural_unconnected]
  - [water_distribution, urban_sewerage]
  - [wastewater_treatment, urban_sewerage]
  - [wastewater_treatment, urban_untreated]
  - [wastewater_treatment, urban_recycle]
  - [wastewater_treatment, rural_sewerage]
  - [wastewater_treatment, rural_untreated]
  - [water_efficiency, ueff1]
  - [water_efficiency, ueff2]
  - [water_efficiency, ueff3]
  - [water_efficiency, reff1]
  - [water_efficiency, reff2]
  - [water_efficiency, reff3]
  - [water_efficiency, ieff1]
  - [water_efficiency, ieff2]
  - [water_efficiency, ieff3]
  #- [water_resource_extraction, extract_upstream_landuse]
  - [water_resource_extraction, extract_surfacewater]
  #- [water_resource_extraction, extract_freshwater_instream]
  - [water_resource_extraction, extract_salinewater]
  - [water_resource_extraction, extract_groundwater]
  - [water_resource_extraction, extract_gw_fossil]
  - [desalination, membrane]
  - [desalination, distillation]

balance_equality:
  add:
  - [freshwater, water_supply]
  - [freshwater_basin, water_supply_basin]
  - [surfacewater_basin, water_avail_basin]
  - [groundwater_basin, water_avail_basin]
  - [salinewater_basin,water_avail_basin]
  - [urban_collected_wst, final]
  - [urban_collected_wst, water_treat]
  - [urban_uncollected_wst, final]
  - [industry_uncollected_wst,final]
  - [rural_collected_wst, final]
  - [rural_collected_wst, water_treat]
  - [rural_uncollected_wst, final]
  - [urban_mw, final]
  - [industry_mw,final]
  - [urban_disconnected, final]
  - [rural_mw, final]
  - [rural_disconnected, final]


shares:
  add:
  - share_basin
```

(continues on next page)

(continued from previous page)

```
    - share_low_lim_GWat
    - share_wat_recycle

  # relation:
  #   add:
    # - gw_share

  unit:
    add:
    - km3/GWa
    - km3
    - km3/year
    - USD/km3

  emission:
    add:
    - fresh_return

  type_emission:
    add:
    - water_consumption

  cat_emission:
    add:
    - [water_consumption, fresh_return]

cooling:
  commodity:
    require:
    - electr

    remove:
    # removing all technologies which are related to water in the previous model
    # Re added in the newer implementation with lesser commodities
    - saline_supply
    #- freshwater_instream
    - freshwater_supply
    - desalinated_water
    - saline_supply_ppl
    - urban_collected_wst
    - urban_uncollected_wst
    - yield_freshwater_suply
    - rural_collected_wst
    - rual_uncollected_wst
    - urban_mw
    - urban_dis
    - rural_mw
    - rural_dis
    - cooling__bio_hpl
    - cooling__bio_ppl
    - cooling__bio_istig
    - cooling__bio_ppl
    - cooling__coal_adv
    - cooling__coal_adv_ccs
    - cooling__coal_ppl
    - cooling__coal_ppl_u
    - cooling__foil_hpl
    - cooling__foil_ppl
    - cooling__gas_cc
    - cooling__gas_cc_ccs
    - cooling__gas_hpl
```

(continues on next page)

(continued from previous page)

```
    - cooling__gas_ppl
    - cooling__geo_hpl
    - cooling__geo_ppl
    - cooling__igcc
    - cooling__igcc_ccs
    - cooling__loil_cc
    - cooling__loil_ppl
    - cooling__nuc_hc
    - cooling__solar_th_ppl

    add:
    - electr
    - salinewater
    - freshwater_instream
    - freshwater
    - freshwater_basin
    - surfacewater_basin
    - groundwater_basin
    - desalinated_water
    - saline_ppl


  level:
    require:
    - secondary

    remove:
    - water_supply
    - cooling
    - desalination_supply
    - water_treat
    add:
    - water_supply
    - saline_supply

  mode:
    require:
    - all

  technology:
    remove:
    - extract__saline_supply
    - bio_hpl__ot_fresh
    - bio_hpl__cl_fresh
    - bio_hpl__cl_fresh
    - bio_hpl__ot_saline
    - bio_hpl__air
    - bio_istig__ot_fresh
    - bio_istig__cl_fresh
    - bio_istig__cl_fresh
    - bio_istig__ot_saline
    - bio_istig__air
    - bio_istig_ccs__ot_fresh
    - bio_istig_ccs__cl_fresh
    - bio_istig_ccs__cl_fresh
    - bio_istig_ccs__ot_saline
    - bio_istig_ccs__air
    - bio_ppl__ot_fresh
    - bio_ppl__cl_fresh
    - bio_ppl__cl_fresh
    - bio_ppl__ot_saline
```

(continues on next page)

(continued from previous page)

```
    - bio_ppl__air
    - coal_adv__ot_fresh
    - coal_adv__cl_fresh
    - coal_adv__cl_fresh
    - coal_adv__ot_saline
    - coal_adv__air
    - coal_adv_ccs__ot_fresh
    - coal_adv_ccs__cl_fresh
    - coal_adv_ccs__cl_fresh
    - coal_adv_ccs__ot_saline
    - coal_ppl__ot_fresh
    - coal_ppl__cl_fresh
    - coal_ppl__cl_fresh
    - coal_ppl__ot_saline
    - coal_ppl__air
    - coal_ppl_u__ot_fresh
    - coal_ppl_u__cl_fresh
    - coal_ppl_u__cl_fresh
    - coal_ppl_u__ot_saline
    - coal_ppl_u__air
    - foil_hpl__ot_fresh
    - foil_hpl__cl_fresh
    - foil_hpl__cl_fresh
    - foil_hpl__ot_saline
    - foil_hpl__air
    - foil_ppl__ot_fresh
    - foil_ppl__cl_fresh
    - foil_ppl__cl_fresh
    - foil_ppl__ot_saline
    - foil_ppl__air
    - gas_cc__ot_fresh
    - gas_cc__cl_fresh
    - gas_cc__cl_fresh
    - gas_cc__ot_saline
    - gas_cc__air
    - gas_cc_ccs__ot_fresh
    - gas_cc_ccs__cl_fresh
    - gas_cc_ccs__cl_fresh
    - gas_cc_ccs__ot_saline
    - gas_hpl__ot_fresh
    - gas_hpl__cl_fresh
    - gas_hpl__cl_fresh
    - gas_hpl__ot_saline
    - gas_hpl__air
    - gas_ppl__ot_fresh
    - gas_ppl__cl_fresh
    - gas_ppl__cl_fresh
    - gas_ppl__ot_saline
    - gas_ppl__air
    - geo_hpl__ot_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__ot_saline
    - geo_hpl__air
    - geo_ppl__ot_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__ot_saline
    - geo_ppl__air
    - igcc__ot_fresh
    - igcc__cl_fresh
```

(continues on next page)

(continued from previous page)

```
    - igcc__cl_fresh
    - igcc__ot_saline
    - igcc__air
    - igcc_ccs__ot_fresh
    - igcc_ccs__cl_fresh
    - igcc_ccs__cl_fresh
    - igcc_ccs__ot_saline
    - loil_cc__ot_fresh
    - loil_cc__cl_fresh
    - loil_cc__cl_fresh
    - loil_cc__ot_saline
    - loil_cc__air
    - loil_ppl__ot_fresh
    - loil_ppl__cl_fresh
    - loil_ppl__cl_fresh
    - loil_ppl__ot_saline
    - loil_ppl__air
    - nuc_hc__ot_fresh
    - nuc_hc__cl_fresh
    - nuc_hc__cl_fresh
    - nuc_hc__ot_saline
    - solar_th_ppl__ot_fresh
    - solar_th_ppl__cl_fresh
    - solar_th_ppl__cl_fresh
    - solar_th_ppl__ot_saline
    - solar_th_ppl__air
    - extract__upstream_landuse
    - extract__saline_supply
    - extract__freshwater_supply
    - extract__freshwater_instream

    #add:
    # Adds all technologies from technologies.yaml file again to reconfigure

  time:
    # NB cannot handle models with sub-annual time resolution
    require:
    - year

  # Addon techs include cooling technologies for now
  addon:
    add:
    - bio_hpl__ot_fresh
    - bio_hpl__cl_fresh
    - bio_hpl__ot_saline
    - bio_hpl__air
    - bio_istig__ot_fresh
    - bio_istig__cl_fresh
    - bio_istig__ot_saline
    - bio_istig__air
    - bio_istig_ccs__ot_fresh
    - bio_istig_ccs__cl_fresh
    - bio_istig_ccs__ot_saline
    - bio_istig_ccs__air
    - bio_ppl__ot_fresh
    - bio_ppl__cl_fresh
    - bio_ppl__ot_saline
    - bio_ppl__air
    - coal_adv__ot_fresh
    - coal_adv__cl_fresh
    - coal_adv__ot_saline
```

(continues on next page)

(continued from previous page)

```
    - coal_adv__air
    - coal_adv_ccs__ot_fresh
    - coal_adv_ccs__cl_fresh
    - coal_adv_ccs__ot_saline
    - coal_adv_ccs__air
    - coal_ppl__ot_fresh
    - coal_ppl__cl_fresh
    - coal_ppl__ot_saline
    - coal_ppl__air
    - coal_ppl_u__ot_fresh
    - coal_ppl_u__cl_fresh
    - coal_ppl_u__ot_saline
    - coal_ppl_u__air
    - foil_ppl__ot_fresh
    - foil_ppl__cl_fresh
    - foil_ppl__ot_saline
    - foil_ppl__air
    - foil_hpl__ot_fresh
    - foil_hpl__cl_fresh
    - foil_hpl__ot_saline
    - foil_hpl__air
    - gas_cc__ot_fresh
    - gas_cc__cl_fresh
    - gas_cc__ot_saline
    - gas_cc__air
    - gas_cc_ccs__ot_fresh
    - gas_cc_ccs__cl_fresh
    - gas_cc_ccs__ot_saline
    - gas_cc_ccs__air
    - gas_hpl__ot_fresh
    - gas_hpl__cl_fresh
    - gas_hpl__ot_saline
    - gas_hpl__air
    - gas_ppl__ot_fresh
    - gas_ppl__cl_fresh
    - gas_ppl__ot_saline
    - gas_ppl__air
    - geo_hpl__ot_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__cl_fresh
    - geo_hpl__ot_saline
    - geo_hpl__air
    - geo_ppl__ot_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__cl_fresh
    - geo_ppl__ot_saline
    - geo_ppl__air
    - igcc__ot_fresh
    - igcc__cl_fresh
    - igcc__ot_saline
    - igcc__air
    - igcc_ccs__ot_fresh
    - igcc_ccs__cl_fresh
    - igcc_ccs__ot_saline
    - igcc_ccs__air
    - loil_cc__ot_fresh
    - loil_cc__cl_fresh
    - loil_cc__ot_saline
    - loil_cc__air
    - loil_ppl__ot_fresh
    - loil_ppl__cl_fresh
```

(continues on next page)

(continued from previous page)

```
    - loil_ppl__ot_saline
    - loil_ppl__air
    - nuc_hc__ot_fresh
    - nuc_hc__cl_fresh
    - nuc_hc__ot_saline
    - nuc_hc__air
    - solar_th_ppl__ot_fresh
    - solar_th_ppl__cl_fresh
    - solar_th_ppl__ot_saline
    - solar_th_ppl__air

  # cat_addon includes cooling technology addons for parent technologies
  type_addon:
    add:
    - cooling__bio_hpl
    - cooling__bio_istig
    - cooling__bio_istig_ccs
    - cooling__bio_ppl
    - cooling__coal_adv
    - cooling__coal_adv_ccs
    - cooling__coal_ppl
    - cooling__coal_ppl_u
    - cooling__foil_hpl
    - cooling__foil_ppl
    - cooling__gas_cc
    - cooling__gas_cc_ccs
    - cooling__gas_hpl
    - cooling__gas_ppl
    - cooling__geo_hpl
    - cooling__geo_ppl
    - cooling__igcc
    - cooling__igcc_ccs
    - cooling__loil_cc
    - cooling__loil_ppl
    - cooling__nuc_hc
    - cooling__nuc_lc
    - cooling__solar_th_ppl

  map_tec_addon:
    add:
    - [bio_hpl,cooling__bio_hpl]
    - [bio_istig, cooling__bio_istig]
    - [bio_istig_ccs, cooling__bio_istig_ccs]
    - [bio_ppl,cooling__bio_ppl]
    - [coal_adv, cooling__coal_adv]
    - [coal_adv_ccs, cooling__coal_adv_ccs]
    - [coal_ppl, cooling__coal_ppl]
    - [coal_ppl_u, cooling__coal_ppl_u]
    - [foil_hpl, cooling__foil_hpl]
    - [foil_ppl, cooling__foil_ppl]
    - [gas_cc, cooling__gas_cc]
    - [gas_cc_ccs, cooling__gas_cc_ccs]
    - [gas_hpl, cooling__gas_hpl]
    - [gas_ppl, cooling__gas_ppl]
    - [geo_hpl, cooling__geo_hpl]
    - [geo_ppl, cooling__geo_ppl]
    - [igcc, cooling__igcc]
    - [igcc_ccs, cooling__igcc_ccs]
    - [loil_cc, cooling__loil_cc]
    - [loil_ppl, cooling__loil_ppl]
    - [nuc_hc, cooling__nuc_hc]
```

(continues on next page)

(continued from previous page)

```
  - [nuc_lc, cooling__nuc_lc]
  - [solar_th_ppl, cooling__solar_th_ppl]


cat_addon:
#   #TODO check whether these needs to be added ?
  add:
  - [cooling__bio_hpl, bio_hpl__ot_fresh]
  - [cooling__bio_hpl, bio_hpl__cl_fresh]
  - [cooling__bio_hpl, bio_hpl__ot_saline]
  - [cooling__bio_hpl, bio_hpl__air]
  - [cooling__bio_istig, bio_istig__ot_fresh]
  - [cooling__bio_istig, bio_istig__cl_fresh]
  - [cooling__bio_istig, bio_istig__ot_saline]
  - [cooling__bio_istig, bio_istig__air]
  - [cooling__bio_istig_ccs, bio_istig_ccs__ot_fresh]
  - [cooling__bio_istig_ccs, bio_istig_ccs__cl_fresh]
  - [cooling__bio_istig_ccs, bio_istig_ccs__ot_saline]
  - [cooling__bio_istig_ccs, bio_istig_ccs__air]
  - [cooling__coal_adv, coal_adv__ot_fresh]
  - [cooling__coal_adv, coal_adv__cl_fresh]
  - [cooling__coal_adv, coal_adv__ot_saline]
  - [cooling__coal_adv, coal_adv__air]
  - [cooling__coal_adv_ccs, coal_adv_ccs__ot_fresh]
  - [cooling__coal_adv_ccs, coal_adv_ccs__cl_fresh]
  - [cooling__coal_adv_ccs, coal_adv_ccs__ot_saline]
  - [cooling__coal_adv_ccs, coal_adv_ccs__air]
  - [cooling__bio_ppl, bio_ppl__ot_fresh]
  - [cooling__bio_ppl, bio_ppl__cl_fresh]
  - [cooling__bio_ppl, bio_ppl__ot_saline]
  - [cooling__bio_ppl, bio_ppl__air]
  - [cooling__coal_ppl, coal_ppl__ot_fresh]
  - [cooling__coal_ppl, coal_ppl__cl_fresh]
  - [cooling__coal_ppl, coal_ppl__ot_saline]
  - [cooling__coal_ppl, coal_ppl__air]
  - [cooling__coal_ppl_u, coal_ppl_u__ot_fresh]
  - [cooling__coal_ppl_u, coal_ppl_u__cl_fresh]
  - [cooling__coal_ppl_u, coal_ppl_u__ot_saline]
  - [cooling__coal_ppl_u, coal_ppl_u__air]
  - [cooling__foil_hpl, foil_hpl__ot_fresh]
  - [cooling__foil_hpl, foil_hpl__cl_fresh]
  - [cooling__foil_hpl, foil_hpl__ot_saline]
  - [cooling__foil_hpl, foil_hpl__air]
  - [cooling__foil_ppl, foil_ppl__ot_fresh]
  - [cooling__foil_ppl, foil_ppl__cl_fresh]
  - [cooling__foil_ppl, foil_ppl__ot_saline]
  - [cooling__foil_ppl, foil_ppl__air]
  - [cooling__gas_cc, gas_cc__ot_fresh]
  - [cooling__gas_cc, gas_cc__cl_fresh]
  - [cooling__gas_cc, gas_cc__ot_saline]
  - [cooling__gas_cc, gas_cc__air]
  - [cooling__gas_cc_ccs, gas_cc_ccs__ot_fresh]
  - [cooling__gas_cc_ccs, gas_cc_ccs__cl_fresh]
  - [cooling__gas_cc_ccs, gas_cc_ccs__ot_saline]
  - [cooling__gas_cc_ccs, gas_cc_ccs__air]
  - [cooling__gas_hpl, gas_hpl__ot_fresh]
  - [cooling__gas_hpl, gas_hpl__cl_fresh]
  - [cooling__gas_hpl, gas_hpl__ot_saline]
  - [cooling__gas_hpl, gas_hpl__air]
  - [cooling__gas_ppl, gas_ppl__ot_fresh]
  - [cooling__gas_ppl, gas_ppl__cl_fresh]
```

(continues on next page)

(continued from previous page)

```
  - [cooling__gas_ppl, gas_ppl__ot_saline]
  - [cooling__gas_ppl, gas_ppl__air]
  - [cooling__geo_hpl, gas_hpl__ot_fresh]
  - [cooling__geo_hpl, gas_hpl__cl_fresh]
  - [cooling__geo_hpl, gas_hpl__ot_saline]
  - [cooling__geo_hpl, gas_hpl__air]
  - [cooling__geo_ppl, geo_ppl__ot_fresh]
  - [cooling__geo_ppl, geo_ppl__cl_fresh]
  - [cooling__geo_ppl, geo_ppl__ot_saline]
  - [cooling__geo_ppl, geo_ppl__air]
  - [cooling__igcc, igcc__ot_fresh]
  - [cooling__igcc, igcc__cl_fresh]
  - [cooling__igcc, igcc__ot_saline]
  - [cooling__igcc, igcc__air]
  - [cooling__igcc_ccs, igcc_ccs__ot_fresh]
  - [cooling__igcc_ccs, igcc_ccs__cl_fresh]
  - [cooling__igcc_ccs, igcc_ccs__ot_saline]
  - [cooling__igcc_ccs, igcc_ccs__air]
  - [cooling__loil_cc, loil_cc__ot_fresh]
  - [cooling__loil_cc, loil_cc__cl_fresh]
  - [cooling__loil_cc, loil_cc__ot_saline]
  - [cooling__loil_cc, loil_cc__air]
  - [cooling__loil_ppl, loil_ppl__ot_fresh]
  - [cooling__loil_ppl, loil_ppl__cl_fresh]
  - [cooling__loil_ppl, loil_ppl__ot_saline]
  - [cooling__loil_ppl, loil_ppl__air]
  - [cooling__nuc_hc, nuc_hc__ot_fresh]
  - [cooling__nuc_hc, nuc_hc__cl_fresh]
  - [cooling__nuc_hc, nuc_hc__ot_saline]
  - [cooling__nuc_hc, nuc_hc__air]
  - [cooling__nuc_lc, nuc_lc__ot_fresh]
  - [cooling__nuc_lc, nuc_lc__cl_fresh]
  - [cooling__nuc_lc, nuc_lc__ot_saline]
  - [cooling__nuc_lc, nuc_lc__air]
  - [cooling__solar_th_ppl, solar_th_ppl__ot_fresh]
  - [cooling__solar_th_ppl, solar_th_ppl__cl_fresh]
  - [cooling__solar_th_ppl, solar_th_ppl__ot_saline]
  - [cooling__solar_th_ppl, solar_th_ppl__air]


unit:
  add:
  - km3/GWa
  - km3
  - km3/year

emission:
  add:
  - fresh_return

type_emission:
  add:
  - water_consumption

cat_emission:
  add:
  - [water_consumption, fresh_return]
```

Configuration (`water/config.yaml`)

```
# Configuration for MESSAGEix-Nexus

# Configuration file for the model
# Added data sources here.


# CSV files containing data for input calculations and assumptions
data files:
- cooltech_cost_and_shares_ssp_msg14
- tech_water_performance_ssp_msg
```

8.16 ADVANCE (`project.advance`)

Although free of charge, the ADVANCE data can not be downloaded automatically. This source requires that users first submit personal information to register before being able to retrieve the data. *message_ix_models* does not circumvent this requirement. Thus:

- A copy of the data are stored in `message_data`.
- *message_ix_models* contains only a 'fuzzed' version of the data (same structure, random values) for testing purposes.

class message_ix_models.project.advance.data.**ADVANCE**(*source*, *source_kw*)

Provider of exogenous data from the ADVANCE project database.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: "ADVANCE"
- *source_kw* including:
  - "model": one of 12 codes including "MESSAGE".
  - "measure": one of 3080 codes for the "VARIABLE" dimension.
  - "scenario": one of 51 codes including "ADV3TRAr2_Base".
  - "name", optional: override *ExoDataSource.name*.
  - "aggregate", optional: if `True`, aggregate data from the ADVANCE native regions using `n::groups` (same behaviour as the base class). Otherwise, do not aggregate.

Example

```
>>> keys = prepare_computer(
...     context,
...     computer,
...     source="ADVANCE",
...     source_kw=dict(
...         measure="Transport|Service demand|Road|Freight",
...         model="MESSAGE",
...         scenario="ADV3TRAr2_Base",
...     ),
... )
>>> result = computer.get(keys[0])
```

Load the metadata packaged with *message_ix_models* to identify usable *source_kw*:

```
>>> import sdmx
>>> from message_ix_models.util import package_data_path
>>>
>>> msg = sdmx.read_sdmx(package_data_path("sdmx", "ADVANCE.xml"))
>>> msg
<sdmx.StructureMessage>
<Header>
    prepared: '2023-11-03T21:51:46.052879'
    source: en: Generated by message_ix_models 2023.9.13.dev57+ga558c0b4.
↪d20231011
    test: False
Codelist (5): MODEL SCENARIO REGION VARIABLE UNIT
>>> msg.codelist["MODEL].items
{'AIM/CGE': <Code AIM/CGE>,
 'DNE21+': <Code DNE21+>,
 'GCAM': <Code GCAM>,
 'GEM-E3': <Code GEM-E3>,
 'IMACLIM V1.1': <Code IMACLIM V1.1>,
 'IMAGE': <Code IMAGE>,
 'MESSAGE': <Code MESSAGE>,
 'POLES ADVANCE': <Code POLES ADVANCE>,
 'REMIND': <Code REMIND>,
 'TIAM-UCL': <Code TIAM-UCL>,
 'WITCH': <Code WITCH>,
 'iPETS V.1.5': <Code iPETS V.1.5>}
```

id: **str** = **'ADVANCE'**

Identifier for this particular source.

transform(*c: Computer*, *base_key: Key*) → Key

Prepare *c* to transform raw data from *base_key*.

Unlike the base class version, this implementation only adds the aggregation step if *aggregate* is True.

8.17 Global Energy Assessment (project.gea)

Handle data from the Global Energy Assessment (GEA).

Although free of charge, the GEA data can not be downloaded automatically. This source requires that users first submit personal information to register before being able to retrieve the data. *message_ix_models* does not circumvent this requirement. Thus:

- A copy of the data are stored in message_data.
- *message_ix_models* contains only a 'fuzzed' version of the data (same structure, random values) for testing purposes.

class message_ix_models.project.gea.data.**GEA**(*source*, *source_kw*)

Provider of exogenous data from the GEA data source.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: "GEA".
- *source_kw* including:
  - *model*, *scenario*: model name and scenario name. See *get_model_scenario()*.
  - *measure*: See the source data for details.
  - *aggregate*, *interpolate*: see *ExoDataSource.transform()*.

aggregate: bool = False

By default, do not aggregate.

id: str = 'GEA'

Identifier for this particular source.

interpolate: bool = False

By default, do not interpolate.

transform(*c: Computer*, *base_key: Key*) → Key

Prepare *c* to transform raw data from *base_key*.

Compared to *ExoDataSource.transform()*, this version:

- Does not perform interpolation.

message_ix_models.project.gea.data.**get_model_scenario**() → Set[Tuple[str, str]]

Return a set of valid GEA (model name, scenario name) combinations.

These are read from `data/gea/model-scenario.json`.

8.18 SHAPE

"Sustainable development pathways achieving Human well-being while safeguarding the climate And Planet Earth"

The SHAPE project introduces the new term **Sustainable Development Pathway** or **SDP** for its scenarios, roughly analogous to SSPs. The SDPs simultaneously achieve the Sustainable Development Goals in 2030, maintain sustainable development thereafter, and meet the climate targets set out in the Paris Agreement.

Uses:

- m-data:/model/buildings
- m-data:/reference/model/material
- m-data:/reference/model/transport

8.18.1 Project information

- Website: http://shape-project.org
- Part of AXIS, an ERA-NET initiated by JPI Climate and funded by DLR/BMBF (DE, Grant No. 01LS1907A-B-C), Formas (SE), FFG/BMWFW (AT), NWO (NL) and RCN (NO) with co-funding by the European Union (Grant No. 776608).
- Project period: 2019-09 to 2022-08.

8.18.2 Data

`data/shape/` contains common data for constructing scenarios. These data quantify the SDP narratives (see link above) as future projections for certain quantities. Modeling tools[1] should use *`exo_data.prepare_computer()`* to configure *`SHAPE`* to retrieve these quantities and select an appropriate scenario's data for model setup calculations.

The files include:

`gdp_v*.mif`, `gdp_v*.xlsx`
Gross Domestic Product.

Generated by Bjoern Soergel at PIK, using stylized GDP/capita growth rate modifications to represent three narratives from the "Economic paradigm, growth, inequality and finance" section of the SHAPE description. The narratives are identified by codes for the IAMC "Scenario" dimension:

- "innovation": "Cornucopia (market-driven innovation)"
- "service": "Human Service Economy (managing the global commons)"
- "society": "A New Public Economy (resilient communities)"

The files have multiple **versions**:

- "v1p0" version 1.0.
- "v1p1" version 1.1.

The extension `.mif` apparently means "modeling interchange file"; this is a CSV file with semicolon (;) delimiters.

`population_v*.mif`, `population_v*.xlsx`
Population.

The values are the same across all SDPs; the single code "all_SHAPE_SDPs" is used for the "Scenario" dimension.

The values are identical to the SSP1 population projection.

Todo: This is may not be true, because the data differ in some (unspecified) way between version 1.0 and 1.1. Locate an accurate description for these data.

Note: If true, this would imply that the same population data could be retrieved with *`SSPOriginal`* and underlying files, also stored in the `message_data` repository; so the files `data/shape/population_*` are duplicates.

`gini_v*.csv`:
Gini coefficients. Generated by Jihoon Min.

- Normative Gini trajectories that will achieve SDG1 (No poverty) and SDG11 (Reduced inequalities) for **most** countries under the corresponding GDP projections (above), also in a stylized way.
- Same dimensions as the GDP data.
- v1.1 uses a slower rate of maximum Gini decrease to alleviate concerns about feasibility.

`urbanization.csv`
Generated by Alessio Mastrucci.

Three different codes are used for the "Scenario" dimension:

- "urb_distr"
- "urb_green"

[1] e.g. the IIASA standalone demand models as well as the MESSAGEix-GLOBIOM family of models.

- "urb_tech"

Todo: Describe what these values represent.

8.18.3 Code reference

Handle data from the SHAPE project.

message_ix_models.project.shape.data.**INFO = {'gdp': {'latest': '1.2', 'suffix': '.mif', 'variable': 'GDP|PPP'}, 'gini': {'drop': ['tgt.achieved', 'Base gini imputed', 'Share of final consumption among GDP imputed'], 'latest': '1.1', 'suffix': '.csv', 'variable': 'Gini'}, 'population': {'latest': '1.2', 'suffix': '.mif', 'variable': 'Population'}, 'urbanisation': {'drop': ['Notes'], 'latest': '1.0', 'suffix': '.csv', 'variable': 'Population|Urban|Share'}}**

Information about data file version, suffixes, "variable" codes, and extra columns to drop.

class message_ix_models.project.shape.data.**SHAPE**(*source*, *source_kw*)

Provider of exogenous data from the SHAPE project data source.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: "SHAPE".
- *source_kw* including:
  - *measure*: one of the keys of *INFO*.
  - *version* (optional): "latest" (default) or a version string like "1.2".
  - *scenario*: one of the SHAPE "SDP" scenario names.
  - *aggregate*, *interpolate*: see *ExoDataSource.transform()*.

id: str = 'SHAPE'

Identifier for this particular source.

message_ix_models.project.shape.data.**UNITS = {'%': '', 'NA': 'dimensionless', 'billion $2005/yr': 'GUSD_2005 / year'}**

Convert unit forms appearing in files to pint-compatible expressions.

8.19 Shared Socioeconomic Pathways (`project.ssp`)

8.19.1 Structure

The enumerations *SSP_2017* and *SSP_2024* contain one member from the corresponding SDMX code lists. These can be used to uniquely identify both an SSP narrative *and* the set in which it occurs, in applications where this distinction is meaningful:

```
>>> from message_ix_models.project.ssp import SSP_2017, SSP_2024
>>> x = SSP_2017["2"]
>>> y = SSP_2024["2"]
>>> str(y)
"ICONICS:SSP(2024).2"
>>> x == y
False
```

message_ix_models.project.ssp.**SSP**
alias of ICONICS:SSP(2017)

message_ix_models.project.ssp.**SSP_2017**
alias of ICONICS:SSP(2017)

message_ix_models.project.ssp.**SSP_2024**
alias of ICONICS:SSP(2024)

message_ix_models.project.ssp.**generate**(*context:* Context, *base_dir: PathLike | None = None*)
Generate SDMX code lists containing the SSPs.

message_ix_models.project.ssp.**parse**(*value: str | ~message_ix_models.util.sdmx.ICONICS:SSP(2017) | ~message_ix_models.util.sdmx.ICONICS:SSP(2024)) -> ~message_ix_models.util.sdmx.ICONICS:SSP(2017) | ~message_ix_models.util.sdmx.ICONICS:SSP(2024)*)
Parse *value* to a member of *SSP_2017* or *SSP_2024*.

class message_ix_models.project.ssp.**ssp_field**(*default: ~message_ix_models.util.sdmx.ICONICS:SSP(2017) | ~message_ix_models.util.sdmx.ICONICS:SSP(2024)*)
SSP field for use in data classes.

8.19.2 Data

Although free of charge, neither the 2017 or 2024 SSP data can be downloaded automatically. Both sources require that users first submit personal information to register before being able to retrieve the data. *message_ix_models* does not circumvent this requirement. Thus:

- A copy of the data are stored in message_data.
- *message_ix_models* contains only a 'fuzzed' version of the data (same structure, random values) for testing purposes.

Todo: Allow users without access to message_data to read a local copy of this data from a *Config.local_data* subdirectory.

SSPOriginal(source, source_kw)	Provider of exogenous data from the original SSP database.
SSPUpdate(source, source_kw)	Provider of exogenous data from the SSP Update database.

class message_ix_models.project.ssp.data.**SSPOriginal**(*source*, *source_kw*)
Provider of exogenous data from the original SSP database.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: Any value from SSP_2017 or equivalent string, for instance "ICONICS:SSP(2017).2". The specific SSP for which data is returned is determined from the value.
- *source_kw* including:
  - "model": one of:
    - IIASA GDP
    - IIASA-WiC POP

    - NCAR
    - OECD Env-Growth
    - PIK GDP-32
  - "measure": The measures available differ according to the model; see the source data for details.

Example

```
>>> keys = prepare_computer(
...     context,
...     computer,
...     source="ICONICS:SSP(2015).3",
...     source_kw=dict(measure="POP", model="IIASA-WiC POP"),
... )
>>> result = computer.get(keys[0])
```

filename = 'SspDb_country_data_2013-06-12.csv.zip'

Name of file containing the data.

id: str = 'SSP'

Identifier for this particular source.

model_date = {'IIASA GDP': '130219', 'IIASA-WiC POP': '130115', 'NCAR': '130115', 'OECD Env-Growth': '130325', 'PIK GDP-32': '130424'}

One-to-one correspondence between "model" codes and date fragments in scenario codes.

replace = {'billion US$2005/yr': 'billion USD_2005/yr'}

Replacements to apply when loading the data.

class message_ix_models.project.ssp.data.**SSPUpdate**(*source*, *source_kw*)

Provider of exogenous data from the SSP Update database.

To use data from this source, call *exo_data.prepare_computer()* with the arguments:

- *source*: Any value from `SSP_2024` or equivalent string, for instance "ICONICS:SSP(2024).2".
- *release*: One of "3.0.1", "3.0", or "preview".

Example

```
>>> keys = prepare_computer(
...     context,
...     computer,
...     source="ICONICS:SSP(2024).3",
...     source_kw=dict(measure="GDP", model="IIASA GDP 2023"),
... )
>>> result = computer.get(keys[0])
```

filename = {'3.0': '1706548837040-ssp_basic_drivers_release_3.0_full.csv.gz', '3.0.1': '1710759470883-ssp_basic_drivers_release_3.0.1_full.csv.gz', 'preview': 'SSP-Review-Phase-1.csv.gz'}

File names containing the data, according to the release.

id: str = 'SSP update'

Identifier for this particular source.

8.20 Node code lists

The codes in these lists denote **regions** and **countries**.

When loaded using *get_codes()*, the `Code.child` attribute is a list of child codes. See the function documentation for how to retrieve these.

See also:

adapt_R11_R12, *adapt_R11_R14*, *identify_nodes()*.

8.20.1 Models with global scope

32-region aggregation (R32)

```yaml
World:
  name: World
  description: |-
    Region code list for the SSP 32-region aggregation.

    Source: https://tntcat.iiasa.ac.at/SspDb/dsd?Action=htmlpage&page=about

R32ANUZ:
  parent: World
  name: Australia & New Zealand
  description: This region includes Australia and New Zealand.
  child: [AUS, NZL]

R32BRA:
  parent: World
  name: Brazil
  child: [BRA]

R32CAN:
  parent: World
  name: Canada
  child: [CAN]
```

(continues on next page)

(continued from previous page)

```
R32CAS:
  parent: World
  name: Central Asia
  description: This region includes the countries of Central Asia.
  child: [ARM, AZE, GEO, KAZ, KGZ, TJK, TKM, UZB]

R32CHN:
  parent: World
  name: China excl. Taiwan
  description: China (Mainland, Hongkong, Macao; excl. Taiwan).
  child: [CHN, HKG, MAC]

R32EEU:
  parent: World
  name: Eastern Europe
  description: |-
    Eastern Europe (excl. former Soviet Union and EU member states).

    The source page describes “the former Yugoslav Republic of Macedonia”; this␣
↪entity was renamed in 2018 to “North Macedonia”.
  child: [ALB, BIH, HRV, MKD, MNE, SRB]

R32EEU-FSU:
  parent: World
  name: Former Soviet Union in Eastern Europe
  description: Eastern Europe, former Soviet Union (excl. Russia and EU members).
  child: [BLR, MDA, UKR]

R32EFTA:
  parent: World
  name: European Free Trade Association
  description: |-
    This region includes Iceland, Norway, Switzerland.

    The source omits Liechtenstein, but it is included as a child.
  child: [ISL, LIE, NOR, CHE]

R32EU12-H:
  parent: World
  name: EU member states new in 2004, high income
  description: New EU member states that joined as of 2004 - high income.
  child: [CYP, CZE, EST, HUN, MLT, POL, SVK, SVN]

R32EU12-M:
  parent: World
  name: EU member states new in 2004, middle income
  description: New EU member states that joined as of 2004 - medium income.
  child: [BGR, LTU, LVA, ROU]

R32EU15:
  parent: World
  name: EU member states pre-2004
  description: This region includes European Union member states that joined prior␣
↪to 2004.
  child: [AUT, BEL, DEU, DNK, ESP, FIN, FRA, GBR, GRC, IRL, ITA, LUX, NLD, PRT,␣
↪SWE]

R32IDN:
  parent: World
  name: Indonesia
  child: [IDN]
```

(continues on next page)

(continued from previous page)

```
R32IND:
  parent: World
  name: India
  child: [IND]

R32JPN:
  parent: World
  name: Japan
  child: [JPN]

R32KOR:
  parent: World
  name: Republic of Korea
  child: [KOR]

R32LAM-L:
  parent: World
  name: Latin America, low income
  description: This region includes the countries of Latin America (excl. Brazil,␣
↪Mexico) - low income.
  child: [BLZ, GTM, HND, HTI, NIC]

R32LAM-M:
  parent: World
  name: Latin America, middle & high income
  description: |-
    This region includes the countries of Latin America (excl. Brazil, Mexico) -␣
↪medium and high income.

    The source includes “Netherlands Antilles” which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT), but is not a country. It was dissolved in 2010 into BES, CUW␣
↪and SXM, also included.
  child: [ABW, AIA, ANT, BES, BHS, BMU, BOL, BRB, CHL, COL, CRI, CUB, CUW, DMA,␣
↪DOM, ECU, GLP, GRD, GUF, GUY, JAM, KNA, LCA, PAN, PER, PRY, MTQ, SLV, SUR, SXM,␣
↪TTO, URY, VCT, VEN]

R32MEA-H:
  parent: World
  name: Middle East & Asia, high income
  description: This region includes the countries of Middle East Asia - high␣
↪income.
  child: [ARE, BHR, ISR, KWT, OMN, QAT, SAU]

R32MEA-M:
  parent: World
  name: Middle East & Asia, low & middle income
  description: This region includes the countries of Middle East Asia - low and␣
↪medium income.
  child: [IRN, IRQ, JOR, LBN, PSE, SYR, YEM]

R32MEX:
  parent: World
  name: Mexico
  child: [MEX]

R32NAF:
  parent: World
  name: North Africa
  description: This region includes the countries of North Africa.
  child: [DZA, EGY, ESH, LBY, MAR, TUN]
```

(continues on next page)

(continued from previous page)

```
R32OAS-CPA:
  parent: World
  name: Other Asia
  description: This region includes the countries of Other Asia - former Centrally␣
↪Planned Asia.
  child: [KHM, LAO, MNG, VNM]

R32OAS-L:
  parent: World
  name: Other Asia, low income
  description: This region includes the countries of Other Asia - low income.
  child: [BGD, FJI, FSM, MMR, NPL, PHL, PNG, PRK, SLB, TLS, TON, VUT, WSM]

R32OAS-M:
  parent: World
  name: Other Asia, middle & high income
  description: This region includes the countries of Other Asia - medium and high␣
↪income.
  child: [BRN, BTN, GUM, LKA, MDV, MYS, NCL, PYF, SGP, THA]

R32PAK:
  parent: World
  name: Pakistan & Afghanistan
  description: This region includes Pakistan and Afghanistan.
  child: [AFG, PAK]

R32RUS:
  parent: World
  name: Russian Federation
  child: [RUS]

R32SAF:
  parent: World
  name: South Africa
  child: [ZAF]

R32SSA-L:
  parent: World
  name: Sub-Saharan Africa, low income
  description: This region includes the countries of Subsahara Africa (excl. South␣
↪Africa) - low income.
  child: [BDI, BEN, BFA, CAF, CIV, CMR, COD, COG, COM, CPV, DJI, ERI, ETH, GHA,␣
↪GIN, GMB, GNB, KEN, LBR, LSO, MDG, MLI, MOZ, MRT, MWI, NER, NGA, RWA, SDN, SEN,␣
↪SLE, SOM, SSD, STP, SWZ, TCD, TGO, TZA, UGA, ZMB, ZWE]

R32SSA-M:
  parent: World
  name: Sub-Saharan Africa, middle & high income
  description: This region includes the countries of Subsahara Africa (excl. South␣
↪Africa) - medium and high income.
  child: [AGO, BWA, GAB, GNQ, MUS, MYT, NAM, REU, SYC]

R32TUR:
  parent: World
  name: Turkey
  child: [TUR]

R32TWN:
  parent: World
  name: Taiwan
```

(continues on next page)

(continued from previous page)

```
  child: [TWN]

R32USA:
  parent: World
  name: United States of America
  description: United States of America.
  child: [PRI, USA, VIR]
```

20-region aggregation (R20)

```
# Region code list
#
# - See message_data.tools.regions.
# - The ISO 3166-1 alpha-3 codes are not defined in this file, but loaded from
#   a copy of the ISO database, e.g. in pycountry.
# - Among others, there are no assignments for:
#   - ATA Antarctica
#   - IOT British Indian Ocean Territory
#   - SGS South Georgia

World:
  name: World
  description: R20 regions

R20_AFR:
  parent: World
  name: Sub-Saharan Africa
  child: [AGO, BDI, BEN, BFA, BWA, CAF, CIV, CMR, COD, COG, COM, CPV, DJI, ERI,␣
↪ETH, GAB, GHA, GIN, GMB, GNB, GNQ, KEN, LBR, LSO, MDG, MLI, MOZ, MRT, MUS, MWI,␣
↪MYT, NAM, NER, NGA, REU, RWA, SEN, SHN, SLE, SOM, STP, SWZ, SYC, TCD, TGO, TZA,␣
↪UGA, ZAF, ZMB, ZWE]
  iea-weo-region: Africa

R20_CHN:
  parent: World
  name: China
  child: [CHN, HKG]
  iea-weo-region: China

R20_PRK:
  parent: World
  name: North Korea
  child: [PRK]
  iea-weo-region: Russia

R20_MNG:
  parent: World
  name: Mongolia
  child: [MNG]
  iea-weo-region: Russia

R20_MSA:
  parent: World
  name: Mainland Southeast Asia
  child: [KHM, LAO, VNM]
  iea-weo-region: India

R20_JPN:
  parent: World
```

(continues on next page)

(continued from previous page)

```
  name: Japan
  child: [JPN]
  iea-weo-region: Japan

R20_AUNZ:
  parent: World
  name: Australia and New Zealand
  child: [AUS, NZL]
  iea-weo-region: Japan

R20_KOR:
  parent: World
  name: South Korea
  child: [KOR]
  iea-weo-region: China

R20_SEA:
  parent: World
  name: South East Asia
  description: >-
    Trust Territory of the Pacific Islands (PCI) still included in this list,
    but it was dissolved into MHL, FSM, MNP and PLW in 1986.
  child: [ASM, BRN, CCK, COK, CXR, FJI, FSM, IDN, KIR, MAC, MHL, MMR, MNP, MYS,␣
↪NCL, NFK, NIU, NRU, PCI, PCN, PHL, PLW, PNG, PYF, SGP, SLB, THA, TKL, TLS, TON,␣
↪TUV, TWN, VUT, WLF, WSM]
  iea-weo-region: India

R20_RUBY:
  parent: World
  name: Russia and Belarus
  child: [RUS, BLR]
  iea-weo-region: Russia

R20_UMBA:
  parent: World
  name: Ukraine, Moldova and Balkans
  description: >-
    Serbia and Montenegro (SCG) and Yugoslavia (YUG) still included in this list,
    even though their ISO 3166-1 codes were deleted in 2006 and 2003, respectively.
  child: [ALB, BGR, BIH, MDA, MKD, MNE, SCG, SRB, UKR, YUG, XKO]
  iea-weo-region: Russia

R20_CAS:
  parent: World
  name: Centra Asia
  child: [ KAZ, KGZ, TJK, TKM, UZB]
  iea-weo-region: Russia

R20_SCST:
  parent: World
  name: South Caucasus and Turkey
  child: [ARM, AZE,  GEO,  TUR]
  iea-weo-region: European Union

R20_EEU27:
  parent: World
  name: Central and Eastern Europe (EU27)
  child: [CZE, EST, HRV, HUN, LTU, LVA, POL, ROU, SVK, SVN]
  iea-weo-region: European Union

R20_LAM:
```

(continues on next page)

(continued from previous page)

```
  parent: World
  name: Latin America and The Caribbean
  description: >-
    The source includes “Netherlands Antilles” which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT),
    but is not a country. It was dissolved in 2010 into BES, CUW and SXM, also␣
↪included.
  child: [ABW, AIA, ANT, ARG, ATG, BES, BHS, BLZ, BMU, BOL, BRA, BRB, CHL, COL,␣
↪CRI, CUB, CUW, CYM, DMA, DOM, ECU, FLK, GLP, GRD, GTM, GUF, GUY, HND, HTI, JAM,␣
↪KNA, LCA, MEX, MSR, MTQ, NIC, PAN, PER, PRY, SLV, SUR, SXM, TCA, TTO, URY, VCT,␣
↪VEN, VGB]
  iea-weo-region: Brazil

R20_MEA:
  parent: World
  name: Middle East and North Africa
  child: [ARE, BHR, DZA, EGY, ESH, IRN, IRQ, ISR, JOR, KWT, LBN, LBY, MAR, OMN,␣
↪PSE, QAT, SAU, SDN, SSD, SYR, TUN, YEM]
  iea-weo-region: Middle East

R20_NAM:
  parent: World
  name: North America
  child: [CAN, GUM, PRI, SPM, USA, VIR]
  iea-weo-region: United States

R20_SAS:
  parent: World
  name: South Asia
  child: [AFG, BGD, BTN, IND, LKA, MDV, NPL, PAK]
  iea-weo-region: India

R20_WEU27:
  parent: World
  name: Western Europe (EU27)
  child: [AND, AUT, BEL, CYP, DEU, DNK, ESP, FIN, FRA, FRO, GIB, GRC, GRL, IMN,␣
↪IRL, ITA, LUX, MCO, MLT, NLD, PRT, SJM, SMR, SWE, VAT]
  iea-weo-region: European Union

R20_UKEFT:
  parent: World
  name: UK and European Free Trade Association
  child: [GBR, ISL, CHE, NOR, LIE]
  iea-weo-region: European Union
```

17-region aggregation (R17)

```
# Region code list
#
# - See message_data.tools.regions.
# - The ISO 3166-1 alpha-3 codes are not defined in this file, but loaded from
#   a copy of the ISO database, e.g. in pycountry.
# - Among others, there are no assignments for:
#   - ATA Antarctica
#   - IOT British Indian Ocean Territory
#   - SGS South Georgia

World:
  name: World
```

(continues on next page)

(continued from previous page)

```
  description: R20 regions

R17_AFR:
  parent: World
  name: Sub-Saharan Africa
  child: [AGO, BDI, BEN, BFA, BWA, CAF, CIV, CMR, COD, COG, COM, CPV, DJI, ERI,␣
↪ETH, GAB, GHA, GIN, GMB, GNB, GNQ, KEN, LBR, LSO, MDG, MLI, MOZ, MRT, MUS, MWI,␣
↪MYT, NAM, NER, NGA, REU, RWA, SEN, SHN, SLE, SOM, STP, SWZ, SYC, TCD, TGO, TZA,␣
↪UGA, ZAF, ZMB, ZWE]

R17_CHN:
  parent: World
  name: China
  child: [CHN, HKG]

R17_PRK:
  parent: World
  name: North Korea
  child: [PRK]

R17_MNG:
  parent: World
  name: Mangolia
  child: [MNG]

R17_RCPA:
  parent: World
  name: East Asia
  child: [KHM, LAO, VNM]

R17_JAP:
  parent: World
  name: Japan
  child: [JPN]

R17_RPAO:
  parent: World
  name: Pacific OECD
  child: [AUS, NZL]

R17_KOR:
  parent: World
  name: South Korea
  child: [KOR]

R17_RPAS:
  parent: World
  name: Pacific Asia
  description: >-
    Trust Territory of the Pacific Islands (PCI) still included in this list,
    but it was dissolved into MHL, FSM, MNP and PLW in 1986.
  child: [ASM, BRN, CCK, COK, CXR, FJI, FSM, IDN, KIR, MAC, MHL, MMR, MNP, MYS,␣
↪NCL, NFK, NIU, NRU, PCI, PCN, PHL, PLW, PNG, PYF, SGP, SLB, THA, TKL, TLS, TON,␣
↪TUV, TWN, VUT, WLF, WSM]

R17_RFSU:
  parent: World
  name: Former Soviet Union
  child: [ARM, AZE, BLR, GEO, KAZ, KGZ, MDA, TJK, TKM, UKR, UZB]

R17_RUS:
```

(continues on next page)

(continued from previous page)

```
  parent: World
  name: Russia
  child: [RUS]

R17_LAM:
  parent: World
  name: Latin America and The Caribbean
  description: >-
   The source includes “Netherlands Antilles” which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT),
    but is not a country. It was dissolved in 2010 into BES, CUW and SXM, also␣
↪included.
  child: [ABW, AIA, ANT, ARG, ATG, BES, BHS, BLZ, BMU, BOL, BRA, BRB, CHL, COL,␣
↪CRI, CUB, CUW, CYM, DMA, DOM, ECU, FLK, GLP, GRD, GTM, GUF, GUY, HND, HTI, JAM,␣
↪KNA, LCA, MEX, MSR, MTQ, NIC, PAN, PER, PRY, SLV, SUR, SXM, TCA, TTO, URY, VCT,␣
↪VEN, VGB]

R17_MEA:
  parent: World
  name: Middle East and North Africa
  child: [ARE, BHR, DZA, EGY, ESH, IRN, IRQ, ISR, JOR, KWT, LBN, LBY, MAR, OMN,␣
↪PSE, QAT, SAU, SDN, SSD, SYR, TUN, YEM]

R17_NAM:
  parent: World
  name: North America
  child: [CAN, GUM, PRI, SPM, USA, VIR]

R17_SAS:
  parent: World
  name: South Asia
  child: [AFG, BGD, BTN, IND, LKA, MDV, NPL, PAK]

R17_WEU:
  parent: World
  name: Western Europe
  child: [AND, AUT, BEL, CHE, CYP, DEU, DNK, ESP, FIN, FRA, FRO, GBR, GIB, GRC,␣
↪GRL, IMN, IRL, ISL, ITA, LIE, LUX, MCO, MLT, NLD, NOR, PRT, SJM, SMR, SWE, TUR,␣
↪VAT]

R17_EEU:
  parent: World
  name: Central and Eastern Europe
  description: >-
   Serbia and Montenegro (SCG) and Yugoslavia (YUG) still included in this list,
    even though their ISO 3166-1 codes were deleted in 2006 and 2003, respectively.
  child: [ALB, BGR, BIH, CZE, EST, HRV, HUN, LTU, LVA, MKD, MNE, POL, ROU, SCG,␣
↪SRB, SVK, SVN, YUG]
```

14-region aggregation (R14)

```
# Region code list
#
# - See message_data.tools.regions.
# - The ISO 3166-1 alpha-3 codes are not defined in this file, but loaded from
#   a copy of the ISO database, e.g. in pycountry.
# - Among others, there are no assignments for:
#   - ATA Antarctica
#   - IOT British Indian Ocean Territory
```

(continues on next page)

(continued from previous page)

```
#   - SGS South Georgia

World:
  name: World
  description: R14 regions

R14_AFR:
  parent: World
  name: Sub-Saharan Africa
  child: [AGO, BDI, BEN, BFA, BWA, CAF, CIV, CMR, COD, COG, COM, CPV, DJI, ERI,␣
↪ETH, GAB, GHA, GIN, GMB, GNB, GNQ, KEN, LBR, LSO, MDG, MLI, MOZ, MRT, MUS, MWI,␣
↪MYT, NAM, NER, NGA, REU, RWA, SDN, SEN, SHN, SLE, SOM, STP, SWZ, SYC, TCD, TGO,␣
↪TZA, UGA, ZAF, ZMB, ZWE]

R14_CAS:
  parent: World
  name: Central Asia
  child: [KAZ, KGZ, TJK, TKM, UZB]

R14_CPA:
  parent: World
  name: Centrally Planned Asia
  child: [CHN, KHM, LAO, MAC, MNG, PRK, TWN, VNM]

R14_EEU:
  parent: World
  name: Central and Eastern Europe
  description: >-
    Serbia and Montenegro (SCG) and Yugoslavia (YUG) still included in this list,
    even though their ISO 3166-1 codes were deleted in 2006 and 2003, respectively.
  child: [ALB, BGR, BIH, CZE, EST, HRV, HUN, LTU, LVA, MKD, MNE, POL, ROU, SCG,␣
↪SRB, SVK, SVN, YUG]

R14_LAM:
  parent: World
  name: Latin America and The Caribbean
  description: >-
    The source includes “Netherlands Antilles” which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT),
    but is not a country. It was dissolved in 2010 into BES, CUW and SXM, also␣
↪included.
  child: [ABW, AIA, ANT, ARG, ATG, BES, BHS, BLZ, BMU, BOL, BRA, BRB, CHL, COL,␣
↪CRI, CUB, CUW, CYM, DMA, DOM, ECU, FLK, GLP, GRD, GTM, GUF, GUY, HND, HTI, JAM,␣
↪LCA, MEX, MSR, MTQ, NIC, PAN, PER, PRI, PRY, SLV, SUR, SXM, TCA, TTO, URY, VCT,␣
↪VEN, VGB]

R14_MEA:
  parent: World
  name: Middle East and North Africa
  child: [ARE, BHR, DZA, EGY, ESH, IRN, IRQ, ISR, JOR, KWT, LBN, LBY, MAR, OMN,␣
↪PSE, QAT, SAU, SDN, SSD, SYR, TUN, YEM]

R14_NAM:
  parent: World
  name: North America
  child: [CAN, GUM, SPM, USA]

R14_PAO:
  parent: World
  name: Pacific OECD
  child: [AUS, JPN, NZL]
```

(continues on next page)

(continued from previous page)

```
R14_PAS:
  parent: World
  name: Other Pacific Asia
  child: [ASM, BRN, CCK, COK, CXR, FJI, FSM, IDN, KIR, KOR, MHL, MMR, MNP, MYS,␣
↪NCL, NFK, NIU, NRU, PCN, PHL, PLW, PNG, PYF, SGP, SLB, THA, TKL, TLS, TON, TUV,␣
↪VUT, WLF, WSM]

R14_RUS:
  parent: World
  name: Russia
  child: [RUS]

R14_SAS:
  parent: World
  name: South Asia
  child: [AFG, BGD, BTN, IND, LKA, MDV, NPL, PAK]

R14_SCS:
  parent: World
  name: Caspian States
  child: [ARM, AZE, GEO]

R14_UBM:
  parent: World
  name: Ukraine, Belarus, and Moldova
  child: [BLR, MDA, UKR]

R14_WEU:
  parent: World
  name: Western Europe
  child: [AND, AUT, BEL, CHE, CYP, DEU, DNK, ESP, FIN, FRA, FRO, GBR, GIB, GRC,␣
↪GRL, IMN, IRL, ISL, ITA, LIE, LUX, MCO, MLT, NLD, NOR, PRT, SJM, SMR, SWE, TUR,␣
↪VAT]
```

11-region aggregation (R11)

```
# Region code list
#
# - See message_data.tools.regions.
# - The ISO 3166-1 alpha-3 codes are not defined in this file, but loaded from
#   a copy of the ISO database, e.g. in pycountry.
# - Among others, there are no assignments for:
#   - ATA Antarctica
#   - IOT British Indian Ocean Territory
#   - SGS South Georgia

World:
  name: World
  description: R11 regions

R11_AFR:
  parent: World
  name: Sub-Saharan Africa
  child: [AGO, BDI, BEN, BFA, BWA, CAF, CIV, CMR, COD, COG, COM, CPV, DJI, ERI,␣
↪ETH, GAB, GHA, GIN, GMB, GNB, GNQ, KEN, LBR, LSO, MDG, MLI, MOZ, MRT, MUS, MWI,␣
↪MYT, NAM, NER, NGA, REU, RWA, SEN, SHN, SLE, SOM, STP, SWZ, SYC, TCD, TGO, TZA,␣
↪UGA, ZAF, ZMB, ZWE]
  iea-weo-region: Africa
```

(continues on next page)

(continued from previous page)

```
R11_CPA:
  parent: World
  name: Centrally Planned Asia
  child: [CHN, HKG, KHM, LAO, MNG, PRK, VNM]
  iea-weo-region: China

R11_EEU:
  parent: World
  name: Central and Eastern Europe
  description: >-
    Serbia and Montenegro (SCG) and Yugoslavia (YUG) still included in this list,
    even though their ISO 3166-1 codes were deleted in 2006 and 2003, respectively.
  child: [ALB, BGR, BIH, CZE, EST, HRV, HUN, LTU, LVA, MKD, MNE, POL, ROU, SCG,␣
↪SRB, SVK, SVN, YUG]
  iea-weo-region: European Union

R11_FSU:
  parent: World
  name: Former Soviet Union
  child: [ARM, AZE, BLR, GEO, KAZ, KGZ, MDA, RUS, TJK, TKM, UKR, UZB]
  iea-weo-region: Russia

R11_LAM:
  parent: World
  name: Latin America and The Caribbean
  description: >-
    The source includes "Netherlands Antilles" which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT),
    but is not a country. It was dissolved in 2010 into BES, CUW and SXM, also␣
↪included.
  child: [ABW, AIA, ANT, ARG, ATG, BES, BHS, BLZ, BMU, BOL, BRA, BRB, CHL, COL,␣
↪CRI, CUB, CUW, CYM, DMA, DOM, ECU, FLK, GLP, GRD, GTM, GUF, GUY, HND, HTI, JAM,␣
↪KNA, LCA, MEX, MSR, MTQ, NIC, PAN, PER, PRY, SLV, SUR, SXM, TCA, TTO, URY, VCT,␣
↪VEN, VGB]
  iea-weo-region: Brazil

R11_MEA:
  parent: World
  name: Middle East and North Africa
  child: [ARE, BHR, DZA, EGY, ESH, IRN, IRQ, ISR, JOR, KWT, LBN, LBY, MAR, OMN,␣
↪PSE, QAT, SAU, SDN, SSD, SYR, TUN, YEM]
  iea-weo-region: Middle East

R11_NAM:
  parent: World
  name: North America
  child: [CAN, GUM, PRI, SPM, USA, VIR]
  iea-weo-region: United States

R11_PAO:
  parent: World
  name: Pacific OECD
  child: [AUS, JPN, NZL]
  iea-weo-region: Japan

R11_PAS:
  parent: World
  name: Other Pacific Asia
  description: >-
    Trust Territory of the Pacific Islands (PCI) still included in this list,
```

(continues on next page)

(continued from previous page)

```
    but it was dissolved into MHL, FSM, MNP and PLW in 1986.
  child: [ASM, BRN, CCK, COK, CXR, FJI, FSM, IDN, KIR, KOR, MAC, MHL, MMR, MNP,␣
↪MYS, NCL, NFK, NIU, NRU, PCI, PCN, PHL, PLW, PNG, PYF, SGP, SLB, THA, TKL, TLS,␣
↪TON, TUV, TWN, VUT, WLF, WSM]
  iea-weo-region: India

R11_SAS:
  parent: World
  name: South Asia
  child: [AFG, BGD, BTN, IND, LKA, MDV, NPL, PAK]
  iea-weo-region: India

R11_WEU:
  parent: World
  name: Western Europe
  child: [AND, AUT, BEL, CHE, CYP, DEU, DNK, ESP, FIN, FRA, FRO, GBR, GIB, GRC,␣
↪GRL, IMN, IRL, ISL, ITA, LIE, LUX, MCO, MLT, NLD, NOR, PRT, SJM, SMR, SWE, TUR,␣
↪VAT]
  iea-weo-region: European Union
```

12-region aggregation (R12)

```
# Region code list
#
# - See message_data.tools.regions.
# - The ISO 3166-1 alpha-3 codes are not defined in this file, but loaded from
#   a copy of the ISO database, e.g. in pycountry.
# - Among others, there are no assignments for:
#   - ATA Antarctica
#   - IOT British Indian Ocean Territory
#   - SGS South Georgia

World:
  name: World
  description: R12 regions

R12_AFR:
  parent: World
  name: Sub-Saharan Africa
  child: [AGO, BDI, BEN, BFA, BWA, CAF, CIV, CMR, COD, COG, COM, CPV, DJI, ERI,␣
↪ETH, GAB, GHA, GIN, GMB, GNB, GNQ, KEN, LBR, LSO, MDG, MLI, MOZ, MRT, MUS, MWI,␣
↪MYT, NAM, NER, NGA, REU, RWA, SEN, SHN, SLE, SOM, STP, SWZ, SYC, TCD, TGO, TZA,␣
↪UGA, ZAF, ZMB, ZWE]
  iea-weo-region: Africa

R12_RCPA:
  parent: World
  name: Rest Centrally Planned Asia
  child: [KHM, LAO, MNG, PRK, VNM]
  iea-weo-region: China

R12_CHN:
  parent: World
  name: China
  child: [CHN, HKG]
  iea-weo-region: China

R12_EEU:
  parent: World
```

(continues on next page)

(continued from previous page)

```
  name: Central and Eastern Europe
  description: >-
    Serbia and Montenegro (SCG) and Yugoslavia (YUG) still included in this list,
    even though their ISO 3166-1 codes were deleted in 2006 and 2003, respectively.
  child: [ALB, BGR, BIH, CZE, EST, HRV, HUN, LTU, LVA, MKD, MNE, POL, ROU, SCG,␣
↪SRB, SVK, SVN, YUG]
  iea-weo-region: European Union

R12_FSU:
  parent: World
  name: Former Soviet Union
  child: [ARM, AZE, BLR, GEO, KAZ, KGZ, MDA, RUS, TJK, TKM, UKR, UZB]
  iea-weo-region: Russia

R12_LAM:
  parent: World
  name: Latin America and The Caribbean
  description: >-
    The source includes "Netherlands Antilles" which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT),
    but is not a country. It was dissolved in 2010 into BES, CUW and SXM, also␣
↪included.
  child: [ABW, AIA, ANT, ARG, ATG, BES, BHS, BLZ, BMU, BOL, BRA, BRB, CHL, COL,␣
↪CRI, CUB, CUW, CYM, DMA, DOM, ECU, FLK, GLP, GRD, GTM, GUF, GUY, HND, HTI, JAM,␣
↪KNA, LCA, MEX, MSR, MTQ, NIC, PAN, PER, PRY, SLV, SUR, SXM, TCA, TTO, URY, VCT,␣
↪VEN, VGB]
  iea-weo-region: Brazil

R12_MEA:
  parent: World
  name: Middle East and North Africa
  child: [ARE, BHR, DZA, EGY, ESH, IRN, IRQ, ISR, JOR, KWT, LBN, LBY, MAR, OMN,␣
↪PSE, QAT, SAU, SDN, SSD, SYR, TUN, YEM]
  iea-weo-region: Middle East

R12_NAM:
  parent: World
  name: North America
  child: [CAN, GUM, PRI, SPM, USA, VIR]
  iea-weo-region: United States

R12_PAO:
  parent: World
  name: Pacific OECD
  child: [AUS, JPN, NZL]
  iea-weo-region: Japan

R12_PAS:
  parent: World
  name: Other Pacific Asia
  description: >-
    Trust Territory of the Pacific Islands (PCI) still included in this list,
    but it was dissolved into MHL, FSM, MNP and PLW in 1986.
  child: [ASM, BRN, CCK, COK, CXR, FJI, FSM, IDN, KIR, KOR, MAC, MHL, MMR, MNP,␣
↪MYS, NCL, NFK, NIU, NRU, PCI, PCN, PHL, PLW, PNG, PYF, SGP, SLB, THA, TKL, TLS,␣
↪TON, TUV, TWN, VUT, WLF, WSM]
  iea-weo-region: India

R12_SAS:
  parent: World
  name: South Asia
```

(continues on next page)

(continued from previous page)

```
  child: [AFG, BGD, BTN, IND, LKA, MDV, NPL, PAK]
  iea-weo-region: India

R12_WEU:
  parent: World
  name: Western Europe
  child: [AND, AUT, BEL, CHE, CYP, DEU, DNK, ESP, FIN, FRA, FRO, GBR, GIB, GRC,␣
↪GRL, IMN, IRL, ISL, ITA, LIE, LUX, MCO, MLT, NLD, NOR, PRT, SJM, SMR, SWE, TUR,␣
↪VAT]
  iea-weo-region: European Union
```

5-region aggregation (RCP)

```
# Codes for the "node" dimension of the Representative Concentration Pathways
#
# - See message_data.tools.regions.
# - Since ixmp does not support the "." character in IDs, the names "R5.2ASIA"
#   are transformed to "R5_ASIA" etc. The original code is left in a
#   description.
# - The ISO 3166-1 alpha-3 codes are not defined in this file, but loaded from
#   a copy of the ISO database, e.g. in pycountry.
# - Among others, there are no assignments for:
#   - ATA Antarctica
#   - IOT British Indian Ocean Territory
#   - SGS South Georgia

World:
  name: World
  description: RCP regions

R5_ASIA:
  parent: World
  description: |-
    Officially "R5.2ASIA".

    Trust Territory of the Pacific Islands (PCI) still included in this list, but␣
↪it was dissolved into MHL, FSM, MNP and PLW in 1986.
  child: [AFG, ASM, BGD, BRN, BTN, CCK, CHN, COK, CXR, FJI, FSM, GUM, HKG, IDN,␣
↪IND, KHM, KIR, KOR, LAO, LKA, MAC, MDV, MHL, MMR, MNG, MNP, MYS, MYT, NCL, NFK,␣
↪NIU, NPL, NRU, PAK, PCI, PCN, PHL, PLW, PNG, PRK, PYF, SGP, SLB, SYC, THA, TKL,␣
↪TLS, TON, TUV, TWN, VNM, VUT, WSM]

R5_LAM:
  parent: World
  description: |-
    Officially "R5.2LAM".

    The source includes “Netherlands Antilles” which has a provisional ISO 3166-2␣
↪alpha-3 code (ANT), but is not a country. It was dissolved in 2010 into BES, CUW␣
↪and SXM, also included.
  child: [ABW, AIA, ANT, ARG, ATG, BES, BHS, BLZ, BMU, BOL, BRA, BRB, CHL, COL,␣
↪CRI, CUB, CUW, CYM, DMA, DOM, ECU, GLP, GRD, GTM, GUF, GUY, HND, HTI, JAM, KNA,␣
↪LCA, MEX, MSR, MTQ, NIC, PAN, PER, PRY, SLV, SUR, SXM, TTO, URY, VCT, VEN]

R5_MAF:
  parent: World
  description: Officially "R5.2MAF".
  child: [AGO, ARE, BDI, BEN, BFA, BHR, BWA, CAF, CIV, CMR, COD, COG, COM, CPV,␣
↪DJI, DZA, EGY, ERI, ESH, ETH, GAB, GHA, GIN, GMB, GNB, GNQ, IRN, IRQ, ISR, JOR,␣
```

(continues on next page)

(continued from previous page)

```
↪KEN, KWT, LBN, LBR, LBY, LSO, MAR, MDG, MLI, MOZ, MRT, MUS, MWI, NAM, NER, NGA,␣
↪OMN, PSE, QAT, REU, RWA, SAU, SDN, SEN, SLE, SOM, SSD, STP, SWZ, SYR, TCD, TGO,␣
↪TUN, TZA, UGA, YEM, ZAF, ZMB, ZWE]

R5_OECD:
  parent: World
  description: |-
    Officially "R5.2OECD".

    Serbia and Montenegro (SCG) and Yugoslavia (YUG) still included in this list,␣
↪even though their ISO 3166-1 codes were deleted in 2006 and 2003, respectively.
  child: [ALB, AND, AUS, AUT, BEL, BGR, BIH, CAN, CHE, CYP, CZE, DEU, DNK, ESP,␣
↪EST, FIN, FLK, FRA, FRO, GBR, GIB, GRC, GRL, HRV, HUN, IMN, IRL, ISL, ITA, JPN,␣
↪LIE, LTU, LUX, LVA, MCO, MKD, MLT, MNE, NLD, NOR, NZL, POL, PRI, PRT, ROU, SCG,␣
↪SHN, SJM, SMR, SPM, SRB, SVK, SVN, SWE, TCA, TUR, USA, VAT, VGB, VIR, WLF, YUG]

R5_REF:
  parent: World
  description: Officially "R5.2REF".
  child: [ARM, AZE, BLR, GEO, KAZ, KGZ, MDA, RUS, TJK, TKM, UKR, UZB]
```

8.20.2 Others

These include models scoped to a single country or region, or a subset of all countries or regions, as well as code lists used in specific data sets from which *message_ix_models* handles data.

ADVANCE project (ADVANCE)

```
# Node code list for the ADVANCE project
#
# Source: https://db1.ene.iiasa.ac.at/ADVANCEDB/dsd?Action=htmlpage&page=10
# Transcribed 2022-07-22 by P.N. Kishimoto
#

World:
  # Countries represented individually
  child: [BRA, CHN, IND, JPN, RUS, USA]

EU:
  description: >-
    European Union (28 member states from the accession of HRV in 2013 to the
    withdrawal of GBR in 2020).
  child: [AUT, BEL, BGR, CYP, CZE, DEU, DNK, ESP, EST, FIN, FRA, GBR, GRC, HRV,␣
↪HUN, IRL, ITA, LTU, LUX, LVA, MLT, NLD, POL, PRT, ROU, SVK, SVN, SWE]

OECD90+EU:
  description: Includes the OECD 1990 countries as well as EU members and␣
↪candidates.
  child: [ALB, AUS, AUT, BEL, BGR, BIH, CAN, CHE, CYP, CZE, DEU, DNK, ESP, EST,␣
↪FIN, FJI, FRA, GBR, GRC, GUM, HRV, HUN, IRL, ISL, ITA, JPN, LTU, LUX, LVA, MKD,␣
↪MLT, MNE, NCL, NLD, NOR, NZL, POL, PRT, PYF, ROU, SLB, SRB, SVK, SVN, SWE, TUR,␣
↪USA, VUT, WSM]

REF:
  description: Countries from the Reforming Ecomonies of the Former Soviet Union.
  child: [ARM, AZE, BLR, GEO, KAZ, KGZ, MDA, RUS, TJK, TKM, UKR, UZB]

ASIA:
```

(continues on next page)

(continued from previous page)

```
  description: The region includes most Asian countries with the exception of the␣
↪Middle East, Japan and Former Soviet Union states.
  child: [AFG, BGD, BRN, BTN, CHN, HKG, IDN, IND, KHM, KOR, LAO, LKA, MAC, MDV,␣
↪MMR, MNG, MYS, NPL, PAK, PHL, PNG, PRK, SGP, THA, TMP, TWN, VNM]

MAF:
  description: This region includes the countries of the Middle East and Africa.
  child: [DZA, AGO, ARE, BDA, BEN, BFA, BHR, BWA, CAF, CIV, CMR, COD, COG, COM,␣
↪CPV, DJI, EGY, ERI, ESH, ETH, GAB, GHA, GIN, GMB, GNB, GNQ, IRN, IRQ, ISR, JOR,␣
↪KEN, KWT, LBN, LBR, LBY, LSO, MAR, MDG, MLI, MOZ, MRT, MUS, MWI, NAM, NER, NGA,␣
↪OMN, QAT, REU, RQA, SAU, SDN, SEN, SLE, SOM, SWZ, SYR, TCD, TGO, TUN, TZA, UGA,␣
↪YEM, ZAF, ZMB, ZWE]

LAM:
  description: This region includes the countries of Latin America and the␣
↪Caribbean.
  child: [ARG, ANT, BHS, BLZ, BOL, BRA, BRB, CHL, COL, CRA, CUB, DOM, EDU, GLP,␣
↪GTM, GUY, HND, HTI, JAM, MEX, MTQ, NIC, PAN, PER, PRI, PRY, SLV, SUR, TTO, URY,␣
↪VEN]
```

Israel (ISR)

```
# Codes for the "node" dimension of the MESSAGE-IL model

World:
  name: World
  description: MESSAGE-IL regions

Israel:
  name: Israel
  parent: World
  child: [ISR]
```

Zambia (ZMB)

```
# Codes for the "node" dimension of the MESSAGE-ZM model

World:
  name: World
  description: MESSAGE-ZM regions

Zambia:
  name: Zambia
  parent: World
  child: [ZMB]
```

8.21 Relations (`relation/*.yaml`)

These lists provide codes (ID, optional name and description) for the `relation` MESSAGEix set. See the MESSAGEix documentation sections on:

- Section of generic relations (linear constraints).
- Parameters of the Relations section.

The codes in these lists have the following annotations:

`group`
: 1 or more strings identifying (a) group(s) to which the relations belong(s).

`parameters`
: 1 or more strings indicating relation parameters for which there should be entries corresponding to this relation ID. For example, "activity, lower" indicates there should be `relation_activity` and `relation_lower` values for this relation ID.

`technology-entry`

Todo: Describe the meaning of this annotation.

8.21.1 List `A`

```yaml
# Relation codes
#
# This set includes all those expected by code in message_data as of 2023-02-04.
# It is the union of entries from:
#
# - data/buildings/set.yaml
# - data/transport/set.yaml
#
# The descriptions are derived from CD-LINKS.yaml

BCA_Emission:
  description: >-
    Emissions of BCA.

    Weight as BCA, kg BCA per kW·a.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

CH4_Emission:
  description: >-
    Emissions of CH4

    Weight as CH4, kg CH4 per kW·a.
  group: emission_factor
  parameters: activity, lower, upper
  technology-entry: 1
```

(continues on next page)

(continued from previous page)

```
CO_Emission:
  description: >-
    Emissions of CO (weight as CO, kg CO per kWyr)
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

CO2_Emission_Global_Total:
  description: >-
    Used by message_data.tools.utilities to constrain global total CO₂ emissions
    to be non-negative.

CO2_r_c:
  description: CO₂ emissions of residential and commercial buildings sector.

CO2_trp:
  description: CO₂ emissions of transport sector.

HFC_Emission:
  description: >-
    Emissions of HFC (weight as HFC, kg HFC-134a-equivalent per kWyr)
  group: emission_factor
  parameters: activity, lower, upper
  technology-entry: 1

HFC_foam_red:
  description: >-
    Constraint to link HFC emissions from insulation foams to reduction
    efficiency of abatement technologies.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

HFC_mobile_red:
  description: >-
    Constraint to link HFC emissions from mobile air conditioning to reduction
    efficiency of abatement technologies.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

HFC_rescom_red:
  description: >-
    Constraint to link HFC emissions from residential & commercial refrigeration
    & AC to reduction efficiency of abatement technologies.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

N2O_Emission:
  description: >-
    Emissions of N₂O

    Weight as N₂O, kg N₂O per kW·a.
  group: emission_factor
  parameters: activity, lower, upper
  technology-entry: 1

NH3_Emission:
  description: >-
```

(continues on next page)

(continued from previous page)

```
    Emissions of NH3

    Weight as NH3, kg NH3 per kW·a.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

NOx_Emission:
  description: >-
    Emissions of nitrous oxides (NOx)

    Weight as NOx, kg NOx per kW·a.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

OCA_Emission:
  description: >-
    Emissions of OCA

    Weight as OCA, kg OCA per kW·a.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

oper_res:
  description: Operating reserve equation for electricity systems integration.
  technology-entry: 1
  parameters: activity, lower
  group: renewable formulation

SF6_Emission:
  description: >-
    Emissions of SF6

    Weight as SF6, kg SF6 per kW·a.
  group: emission_factor
  parameters: activity, lower, upper
  technology-entry: 1

SF6_mag_red:
  description: >-
    Constraint to link SF6 emissions from electrical equipment to reduction
    efficiency of abatement technologies.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

SO2_Emission:
  description: >-
    Emissions of SO2

    Weight as SO2, kg SO2 per kW·a.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1

solar_th_pot:
  description: Solar thermal potential
  technology-entry: 1
  parameters: activity, upper
```

(continues on next page)

(continued from previous page)

```
  group: renewable formulation

VOC_Emission:
  description: >-
    Emissions of volatile organic compounds (VOCs)

    Weight as VOC, kg VOC per kW·a.
  group: emission_factor
  parameters: activity, lower
  technology-entry: 1
```

8.21.2 List B

```
# Relation codes
#
# This set includes all values appearing in:
# ixmp://ixmp-dev/MESSAGEix-GLOBIOM 1.1-BMT-R12 (NAVIGATE)/NPi-ref#2

- BCA_Emission
- BCA_Emission_bunkers
- CF4_Emission
- CF4_alm_red
- CH4_Emission
- CH4_Emission_bunkers
- CH4_RemainingLink
- CH4_new_Emission
- CH4_nonenergy
- CH4n_landfills_red
- CO2_Emission
- CO2_Emission_Global_Total
- CO2_cc
- CO2_feedstocks
- CO2_ind
- CO2_r_c
- CO2_shipping
- CO2_trade
- CO2_trp
- CO_Emission
- CO_Emission_bunkers
- CO_nonenergy
- FE_coal
- FE_electricity+dh
- FE_feedstock
- FE_final_energy
- FE_gaseous
- FE_industry
↪                              [266/1626]
- FE_liquids
- FE_new_biomass
- FE_old_biomass
- FE_resid_comm
- FE_transport
- HFC_Emission
- HFC_foam_red
- HFC_mobile_red
- HFC_rescom_red
- HFCequivOther
- IndGDPAdLink
- IndGDPNiLink
```

(continues on next page)

(continued from previous page)

```
- IndThermDemLink
- N2O_Emission
- N2O_Emission_bunkers
- N2O_nonenergy
- N2On_adipic_red
- N2On_nitric_red
- N2Oother
- NH3_Emission
- NOx_Emission
- NOx_Emission_bunkers
- NOx_nonenergy
- OCA_Emission
- OCA_Emission_bunkers                                                    ␣
↪                              [241/1626]
- OilPrices
- PE_BACKSTOPS
- PE_coal
- PE_domestic_total
- PE_export_total
- PE_gas
- PE_hydro
- PE_import_share
- PE_import_total
- PE_new_renewables
- PE_nuclear
- PE_oil
- PE_old_renewables
- PE_synliquids
- PE_total_direct
- PE_total_engineering
- PE_total_traditional
- PM2_Emission
- POPtoAeroNonMDILink
- POPtoAerosolMDILink
- POPtoDomWasteWaLink
- POPtoFireExtLink
- POPtoHuSewageLink
- POPtoSolvent
- RES_variable_limt
- SF6_Emission
- SF6_elec_red                                                            ␣
↪                              [214/1626]
- SF6_mag_red
- SO2_Emission
- SO2_Emission_bunkers
- SO2_elec
- SO2_import
- SO2_ind
- SO2_r_c
- SO2_red_ref
- SO2_red_synf
- SolWaPOPLink
- TCE_Emission
- TCE_Emission_AFR_CPA_LAC_MEA_PAS_SAS
- TCE_Emission_FSU_MEA
- TCE_Emission_NAM_PAO_EEU_WEU
- TCE_Emission_NAM_PAO_EEU_WEU_FSU
- TCE_regional
- TCE_rel2020
- TCE_trade
- TCEl
```

(continues on next page)

(continued from previous page)

```
- TCF4_Emission
- TCH4_Emission
- TCO2_Emission
- THFC_Emission
- TN2O_Emission
- TSF6_Emission
- UE_feedstock
- UE_feedstock_coal                                                              ␣
↪                              [187/1626]
- UE_feedstock_foil
- UE_feedstock_gas
- UE_feedstock_liquid
- UE_industry_sp
- UE_industry_sp_fc
- UE_industry_sp_liquid
- UE_industry_sp_solar
- UE_industry_th
- UE_industry_th_electric
- UE_industry_th_foil
- UE_industry_th_gas
- UE_industry_th_hydrogen
- UE_industry_th_liquid
- UE_industry_th_low_temp_heat
- UE_industry_th_solar
- UE_industry_th_solid
- UE_res_comm_sp
- UE_res_comm_sp_fc
- UE_res_comm_sp_solar
- UE_res_comm_th
- UE_res_comm_th_biomass
- UE_res_comm_th_electric
- UE_res_comm_th_foil
- UE_res_comm_th_gas
- UE_res_comm_th_heat
- UE_res_comm_th_hp
- UE_res_comm_th_hydrogen                                                        ␣
↪                              [160/1626]
- UE_res_comm_th_liquid
- UE_res_comm_th_solar
- UE_res_comm_th_solids
- UE_transport
- UE_transport_electric
- UE_transport_electric_Minimum
- UE_transport_fc
- UE_transport_foil
- UE_transport_gas
- UE_transport_liquid
- VOC_Emission
- VOC_Emission_bunkers
- VOC_IndThermLink
- VOC_nonenergy
- WasteGenToBCALink
- WasteGenToCH4Link
- WasteGenToCOLink
- WasteGenToNOxLink
- WasteGenToOCALink
- WasteGenToPM2Link
- WasteGenToSO2Link
- WasteGenToVOCLink
- bco2_trans_disp
- bio_extr_mpen_c                                                                ␣
```

(continues on next page)

(continued from previous page)

```
↪                              [136/1626]
- bio_scrub_lim
- co2_trans_disp
- co_gen_limit
- coal-exp
- coal_extr
- coal_extr_mpen_c
- coal_scrub_lim
- csp_japan_limit
- demF_limit
- demI_limit
- demR_limit
- demb_limit
- demp_limit
- demt_limit
- domestic_coal
- domestic_gas
- ele_nuc
- elec_coal
- elec_gas
- elec_hydro
- elec_nuclear
- elec_relations2_coal
- elec_relations2_gas
- elec_relations2_hydro
- elec_relations2_nuclear
- elec_relations2_oil
- elec_relations2_renewable                                                    ⌙
↪                              [109/1626]
- extraction_coal
- extraction_gas
- extraction_oil
- foil_prod
- gas_export
- gas_extr_mpen_c
- gas_mix_lim
- gas_prod
- gas_scrub_lim
- gas_util
- global_GCO2
- global_GSO2
- h2_exp_limit
- h2_scrub_limit
- h2mix_direct
- hydro_min
- hydro_pot
- import_dependence
- lfil_limit
- lim_exp_afr_LNG
- lim_exp_afr_coal
- lim_exp_afr_eth
- lim_exp_afr_meth
- lim_exp_cpa_oil
- lim_exp_eeu_eth
- lim_exp_eeu_meth
- lim_exp_fsu_LNG                                                              ⌙
↪                               [82/1626]
- lim_exp_fsu_coal
- lim_exp_fsu_eth
- lim_exp_fsu_meth
- lim_exp_fsu_oil
```

(continues on next page)

(continued from previous page)

```
- lim_exp_lam_LNG
- lim_exp_lam_oil
- lim_exp_mea_LNG
- lim_exp_mea_oil
- lim_exp_nam_LNG
- lim_exp_nam_coal
- lim_exp_pao_coal
- lim_exp_pao_eth
- lim_exp_pao_meth
- lim_exp_pas_LNG
- lim_exp_weu_eth
- lim_exp_weu_meth
- limit_tot_BECCS
- max_coal_ind
- meth_exp_limit
- min-liquids_industry
- min-liquids_res-com
- minimum_recycling_aluminum
- minimum_recycling_steel
- nica_limit
- nuc_lc_in_el
- oil_based_elec_gen
- oil_extr_mpen_c                                                    ⌐
↪                              [55/1626]
- oil_imp_c
- oil_prod
- oil_trd
- oper_res
- pass_out_trb
- prod_low_lim_BIEL
- prod_low_lim_BIFU
- prod_low_lim_BIHE
- prod_low_lim_COEL
- prod_low_lim_GAEL
- prod_low_lim_GEEL
- prod_low_lim_HYEL
- prod_low_lim_NUEL
- prod_low_lim_OIEL
- prod_low_lim_SOEL
- prod_low_lim_WIEL
- res_japan_limit
- res_marg
- share_low_lim_NFEG
- share_low_lim_NFPE
- share_low_lim_PEGas
- share_low_lim_REEG
- share_low_lim_REFE
- share_low_lim_REPE
- sm1_ppl_res
- sm3_ppl_res
- solar_csp_pot
- solar_csp_pot1                                                     ⌐
↪                              [27/1626]
- solar_csp_pot2
- solar_csp_pot3
- solar_csp_pot4
- solar_csp_pot5
- solar_csp_pot6
- solar_csp_pot7
- solar_curtailment_1
- solar_curtailment_2
```

(continues on next page)

(continued from previous page)

```
- solar_curtailment_3
- solar_pot
- solar_pot2
- solar_pot3
- solar_pot4
- solar_pot5
- solar_pot6
- solar_pot7
- solar_pot8
- solar_pot_hist_2010
- solar_pot_hist_2015
- solar_pot_hist_2020
- solar_ppl_res
- solar_res_cv
- solar_step
- solar_step2
- solar_step3
- solar_th_pot'
- syn_liq_exp
- total_TCO2
- trp_energy
- unabated_coal_lim
- wind_curtailment_1
- wind_curtailment_2
- wind_curtailment_3
- wind_pof
- wind_pof2
- wind_pof3
- wind_pof4
- wind_pof5
- wind_pot
- wind_pot2
- wind_pot3
- wind_pot4
- wind_pot_hist_2005
- wind_pot_hist_2010
- wind_pot_hist_2015
- wind_pot_hist_2020
- wind_ppl_offshore_res
- wind_ppl_res
- wind_res_cv
- wind_step
- wind_step2
- wind_step3
```

8.21.3 List `CD-LINKS`

```
# Relation codes
#
# The IDs, descriptions, and other annotations are derived from a spreadsheet
# appearing in IIASA ECE SharePoint, related to the CD-LINKS project.
#
# The entries appear in the same order as in the file.

BCA_Emission_bunkers:
  technology-entry: 1
  parameters: activity
  description: >-
    BC emission from international shipping (bunker fuel)
```

(continues on next page)

(continued from previous page)

```
  group: removed
CO_Emission_bunkers:
  technology-entry: 1
  parameters: activity
  description: >-
    CO emission from international shipping (bunker fuel)
  group: removed
CO2_cc:
  technology-entry: 1
  parameters: activity
  description: >-
    Emissions of CO2 in central conversion and extraction
  group: removed
CO2_ind:
  technology-entry: 1
  parameters: activity
  description: >-
    CO2 emissions of industry sector
  group: removed
CO2_r_c:
  technology-entry: 1
  parameters: activity
  description: >-
    CO2 emissions of residential and commercial sector
  # NB despite this annotation, the relation still has entries in current
  #    scenarios; see A.yaml and B.yaml
  group: removed
CO2_trp:
  technology-entry: 1
  parameters: activity
  description: >-
    CO2 esmissions of transport sector
  # NB despite this annotation, the relation still has entries in current
  #    scenarios; see A.yaml and B.yaml
  group: removed
coal_extr:
  technology-entry: 1
  parameters: activity
  description: >-
    In R11 GLB only - used for accounting or limiting global coal extraction
    (currently inactive as only positive entries)
  group: removed
elec_relations2_coal:
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - electricty from coal (inactive as no -1)
  group: removed
elec_relations2_gas:
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - electricty from gas (inactive as no -1)
  group: removed
elec_relations2_hydro:
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - electricty from hydro (inactive as no -1)
  group: removed
elec_relations2_nuclear:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - electricty from nuclear (inactive as no -1)
  group: removed
elec_relations2_oil:
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - electricty from oil (inactive as no -1)
  group: removed
elec_relations2_renewable:
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - electricty from renewable (inactive as no -1)
  group: removed
extraction_coal:
  technology-entry: 1
  parameters: activity
  description: >-
    Constrains/calibrates historical coal extraction (inactive as no -1)
  group: removed
extraction_gas:
  technology-entry: 1
  parameters: activity
  description: >-
    Constrains/calibrates historical gas extraction (inactive as no -1)
  group: removed
extraction_oil:
  technology-entry: 1
  parameters: activity
  description: >-
    Constrains/calibrates historical oil extraction (inactive as no -1)
  group: removed
FE_coal:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_electricity:
  kind:
  technology-entry: 1
  parameters2: upper
  description: >-
    final energy accounting equation
  group: removed
FE_feedstock:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_final_energy:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_gaseous:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_industry:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_liquids:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_new_biomass:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_old_biomass:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_resid_comm:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy accounting equation
  group: removed
FE_transport:
  technology-entry: 1
  parameters: activity
  description: >-
    final energy use in transport sector
  group: removed
global_GCO2:
  technology-entry: 1
  parameters: activity
  description: >-
    R11-GLB only - Total CO2 emissions
  group: removed
global_GSO2:
  technology-entry: 1
  parameters: activity
  description: >-
    R11-GLB only - Total SO2 Emissions
  group: removed
import_dependence:
  technology-entry: 1
  parameters: activity
  description: >-
    maximum import dependence
  group: removed
NOx_Emission_bunkers:
  technology-entry: 1
```

(continues on next page)

(continued from previous page)

```
  parameters: activity
  description: >-
    NOx emission from international shipping (bunker fuel)
  group: removed
nuc_lc_in_el:
  technology-entry: 1
  parameters: activity
  description: >-
    Constrains nuc_lc contribution to electricity production
  group: removed
OCA_Emission_bunkers:
  technology-entry: 1
  parameters: activity
  description: >-
    OC emission from international shipping (bunker fuel)
  group: removed
oil_prod:
  technology-entry: 1
  parameters: activity
  description: >-
    R11-NAM only - Limits import dependence of crude oil to max 60% (inactive
    as there are no negative entries and no limits)
  group: removed
oil_trd:
  technology-entry: 1
  parameters: activity
  description: >-
    In R11 GLB only - penalty for international trade introduced to reflect
    costs for importing region. In SOME other regions - links/limits net oil
    product imports to fraction of net crude imports.
  group: removed
OilPrices:
  technology-entry: 1
  parameters: activity
  description: >-
    crude oil import prices
  group: removed
PE_BACKSTOPS:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_coal:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_gas:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_hydro:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
```

(continues on next page)

(continued from previous page)

```
PE_new_renewables:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_nuclear:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_oil:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_old_renewables:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_synliquids:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_total_direct:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_total_engineering:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
PE_total_traditional:
  technology-entry: 1
  parameters: activity
  description: >-
    primary energy accounting equation
  group: removed
prod_low_lim_BIEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_BIFU:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_BIHE:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_COEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_GAEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_GEEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_HYEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_NUEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_OIEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_SOEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
prod_low_lim_WIEL:
  technology-entry: 0
  parameters: ""
  description: >-
    INDC Policy constraint
  group: removed
share_low_lim_NFEG:
  technology-entry: 0
  parameters: ""
  description: >-
    removed
share_low_lim_NFPE:
  technology-entry: 0
  parameters: ""
```

(continues on next page)

(continued from previous page)

```
  description: >-
    removed
share_low_lim_PEGas:
  technology-entry: 0
  parameters: ""
  description: >-
    removed
share_low_lim_REEG:
  technology-entry: 0
  parameters: ""
  group: removed
share_low_lim_REFE:
  technology-entry: 0
  parameters: ""
  group: removed
share_low_lim_REPE:
  technology-entry: 0
  parameters: ""
  group: removed
SO2_Emission_bunkers:
  technology-entry: 1
  parameters: activity
  description: >-
    SO2 emission from international shipping (bunker fuel)
  group: removed
TCE_Emission_AFR_CPA_LAC_MEA_PAS_SAS:
  technology-entry: 1
  parameters: activity
  description: >-
    total carbon-equivalent emissions for set of regions
  group: removed
TCE_Emission_FSU_MEA:
  technology-entry: 1
  parameters: activity
  description: >-
    total carbon-equivalent emissions for set of regions
  group: removed
TCE_Emission_NAM_PAO_EEU_WEU:
  technology-entry: 1
  parameters: activity
  description: >-
    total carbon-equivalent emissions for set of regions
  group: removed
TCE_Emission_NAM_PAO_EEU_WEU_FSU:
  technology-entry: 1
  parameters: activity
  description: >-
    total carbon-equivalent emissions for set of regions
  group: removed
TCE_regional:
  technology-entry: 1
  parameters: activity
  description: >-
    total carbon-equivalent emission equation (regional)
  group: removed
TCE_rel2020:
  technology-entry: 1
  parameters: activity
  description: >-
    "(?) Counter - TCE in 2020
  group: removed
```

(continues on next page)

(continued from previous page)

```yaml
TCE_trade:
  technology-entry: 1
  parameters: activity
  description: >-
    trade in carbon-equivalent emissions
  group: removed
TCF4_Emission:
  technology-entry: 1
  parameters: activity
  description: >-
    R11-GLB only - Total Emissions of CF4 (weight as CF4, kg CF4 per kWyr)
  group: removed
TCH4_Emission:
  technology-entry: 1
  parameters: activity
  description: >-
    total CH4 emission equation
  group: removed
TCO2_Emission:
  technology-entry: 1
  parameters: activity
  description: >-
    total CO2 emission equation
  group: removed
THFC_Emission:
  technology-entry: 1
  parameters: activity
  description: >-
    R11-GLB only - Counter for HFC emissions
  group: removed
TN2O_Emission:
  technology-entry: 1
  parameters: activity
  description: >-
    total N2O emission equation
  group: removed
total_TCO2:
  technology-entry: 1
  parameters: activity
  description: >-
    Total CO2 emissions including feedstocks
  group: removed
trp_energy:
  technology-entry: 1
  parameters: activity
  description: >-
    limits total final energy in transport, if necessary (inactive as set to
    free)
  group: removed
TSF6_Emission:
  technology-entry: 1
  parameters: activity
  description: >-
    R11-GLB only - Total Emissions of SF6 (weight as SF6, kg SF6 per kWyr)
  group: removed
VOC_Emission_bunkers:
  technology-entry: 1
  parameters: activity
  description: >-
    VOC emission from international shipping (bunker fuel)
  group: removed
```

(continues on next page)

(continued from previous page)

```
VOC_IndThermLink:
  technology-entry: 1
  parameters: activity
  description: >-
    Constraint links VOC solvent emissions to its driver: the industrial themal
    demand.
  group: removed
BCA_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of BCA (weight as BCA, kg BCA per kWyr)
  group: emission_factor
bco2_trans_disp:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    CO2 transport and injection from BECCS
bio_extr_mpen_c:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link equation for joint diffusion constraint of biomass extraction␣
↪technologies (deprecated?)
bio_scrub_lim:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on CO2 scrubber application (CCS add-on) for biomass power plants
cement_pro:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    relation for cement production linked to ind thermal
cement_scrub_lim:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Constraint limiting cement scrubber activity to cement CO2 production
CF4_alm_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link CF4 emissions from aluminum to reduction efficiency of
    abatement technologies.
  group: add-on
CF4_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of CF4 (weight as CF4, kg CF4 per kWyr)
  group: emission_factor
CH4_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of CH4 (weight as CH4, kg CH4 per kWyr)
  group: emission_factor
CH4_Emission_bunkers:
  technology-entry: 1
  parameters: activity, lower, upper
```

(continues on next page)

(continued from previous page)

```
  description: >-
    Counter - R11-GLB only - CH4 emissions of international shipping
  group: emission_factor
CH4_new_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Counter for CH4 emissions from newly added sources
  group: emission_factor
CH4_nonenergy:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of CH4 from nonenergy sources (weight as CH4, kg CH4 per kWyr)
  group: emission_factor
CH4_RemainingLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint links the remaining CH4 emissions to oil extraction
  group: emission_factor
CH4n_landfills_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link landfill emissions to reduction efficiency of CH4
    abatement technologies.
  group: add-on
CO_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of CO (weight as CO, kg CO per kWyr)
  group: emission_factor
co_gen_limit:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Limits cogeneration as share of total central heat production
  group: share-constraint formulation
CO_nonenergy:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    CO emission from non-energy sectors
  group: emission_factor
CO2_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of CO2 (weight as carbon, ton C/kWyr)
  group: emission_factor
CO2_feedstocks:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    CO2 emissions of feedstock use (non-energy use)
  group: emission_factor
CO2_shipping:
  technology-entry: 1
  parameters: activity, lower, upper
```

(continues on next page)

(continued from previous page)

```
  description: >-
    Counter - R11-GLB only - CO2 emissions of international shipping
  group: emission_factor
CO2_trade:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Counter - R11-GLB only - CO2 emissions of energy trade
  group: emission_factor
co2_trans_disp:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint to connect captured carbon to transportation and disposal
  group: add-on
coal_extr_mpen_c:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link equation for joint diffusion constraint of coal extraction technologies
  group: fossil resources
coal_scrub_lim:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on CO2 scrubber application (CCS add-on) for coal power plants
  group: add-on
coal-exp:
  parameters: activity, upper
  technology-entry: 1
  description: >-
    Restricts coal exports to hard coal only  by limiting lignite to power
    plants and other central conversion.
  group: share-constraint formulation
csp_japan_limit:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on renewable resource use of Japan in PAO region
  group: renewable formulation
demb_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint linking mpa for additional VOM costs for breaching share
    constraints related to Industrial Thermal sector to growth rate
  group: share-constraint formulation
demF_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint linking mpa for additional VOM costs for breaching share
    constraints related to Feedstock sector to growth rate
  group: share-constraint formulation
demI_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint linking mpa for additional VOM costs for breaching share
    constraints related to Industrial Specific sector to growth rate (currently
    inactive as all technologies have bda up 0)
```

(continues on next page)

(continued from previous page)

```
  group: share-constraint formulation
demp_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint linking mpa for additional VOM costs for breaching share
    constraints related to Transport sector to growth rate
  group: share-constraint formulation
demR_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint linking mpa for additional VOM costs for breaching share
    constraints related to Residential Commercial Specific sector to growth rate
  group: share-constraint formulation
demt_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint linking mpa for additional VOM costs for breaching share
    constraints related to Residential Commercial Thermal sector to growth rate
  group: share-constraint formulation
domestic_coal:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11-PAO only - Constraint - limits domestic coal use to x% of total primary
    energy.
  group: share-constraint formulation
domestic_gas:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11-PAO only - Constraint - limits domestic coal use to x% of total primary
    energy.
  group: share-constraint formulation
ele_nuc:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constrains/calibrates historical nuclear
  group: hisc
elec_coal:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constrains/calibrates historical electricity production from coal
  group: hisc
elec_gas:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constrains/calibrates historical electricity production from natural gas
  group: hisc
elec_hydro:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constrains/calibrates historical electricity production from hydro
  group: hisc
elec_nuclear:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constrains/calibrates historical electricity production from nuclear
  group: hisc
foil_prod:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constrains/calibrates historical refinery output
  group: hisc
gas_export:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constrains/calibrates historical gas exports
  group: hisc
gas_extr_mpen_c:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link equation for joint diffusion constraint of gas extraction technologies
  group: fossil resources
gas_mix_lim:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11-AFR only - Constrains hydrogen contribution to final gas usage
  group: share-constraint formulation
gas_prod:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    links and stabilizes gas production (currently inactive - no positive entry)
  group: obsolete (?)
gas_scrub_lim:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on CO2 scrubber application (CCS add-on) for gas power plants
  group: add-on
gas_util:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Links gas infrastructure growth between utilities versus residential and
    industry (inactive as not negative entry).
  group: obsolete (?)
h2_exp_limit:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    limits hydrogen exports as share of total hydrogen production
  group: share-constraint formulation
h2_scrub_limit:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    scrubbing limit of CO2 from natural gas due to hydrogen blending
  group: add-on
h2mix_direct:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity, lower
  description: >-
    blending of hydrogen into gas grid
  group: RES
HFC_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of HFC (weight as HFC, kg HFC-134a-equivalent per kWyr)
  group: emission_factor
HFC_foam_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link HFC emissions from insulation foams to reduction
    efficiency of abatement technologies.
  group: emission_factor
HFC_mobile_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link HFC emissions from mobile airconditioning to reduction
    efficiency of abatement technologies.
  group: emission_factor
HFC_rescom_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link HFC emissions from residential & commercial refrigeration
    & AC to reduction efficiency of abatement technologies.
  group: emission_factor
HFCequivOther:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Counter for the HFC-134 equiv. emissions from solvents, fire extinguishers,
    aerosols MDI, aerosols non-MDI.
  group: emission_factor
hydro_min:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint limiting minimum hydro production
  group: hydro resource
hydro_pot:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Constraint limiting maximum hydro potential
  group: hydro resource
IndGDPAdLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link of GDP to adipic acid
  group: emission_factor
IndGDPNiLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
```

(continues on next page)

(continued from previous page)

```
    link of GDP to nitric acid
  group: emission_factor
IndThermDemLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint links industry waste water CH4 emissions to its driver: the
    industrial themal demand.
  group: emission_factor
lfil_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint limiting gr of landfill mitigaiton technology activty (mpa)
  group: emission_factor
lim_exp_afr_coal:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on coal exports
  group: share-constraint formulation
lim_exp_afr_eth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on hydrogen exports
  group: share-constraint formulation
lim_exp_afr_LNG:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11_GLB only - limits LNG exports from afr
  group: share-constraint formulation
lim_exp_afr_meth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on methanol (synthetic fossil fuel) exports
  group: share-constraint formulation
lim_exp_cpa_oil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on oil exports
  group: share-constraint formulation
lim_exp_eeu_eth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on ethanol (biofuel) exports
  group: share-constraint formulation
lim_exp_eeu_meth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on methanol (synthetic fossil fuel) exports
  group: share-constraint formulation
lim_exp_fsu_coal:
  technology-entry: 1
  parameters: activity, upper
  description: >-
```

(continues on next page)

(continued from previous page)

```
    limit on coal exports
  group: share-constraint formulation
lim_exp_fsu_eth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on ethanol (biofuel) exports
  group: share-constraint formulation
lim_exp_fsu_LNG:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11_GLB only - limits LNG exports from fsu
  group: share-constraint formulation
lim_exp_fsu_meth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on methanol (synthetic fossil fuel) exports
  group: share-constraint formulation
lim_exp_fsu_oil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on oil exports
  group: share-constraint formulation
lim_exp_lam_LNG:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11_GLB only - limits LNG exports from lam
  group: share-constraint formulation
lim_exp_lam_oil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on oil exports
  group: share-constraint formulation
lim_exp_mea_LNG:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11_GLB only - limits LNG exports from mea
  group: share-constraint formulation
lim_exp_mea_oil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on oil exports
  group: share-constraint formulation
lim_exp_nam_coal:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on coal exports
  group: share-constraint formulation
lim_exp_nam_LNG:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11_GLB only - limits LNG exports from nam
```

(continues on next page)

(continued from previous page)

```
  group: share-constraint formulation
lim_exp_pao_coal:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on coal exports
  group: share-constraint formulation
lim_exp_pao_eth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on ethanol (biofuel) exports
  group: share-constraint formulation
lim_exp_pao_meth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on methanol (synthetic fossil fuel) exports
  group: share-constraint formulation
lim_exp_pas_LNG:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11_GLB only - limits LNG exports from pas
  group: share-constraint formulation
lim_exp_weu_eth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on ethanol (biofuel) exports
  group: share-constraint formulation
lim_exp_weu_meth:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on methanol (synthetic fossil fuel) exports
  group: share-constraint formulation
limit_tot_BECCS:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    limit on BECCS application
  group: share-constraint formulation
max_coal_ind:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    R11-NAM only - limits coal to max of 50% of gas in industries
  group: share-constraint formulation
meth_exp_limit:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    limits methanol exports as share of total methanol production
  group: share-constraint formulation
min-liquids_industry:
  technology-entry: 1
  parameters:  activity, lower
  group: share-constraint formulation
min-liquids_res_com:
  technology-entry: 1
```

(continues on next page)

(continued from previous page)

```
  parameters: activity, lower
  group: share-constraint formulation
N2O_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of N2O (weight as N2O, kg N2O per kWyr)
  group: emission_factor
N2O_Emission_bunkers:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Counter - R11-GLB only - N2O emissions of international shipping
  group: emission_factor
N2O_nonenergy:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of N2O from nonenergy sources (weight as N2O, kg N2O per kWyr)
  group: emission_factor
N2On_adipic_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link adipic acid emissions to reduction efficiency of N2O
    abatement technologies.
  group: emission_factor
N2On_nitric_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link nitric acid emissions to reduction efficiency of N2O
    abatement technologies.
  group: emission_factor
N2Oother:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Counter for the N2O emissions from Manure Management, Human Sewage, Other
    Agricultural sources, Other Non Ag Sources.
  group: emission_factor
NH3_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of NH3 (weight as NH3, kg NH3 per kWyr)
  group: emission_factor
nica_limit:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Constraint limiting gr of catalytic converter (mitigaiton) technology
    activty (mpa).
  group: emission_factor
NOx_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of NOx (weight as NOx, kg NOx per kWyr)
  group: emission_factor
NOx_nonenergy:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity, lower
  description: >-
    NOx emission from non-energy sectors
  group: emission_factor
OCA_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of OCA (weight as OCA, kg OCA per kWyr)
  group: emission_factor
oil_based_elec_gen:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constrains/calibrates historical minimum share of oil for electricity
    generation.
  group: hisc
oil_extr_mpen_c:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link equation for joint diffusion constraint of oil extraction technologies
  group: fossil resources
oil_imp_c:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11-NAM only - Limits import dependence of crude oil in relation to oil
    extraction.
  group: share-constraint formulation
oper_res:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    operating reserve equation for electricity systems integration
  group: renewable formulation
pass_out_trb:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    link to use of passout turbine from thermal generators
  group: add-on
PE_domestic_total:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    primary energy accounting equation
  group: obsolete (?)
PE_export_total:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    primary energy accounting equation
  group: obsolete (?)
PE_import_share:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    primary energy accounting equation
  group: obsolete (?)
```

(continues on next page)

(continued from previous page)

```
PE_import_total:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    primary energy accounting equation
  group: obsolete (?)
PM2_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    PM2.5 emission
  group: emission_factor
POPtoAeroNonMDILink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to aerosol non-medical use (driver for F gas emission)
  group: emission_factor
POPtoAerosolMDILink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to aerosol medical use (driver for F gas emission)
  group: emission_factor
POPtoDomWasteWaLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to domestic waste water
  group: emission_factor
POPtoFireExtLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to fire extiguisher (driver for F gas emission)
  group: emission_factor
POPtoHuSewageLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to human sewage
  group: emission_factor
POPtoSolvent:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to solvent use
  group: emission_factor
res_japan_limit:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    limit on renewable resource use of Japan in PAO region
  group: fossil resources
res_marg:
  parameters: activity, new_capacity, total_capacity
  technology-entry: 3
  parameters2: lower
  description: >-
    electricity systems integration - reserve margin
  group: renewable formulation
```

(continues on next page)

(continued from previous page)

```
RES_variable_limt:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    variable renewable energy limit
  group: "share-constraint formulation, renewable formulation"
SF6_elec_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link SF6 emissions from electrical equipment to reduction
    efficiency of abatement technologies.
  group: emission_factor
SF6_Emission:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Emissions of SF6 (weight as SF6, kg SF6 per kWyr)
  group: emission_factor
SF6_mag_red:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Constraint to link SF6 emissions from electrical equipment to reduction
    efficiency of abatement technologies.
  group: emission_factor
sm1_ppl_res:
  technology-entry: 1
  parameters: total_capacity, lower, upper
  description: >-
    renewable formulation
sm3_ppl_res:
  technology-entry: 1
  parameters: total_capacity, lower, upper
  description: >-
    renewable formulation
SO2_elec:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of SO2 in ppl, counter for DESOX (formerly SO2_emi_el)
  group: emission_factor
SO2_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of SO2 (weight as SO2, kg SO2 per kWyr)
  group: emission_factor
SO2_import:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Import of of low sulfur coal (0.6% S)
  group: emission_factor
SO2_ind:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Counter - Emissions of CO2 in industrial sector
  group: obsolete (?)
SO2_r_c:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Counter - Emissions of CO2 in residential/commercial sector
  group: obsolete (?)
SO2_red_ref:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Limit on SO2 Emission reduction in refinery products
  group: emission_factor
SO2_red_synf:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Limit on SO2 Emission reduction in synfuel production
  group: emission_factor
solar_csp_pot:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot1:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot2:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot3:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot4:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot5:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot6:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_csp_pot7:
  technology-entry: 1
```

(continues on next page)

(continued from previous page)

```
  parameters: activity, upper
  description: >-
    CSP resource potential category
  group: renewable formulation
solar_curtailment_1:
  parameters: activity, total_capacity
  technology-entry: 2
  parameters2: upper
  description: >-
    electricity systems integration constraint (curtailment)
  group: renewable formulation
solar_curtailment_2:
  parameters: activity, total_capacity
  technology-entry: 2
  parameters2: upper
  description: >-
    electricity systems integration constraint (curtailment)
  group: renewable formulation
solar_curtailment_3:
  parameters: activity, total_capacity
  technology-entry: 2
  parameters2: upper
  description: >-
    electricity systems integration constraint (curtailment)
  group: renewable formulation
solar_pot:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_pot2:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_pot3:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_pot4:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_pot5:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_pot6:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
```

(continues on next page)

(continued from previous page)

```
solar_pot7:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_pot8:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar PV resource potential
  group: renewable formulation
solar_ppl_res:
  technology-entry: 1
  parameters: total_capacity, lower, upper
  group: renewable formulation
solar_res_cv:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    balances the solar_res and solar_cv technologies
  group: renewable formulation
solar_step:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    first step of solar PV generation is limited to a certain fraction of load
    (higher steps have worse oper and resm).
  group: renewable formulation
solar_step2:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    second step of solar PV generation is limited to a certain fraction of load
    (higher steps have worse oper and resm).
  group: renewable formulation
solar_step3:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    third step of solar PV generation is limited to a certain fraction of load
    (higher steps have worse oper and resm).
  group: renewable formulation
solar_th_pot:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    solar thermal potential
  group: renewable formulation
SolWaPOPLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from population to solid waste
  group: emission_factor
syn_liq_exp:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11-FSU only - constraints the export of synthetic liquids to x% of
    indigenous oil use.
```

(continues on next page)

(continued from previous page)

```
  group: share-constraint formulation
TCE_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    R11-only - total carbon-equivalent emission equation
  group: emission_factor
TCEl:
  technology-entry: 0
  parameters: lower
  group: obsolete (?)
UE_feedstock:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Feedstock share constraint - connects all relevant technologies
  group: share-constraint formulation
UE_feedstock_coal:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Feedstock share constraint - restrict contribution of coal to x % of
    feedstock demand.
  group: share-constraint formulation
UE_feedstock_foil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Feedstock share constraint - restrict contribution of liquid fuels to x % of
    feedstock demand.
  group: share-constraint formulation
UE_feedstock_gas:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Feedstock share constraint - restrict contribution of heavy fuel oil to x %
    of feedstock demand.
  group: share-constraint formulation
UE_feedstock_liquid:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Feedstock share constraint - restrict contribution of gaseous fuels to x %
    of feedstock demand.
  group: share-constraint formulation
UE_industry_sp:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    share constraint for useful energy industry specific
  group: share-constraint formulation
UE_industry_sp_fc:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for fuel cell (onsite FC) useful energy industry specific
  group: share-constraint formulation
UE_industry_sp_liquid:
  technology-entry: 1
  parameters: activity, upper
  description: >-
```

(continues on next page)

(continued from previous page)

```
    share constraint for liquids (onsite IC generator) useful energy industry
    specific.
  group: share-constraint formulation
UE_industry_sp_solar:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Industry Specific sahre constraint - restrict contribution of solar
    technologies to x % of industry thermal energy demand.
  group: share-constraint formulation
UE_industry_th:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    share constraint for useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_electric:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for electric useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_foil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for fuel oil useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_gas:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for gas useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_hydrogen:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for hydrogen useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_liquid:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for liquids useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_low_temp_heat:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for low temperature heat useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_solar:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    share constraint for solar thermal useful energy industry thermal
  group: share-constraint formulation
UE_industry_th_solid:
  technology-entry: 1
  parameters: activity, upper
```

(continues on next page)

(continued from previous page)

```
  description: >-
    share constraint for solids useful energy industry thermal
  group: share-constraint formulation
UE_res_comm_sp:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Residential&Commercial Specific share constraint  - connects all relevant
    technologies.
  group: share-constraint formulation
UE_res_comm_sp_fc:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Specific share constraint - restrict contribution of
    fuel cell technologies to x % of residential specific energy demand.
  group: share-constraint formulation
UE_res_comm_sp_solar:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Specific share constraint - restrict contribution of
    solar technologies to x % of residential specific energy demand.
  group: share-constraint formulation
UE_res_comm_th:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Residential&Commercial Thermal share constraint  - connects all relevant
    technologies.
  group: share-constraint formulation
UE_res_comm_th_biomass:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    biomass technologies to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_electric:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    electricity to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_foil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    heavy fuel oil to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_gas:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    gaseous fuels to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_heat:
  technology-entry: 1
```

(continues on next page)

(continued from previous page)

```
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    district heat to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_hp:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    heat pumps to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_hydrogen:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    hydrogen to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_liquid:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    liquid fuels to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_solar:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    solar technologies to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_res_comm_th_solids:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Residential&Commercial Thermal share constraint  - restrict contribution of
    solid fuels to x % of residential/commercial thermal energy demand.
  group: share-constraint formulation
UE_transport:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    Transport share constraint  - connects all relevant technologies
  group: share-constraint formulation
UE_transport_electric:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Transport share constraint  - restrict contribution of electric technologies
    to x % of transport energy demand.
  group: share-constraint formulation
UE_transport_electric_Minimum:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Transport share constraint  - restrict LOWER contribution of electric
    technologies to x % of transport energy demand.
  group: share-constraint formulation
UE_transport_fc:
```

(continues on next page)

(continued from previous page)

```
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Transport share constraint  - restrict contribution of fuel cell
    technologies to x % of transport energy demand.
  group: share-constraint formulation
UE_transport_foil:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Transport share constraint  - restrict contribution of heavy fuel oil to x %
    of transport energy demand.
  group: share-constraint formulation
UE_transport_gas:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Transport share constraint  - restrict contribution of gaseous fuels to x %
    of transport energy demand.
  group: share-constraint formulation
UE_transport_liquid:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    Transport share constraint  - restrict contribution of liquid fuels to x %
    of transport energy demand.
  group: share-constraint formulation
unabated_coal_lim:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    R11-PAO, R11-NAM, R11-WEU only - Limits unabated coal power plants to
    present share.
  group: share-constraint formulation
VOC_Emission:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    Emissions of VOC (weight as VOC, kg VOC per kWyr)
  group: emission_factor
VOC_nonenergy:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    VOC emission from non-energy sectors
  group: emission_factor
WasteGenToBCALink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to BC
  group: emission_factor
WasteGenToCH4Link:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to CH4
  group: emission_factor
WasteGenToCOLink:
  technology-entry: 1
  parameters: activity, lower, upper
```

(continues on next page)

(continued from previous page)

```
  description: >-
    link from waste generation to CO
  group: emission_factor
WasteGenToNOxLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to Nox
  group: emission_factor
WasteGenToOCALink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to OC
  group: emission_factor
WasteGenToPM2Link:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to PM2.5
  group: emission_factor
WasteGenToSO2Link:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to SO2
  group: emission_factor
WasteGenToVOCLink:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    link from waste generation to VOC
  group: emission_factor
weight_ind:
  technology-entry: 1
  parameters: activity, cost
  description: >-
    Weight factor representing costs, inconvenience, etc  in lieu of end-use
    technology costs in industries.
weight_rc:
  technology-entry: 1
  parameters: activity, cost
  description: >-
    inconvenience costs for residential and commercial sector
weight_trp:
  technology-entry: 1
  parameters: activity, cost
  description: >-
    inconvenience costs for transport sector
wind_curtailment_1:
  parameters: activity, total_capacity
  technology-entry: 2
  parameters2: upper
  description: >-
    electricity systems integration constraint (curtailment)
  group: renewable formulation
wind_curtailment_2:
  parameters: activity, total_capacity
  technology-entry: 2
  parameters2: upper
  description: >-
```

(continues on next page)

(continued from previous page)

```
    electricity systems integration constraint (curtailment)
  group: renewable formulation
wind_curtailment_3:
  parameters: activity, total_capacity
  technology-entry: 2
  parameters2: upper
  description: >-
    electricity systems integration constraint (curtailment)
  group: renewable formulation
wind_pof:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    offshore wind resource potential
  group: renewable formulation
wind_pof2:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    offshore wind resource potential
  group: renewable formulation
wind_pof3:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    offshore wind resource potential
  group: renewable formulation
wind_pof4:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    offshore wind resource potential
  group: renewable formulation
wind_pof5:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    offshore wind resource potential
  group: renewable formulation
wind_pot:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    onshore wind resource potential
  group: renewable formulation
wind_pot2:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    onshore wind resource potential
  group: renewable formulation
wind_pot3:
  technology-entry: 1
  parameters: activity, upper
  description: >-
    onshore wind resource potential
  group: renewable formulation
wind_pot4:
  technology-entry: 1
  parameters: activity, upper
  description: >-
```

(continues on next page)

(continued from previous page)

```
    onshore wind resource potential
  group: renewable formulation
wind_ppl_offshore_res:
  technology-entry: 1
  parameters: total_capacity, lower, upper
  group: renewable formulation
wind_ppl_res:
  technology-entry: 1
  parameters: total_capacity, lower, upper
  group: renewable formulation
wind_res_cv:
  technology-entry: 1
  parameters: activity, lower, upper
  description: >-
    balances the wind_res and wind_cv technologies
  group: renewable formulation
wind_step:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    first step of wind generation is limited to a certain fraction of load
    (higher steps have worse oper and resm).
  group: renewable formulation
wind_step2:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    second step of wind generation is limited to a certain fraction of load
    (higher steps have worse oper and resm).
  group: renewable formulation
wind_step3:
  technology-entry: 1
  parameters: activity, lower
  description: >-
    third step of wind generation is limited to a certain fraction of load
    (higher steps have worse oper and resm).
  group: renewable formulation
```

8.22 Years or time periods (`year/*.yaml`)

- See also:
  - The discussion of Years, periods, and time slices in the `message_ix` documentation, which explains the standard sense of time periods used across the MESSAGEix framework and specific models based on it.
  - *`ScenarioInfo.year_from_codes()`*
- These are not the only possible meanings of these codes; others may be used in data from other sources.

  For instance, the ID `2020` could be used to represent the period from 2017-07-01 to 2022-06-30. These lists alone cannot resolve these differences; they exist only to provide clarity about the sense used in *`message_ix_models`*.
- When working with data from other sources, *`message_ix_models`* code **must**:
  - Explicitly note (e.g. in comments or docstrings) any differing time discretization used in the other data.
  - Perform appropriate conversion *or* record a decision to use the data directly, without conversion.
- It is **optional** for code to fill Scenario parameters for the full set of historical years.

For instance, when working with list `B`, code for a model variant or project could only populate parameter values for the historical periods `2010` and `2015`, but not `2005` and earlier. This **should** be described and documented on at the scope (function or module) where such subsets are selected from the full codelist.

8.22.1 List `A`

```
# Time periods used in CD-LINKS, inter alia

1960:
  description: Period from 1951-01-01 to 1960-12-31.
  # Durations of subsequent periods are implied by the prior period
  duration_period: 10

1970:
  description: Period from 1961-01-01 to 1970-12-31.
1980:
  description: Period from 1971-01-01 to 1980-12-31.
1990:
  description: Period from 1981-01-01 to 1990-12-31.
2000:
  description: Period from 1991-01-01 to 2000-12-31.
2010:
  description: Period from 2001-01-01 to 2010-12-31.

2020:
  description: Period from 2011-01-01 to 2020-12-31.
  # Periods before this are historical
  firstmodelyear: true

2030:
  description: Period from 2021-01-01 to 2030-12-31.
2040:
  description: Period from 2031-01-01 to 2040-12-31.
2050:
  description: Period from 2041-01-01 to 2050-12-31.
2060:
  description: Period from 2051-01-01 to 2060-12-31.
2070:
  description: Period from 2061-01-01 to 2070-12-31.
2080:
  description: Period from 2071-01-01 to 2080-12-31.
2090:
  description: Period from 2081-01-01 to 2090-12-31.
2100:
  description: Period from 2091-01-01 to 2100-12-31.
2110:
  description: Period from 2101-01-01 to 2110-12-31.
```

8.22.2 List `B`

```
# Time periods used in ENGAGE, inter alia

1950:
  description: Period from 1946-01-01 to 1950-12-31.
  # Durations of subsequent periods are implied by the prior period
  duration_period: 5
1955:
  description: Period from 1951-01-01 to 1955-12-31.
1960:
```

(continues on next page)

(continued from previous page)

```
  description: Period from 1956-01-01 to 1960-12-31.
1965:
  description: Period from 1961-01-01 to 1965-12-31.
1970:
  description: Period from 1966-01-01 to 1970-12-31.
1975:
  description: Period from 1971-01-01 to 1975-12-31.
1980:
  description: Period from 1976-01-01 to 1980-12-31.
1985:
  description: Period from 1981-01-01 to 1985-12-31.
1990:
  description: Period from 1986-01-01 to 1990-12-31.
1995:
  description: Period from 1991-01-01 to 1995-12-31.
2000:
  description: Period from 1996-01-01 to 2000-12-31.
2005:
  description: Period from 2001-01-01 to 2005-12-31.
2010:
  description: Period from 2006-01-01 to 2010-12-31.
2015:
  description: Period from 2011-01-01 to 2015-12-31.

2020:
  description: Period from 2016-01-01 to 2020-12-31.
  # Periods before this are historical
  firstmodelyear: true

2025:
  description: Period from 2021-01-01 to 2025-12-31.
2030:
  description: Period from 2026-01-01 to 2030-12-31.
2035:
  description: Period from 2031-01-01 to 2035-12-31.
2040:
  description: Period from 2036-01-01 to 2040-12-31.
2045:
  description: Period from 2041-01-01 to 2045-12-31.
2050:
  description: Period from 2046-01-01 to 2050-12-31.
2055:
  description: Period from 2046-01-01 to 2055-12-31.
2060:
  description: Period from 2056-01-01 to 2060-12-31.
2070:
  description: Period from 2061-01-01 to 2070-12-31.
2080:
  description: Period from 2071-01-01 to 2080-12-31.
2090:
  description: Period from 2081-01-01 to 2090-12-31.
2100:
  description: Period from 2091-01-01 to 2100-12-31.
2110:
  description: Period from 2101-01-01 to 2110-12-31.
```

8.23 Other code lists

These codelists correspond to sets in the generic MESSAGE IAM formulation with the same names.

8.23.1 Commodities (`commodity.yaml`)

These codes have the following annotations:

`level` (mandatory)
Level where this commodity typically (not exclusively) occurs.

`units` (mandatory)
Units typically associated with this commodity.

`iea-eweb-flow` (optional)
List of `FLOW` codes from the IEA *(Extended) World Energy Balances (tools.iea.web)* associated with this MESSAGEix-GLOBIOM commodity.

`iea-eweb-product` (optional)
List of `PRODUCT` codes from the IEA *(Extended) World Energy Balances (tools.iea.web)* associated with this MESSAGEix-GLOBIOM commodity.

```yaml
biomass:
  units: GWa
  report: Solids|Biomass
  ipcc-1996-name: "Solid Biomass"

coal:
  name: Coal
  units: GWa
  report: Solids|Fossil
  # NB same value as "Coking coal"; this choice is arbitrary
  # NB in message_doc, this appears as "Hard coal", but this
  #    term usually refers to anthracite, which has a higher
  #    carbon emission factor.
  ipcc-1996-name: "Other Bituminous Coal"
  iea-eweb-product:
  - ANTCOAL
  - BITCOAL
  - BKB
  - BLFURGS
  - BROWN
  - COALTAR
  - COKCOAL
  - COKEOVGS
  - GASCOKE
  - GASWKSGS
  - HARDCOAL
  - INDWASTE
  - LIGNITE
  - MANGAS
  - MUNWASTEN
```

(continues on next page)

(continued from previous page)

```
  - OVENCOKE
  - PATFUEL
  - PEAT
  - SUBCOAL

crudeoil:
  name: Crude oil
  description: >-
    For secondary energy, use 'fueloil', 'lightoil', etc.
  level: primary
  units: GWa
  report: Oil
  ipcc-1996-name: "Crude Oil"

d_heat:
  name: (?) District heat
  description: >-
    FIXME provide an unambiguous description of what this commodity represents.
  report: Heat

electr:
  name: Electricity
  units: GWa
  report: Electricity
  iea-eweb-product: [ELECTR]

ethanol:
  name: Ethanol
  units: GWa
  report: Liquids|Biomass
  iea-eweb-product: [BIODIESEL, BIOGASOL, BIOJETKERO, OBIOLIQ]

freshwater_supply:
  name: (?) Fresh water
  description: >-
    FIXME provide an unambiguous description of what this commodity represents.

fueloil:
  name: Fuel oil
  description: Heavy fuel oil
  level: secondary
  units: GWa
  ipcc-1996-name: "Residual Fuel Oil"
  iea-eweb-product: [BITUMEN, PARWAX, PETCOKE, RESFUEL]

gas:
  name: Natural Gas
  units: GWa
  report: Gases
  ipcc-1996-name: "Natural Gas (Dry)"
  iea-eweb-product: [NATGAS]

hydrogen:
  name: Gaseous hydrogen
  units: GWa

lh2:
  name: Liquid hydrogen
  units: GWa

lightoil:
```

(continues on next page)

(continued from previous page)

```
  name: Light oil
  description: Includes gasoline, diesel oil.
  # level: secondary
  units: GWa
  report: Liquids|Oil
  # NB same value as "Crude Oil" and several others; this choice is arbitrary
  ipcc-1996-name: "Other Oil"
  iea-eweb-product:
  - AVGAS
  - ETHANE
  - JETGAS
  - LPG
  - LUBRIC
  - NAPHTHA
  - NONBIODIES
  - NONBIOGASO
  - NONBIOJETK
  - ONONSPEC
  - OTHKERO
  - REFINGAS
  - WHITESP

lignite:
  name: Lignite
  ipcc-1996-name: "Lignite"

methanol:
  name: Methanol
  units: GWa
  report: Liquids|Coal
  # This does not appear in the referenced source; value is 17.4
  # ipcc-1996-name: MISSING

non-comm:
  name: Non-commercial biomass
  units: GWa
  report: Solids|Biomass|Traditional

rc_spec:
  name: Residential and commercial non-substitutable fuels

rc_therm:
  name: Residential and commercial thermal

transport:
  name: Transportation
  description: >-
    For MESSAGEix-Transport, this commodity is not used; it is replaced by a
    disaggregated set of transport service demands (representing e.g. light-
    duty vehicles, civil aviation, freight transport, etc.)
  level: useful
  units: GWa
  iea-eweb-flow: [DOMESAIR, DOMESNAV, RAIL, ROAD, TRNONSPE]

# The following codes also appear in a recent (2020-02-28) SSP2 scenario, but
# are not currently used by model.bare.create_res.
#
# Aff_CO2_G4M
# Agri_CH4
# Agri_N2O
# Agri_N2O_calc
```

(continues on next page)

(continued from previous page)

```
# Agricultural Demand
# Agricultural Demand|Bioenergy
# Agricultural Demand|Bioenergy|1st generation
# Agricultural Demand|Bioenergy|2nd generation
# Agricultural Demand|Feed
# Agricultural Demand|Feed|Crops
# Agricultural Demand|Food
# Agricultural Demand|Food|Crops
# Agricultural Demand|Food|Livestock
# Agricultural Demand|Non-Food
# Agricultural Demand|Non-Food|Crops
# Agricultural Demand|Non-Food|Livestock
# Agricultural Production
# Agricultural Production|Energy Crops
# Agricultural Production|Livestock
# Agricultural Production|Non-Energy Crops
# Agricultural Production|Non-Energy Crops|Cereals
# BCA_LandUseChangeEM
# BCA_SavanBurnEM
# Biodiesel_G1
# bioenergy
# Bioethanol_G1
# CalAnim
# CalCrop
# CalTot
# CH4_LandUseChangeEM
# CH4_SavanBurnEM
# CO_LandUseChangeEM
# CO_SavanBurnEM
# CO2_oil
# CO2_rem
# cooling__bio_hpl
# cooling__bio_istig
# cooling__bio_istig_ccs
# cooling__bio_ppl
# cooling__coal_adv
# cooling__coal_adv_ccs
# cooling__coal_ppl
# cooling__coal_ppl_u
# cooling__foil_hpl
# cooling__foil_ppl
# cooling__gas_cc
# cooling__gas_cc_ccs
# cooling__gas_hpl
# cooling__gas_ppl
# cooling__geo_hpl
# cooling__geo_ppl
# cooling__igcc
# cooling__igcc_ccs
# cooling__loil_cc
# cooling__loil_ppl
# cooling__nuc_hc
# cooling__nuc_lc
# cooling__solar_th_ppl
# CrpLnd
# crude_1
# crude_2
# crude_3
# crude_4
# crude_5
# crude_6
```

(continues on next page)

(continued from previous page)

```
# crude_7
# crude_8
# Def_CO2_G4M
# Def_CO2_GLO
# dumagr
# dumfert
# Emissions|CH4|Land Use
# Emissions|CH4|Land Use|Agricultural Waste Burning
# Emissions|CH4|Land Use|Agriculture
# Emissions|CH4|Land Use|Agriculture|AWM
# Emissions|CH4|Land Use|Agriculture|Enteric Fermentation
# Emissions|CH4|Land Use|Agriculture|Rice
# Emissions|CH4|Land Use|Savannah Burning
# Emissions|CO2|Land Use
# Emissions|CO2|Land Use|Negative
# Emissions|CO2|Land Use|Positive
# Emissions|N2O|Land Use
# Emissions|N2O|Land Use|Agricultural Waste Burning
# Emissions|N2O|Land Use|Agriculture
# Emissions|N2O|Land Use|Agriculture|AWM
# Emissions|N2O|Land Use|Agriculture|Cropland Soils
# Emissions|N2O|Land Use|Agriculture|Pasture
# Emissions|N2O|Land Use|Savannah Burning
# EnergyRoundwood
# exports
# Fertilizer Use|Nitrogen
# Fertilizer Use|Phosphorus
# Fmg_CO2_G4M
# Food Demand
# Food Demand|Crops
# Food Demand|Livestock
# Food Energy Demand
# Food Energy Demand|Livestock
# ForestBiomass
# ForestBiomass_G4M
# ForestHarvestDF_G4M
# ForestHarvestFM_G4M
# ForestHarvestTot_G4M
# ForestHarvestTot_GLO
# Forestry Demand|Roundwood
# Forestry Demand|Roundwood|Industrial Roundwood
# Forestry Demand|Roundwood|Wood Fuel
# Forestry Production|Forest Residues
# Forestry Production|Roundwood
# Forestry Production|Roundwood|Industrial Roundwood
# Forestry Production|Roundwood|Wood Fuel
# freshwater_instream
# FuelWood
# FuelWood_G4M
# gas
# gas_1
# gas_2
# gas_3
# gas_4
# gas_5
# gas_6
# gas_7
# gas_8
# gas_afr
# gas_cpa
# gas_eeu
```

(continues on next page)

(continued from previous page)

```
# gas_nam
# gas_pao
# gas_pas
# gas_sas
# gas_weu
# GrsLnd
# i_feed
# i_spec
# i_therm
# IrriWithdrawal
# Land Cover
# Land Cover|Cropland
# Land Cover|Cropland|Cereals
# Land Cover|Cropland|Energy Crops
# Land Cover|Cropland|Irrigated
# Land Cover|Forest
# Land Cover|Forest|Afforestation and Reforestation
# Land Cover|Forest|Forestry
# Land Cover|Forest|Managed
# Land Cover|Forest|Natural Forest
# Land Cover|Other Natural Land
# Land Cover|Pasture
# LiquidTotal
# LNG
# LoggingResidues
# LU_CO2
# LU_GHG
# LucGrs_CO2
# LucOth_CO2
# NewFor_G4M
# NH3_LandUseChangeEM
# NH3_ManureEM
# NH3_RiceEM
# NH3_SavanBurnEM
# NH3_SoilEM
# NOx_LandUseChangeEM
# NOx_SavanBurnEM
# NOx_SoilEM
# nucfuel
# OCA_LandUseChangeEM
# OCA_SavanBurnEM
# oil_st
# Olc_CO2_GLO
# Olc_CO2_GLO_neg
# Olc_CO2_GLO_pos
# OldFor_G4M
# OtherLnd
# OthSolidNonComm
# PlantationBiomass
# PlantationHarvest_GLO
# PltArt
# PltFor
# PltFor_gr
# Price_BIO
# Price_CO2
# Price|Agriculture|Non-Energy Crops and Livestock|Index
# Price|Agriculture|Non-Energy Crops|Index
# Price|Primary Energy|Biomass
# pu
# puq
# puq2
```

(continues on next page)

(continued from previous page)

```
# saline_supply
# SawmillResidues
# SawmillResidues_G4M
# shipping
# SO2_LandUseChangeEM
# SO2_SavanBurnEM
# SolidExogenous
# SolidExogenous_G4M
# SolidTotal
# SolidTotal_G4M
# TCE
# TimberIndust
# TimberIndust_G4M
# Tot_CO2_G4M
# total_cost
# TotalLnd
# TotFor_G4M
# u5
# u5q
# u5t
# upstream_landuse
# uq
# uranium
# VOC_LandUseChangeEM
# VOC_SavanBurnEM
# water_constraint
# Water|Withdrawal|Irrigation
# Yield|Cereal
# Yield|Oilcrops
# Yield|Sugarcrops
```

8.23.2 Levels (`level.yaml`)

This code list has no annotations and no hierarchy.

```
primary:
  name: Primary Energy
  description: >-
    A form found in nature that has not been subjected to any human engineered
    conversion process.

secondary:
  name: Secondary Energy
  description: Forms which have been transformed from primary energy.

final:
  name: Final Energy
  description: >-
    Represents end-use demands of the end-use sectors (e.g. industry, transport,
    residential, commercial and agriculture).

import:
  name: Imports

useful:
  name: Useful Energy
  description: >-
    Represents energy-service demands (or activity levels) of the end-use
    sectors in non-energy units.
```

(continues on next page)

(continued from previous page)

```
water_supply:
  name: Water Supply
  description: >-
    FIXME provide an unambiguous description of what this level represents.

# The following codes also appear in a recent (2020-02-28) SSP2 scenario, but
# are not currently used by model.bare.create_res.
#
# cooling
# export
# land_use_reporting
# land_use
# piped-gas
# resource
# stocks
# water_supply_constraint
```

8.23.3 Technologies (`technology.yaml`)

Warning: This list is *only for reference*; particular MESSAGE-GLOBIOM scenarios may not contain all these technologies, or may contain other technologies not listed.

Each of these codes has the following annotations:

`sector`
: A categorization of the technology.

`input`
: (`commodity`, `level`) for input to the technology.

`output`
: (`commodity`, `level`) for output from the technology.

`vintaged`
: `True` if the technology is subject to vintaging.

`type`
: Same as `output[1]`.

```
# This file describes a possible set of base technologies to be used in the
# global model. It will be usable by both model creation and reporting code.
#
# Each entry includes the following fields:
# - name, description: required.
# - sector: a label to group multiple technologies to a notional sector.
#   Required.
# - output (required) and input (optional): either
#   - a list of [commodity, level] giving the output generated or input used by
#     the technology; or,
#   - a list of 2 or more such lists, if the technology has multiple inputs or
#     outputs.
# - vintaged: True if the technology's properties vary by year_vintage.
#   Optional; False if omitted.
# - type: In the Excel file used to create this YAML file (see
#   https://github.com/iiasa/message_data/issues/74), 'type' appears to be
#   always the same as 'output'/level; *unless* the 'output'/commodity is a
#   dummy commodity, in which case it is 'dummy'.
```

(continues on next page)

(continued from previous page)

```
#
#   TODO if this is the case, remove 'type' from this file, and generate 'type'
#        in tools.technologies.get_info.

CF4_TCE:
  name: CF4_TCE
  description: Tetrafluoromethane (CF4) Total Carbon Emissions
  type: primary
  sector: dummy
  output: [dummy, primary]

CH4_TCE:
  name: CH4_TCE
  description: Methane total carbon equivalent emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

CH4g_TCE:
  name: CH4g_TCE
  description: CH4 emissions from animals directly in Total Carbon Equivalent␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy agriculture, primary]

CH4n_TCE:
  name: CH4n_TCE
  description: CH4 emissions from anaerobic waste decomposition in Total Carbon␣
↪Equivalent emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

CH4o_TCE:
  name: CH4o_TCE
  description: Dummy technology converting CH4 emissions from industrial and␣
↪domestic wastewater, non energy biomass burning and other CH4 emissions, to␣
↪total carbon equivalent emissions (TCE)
  type: dummy
  sector: dummy
  output: [dummy, primary]

CO2_TCE:
  name: CO2_TCE
  description: CO2 total carbon equivalent emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

dom_total:
  name: dom_total
  description: Used in balance equation for domestic energy supply and in the␣
↪equation for constraining net imports
  type: secondary
  sector: dummy
  output: [exports, secondary]

dummy_producer:
  name: dummy_producer
  description: Technology added to produce dummy energy to avoid infeasibility if␣
```

(continues on next page)

(continued from previous page)

```
↪needed (e.g. when CO2_TCE needs to go negative)
  type: primary
  sector: dummy
  output: [dummy, primary]

exp_total:
  name: exp_total
  description: Used in balance equation for exported energy supply and in the␣
↪equation for constraining net imports
  type: secondary
  sector: dummy
  output: [exports, secondary]

HFC_TCE:
  name: HFC_TCE
  description: HFC total carbon equivalent emissions
  type: primary
  sector: dummy
  output: [dummy, primary]

HFCo_TCE:
  name: HFCo_TCE
  description: Dummy technology converting HFC equiv emissions from solvents, fire␣
↪extinguishers, aerosols MDI, aerosols non-MDI to total carbon equivalent␣
↪emissions (TCE)
  type: primary
  sector: dummy
  output: [dummy, primary]

imp_total:
  name: imp_total
  description: Used in balance equation for imported energy supply
  type: exports
  sector: dummy
  output: [exports, secondary]

N2O_TCE:
  name: N2O_TCE
  description: N2O total carbon equivalent emissions
  type: dummy
  sector: dummy
  output: [dummy]

N2OG_TCE:
  name: N2OG_TCE
  description: N2O soil emissions total carbon equivalent emissions
  type: dummy
  sector: dummy
  output: [dummy]

N2On_TCE:
  name: N2On_TCE
  description: N2O adipic acid total carbon equivalent emissions
  type: dummy
  sector: dummy
  output: [dummy]

N2Oo_TCE:
  name: N2Oo_TCE
  description: Dummy technology converting N2O emissions from Manure Management,␣
↪Human Sewage, Other Agricultural sources, Other Non Ag Sources to total carbon␣
```

(continues on next page)

(continued from previous page)

```
↪equivalent emissions (TCE)
  type: dummy
  sector: dummy
  output: [dummy]

nica_con:
  name: nica_con
  type: dummy
  sector: dummy
  output: [dummy]

nitric_catalytic1:
  name: nitric_catalytic1
  description: Mitigation technology (catalytic converter) category 2 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

nitric_catalytic2:
  name: nitric_catalytic2
  description: Mitigation technology (catalytic converter) category 1 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

nitric_catalytic3:
  name: nitric_catalytic3
  description: Mitigation technology (catalytic converter) category 3 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

nitric_catalytic4:
  name: nitric_catalytic4
  description: Mitigation technology (catalytic converter) category 4 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

nitric_catalytic5:
  name: nitric_catalytic5
  description: Mitigation technology (catalytic converter) category 5 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

nitric_catalytic6:
  name: nitric_catalytic6
  description: Mitigation technology (catalytic converter) category 6 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

nitric_catalytic7:
  name: nitric_catalytic7
```

(continues on next page)

(continued from previous page)

```
  description: Mitigation technology (catalytic converter) category 7 for N2O␣
↪emissions
  type: dummy
  sector: dummy
  output: [dummy, primary]

SF6_TCE:
  name: SF6_TCE
  description: SF6 total carbon equivalent emissions
  type: primary
  sector: dummy
  output: [dummy, primary]

useful_feedstock:
  name: useful_feedstock
  description: Share constraint for industry feedstocks
  type: dummy
  sector: dummy
  output: [dummy_useful, primary]

useful_industry_sp:
  name: useful_industry_sp
  description: Share constraint for Industry Specific
  type: dummy
  sector: dummy
  output: [dummy_useful, primary]

useful_industry_th:
  name: useful_industry_th
  description: Share constraint for Industry Thermal
  type: dummy
  sector: dummy
  output: [dummy_useful, primary]

useful_res/comm_sp:
  name: useful_res/comm_sp
  description: Share constraint for Residential and Commercial Specific
  type: dummy
  sector: dummy
  output: [dummy_useful, primary]

useful_res/comm_th:
  name: useful_res/comm_th
  description: Share constraint for Residential and Commercial Thermal
  type: dummy
  sector: dummy
  output: [dummy_useful, primary]

useful_transport:
  name: useful_transport
  description: Share constraint for Transport
  type: dummy
  sector: dummy
  output: [dummy_useful, primary]

bio_istig:
  name: bio_istig
  description: Advanced biomass power plant- gasified biomass is burned in gas␣
↪turbine plant - modes with and without net carbon release
  type: secondary
  vintaged: TRUE
```

(continues on next page)

(continued from previous page)

```
  sector: electricity
  input: [biomass, primary]
  output: [electr, secondary]

bio_istig_ccs:
  name: bio_istig_ccs
  description: Advanced biomass power plant with carbon capture and storage-␣
↪gasified biomass is burned in gas turbine plant - modes with and without net␣
↪carbon release
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [biomass, primary]
  output: [electr, secondary]

bio_ppl:
  name: bio_ppl
  description: Bio powerplant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input:
    - [biomass, primary]
    - [cooling__bio_ppl, cooling]
    - [freshwater_supply, water_supply]
  output: [electr, secondary]

coal_adv:
  name: coal_adv
  description: Advanced coal power plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [coal, secondary]
  output: [electr, secondary]

coal_adv_ccs:
  name: coal_adv_ccs
  description: Advanced coal power plant with carbon capture and storage
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [coal, secondary]
  output: [electr, secondary]

coal_ppl:
  name: coal_ppl
  description: Coal power-plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [coal, secondary]
  output: [electr, secondary]

coal_ppl_u:
  name: coal_ppl_u
  description: Coal power plant without abatement measures
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [coal, secondary]
```

(continues on next page)

(continued from previous page)

```
  output: [electr, secondary]

elec_exp:
  name: elec_exp
  description: Net export of electricity
  type: exports
  sector: electricity
  input: [electr, secondary]
  output: [electr, exports]

elec_imp:
  name: elec_imp
  description: Net import of electricity
  type: secondary
  sector: electricity
  input: [electr, imports]
  output: [electr, secondary]

elec_t/d:
  name: elec_t/d
  description: Grid technology cost converted to 2005$
  type: final
  vintaged: TRUE
  sector: electricity
  input: [electr, secondary]
  output: [electr, final]

foil_ppl:
  name: foil_ppl
  description: New standard oil power plant, Rankine cycle
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [fueloil, secondary]
  output: [electr, secondary]

gas_cc:
  name: gas_cc
  description: Gas combined cycle power-plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [gas, secondary]
  output: [electr, secondary]

gas_cc_ccs:
  name: gas_cc_ccs
  description: Gas combined cycle power-plant with carbon capture and storage
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [gas, secondary]
  output: [electr, secondary]

gas_ct:
  name: gas_ct
  description: Gas combustion-turbine power plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [gas, secondary]
```

(continues on next page)

(continued from previous page)

```
  output: [electr, secondary]

gas_htfc:
  name: gas_htfc
  description: High temperature fuel cell powered with natural gas
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [gas, secondary]
  output: [electr, secondary]

gas_ppl:
  name: gas_ppl
  description: Gas power plant, Rankine cycle
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [gas, secondary]
  output: [electr, secondary]

geo_ppl:
  name: geo_ppl
  description: Geothermal power plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [electr, secondary]

glb_elec_exp:
  name: glb_elec_exp
  description: Global net export of electricity
  type: imports
  sector: electricity
  output: [electr, imports]

glb_elec_imp:
  name: glb_elec_imp
  description: Global net import of electricity
  type: exports
  sector: electricity
  input: [electr, exports]
  output: [exports]

hydro_hc:
  name: hydro_hc
  description: High cost hydro power plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [electr, secondary]

hydro_lc:
  name: hydro_lc
  description: Low cost hydro power plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [electr, secondary]

igcc:
  name: igcc
```

(continues on next page)

(continued from previous page)

```
  description: Integrated gasification combined cycle (IGCC) power plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [coal, secondary]
  output: [electr, secondary]

igcc_ccs:
  name: igcc_ccs
  description: Integrated gasification combined cycle (IGCC) power plant with␣
↪carbon capture and storage
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [coal, secondary]
  output: [electr, secondary]

igcc_co2scr:
  name: igcc_co2scr
  description: New coal scrubber for igcc plants
  type: exports
  vintaged: TRUE
  sector: electricity
  output: [exports, secondary]

loil_cc:
  name: loil_cc
  description: Light oil combined cycle
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [lightoil, secondary]
  output: [electr, secondary]

loil_ppl:
  name: loil_ppl
  description: Existing light oil power-plant
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [lightoil, secondary]
  output: [electr, secondary]

nuc_hc:
  name: nuc_hc
  description: Nuclear power plant (~GEN III+), high cost
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [uranium, stocks]
  output: [electr, secondary]

nuc_lc:
  name: nuc_lc
  description: Nuclear power plant (~GEN II), low cost
  type: secondary
  vintaged: TRUE
  sector: electricity
  input: [uranium, stocks]
  output: [electr, secondary]
```

(continues on next page)

(continued from previous page)

```
solar_curtailment1:
  name: solar_curtailment1
  description: Solar PV curtailment steps
  type: dummy
  sector: electricity
  input: [electr, secondary]
  output: [dummy renewable, secondary]

solar_curtailment2:
  name: solar_curtailment2
  description: Solar PV curtailment steps
  type: dummy
  sector: electricity
  input: [electr, secondary]
  output: [dummy renewable, secondary]

solar_curtailment3:
  name: solar_curtailment3
  description: Solar PV curtailment steps
  type: dummy
  sector: electricity
  input: [electr, secondary]
  output: [dummy renewable, secondary]

solar_cv1:
  name: solar_cv1
  description: Quadratic systems integration costs added to solar PV
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

solar_cv2:
  name: solar_cv2
  description: Quadratic systems integration costs added to solar PV
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

solar_cv3:
  name: solar_cv3
  description: Quadratic systems integration costs added to solar PV
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

solar_cv4:
  name: solar_cv4
  description: Quadratic systems integration costs added to solar PV
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

solar_pv_ppl:
  name: solar_pv_ppl
  description: Solar photovoltaic power plant (no storage)
  type: dummy
  vintaged: TRUE
  sector: electricity
  output: [dummy renewable, secondary]

solar_res1:
```

(continues on next page)

(continued from previous page)

```
  name: solar_res1
  description: Maximum solar electricity potential 1
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_res2:
  name: solar_res2
  description: Maximum solar electricity potential 2
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_res3:
  name: solar_res3
  description: Maximum solar electricity potential 3
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_res4:
  name: solar_res4
  description: Maximum solar electricity potential 4
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_res5:
  name: solar_res5
  description: Maximum solar electricity potential 5
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_res6:
  name: solar_res6
  description: Maximum solar electricity potential 6
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_res7:
  name: solar_res7
  description: Maximum solar electricity potential 7
  type: secondary
  sector: electricity
  output: [electr, secondary]

solar_th_ppl:
  name: solar_th_ppl
  description: Solar thermal power plant with storage
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [electr, secondary]

stor_ppl:
  name: stor_ppl
  description: Generic electric storage
  type: secondary
  vintaged: TRUE
```

(continues on next page)

(continued from previous page)

```
  sector: electricity
  input: [electr, secondary]
  output: [exports, secondary]

wind_curtailment1:
  name: wind_curtailment1
  description: Wind curtailment steps
  type: dummy
  sector: electricity
  input: [secondary, electricity]
  output: [dummy renewable, secondary]

wind_curtailment2:
  name: wind_curtailment2
  description: Wind curtailment steps
  type: dummy
  sector: electricity
  input: [secondary, electricity]
  output: [dummy renewable, secondary]

wind_curtailment3:
  name: wind_curtailment3
  description: Wind curtailment steps
  type: dummy
  sector: electricity
  input: [secondary, electricity]
  output: [dummy renewable, secondary]

wind_cv1:
  name: wind_cv1
  description: Wind flexibility requirement and firm capacity contribution,␣
↪quadratic systems integration costs added to wind
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

wind_cv2:
  name: wind_cv2
  description: Wind flexibility requirement and firm capacity contribution,␣
↪quadratic systems integration costs added to wind
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

wind_cv3:
  name: wind_cv3
  description: Wind flexibility requirement and firm capacity contribution,␣
↪quadratic systems integration costs added to wind
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

wind_cv4:
  name: wind_cv4
  description: Wind flexibility requirement and firm capacity contribution,␣
↪quadratic systems integration costs added to wind
  type: dummy
  sector: electricity
  output: [dummy renewable, secondary]

wind_ppl:
```

(continues on next page)

(continued from previous page)

```
  name: wind_ppl
  description: Wind power plant onshore (provides capacity only)
  type: dummy
  vintaged: TRUE
  sector: electricity
  output: [dummy renewable, secondary]

wind_res1:
  name: wind_res1
  description: Wind onshore potential and generation 1
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_res2:
  name: wind_res2
  description: Wind onshore potential and generation 2
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_res3:
  name: wind_res3
  description: Wind onshore potential and generation 3
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_res4:
  name: wind_res4
  description: Wind onshore potential and generation 4
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_ppf:
  name: wind_ppf
  description: Wind power plant offshore (provides capacity only)
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [dummy renewable, secondary]

wind_ref1:
  name: wind_ref1
  description: Wind offshore potential and generation 1
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_ref2:
  name: wind_ref2
  description: Wind offshore potential and generation 2
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_ref3:
  name: wind_ref3
  description: Wind offshore potential and generation 3
  type: secondary
```

(continues on next page)

(continued from previous page)

```
  sector: electricity
  output: [electr, secondary]

wind_ref4:
  name: wind_ref4
  description: Wind offshore potential and generation 4
  type: secondary
  sector: electricity
  output: [electr, secondary]

wind_ref5:
  name: wind_ref5
  description: Wind offshore potential and generation 5
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res:
  name: csp_sm3_res
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 1
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res1:
  name: csp_sm3_res1
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 2
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res2:
  name: csp_sm3_res2
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 3
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res3:
  name: csp_sm3_res3
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 4
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res4:
  name: csp_sm3_res4
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 5
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res5:
  name: csp_sm3_res5
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 6
```

(continues on next page)

(continued from previous page)

```
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res6:
  name: csp_sm3_res6
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 7
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_res7:
  name: csp_sm3_res7
  description: Concentrating solar power (CSP) with solar multiple of 3 potential␣
↪and generation 8
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm3_ppl:
  name: csp_sm3_ppl
  description: Concentrating solar power (CSP) with solar multiple of 3 (provides␣
↪capacity only)
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [dummy renewable, secondary]

csp_sm1_res:
  name: csp_sm1_res
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 1
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_res1:
  name: csp_sm1_res1
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 2
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_res2:
  name: csp_sm1_res2
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 3
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_res3:
  name: csp_sm1_res3
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 4
  type: secondary
  sector: electricity
  output: [electr, secondary]
```

(continues on next page)

(continued from previous page)

```
csp_sm1_res4:
  name: csp_sm1_res4
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 5
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_res5:
  name: csp_sm1_res5
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 6
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_res6:
  name: csp_sm1_res6
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 7
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_res7:
  name: csp_sm1_res7
  description: Concentrating solar power (CSP) with solar multiple of 1 potential␣
↪and generation 8
  type: secondary
  sector: electricity
  output: [electr, secondary]

csp_sm1_ppl:
  name: csp_sm1_ppl
  description: Concentrating solar power (CSP) with solar multiple of 1 (provides␣
↪capacity only)
  type: secondary
  vintaged: TRUE
  sector: electricity
  output: [dummy renewable, secondary]

bio_extr_1:
  name: bio_extr_1
  description: Biomass Extraction
  type: primary
  sector: extraction
  output: [biomass, primary]

bio_extr_2:
  name: bio_extr_2
  description: Biomass Extraction
  type: primary
  sector: extraction
  output: [biomass, primary]

bio_extr_3:
  name: bio_extr_3
  description: Biomass Extraction
  type: primary
  sector: extraction
  output: [biomass, primary]
```

(continues on next page)

(continued from previous page)

```
bio_extr_4:
  name: bio_extr_4
  description: Biomass Extraction
  type: primary
  sector: extraction
  output: [biomass, primary]

bio_extr_5:
  name: bio_extr_5
  description: Biomass Extraction
  type: primary
  sector: extraction
  output: [biomass, primary]

bio_extr_6:
  name: bio_extr_6
  description: Biomass Extraction
  type: primary
  sector: extraction
  output: [biomass, primary]

coal_extr:
  name: coal_extr
  description: Hard coal extraction, world average grade A
  type: primary
  sector: extraction
  input: [coal, resource]
  output: [coal, primary]

coal_extr_ch4:
  name: coal_extr_ch4
  description: Describes efforts in CH4-reduction from coal mining
  type: primary
  vintaged: TRUE
  sector: extraction
  input: [coal, resource]
  output: [coal, primary]

flaring_CO2:
  name: flaring_CO2
  description: Co2 emissions from gas flaring
  type: exports
  sector: extraction
  output: [exports, exports]

gas_extr_1:
  name: gas_extr_1
  description: Natural gas extraction, Cat I = Master et al.14.WPC "Identified␣
↪Reserves"
  type: primary
  sector: extraction
  input: [resource]
  output: [primary]

gas_extr_2:
  name: gas_extr_2
  description: Natural gas extraction, Cat II = Master et al.14.WPC "Mode"␣
↪undiscovered natural gas
  type: primary
  sector: extraction
```

(continues on next page)

(continued from previous page)

```
  input: [resource]
  output: [primary]

gas_extr_3:
  name: gas_extr_3
  description: Natural gas extraction, Cat III = Masters  et al.14.WPC Difference␣
↪between "Mode and 5%"  undiscovered natural gas
  type: primary
  sector: extraction
  input: [resource]
  output: [primary]

gas_extr_4:
  name: gas_extr_4
  description: Natural gas extraction, Cat IV = Estimated enhanced Recovery (30%␣
↪of Resources I+II+III)  plus 15% of historical production
  type: primary
  sector: extraction
  input: [resource]
  output: [primary]

gas_extr_5:
  name: gas_extr_5
  description: Natural gas extraction, Cat V = Non-conventional reserves (20% of␣
↪Coal bed; 15% of fractured Shale; 15% of Tight formation)
  type: primary
  sector: extraction
  input: [resource]
  output: [primary]

gas_extr_6:
  name: gas_extr_6
  description: Natural gas extraction, Cat VI -VII= Non-conventional resources.␣
↪Rest of Coal bed (80%), fractured Shale (85%) and Tight formation (85%) were␣
↪aggregated and then distributed to VI (40%) nd VII (60%)
  type: primary
  sector: extraction
  input: [resource]
  output: [primary]

gas_extr_mpen:
  name: gas_extr_mpen
  description: Common Market penetration for all gas extraction technologies
  type: secondary
  sector: extraction
  output: [exports, secondary]

lignite_extr:
  name: lignite_extr
  description: Lignite extraction, world average grade A
  type: primary
  sector: extraction
  input: [lignite, resource]
  output: [coal, primary]

oil_extr_1:
  name: oil_extr_1
  description: Crude oil extraction, Cat I = Masters 14 WPC conv. oil reserves
  type: primary
  sector: extraction
  input: [crude 1 resource, primary]
```

(continues on next page)

(continued from previous page)

```
  output: [crude oil, primary]

oil_extr_1_ch4:
  name: oil_extr_1_ch4
  description: Describes efforts in CH4-reduction from oil extraction of Cat I
  type: primary
  sector: extraction
  input: [crude 1 resource, primary]
  output: [crude oil, primary]

oil_extr_2:
  name: oil_extr_2
  description: Crude oil extraction, Cat II = Masters mode undiscovered conv. oil␣
↪(incl. NGL)
  type: primary
  sector: extraction
  input: [crude 2 resource, primary]
  output: [crude oil, primary]

oil_extr_2_ch4:
  name: oil_extr_2_ch4
  description: Describes efforts in CH4-reduction from oil extraction of Cat II
  type: primary
  sector: extraction
  input: [crude 2 resource, primary]
  output: [crude oil, primary]

oil_extr_3:
  name: oil_extr_3
  description: Crude oil extraction, Cat III = Masters 5% - Masters 50%
  type: primary
  sector: extraction
  input: [crude 3 resource, primary]
  output: [crude oil, primary]

oil_extr_3_ch4:
  name: oil_extr_3_ch4
  description: Describes efforts in CH4-reduction from oil extraction of Cat III
  type: primary
  sector: extraction
  input: [crude 3 resource, primary]
  output: [crude oil, primary]

oil_extr_4:
  name: oil_extr_4
  description: Crude oil extraction, Cat IV = Recoverable "reserves" non-
↪conventional oil
  type: primary
  sector: extraction
  input: [crude 4 resource, primary]
  output: [crude oil, primary]

oil_extr_4_ch4:
  name: oil_extr_4_ch4
  description: Describes efforts in CH4-reduction from oil extraction of Cat IV
  type: primary
  sector: extraction
  input: [crude 4 resource, primary]
  output: [crude oil, primary]

oil_extr_5:
```

(continues on next page)

(continued from previous page)

```
  name: oil_extr_5
  description: Crude oil extraction, Cat V = Recoverable reserves of␣
↪nonconventional oil = Shale, tarsands/bitumen and heavy oils
  type: primary
  sector: extraction
  input: [crude 5 resource, primary]
  output: [crude oil, primary]

oil_extr_6:
  name: oil_extr_6
  description: Crude oil extraction, Cat VI = 20% of estimated occurrences (-
↪reserves) of shale, heavy oils, tarsands/bitumen
  type: primary
  sector: extraction
  input: [crude 6 resource, primary]
  output: [crude oil, primary]

oil_extr_mpen:
  name: oil_extr_mpen
  description: Common Market penetration for all oil extraction technologies
  type: secondary
  sector: extraction
  output: [exports, secondary]

Feeds_1:
  name: Feeds_1
  description: Conservation cost curve step for industry feedstock demand
  type: useful
  sector: feedstock
  output: [i_feed, useful]

Feeds_2:
  name: Feeds_2
  description: Conservation cost curve step for industry feedstock demand
  type: useful
  sector: feedstock
  output: [i_feed, useful]

Feeds_3:
  name: Feeds_3
  description: Conservation cost curve step for industry feedstock demand
  type: useful
  sector: feedstock
  output: [i_feed, useful]

Feeds_4:
  name: Feeds_4
  description: Conservation cost curve step for industry feedstock demand
  type: useful
  sector: feedstock
  output: [i_feed, useful]

Feeds_5:
  name: Feeds_5
  description: Conservation cost curve step for industry feedstock demand
  type: useful
  sector: feedstock
  output: [i_feed, useful]

Feeds_con:
  name: Feeds_con
```

(continues on next page)

(continued from previous page)

```
  description: Joint diffusion constraint for feedstock conservation cost curve␣
↪steps
  type: primary
  sector: feedstock
  output: [dummy agriculture, primary]

coal_fs:
  name: coal_fs
  description: Coal as industry feedstock
  type: useful
  sector: feedstock
  input: [coal, final]
  output: [i_feed, useful]

ethanol_fs:
  name: ethanol_fs
  description: Ethanol as industry feedstock
  type: useful
  sector: feedstock
  input: [ethanol, final]
  output: [i_feed, useful]

foil_fs:
  name: foil_fs
  description: Fuel oil as industry feedstock
  type: useful
  sector: feedstock
  input: [fueloil, final]
  output: [i_feed, useful]

gas_fs:
  name: gas_fs
  description: Gas as industry feedstock
  type: useful
  sector: feedstock
  input: [gas, final]
  output: [i_feed, useful]

loil_fs:
  name: loil_fs
  description: Lightoil as industry feedstock
  type: useful
  sector: feedstock
  input: [lightoil, final]
  output: [i_feed, useful]

methanol_fs:
  name: methanol_fs
  description: Methanol as industry feedstock
  type: useful
  sector: feedstock
  input: [methanol, final]
  output: [i_feed, useful]

coal_gas:
  name: coal_gas
  description: Hard coal gasification
  type: secondary
  vintaged: TRUE
  sector: gas
  input:
```

(continues on next page)

(continued from previous page)

```
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output: [gas, secondary]

g_ppl_co2scr:
  name: g_ppl_co2scr
  description: CO2 scrubber for natural gas power plant
  type: secondary
  vintaged: TRUE
  sector: gas
  output: [exports, secondary]

gas_bal:
  name: gas_bal
  description: Link technology to stabilize gas production
  type: secondary
  sector: gas
  input: [gas, primary]
  output: [gas, secondary]

gas_bio:
  name: gas_bio
  description: Synthesis gas production from biomass
  type: secondary
  vintaged: TRUE
  sector: gas
  input:
    - [biomass, primary]
    - [electr, secondary]
    - [freshwater_supply, water_supply]
  output: [gas, secondary]

gas_imp:
  name: gas_imp
  description: Piped Gas imports
  type: secondary
  sector: gas
  input: [gas, exports]
  output: [gas, secondary]

gas_rc:
  name: gas_rc
  description: Gas heating in residential/commercial sector
  type: final
  sector: residential/commercial
  input: [gas, secondary]
  output: [gas, final]

gas_t_d:
  name: gas_t_d
  description: Transmission/Distribution of gas
  type: final
  vintaged: TRUE
  sector: gas
  input: [gas, secondary]
  output: [gas, final]

gas_t_d_ch4:
  name: gas_t_d_ch4
  description: Transmission/Distribution of gas with CH4 mitigation
  type: final
```

(continues on next page)

(continued from previous page)

```
  vintaged: TRUE
  sector: gas
  input: [gas, secondary]
  output: [gas, final]

gfc_co2scr:
  name: gfc_co2scr
  description: New co2 scrubber for gas fuel cells
  type: secondary
  vintaged: TRUE
  sector: gas
  output: [exports, secondary]

glb_gas_exp:
  name: glb_gas_exp
  description: Global net export of gas
  type: exports
  sector: gas
  output: [gas, exports]

glb_LNG_exp:
  name: glb_LNG_exp
  description: Global net export of liquified natural gas
  type: imports
  sector: gas
  output: [LNG, imports]

h2_mix:
  name: h2_mix
  description: Hydrogen injection into the natural gas system
  type: secondary
  sector: gas
  input: [hydrogen, secondary]
  output: [gas, secondary]

LNG_bal:
  name: LNG_bal
  description: Link technology to stabilize liquified natural gas production
  type: secondary
  sector: gas
  input: [LNG, primary]
  output: [LNG, secondary]

LNG_imp:
  name: LNG_imp
  description: LNG Imports
  type: imports
  sector: gas
  input: [LNG, imports]
  output: [LNG, secondary]

LNG_regas:
  name: LNG_regas
  description: LNG regasification (just link; losses are in trade)
  type: secondary
  vintaged: TRUE
  sector: gas
  input: [LNG, secondary]
  output: [gas, secondary]

bio_hpl:
```

(continues on next page)

(continued from previous page)

```
  name: bio_hpl
  description: Biomass heating plant
  type: secondary
  vintaged: TRUE
  sector: heat
  input:
    - [biomass, primary]
    - [cooling__bio_hpl, cooling]
    - [freshwater_supply, water_supply]
  output: [d_heat, secondary]

coal_hpl:
  name: coal_hpl
  description: Coal heating plant
  type: secondary
  vintaged: TRUE
  sector: heat
  input: [coal, secondary]
  output: [d_heat, secondary]

foil_hpl:
  name: foil_hpl
  description: Fuel oil heating plant
  type: secondary
  vintaged: TRUE
  sector: heat
  input: [fueloil, secondary]
  output: [d_heat, secondary]

gas_hpl:
  name: gas_hpl
  description: Natural gas heating plant
  type: secondary
  vintaged: TRUE
  sector: heat
  input: [gas, secondary]
  output: [d_heat, secondary]

geo_hpl:
  name: geo_hpl
  description: Geothermal heat plant
  type: secondary
  vintaged: TRUE
  sector: heat
  output: [d_heat, secondary]

heat_t/d:
  name: heat_t/d
  description: Transmission/Distribution of district heat
  type: final
  vintaged: TRUE
  sector: heat
  input: [d_heat, secondary]
  output: [d_heat, final]

po_turbine:
  name: po_turbine
  description: Pass out turbine
  type: secondary
  vintaged: TRUE
  sector: heat
```

(continues on next page)

(continued from previous page)

```
  input: [electr, secondary]
  output: [d_heat, secondary]

glb_lh2_imp:
  name: glb_lh2_imp
  description: Global net import of liquid hydrogen
  type: exports
  sector: hydrogen
  input: [liquid hydrogen, exports]
  output: [exports]

h2_bio:
  name: h2_bio
  description: Hydrogen production from biomass with C (via gasification)
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [biomass, primary]
    - [freshwater_supply, water_supply]
  output:
    - [hydrogen, secondary]
    - [electr, secondary]

h2_bio_ccs:
  name: h2_bio_ccs
  description: Hydrogen production from biomass with C (via gasification) with␣
↪carbon capture and storage
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [biomass, primary]
    - [freshwater_supply, water_supply]
  output:
    - [hydrogen, secondary]
    - [electr, secondary]

h2_co2_scrub:
  name: h2_co2_scrub
  description: CO2 scrubber for h2 production from coal and gas
  type: exports
  vintaged: TRUE
  sector: hydrogen
  input: [electr, secondary]
  output: [exports, secondary]

h2_coal:
  name: h2_coal
  description: Hydrogen production via coal gasification
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [hydrogen, secondary]
    - [electr, secondary]

h2_coal_ccs:
```

(continues on next page)

(continued from previous page)

```
  name: h2_coal_ccs
  description: Hydrogen production via coal gasification with carbon capture and␣
↪storage
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [hydrogen, secondary]
    - [electr, secondary]

h2_elec:
  name: h2_elec
  description: Hydrogen production via electrolysis
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input: [electr, secondary]
  output: [hydrogen, secondary]

h2_liq:
  name: h2_liq
  description: Hydrogen liquefaction
  type: primary
  vintaged: TRUE
  sector: hydrogen
  input: [hydrogen, primary]
  output: [liquid hydrogen, primary]

h2_smr:
  name: h2_smr
  description: Hydrogen production via steam-methane reforming of natural gas
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [gas, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [hydrogen, secondary]
    - [electr, secondary]

h2_smr_ccs:
  name: h2_smr_ccs
  description: Hydrogen production via steam-methane reforming of natural gas with␣
↪carbon capture and storage
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [gas, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [hydrogen, secondary]
    - [electr, secondary]

h2_t/d:
  name: h2_t/d
  description: Transmission/Distribution of gaseous hydrogen (just linking␣
```

(continues on next page)

(continued from previous page)

```
↪technology)
  type: final
  vintaged: TRUE
  sector: hydrogen
  input: [hydrogen, secondary]
  output: [hydrogen, final]

h2b_co2_scrub:
  name: h2b_co2_scrub
  description: CO2 scrubber for h2 production from biomass
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input: [electr, secondary]
  output: [exports, secondary]

lh2_bal:
  name: lh2_bal
  description: Link technology to stabilize liquid hydrogen production
  type: secondary
  sector: hydrogen
  input: [liquid hydrogen, primary]
  output: [liquid hydrogen, secondary]

lh2_exp:
  name: lh2_exp
  description: Exports of liquid hydrogen
  type: exports
  sector: hydrogen
  input: [liquid hydrogen, primary]
  output: [liquid hydrogen, exports]

lh2_imp:
  name: lh2_imp
  description: Imports of liquid hydrogen
  type: secondary
  sector: hydrogen
  input: [liquid hydrogen, exports]
  output: [liquid hydrogen, secondary]

lh2_regas:
  name: lh2_regas
  description: Regasification of liquid hydrogen
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input: [liquid hydrogen, secondary]
  output: [hydrogen, secondary]

lh2_t/d:
  name: lh2_t/d
  description: Transmission/Distribution of liquid hydrogen
  type: final
  vintaged: TRUE
  sector: hydrogen
  input: [liquid hydrogen, secondary]
  output: [liquid hydrogen, final]

back_bio_ind:
  name: back_bio_ind
  description: Backstop for diagnosing model infeasibility
```

(continues on next page)

(continued from previous page)

```
  type: final
  sector: industry
  output: [biomass, useful]

back_fs:
  name: back_fs
  description: Backstop for diagnosing model infeasibility
  type: useful
  sector: industry
  output: [i_feed, useful]

back_I:
  name: back_I
  description: Backstop for diagnosing model infeasibility
  type: useful
  sector: industry
  output: [i_spec, useful]

cement_CO2:
  name: cement_CO2
  description: Co2 emissions from cement production
  type: secondary
  sector: industry
  output: [exports, secondary]

cement_co2scr:
  name: cement_co2scr
  description: Cement CO2 scrubber (CCS)
  type: secondary
  vintaged: TRUE
  sector: industry
  output: [exports, secondary]

coal_i:
  name: coal_i
  description: Coal in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [coal, final]
  output: [i_therm, useful]

elec_i:
  name: elec_i
  description: Electricity in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [electr, final]
  output: [i_therm, useful]

eth_i:
  name: eth_i
  description: Ethanol (without C) replacement for use as liquid fuel in industry␣
↪thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [ethanol, final]
  output: [i_therm, useful]
```

(continues on next page)

(continued from previous page)

```
foil_i:
  name: foil_i
  description: Fuel oil for thermal uses in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [fueloil, final]
  output: [i_therm, useful]

gas_i:
  name: gas_i
  description: Gas for thermal uses in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [gas, final]
  output: [i_therm, useful]

h2_i:
  name: h2_i
  description: Gaseous hydrogen in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [hydrogen, final]
  output: [i_therm, useful]

heat_i:
  name: heat_i
  description: District heating for thermal uses in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [d_heat, final]
  output: [i_therm, useful]

hp_el_i:
  name: hp_el_i
  description: Electric heat pump in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [electr, final]
  output: [i_therm, useful]

hp_gas_i:
  name: hp_gas_i
  description: Natural gas heat pump in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [gas, final]
  output: [i_therm, useful]

loil_i:
  name: loil_i
  description: Lightoil for thermal uses in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [lightoil, final]
```

(continues on next page)

(continued from previous page)

```
  output: [i_therm, useful]

meth_i:
  name: meth_i
  description: Methanol (with C) replacement for use as liquid fuel in industry␣
↪thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [methanol, final]
  output: [i_therm, useful]

solar_i:
  name: solar_i
  description: Solar thermal in industry thermal sector
  type: useful
  vintaged: TRUE
  sector: industry
  output: [i_therm, useful]

Ispec_1:
  name: Ispec_1
  description: Conservation cost curve step for industry specific demand
  type: useful
  sector: industry
  output: [i_spec, useful]

Ispec_2:
  name: Ispec_2
  description: Conservation cost curve step for industry specific demand
  type: useful
  sector: industry
  output: [i_spec, useful]

Ispec_3:
  name: Ispec_3
  description: Conservation cost curve step for industry specific demand
  type: useful
  sector: industry
  output: [i_spec, useful]

Ispec_4:
  name: Ispec_4
  description: Conservation cost curve step for industry specific demand
  type: useful
  sector: industry
  output: [i_spec, useful]

Ispec_5:
  name: Ispec_5
  description: Conservation cost curve step for industry specific demand
  type: useful
  sector: industry
  output: [i_spec, useful]

Ispec_con:
  name: Ispec_con
  description: Joint diffusion constraint for industry specific conservation cost␣
↪curve steps
  type: primary
  sector: industry
```

(continues on next page)

(continued from previous page)

```
  output: [dummy agriculture, primary]

Itherm_1:
  name: Itherm_1
  description: Conservation cost curve step for industry thermal demand
  type: useful
  sector: industry
  output: [i_therm, useful]

Itherm_2:
  name: Itherm_2
  description: Conservation cost curve step for industry thermal demand
  type: useful
  sector: industry
  output: [i_therm, useful]

Itherm_3:
  name: Itherm_3
  description: Conservation cost curve step for industry thermal demand
  type: useful
  sector: industry
  output: [i_therm, useful]

Itherm_4:
  name: Itherm_4
  description: Conservation cost curve step for industry thermal demand
  type: useful
  sector: industry
  output: [i_therm, useful]

Itherm_5:
  name: Itherm_5
  description: Conservation cost curve step for industry thermal demand
  type: useful
  sector: industry
  output: [i_therm, useful]

Itherm_con:
  name: Itherm_con
  description: Joint diffusion constraint for industry thermal conservation cost␣
↪curve steps
  type: dummy
  sector: industry
  output: [dummy agriculture, primary]

solar_pv_I:
  name: solar_pv_I
  description: On-site solar photovoltaic power plant (no storage) in industry␣
↪specific
  type: useful
  vintaged: TRUE
  sector: industry
  output: [i_spec, useful]

h2_fc_I:
  name: h2_fc_I
  description: Hydrogen fuel cell cogeneration system for industry specific
  type: useful
  vintaged: TRUE
  sector: industry
  input: [hydrogen, final]
```

(continues on next page)

(continued from previous page)

```
  output: [i_spec, useful]

sp_coal_I:
  name: sp_coal_I
  description: Specific use of coal in industry
  type: useful
  sector: industry
  input: [coal, final]
  output: [i_spec, useful]

sp_el_I:
  name: sp_el_I
  description: Specific use of electricity in industry
  type: useful
  sector: industry
  input: [electr, final]
  output: [i_spec, useful]

sp_eth_I:
  name: sp_eth_I
  description: Ethanol (without net C) replacement for specific use of light oil␣
↪in industry
  type: useful
  sector: industry
  input: [ethanol, final]
  output: [i_spec, useful]

sp_liq_I:
  name: sp_liq_I
  description: Specific use of light oil in industry
  type: useful
  sector: industry
  input: [lightoil, final]
  output: [i_spec, useful]

sp_meth_I:
  name: sp_meth_I
  description: Methanol (with C) replacement for  specific use of light oil in␣
↪industry
  type: useful
  sector: industry
  input: [methanol, final]
  output: [i_spec, useful]

bio_extr_mpen:
  name: bio_extr_mpen
  description: Slack primary biomass created with new implementation of non␣
↪commercial biomass
  type: secondary
  sector: land
  output: [exports, secondary]

forest_CO2:
  name: forest_CO2
  description: Co2 emissions from forests
  type: secondary
  sector: land
  output: [exports, secondary]

sinks_1:
  name: sinks_1
```

(continues on next page)

(continued from previous page)

```
  description: Potential for sinks (50. US$)
  type: secondary
  sector: land
  output: [exports, secondary]

sinks_2:
  name: sinks_2
  description: Potential for sinks (100. US$)
  type: secondary
  sector: land
  output: [exports, secondary]

sinks_3:
  name: sinks_3
  description: Potential for sinks (200. US$)
  type: secondary
  sector: land
  output: [exports, secondary]

sinks_4:
  name: sinks_4
  description: Potential for sinks (300. US$)
  type: secondary
  sector: land
  output: [exports, secondary]

eth_bal:
  name: eth_bal
  description: Link technology to stabilize ethanol production
  type: secondary
  sector: liquids
  input: [ethanol, primary]
  output: [ethanol, secondary]

eth_bio:
  name: eth_bio
  description: Ethanol synthesis via biomass gasification
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [biomass, primary]
    - [freshwater_supply, water_supply]
  output:
    - [ethanol, primary]
    - [electr, secondary]

eth_bio_ccs:
  name: eth_bio_ccs
  description: Ethanol synthesis via biomass gasification with carbon capture and␣
↪storage
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [biomass, primary]
    - [freshwater_supply, water_supply]
  output:
    - [ethanol, primary]
    - [electr, secondary]
```

(continues on next page)

(continued from previous page)

```
eth_exp:
  name: eth_exp
  description: Exports of ethanol (no accounting of CO2 transferred)
  type: exports
  sector: liquids
  input: [ethanol, primary]
  output: [ethanol, exports]

eth_imp:
  name: eth_imp
  description: Imports of ethanol (no accounting of CO2 transferred)
  type: secondary
  sector: liquids
  input: [ethanol, exports]
  output: [ethanol, secondary]

eth_t/d:
  name: eth_t/d
  description: Transmission/Distribution of methanol without net C (just linking␣
↪technology)
  type: final
  sector: liquids
  input: [ethanol, secondary]
  output: [ethanol, final]

foil_exp:
  name: foil_exp
  description: Net exports of crude oil at 95% of oil price
  type: exports
  sector: liquids
  input: [fueloil, secondary]
  output: [fueloil, exports]

foil_imp:
  name: foil_imp
  description: Net imports of residual oil at 95% of oil price
  type: secondary
  sector: liquids
  input: [fueloil, imports]
  output: [fueloil, secondary]

foil_t/d:
  name: foil_t/d
  description: Transmission/Distribution of fueloil  (just linking technology)
  type: final
  sector: liquids
  input: [fueloil, secondary]
  output: [fueloil, final]

glb_eth_imp:
  name: glb_eth_imp
  description: Global net import of ethanol
  type: exports
  sector: liquids
  input: [ethanol, exports]
  output: [exports]

glb_foil_exp:
  name: glb_foil_exp
  description: Global net export of fuel oil
  type: imports
```

(continues on next page)

(continued from previous page)

```
  sector: liquids
  output: [fueloil, imports]

glb_foil_imp:
  name: glb_foil_imp
  description: Global net import of fuel oil
  type: exports
  sector: liquids
  input: [fueloil, exports]
  output: [exports]

glb_loil_exp:
  name: glb_loil_exp
  description: Global net export of light oil
  type: imports
  sector: liquids
  output: [loil, imports]

glb_loil_imp:
  name: glb_loil_imp
  description: Global net import of light oil
  type: exports
  sector: liquids
  input: [loil, exports]
  output: [exports]

glb_meth_imp:
  name: glb_meth_imp
  description: Global net import of methanol
  type: exports
  sector: liquids
  input: [methanol, exports]
  output: [exports]

glb_oil_exp:
  name: glb_oil_exp
  description: Global net export of crude oil
  type: imports
  sector: liquids
  output: [oil, imports]

glb_oil_imp:
  name: glb_oil_imp
  description: Global net import of crude oil
  type: exports
  sector: liquids
  input: [oil, exports]
  output: [exports]

liq_bio:
  name: liq_bio
  description: Second Generation Ethanol Production based on Biomass to FTL
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [biomass, primary]
    - [freshwater_supply, water_supply]
  output:
    - [ethanol, primary]
    - [electr, secondary]
```

(continues on next page)

(continued from previous page)

```
liq_bio_ccs:
  name: liq_bio_ccs
  description: Second Generation Ethanol Production with carbon capture and␣
↪storage based on Biomass to FTL
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [biomass, primary]
    - [freshwater_supply, water_supply]
  output:
    - [ethanol, primary]
    - [electr, secondary]

loil_exp:
  name: loil_exp
  description: Net exports of crude oil at 15% above oil price
  type: exports
  sector: liquids
  input: [lightoil, secondary]
  output: [lightoil, exports]

loil_imp:
  name: loil_imp
  description: Net imports of light oil at 15% above crude price
  type: secondary
  sector: liquids
  input: [lightoil, imports]
  output: [lightoil, secondary]

loil_std:
  name: loil_std
  description: Standard light oil power-plant
  type: secondary
  vintaged: TRUE
  sector: liquids
  output: [lightoil, secondary]

loil_t/d:
  name: loil_t/d
  description: Transmission/Distribution of light oil (just linking technology)
  type: final
  sector: liquids
  input: [lightoil, secondary]
  output: [lightoil, final]

meth_coal:
  name: meth_coal
  description: Methanol synthesis via coal gasification
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [methanol, primary]
    - [electr, secondary]

meth_coal_ccs:
```

(continues on next page)

(continued from previous page)

```
  name: meth_coal_ccs
  description: Methanol synthesis via coal gasification with carbon capture and␣
↪storage
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [methanol, primary]
    - [electr, secondary]

meth_exp:
  name: meth_exp
  description: Exports of methanol (no accounting of CO2 transferred)
  type: exports
  sector: liquids
  input: [methanol, primary]
  output: [methanol, exports]

meth_imp:
  name: meth_imp
  description: Imports of methanol (no accounting of CO2 transferred)
  type: secondary
  sector: liquids
  input: [methanol, exports]
  output: [methanol, secondary]

meth_ng:
  name: meth_ng
  description: Methanol synthesis via natural gas
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [gas, secondary]
    - [freshwater_supply, water_supply]
  output: [methanol, primary]

meth_ng_ccs:
  name: meth_ng_ccs
  description: Methanol synthesis via natural gas with carbon capture and storage
  type: primary
  vintaged: TRUE
  sector: liquids
  input:
    - [gas, secondary]
    - [freshwater_supply, water_supply]
  output: [methanol, primary]

meth_t/d:
  name: meth_t/d
  description: Transmission/Distribution of methanol with C
  type: final
  sector: liquids
  input: [methanol, secondary]
  output: [methanol, final]

oil_bal:
  name: oil_bal
```

(continues on next page)

(continued from previous page)

```
  description: Link technology to stabilize crude oil production
  type: secondary
  sector: liquids
  input: [crude oil, primary]
  output: [crude oil, secondary]

oil_exp:
  name: oil_exp
  description: Net exports of crude oil at 100% of oil price
  type: exports
  sector: liquids
  input: [crude oil, primary]
  output: [oil, exports]

oil_imp:
  name: oil_imp
  description: Net imports of oil
  type: secondary
  sector: liquids
  input: [oil, imports]
  output: [crude oil, secondary]

plutonium_prod:
  name: plutonium_prod
  description: Plutonium production
  type: stocks
  sector: liquids
  input: [plutonium, stocks]
  output: [plutonium, stocks]

ref_hil:
  name: ref_hil
  description: New deeply upgraded refineries
  type: secondary
  vintaged: TRUE
  sector: liquids
  input: [crude oil, secondary]
  output: [fueloil, secondary]

ref_lol:
  name: ref_lol
  description: Existing refineries (low yield)
  type: secondary
  vintaged: TRUE
  sector: liquids
  input: [crude oil, secondary]
  output: [fueloil, secondary]

SO2_scrub_ref:
  name: SO2_scrub_ref
  description: SO2 scrubber for refineries
  type: secondary
  vintaged: TRUE
  sector: liquids
  output: [exports, secondary]

syn_liq:
  name: syn_liq
  description: Coal liquefaction and light oil synthesis
  type: secondary
  vintaged: TRUE
```

(continues on next page)

(continued from previous page)

```
  sector: liquids
  input:
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [lightoil, secondary]
    - [electr, secondary]

syn_liq_ccs:
  name: syn_liq_ccs
  description: Coal liquefaction and light oil synthesis with carbon capture and␣
↪storage
  type: secondary
  vintaged: TRUE
  sector: liquids
  input:
    - [coal, secondary]
    - [freshwater_supply, water_supply]
  output:
    - [lightoil, secondary]
    - [electr, secondary]

adipic_thermal:
  name: adipic_thermal
  description: Thermal destruction tech for adipic acid sector
  type: dummy
  sector: non-co2
  output: [dummy, primary]

ammonia_secloop:
  name: ammonia_secloop
  description: Ammonia Secondary Loop Systems
  type: dummy
  sector: non-co2
  output: [dummy, primary]

enre_con:
  name: enre_con
  description: Joint diffusion constraint for enteric fermentation mitigation␣
↪technologies
  type: dummy
  sector: non-co2
  output: [dummy agriculture, primary]

ent_red1:
  name: ent_red1
  description: Mitigation for CH4 emissions from animals directly
  type: dummy
  sector: non-co2
  output: [dummy agriculture, primary]

ent_red2:
  name: ent_red2
  description: Mitigation for CH4 emissions from animals directly
  type: dummy
  sector: non-co2
  output: [dummy agriculture, primary]

ent_red3:
  name: ent_red3
  description: Mitigation for CH4 emissions from animals directly
```

(continues on next page)

(continued from previous page)

```
  type: dummy
  sector: non-co2
  output: [dummy agriculture, primary]

landfill_compost1:
  name: landfill_compost1
  description: Landfill mitigation technology (composting) technology 1 for CH4␣
↪mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

landfill_compost2:
  name: landfill_compost2
  description: Landfill mitigation technology (composting) technology 2 for CH4␣
↪mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

landfill_direct1:
  name: landfill_direct1
  description: Landfill mitigation technology (direct) 1 for CH4 mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

landfill_direct2:
  name: landfill_direct2
  description: Landfill mitigation technology (direct) 1 for CH4 mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

landfill_ele:
  name: landfill_ele
  description: Landfill mitigation technology (ele) 1 for CH4 mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

landfill_flaring:
  name: landfill_flaring
  description: Landfill mitigation technology (flaring) 1 for CH4 mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

landfill_heatprdn:
  name: landfill_heatprdn
  description: Landfill mitigation technology (heat production) 1 for CH4␣
↪mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
```

(continues on next page)

(continued from previous page)

```
  output: [dummy, primary]

landfill_oxdn:
  name: landfill_oxdn
  description: Landfill mitigation technology (oxidation) 1 for CH4 mitigation
  type: dummy
  vintaged: TRUE
  sector: non-co2
  output: [dummy, primary]

leak_repair:
  name: leak_repair
  description: Leak-repairs, HFC-134a from refrigeration & AC(SR)
  type: dummy
  sector: non-co2
  output: [dummy, primary]

leak_repairsf6:
  name: leak_repairsf6
  description: Recycling gas carts for SF6 recovery during assembly of gas␣
↪insulated equipment
  type: dummy
  sector: non-co2
  output: [dummy, primary]

lfil_con:
  name: lfil_con
  description: Joint diffusion constraint for landfill mitigation technologies
  type: dummy
  sector: non-co2
  output: [dummy agriculture, primary]

manu_con:
  name: manu_con
  description: Joint diffusion constraint for manure management+B249 mitigation␣
↪technologies
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

meth_bal:
  name: meth_bal
  description: Link technology to stabilize methanol production
  type: primary
  sector: non-co2
  input: [methanol, primary]
  output: [methanol, secondary]

mvac_co2:
  name: mvac_co2
  description: Transcritical vapor cycle CO2 systems for mobile vehicle air␣
↪conditioners
  type: dummy
  sector: non-co2
  output: [dummy]

recycling_gas1:
  name: recycling_gas1
  description: Recycling gas carts for SF6 recovery during maintenance of gas␣
↪insulated equipment(SR)
  type: dummy
```

(continues on next page)

(continued from previous page)

```
  sector: non-co2
  output: [dummy, primary]

refrigerant_recover:
  name: refrigerant_recover
  description: Recovery of refrigerant, HFC-134a from refrigeration and AC (SR)
  type: primary
  sector: non-co2
  output: [dummy, primary]

repl_hc:
  name: repl_hc
  description: Replacement with HC for foams
  type: primary
  sector: non-co2
  output: [dummy, primary]

replacement_so2:
  name: replacement_so2
  description: Replacing SF6 by SO2(SR)
  type: primary
  sector: non-co2
  output: [dummy, primary]

rice_red1:
  name: rice_red1
  description: Mitigation for CH4 emissions from rice
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

rice_red2:
  name: rice_red2
  description: Mitigation for CH4 emissions from rice
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

rice_red3:
  name: rice_red3
  description: Mitigation for CH4 emissions from rice
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

rire_con:
  name: rire_con
  description: Joint diffusion constraint for rice cultivation mitigation␣
↪technologies
  type: primary
  sector: non-co2
  output: [dummy  agriculture, primary]

soil_red1:
  name: soil_red1
  description: Mitigation for N2Oemissions from soil
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

soil_red2:
```

(continues on next page)

(continued from previous page)

```
  name: soil_red2
  description: Mitigation for N2Oemissions from soil
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

soil_red3:
  name: soil_red3
  description: Mitigation for N2Oemissions from soil
  type: primary
  sector: non-co2
  output: [dummy agriculture, primary]

vertical_stud:
  name: vertical_stud
  description: Soderberg process for CF4 from aluminum (SR)
  type: dummy
  sector: non-co2
  output: [dummy, primary]

u5-reproc:
  name: u5-reproc
  description: Uranium reprocessing
  type: stocks
  vintaged: TRUE
  sector: nuclear
  input: [uranium, stocks]
  output: [uranium, stocks]

Uran_extr:
  name: Uran_extr
  description: Uranium extraction, milling for FBR blanket
  type: stocks
  sector: nuclear
  input: [uranium, resource]
  output: [uranium, stocks]

uran2u5:
  name: uran2u5
  description: Uranium extraction, milling, fluorination and enrichment per t U5
  type: stocks
  sector: nuclear
  input: [uranium, resource]
  output: [uranium, stocks]

bco2_tr_dis:
  name: bco2_tr_dis
  description: Technology for co2 transportation and disposal from biomass␣
↪technologies
  type: secondary
  sector: other conversion
  output: [exports, secondary]

co2_tr_dis:
  name: co2_tr_dis
  description: Technology for co2 transportation and disposal
  type: secondary
  sector: other conversion
  output: [exports, secondary]

back_rc:
```

(continues on next page)

(continued from previous page)

```
  name: back_RC
  description: Backstop for diagnosing model infeasibility
  type: useful
  sector: residential/commercial
  output: [rc_spec, useful]

solar_rc:
  name: solar_rc
  description: Solar thermal in residential/commercial sector
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]

biomass_rc:
  name: biomass_rc
  description: Biomass with C for heating in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [biomass, final]
  output: [rc_therm, useful]

coal_rc:
  name: coal_rc
  description: Coal heating in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [coal, final]
  output: [rc_therm, useful]

elec_rc:
  name: elec_rc
  description: Electricity heating in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [electr, final]
  output: [rc_therm, useful]

eth_rc:
  name: eth_rc
  description: Ethanol (without C) replacement for use as liquid fuel in␣
↪residential/commercial
  type: useful
  sector: residential/commercial
  input: [ethanol, final]
  output: [rc_therm, useful]

foil_rc:
  name: foil_rc
  description: Fuel oil heating in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [fueloil, final]
  output: [rc_therm, useful]

h2_rc:
  name: h2_rc
  description: Hydrogen (gaseous) catalytic heating in the residential/commercial␣
↪sector
  type: secondary
  sector: residential/commercial
  input: [hydrogen, final]
```

(continues on next page)

(continued from previous page)

```
  output: [rc_therm, useful]

heat_rc:
  name: heat_rc
  description: District heating in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [d_heat, final]
  output: [rc_therm, useful]

hp_el_rc:
  name: hp_el_rc
  description: Electric heat pump in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [electr, final]
  output: [rc_therm, useful]

hp_gas_rc:
  name: hp_gas_rc
  description: Natural gas heat pump in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [gas, final]
  output: [rc_therm, useful]

loil_rc:
  name: loil_rc
  description: Lightoil heating in residential/commercial sector
  type: useful
  sector: residential/commercial
  input: [lightoil, final]
  output: [rc_therm, useful]

meth_rc:
  name: meth_rc
  description: Methanol (with C) replacement for use as liquid fuel in residential/
↪commercial
  type: useful
  sector: residential/commercial
  input: [methanol, final]
  output: [rc_therm, useful]

RCspec_1:
  name: RCspec_1
  description: Conservation cost curve step for residential/commercial specific␣
↪demand
  type: useful
  sector: residential/commercial
  output: [useful]

RCspec_2:
  name: RCspec_2
  description: Conservation cost curve step for residential/commercial specific␣
↪demand
  type: useful
  sector: residential/commercial
  output: [useful]

RCspec_3:
  name: RCspec_3
```

(continues on next page)

(continued from previous page)

```
  description: Conservation cost curve step for residential/commercial specific␣
↪demand
  type: useful
  sector: residential/commercial
  output: [useful]

RCspec_4:
  name: RCspec_4
  description: Conservation cost curve step for residential/commercial specific␣
↪demand
  type: useful
  sector: residential/commercial
  output: [useful]

RCspec_5:
  name: RCspec_5
  description: Conservation cost curve step for residential/commercial specific␣
↪demand
  type: useful
  sector: residential/commercial
  output: [useful]

RCspec_con:
  name: RCspec_con
  description: Joint diffusion constraint for residential/commercial specific␣
↪conservation cost curve steps
  type: dummy
  sector: residential/commercial
  output: [dummy agriculture, primary]

RCtherm_1:
  name: RCtherm_1
  description: Conservation cost curve step for residential/commercial thermal␣
↪demand
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]

RCtherm_2:
  name: RCtherm_2
  description: Conservation cost curve step for residential/commercial thermal␣
↪demand
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]

RCtherm_3:
  name: RCtherm_3
  description: Conservation cost curve step for residential/commercial thermal␣
↪demand
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]

RCtherm_4:
  name: RCtherm_4
  description: Conservation cost curve step for residential/commercial thermal␣
↪demand
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]
```

(continues on next page)

(continued from previous page)

```
RCtherm_5:
  name: RCtherm_5
  description: Conservation cost curve step for residential/commercial thermal␣
↪demand
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]

RCtherm_con:
  name: RCtherm_con
  description: Joint diffusion constraint for residential/commercial thermal␣
↪conservation cost curve steps
  type: dummy
  sector: residential/commercial
  output: [dummy agriculture, primary]

solar_pv_RC:
  name: solar_pv_RC
  description: Specific use of on-site solar photovoltaic power plant (no storage)␣
↪in residential/commercial
  type: useful
  vintaged: TRUE
  sector: residential/commercial
  output: [rc_spec, useful]

sp_el_RC:
  name: sp_el_RC
  description: Specific use of electricity in residential/commercial
  type: useful
  sector: residential/commercial
  input: [electr, final]
  output: [rc_spec, useful]

h2_fc_RC:
  name: h2_fc_RC
  description: Specific use of hydrogen fuel cell cogeneration system in res/comm
  type: useful
  vintaged: TRUE
  sector: residential/commercial
  input: [hydrogen, final]
  output: [rc_spec, useful]

bio_extr_chp:
  name: bio_extr_chp
  description: Supply of biomass without net C emissions;  defined here as␣
↪agricultural wastes, and other crops with cycle of 1 year or less
  type: primary
  sector: solids
  output: [biomass, primary]

bio_ppl_co2scr:
  name: bio_ppl_co2scr
  description: New coal scrubber for power plants
  type: secondary
  vintaged: TRUE
  sector: solids
  output: [exports, secondary]

biomass_i:
  name: biomass_i
```

(continues on next page)

(continued from previous page)

```
  description: Biomass with C in industry thermal
  type: useful
  vintaged: TRUE
  sector: solids
  input: [biomass, final]
  output: [i_therm, useful]

biomass_nc:
  name: biomass_nc
  description: Non-commercial biomass with C
  type: useful
  vintaged: TRUE
  sector: solids
  input: [biomass, primary]
  output: [non-comm, useful]

biomass_t/d:
  name: biomass_t/d
  description: Transmission/Distribution of biomass with C
  type: useful
  sector: solids
  input: [biomass, primary]
  output: [biomass, final]

c_ppl_co2scr:
  name: c_ppl_co2scr
  description: New coal scrubber for coal power plants
  type: secondary
  vintaged: TRUE
  sector: solids
  output: [exports, secondary]

cfc_co2scr:
  name: cfc_co2scr
  description: Co2 scrubber for coal fuel cells (CCS)
  type: secondary
  vintaged: TRUE
  sector: solids
  output: [exports, secondary]

coal_bal:
  name: coal_bal
  description: Link technology to stabilize coal production
  type: secondary
  sector: solids
  input: [coal, primary]
  output: [coal, secondary]

coal_exp:
  name: coal_exp
  description: Net exports of coal at fixed price of about US$ 42/tce
  type: exports
  sector: solids
  input: [coal, primary]
  output: [coal, exports]

coal_imp:
  name: coal_imp
  description: Net imports of coal
  type: secondary
  sector: solids
```

(continues on next page)

(continued from previous page)

```
  input: [coal, imports]
  output: [coal, secondary]

coal_t/d:
  name: coal_t/d
  description: Transmission/Distribution of coal
  type: final
  sector: solids
  input: [coal, secondary]
  output: [coal, final]

coal_t/d-in-06%:
  name: coal_t/d-in-06%
  description: Transmission/Distribution of coal industry like coal_t/d but␣
↪imported coal  with 0.6% S content
  type: final
  sector: solids
  input: [coal, secondary]
  output: [coal, final]

coal_t/d-in-SO2:
  name: coal_t/d-in-SO2
  description: Transmission/Distribution of coal industry like coal_t/d but␣
↪processed coal
  type: final
  sector: solids
  input: [coal, secondary]
  output: [coal, final]

coal_t/d-rc-06%:
  name: coal_t/d-rc-06%
  description: Transmission/Distribution of coal residential/commercial like coal_
↪t/d but imported coal with 0.6% S content
  type: final
  sector: solids
  input: [coal, secondary]
  output: [coal, final]

coal_t/d-rc-SO2:
  name: coal_t/d-rc-SO2
  description: Transmission/Distribution of coal like coal_t/d but processed coal
  type: final
  sector: solids
  input: [coal, secondary]
  output: [coal, final]

glb_coal_exp:
  name: glb_coal_exp
  description: Global net export of coal
  type: imports
  sector: solids
  output: [coal, imports]

glb_coal_imp:
  name: glb_coal_imp
  description: Global net import of coal
  type: exports
  sector: solids
  input: [electr, exports]
  output: [exports]
```

(continues on next page)

(continued from previous page)

```
back_trp:
  name: back_trp
  description: Backstop for diagnosing model infeasibility
  type: useful
  sector: transport
  output: [transport, useful]

coal_trp:
  name: coal_trp
  description: Coal-based transport
  type: useful
  sector: transport
  input: [coal, final]
  output: [transport, useful]

elec_trp:
  name: elec_trp
  description: Electricity-based transport
  type: useful
  sector: transport
  input: [electr, final]
  output: [transport, useful]

eth_fc_trp:
  name: eth_fc_trp
  description: Ethanol (without net C) fuel cell-based transport
  type: useful
  sector: transport
  input: [ethanol, final]
  output: [transport, useful]

eth_ic_trp:
  name: eth_ic_trp
  description: Ethanol (without net C) ic-engine-based transport
  type: useful
  sector: transport
  input: [ethanol, final]
  output: [transport, useful]

foil_trp:
  name: foil_trp
  description: Fueloil-based transport
  type: useful
  sector: transport
  input: [fueloil, final]
  output: [transport, useful]

gas_trp:
  name: gas_trp
  description: Gas-based transport
  type: useful
  sector: transport
  input: [gas, final]
  output: [transport, useful]

h2_fc_trp:
  name: h2_fc_trp
  description: Hydrogen fuel cell-based transport (plus off-hours electricity␣
↪generation)
  type: useful
  vintaged: TRUE
```

(continues on next page)

(continued from previous page)

```
  sector: transport
  input: [liquid hydrogen, final]
  output: [transport, useful]

loil_trp:
  name: loil_trp
  description: Lightoil-based transport
  type: useful
  sector: transport
  input: [lightoil, final]
  output: [transport, useful]

meth_fc_trp:
  name: meth_fc_trp
  description: Methanol (with C) fuel cell-based transport
  type: useful
  sector: transport
  input: [methanol, final]
  output: [transport, useful]

meth_ic_trp:
  name: meth_ic_trp
  description: Methanol (with C) ic-engine-based transport
  type: useful
  sector: transport
  input: [methanol, final]
  output: [transport, useful]

Trans_1:
  name: Trans_1
  description: Conservation cost curve step for transport demand
  type: useful
  sector: transport
  output: [transport, useful]

Trans_2:
  name: Trans_2
  description: Conservation cost curve step for transport demand
  type: useful
  sector: transport
  output: [transport, useful]

Trans_3:
  name: Trans_3
  description: Conservation cost curve step for transport demand
  type: useful
  sector: transport
  output: [transport, useful]

Trans_4:
  name: Trans_4
  description: Conservation cost curve step for transport demand
  type: useful
  sector: transport
  output: [transport, useful]

Trans_5:
  name: Trans_5
  description: Conservation cost curve step for transport demand
  type: useful
  sector: transport
```

(continues on next page)

(continued from previous page)

```
  output: [transport, useful]

Trans_con:
  name: Trans_con
  description: Joint diffusion constraint for transport conservation cost curve␣
↪steps
  type: primary
  sector: transport
  output: [dummy agriculture, primary]

foil_bunker:
  name: foil_bunker
  description: Fuel oil demand for international shipping bunkers (global trade)
  type: useful
  sector: shipping
  input: [fueloil, final]
  output: [shipping, useful]

loil_bunker:
  name: loil_bunker
  description: Light oil demand for international shipping bunkers (global trade)
  type: useful
  sector: shipping
  input: [lightoil, final]
  output: [shipping, useful]

meth_bunker:
  name: meth_bunker
  description: Methanol demand for international shipping bunkers (global trade)
  type: useful
  sector: shipping
  input: [methanol, final]
  output: [shipping, useful]

eth_bunker:
  name: eth_bunker
  description: Ethanol demand for international shipping bunkers (global trade)
  type: useful
  sector: shipping
  input: [ethanol, final]
  output: [shipping, useful]

LNG_bunker:
  name: LNG_bunker
  description: Liquified natural gas demand for international shipping bunkers␣
↪(global trade)
  type: useful
  sector: shipping
  input: [LNG, final]
  output: [shipping, useful]

LH2_bunker:
  name: LH2_bunker
  description: Liquified hydrogen  demand for international shipping bunkers␣
↪(global trade)
  type: useful
  sector: shipping
  input: [lh2, final]
  output: [shipping, useful]
```

8.23.4 Others

```
<mes:Structure xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:com=
↪"http://www.sdmx.org/resources/sdmxml/schemas/v2_1/common" xmlns:data="http://
↪www.sdmx.org/resources/sdmxml/schemas/v2_1/data/structurespecific" xmlns:str=
↪"http://www.sdmx.org/resources/sdmxml/schemas/v2_1/structure" xmlns:mes="http://
↪www.sdmx.org/resources/sdmxml/schemas/v2_1/message" xmlns:gen="http://www.sdmx.
↪org/resources/sdmxml/schemas/v2_1/data/generic" xmlns:footer="http://www.sdmx.
↪org/resources/sdmxml/schemas/v2_1/message/footer">
  <mes:Header>
    <mes:Test>false</mes:Test>
    <mes:Prepared>2023-09-04T16:31:44.701872</mes:Prepared>
    <mes:Source xml:lang="en">Generated by message_ix_models 2023.5.32.
↪dev20+g8d51636</mes:Source>
  </mes:Header>
  <mes:Structures>
    <str:Codelists>
      <str:Codelist version="2017" isExternalReference="none" isFinal="none"␣
↪agencyID="ICONICS" id="SSP" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Codelist=ICONICS:SSP(2017)">
        <com:Name xml:lang="en">Shared Socioeconomic Pathways (2017 edition)</
↪com:Name>
        <com:Description xml:lang="en">This code list is *not* officially␣
↪published by ICONICS; rather, it is a rendering into SDMX of structural␣
↪information that is provided by ICONICS participants. It may change or be␣
↪superseded at any time.  Each code has a single digit ID like "1"; the "original-
↪id" annotation (and the start of the name) give a string like "SSP1". These␣
↪original IDs are *only* unique if using data enumerated by solely by one code␣
↪list or the other; if mixing the two, then they will be ambiguous. The URNs of␣
↪the codes or parts thereof (for instance, "ICONICS:SSP(2017).1") are, by␣
↪construction, unique.

This is the original set of SSPs as described in https://doi.org/10.1016/j.
↪gloenvcha.2016.05.009.</com:Description>
        <str:Code id="1" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2017).1">
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP1</com:AnnotationText>
            </com:Annotation>
          </com:Annotations>
          <com:Name xml:lang="en">SSP1: Sustainability</com:Name>
          <com:Description xml:lang="en">SSP1: Sustainability &#8211; Taking the␣
↪Green Road

Low challenges to mitigation and adaptation.

The world shifts gradually, but pervasively, toward a more sustainable path,␣
↪emphasizing
more inclusive development that respects perceived environmental boundaries.␣
↪Management
of the global commons slowly improves, educational and health investments␣
↪accelerate the
demographic transition, and the emphasis on economic growth shifts toward a broader
emphasis on human well-being. Driven by an increasing commitment to achieving
development goals, inequality is reduced both across and within countries.␣
↪Consumption
is oriented toward low material growth and lower resource and energy intensity.</
↪com:Description>
        </str:Code>
        <str:Code id="2" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2017).2">
```

(continues on next page)

(continued from previous page)

```
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP2</com:AnnotationText>
            </com:Annotation>
          </com:Annotations>
          <com:Name xml:lang="en">SSP2: Middle of the Road</com:Name>
          <com:Description xml:lang="en">SSP2: Middle of the Road

Medium challenges to mitigation and adaptation.

The world follows a path in which social, economic, and technological trends do not
shift markedly from historical patterns. Development and income growth proceeds
unevenly, with some countries making relatively good progress while others fall␣
↪short of
expectations. Global and national institutions work toward but make slow progress␣
↪in
achieving sustainable development goals. Environmental systems experience␣
↪degradation,
although there are some improvements and overall the intensity of resource and␣
↪energy
use declines. Global population growth is moderate and levels off in the second␣
↪half of
the century. Income inequality persists or improves only slowly and challenges to
reducing vulnerability to societal and environmental changes remain.</
↪com:Description>
        </str:Code>
        <str:Code id="3" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2017).3">
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP3</com:AnnotationText>
            </com:Annotation>
          </com:Annotations>
          <com:Name xml:lang="en">SSP3: Regional Rivalry</com:Name>
          <com:Description xml:lang="en">SSP3: Regional Rivalry &#8211; A Rocky␣
↪Road

High challenges to mitigation and adaptation.

A resurgent nationalism, concerns about competitiveness and security, and regional
conflicts push countries to increasingly focus on domestic or, at most, regional␣
↪issues.
Policies shift over time to become increasingly oriented toward national and␣
↪regional
security issues. Countries focus on achieving energy and food security goals within
their own regions at the expense of broader-based development. Investments in␣
↪education
and technological development decline. Economic development is slow, consumption is
material-intensive, and inequalities persist or worsen over time. Population␣
↪growth is
low in industrialized and high in developing countries. A low international␣
↪priority for
addressing environmental concerns leads to strong environmental degradation in some
regions.</com:Description>
        </str:Code>
        <str:Code id="4" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2017).4">
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP4</com:AnnotationText>
            </com:Annotation>
```

(continues on next page)

(continued from previous page)

```
          </com:Annotations>
          <com:Name xml:lang="en">SSP4: Inequality</com:Name>
          <com:Description xml:lang="en">SSP4: Inequality &#8211; A Road Divided

Low challenges to mitigation, high challenges to adaptation.

Highly unequal investments in human capital, combined with increasing disparities␣
↪in
economic opportunity and political power, lead to increasing inequalities and
stratification both across and within countries. Over time, a gap widens between an
internationally-connected society that contributes to knowledge- and capital-
↪intensive
sectors of the global economy, and a fragmented collection of lower-income, poorly
educated societies that work in a labor intensive, low-tech economy. Social␣
↪cohesion
degrades and conflict and unrest become increasingly common. Technology␣
↪development is
high in the high-tech economy and sectors. The globally connected energy sector
diversifies, with investments in both carbon-intensive fuels like coal and
unconventional oil, but also low-carbon energy sources. Environmental policies␣
↪focus on
local issues around middle and high income areas.</com:Description>
        </str:Code>
        <str:Code id="5" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2017).5">
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP5</com:AnnotationText>
            </com:Annotation>
          </com:Annotations>
          <com:Name xml:lang="en">SSP5: Fossil-fueled Development</com:Name>
          <com:Description xml:lang="en">SSP5: Fossil-fueled Development &#8211;␣
↪Taking the Highway

High challenges to mitigation, low challenges to adaptation.

This world places increasing faith in competitive markets, innovation and␣
↪participatory
societies to produce rapid technological progress and development of human capital␣
↪as
the path to sustainable development. Global markets are increasingly integrated.␣
↪There
are also strong investments in health, education, and institutions to enhance␣
↪human and
social capital. At the same time, the push for economic and social development is
coupled with the exploitation of abundant fossil fuel resources and the adoption of
resource and energy intensive lifestyles around the world. All these factors lead␣
↪to
rapid growth of the global economy, while global population peaks and declines in␣
↪the
21st century. Local environmental problems like air pollution are successfully␣
↪managed.
There is faith in the ability to effectively manage social and ecological systems,
including by geo-engineering if necessary.</com:Description>
        </str:Code>
      </str:Codelist>
    </str:Codelists>
  </mes:Structures>
</mes:Structure>
```

```
<mes:Structure xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:com=
↪"http://www.sdmx.org/resources/sdmxml/schemas/v2_1/common" xmlns:data="http://
↪www.sdmx.org/resources/sdmxml/schemas/v2_1/data/structurespecific" xmlns:str=
↪"http://www.sdmx.org/resources/sdmxml/schemas/v2_1/structure" xmlns:mes="http://
↪www.sdmx.org/resources/sdmxml/schemas/v2_1/message" xmlns:gen="http://www.sdmx.
↪org/resources/sdmxml/schemas/v2_1/data/generic" xmlns:footer="http://www.sdmx.
↪org/resources/sdmxml/schemas/v2_1/message/footer">
  <mes:Header>
    <mes:Test>false</mes:Test>
    <mes:Prepared>2023-09-04T16:31:44.703451</mes:Prepared>
    <mes:Source xml:lang="en">Generated by message_ix_models 2023.5.32.
↪dev20+g8d51636</mes:Source>
  </mes:Header>
  <mes:Structures>
    <str:Codelists>
      <str:Codelist version="2024" isExternalReference="none" isFinal="none"␣
↪agencyID="ICONICS" id="SSP" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Codelist=ICONICS:SSP(2024)">
        <com:Name xml:lang="en">Shared Socioeconomic Pathways (2024 edition)</
↪com:Name>
        <com:Description xml:lang="en">This code list is *not* officially␣
↪published by ICONICS; rather, it is a rendering into SDMX of structural␣
↪information that is provided by ICONICS participants. It may change or be␣
↪superseded at any time.  Each code has a single digit ID like "1"; the "original-
↪id" annotation (and the start of the name) give a string like "SSP1". These␣
↪original IDs are *only* unique if using data enumerated by solely by one code␣
↪list or the other; if mixing the two, then they will be ambiguous. The URNs of␣
↪the codes or parts thereof (for instance, "ICONICS:SSP(2017).1") are, by␣
↪construction, unique.

This set of SSPs is currently under development; for details, see https://depts.
↪washington.edu/iconics/.</com:Description>
        <str:Code id="1" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2024).1">
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP1</com:AnnotationText>
            </com:Annotation>
          </com:Annotations>
          <com:Name xml:lang="en">SSP1: Sustainability</com:Name>
          <com:Description xml:lang="en">SSP1: Sustainability &#8211; Taking the␣
↪Green Road

Low challenges to mitigation and adaptation.

The world shifts gradually, but pervasively, toward a more sustainable path,␣
↪emphasizing
more inclusive development that respects perceived environmental boundaries.␣
↪Management
of the global commons slowly improves, educational and health investments␣
↪accelerate the
demographic transition, and the emphasis on economic growth shifts toward a broader
emphasis on human well-being. Driven by an increasing commitment to achieving
development goals, inequality is reduced both across and within countries.␣
↪Consumption
is oriented toward low material growth and lower resource and energy intensity.</
↪com:Description>
        </str:Code>
        <str:Code id="2" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2024).2">
          <com:Annotations>
            <com:Annotation id="original-id">
```

(continues on next page)

(continued from previous page)

```
            <com:AnnotationText xml:lang="en">SSP2</com:AnnotationText>
          </com:Annotation>
        </com:Annotations>
        <com:Name xml:lang="en">SSP2: Middle of the Road</com:Name>
        <com:Description xml:lang="en">SSP2: Middle of the Road

Medium challenges to mitigation and adaptation.

The world follows a path in which social, economic, and technological trends do not
shift markedly from historical patterns. Development and income growth proceeds
unevenly, with some countries making relatively good progress while others fall␣
↪short of
expectations. Global and national institutions work toward but make slow progress␣
↪in
achieving sustainable development goals. Environmental systems experience␣
↪degradation,
although there are some improvements and overall the intensity of resource and␣
↪energy
use declines. Global population growth is moderate and levels off in the second␣
↪half of
the century. Income inequality persists or improves only slowly and challenges to
reducing vulnerability to societal and environmental changes remain.</
↪com:Description>
      </str:Code>
      <str:Code id="3" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2024).3">
        <com:Annotations>
          <com:Annotation id="original-id">
            <com:AnnotationText xml:lang="en">SSP3</com:AnnotationText>
          </com:Annotation>
        </com:Annotations>
        <com:Name xml:lang="en">SSP3: Regional Rivalry</com:Name>
        <com:Description xml:lang="en">SSP3: Regional Rivalry &#8211; A Rocky␣
↪Road

High challenges to mitigation and adaptation.

A resurgent nationalism, concerns about competitiveness and security, and regional
conflicts push countries to increasingly focus on domestic or, at most, regional␣
↪issues.
Policies shift over time to become increasingly oriented toward national and␣
↪regional
security issues. Countries focus on achieving energy and food security goals within
their own regions at the expense of broader-based development. Investments in␣
↪education
and technological development decline. Economic development is slow, consumption is
material-intensive, and inequalities persist or worsen over time. Population␣
↪growth is
low in industrialized and high in developing countries. A low international␣
↪priority for
addressing environmental concerns leads to strong environmental degradation in some
regions.</com:Description>
      </str:Code>
      <str:Code id="4" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2024).4">
        <com:Annotations>
          <com:Annotation id="original-id">
            <com:AnnotationText xml:lang="en">SSP4</com:AnnotationText>
          </com:Annotation>
        </com:Annotations>
        <com:Name xml:lang="en">SSP4: Inequality</com:Name>
```

(continues on next page)

(continued from previous page)

```
          <com:Description xml:lang="en">SSP4: Inequality &#8211; A Road Divided

Low challenges to mitigation, high challenges to adaptation.

Highly unequal investments in human capital, combined with increasing disparities␣
↪in
economic opportunity and political power, lead to increasing inequalities and
stratification both across and within countries. Over time, a gap widens between an
internationally-connected society that contributes to knowledge- and capital-
↪intensive
sectors of the global economy, and a fragmented collection of lower-income, poorly
educated societies that work in a labor intensive, low-tech economy. Social␣
↪cohesion
degrades and conflict and unrest become increasingly common. Technology␣
↪development is
high in the high-tech economy and sectors. The globally connected energy sector
diversifies, with investments in both carbon-intensive fuels like coal and
unconventional oil, but also low-carbon energy sources. Environmental policies␣
↪focus on
local issues around middle and high income areas.</com:Description>
        </str:Code>
        <str:Code id="5" urn="urn:sdmx:org.sdmx.infomodel.codelist.
↪Code=ICONICS:SSP(2024).5">
          <com:Annotations>
            <com:Annotation id="original-id">
              <com:AnnotationText xml:lang="en">SSP5</com:AnnotationText>
            </com:Annotation>
          </com:Annotations>
          <com:Name xml:lang="en">SSP5: Fossil-fueled Development</com:Name>
          <com:Description xml:lang="en">SSP5: Fossil-fueled Development &#8211;␣
↪Taking the Highway

High challenges to mitigation, low challenges to adaptation.

This world places increasing faith in competitive markets, innovation and␣
↪participatory
societies to produce rapid technological progress and development of human capital␣
↪as
the path to sustainable development. Global markets are increasingly integrated.␣
↪There
are also strong investments in health, education, and institutions to enhance␣
↪human and
social capital. At the same time, the push for economic and social development is
coupled with the exploitation of abundant fossil fuel resources and the adoption of
resource and energy intensive lifestyles around the world. All these factors lead␣
↪to
rapid growth of the global economy, while global population peaks and declines in␣
↪the
21st century. Local environmental problems like air pollution are successfully␣
↪managed.
There is faith in the ability to effectively manage social and ecological systems,
including by geo-engineering if necessary.</com:Description>
        </str:Code>
      </str:Codelist>
    </str:Codelists>
  </mes:Structures>
</mes:Structure>
```

```
<mes:Structure xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:com=
↪"http://www.sdmx.org/resources/sdmxml/schemas/v2_1/common" xmlns:data="http://
↪www.sdmx.org/resources/sdmxml/schemas/v2_1/data/structurespecific" xmlns:str=
```

(continues on next page)

(continued from previous page)

```
↪"http://www.sdmx.org/resources/sdmxml/schemas/v2_1/structure" xmlns:mes="http://
↪www.sdmx.org/resources/sdmxml/schemas/v2_1/message" xmlns:gen="http://www.sdmx.
↪org/resources/sdmxml/schemas/v2_1/data/generic" xmlns:footer="http://www.sdmx.
↪org/resources/sdmxml/schemas/v2_1/message/footer">
  <mes:Header>
    <mes:Test>false</mes:Test>
    <mes:Prepared>2023-09-04T16:31:44.700655</mes:Prepared>
    <mes:Source xml:lang="en">Generated by message_ix_models 2023.5.32.
↪dev20+g8d51636</mes:Source>
  </mes:Header>
  <mes:Structures>
    <str:OrganisationSchemes>
      <str:AgencyScheme version="0.1" isExternalReference="false" isFinal="false"␣
↪agencyID="IIASA_ECE" id="AGENCIES" urn="urn:sdmx:org.sdmx.infomodel.base.
↪AgencyScheme=IIASA_ECE:AGENCIES(0.1)">
        <com:Description xml:lang="en">Agencies referenced by data structures in␣
↪message_ix_models</com:Description>
        <str:Agency id="IIASA_ECE">
          <com:Name xml:lang="en">IIASA Energy, Climate, and Environment Program</
↪com:Name>
        </str:Agency>
        <str:Agency id="ICONICS">
          <com:Name xml:lang="en">International Committee on New Integrated␣
↪Climate Change Assessment Scenarios</com:Name>
          <str:Contact>
            <str:URI>https://depts.washington.edu/iconics/</str:URI>
          </str:Contact>
        </str:Agency>
        <str:Agency id="IEA">
          <com:Name xml:lang="en">International Energy Agency</com:Name>
          <str:Contact>
            <str:URI>https://iea.org</str:URI>
          </str:Contact>
        </str:Agency>
      </str:AgencyScheme>
    </str:OrganisationSchemes>
  </mes:Structures>
</mes:Structure>
```

8.24 IIASA Scenario Explorer metadata

Each instance of the IIASA Scenario Explorer (e.g. https://data.ene.iiasa.ac.at/ar6/) requires a distinct collection of information about the structure of model output data to be accepted. The process of providing this structure information is termed **"model registration,"** and is partly documented at https://nomenclature-iamc.readthedocs.io, including the expected directory and file layout. The process must be repeated for additional Scenario Explorer instances.

message_ix_models includes, in `data/iiasa-se/`, files that can be used for such registration. As of 2022-08-15, these are exactly the files use in the NAVIGATE project instance of the Scenario Explorer. These files **should**:

- …match the actual structure and metadata of models/scenarios generated by *message_ix_models*,
- …be used as the basis for, and/or be updated to match, files used for registration in other Scenario Explorer instances.

- *mappings-R12.yaml*

- *def-regions.yaml*

8.24.1 mappings-R12.yaml

This file would be placed at e.g. `mappings/MESSAGEix-GLOBIOM_1.1-R12.yaml` in a registration repository.

```
# Model 'registration' file for use with `nomenclature` package/IIASA Scenario␣
↪Explorer
# workflows
#
# TODO generate programmatically from node codelists

model:
- MESSAGEix-GLOBIOM 1.1-BM-R12
- MESSAGEix-GLOBIOM 1.1-BMT-R12
- MESSAGEix-GLOBIOM 1.1-M-R12
- MESSAGEix-GLOBIOM 1.1-MT-R12

native_regions:
- R12_AFR: MESSAGEix-GLOBIOM 1.1-R12|Sub-saharan Africa
- R12_CHN: MESSAGEix-GLOBIOM 1.1-R12|China
- R12_EEU: MESSAGEix-GLOBIOM 1.1-R12|Eastern Europe
- R12_FSU: MESSAGEix-GLOBIOM 1.1-R12|Former Soviet Union
- R12_LAM: MESSAGEix-GLOBIOM 1.1-R12|Latin America and the Caribbean
- R12_MEA: MESSAGEix-GLOBIOM 1.1-R12|Middle East and North Africa
- R12_NAM: MESSAGEix-GLOBIOM 1.1-R12|North America
- R12_PAO: MESSAGEix-GLOBIOM 1.1-R12|Pacific OECD
- R12_PAS: MESSAGEix-GLOBIOM 1.1-R12|Other Pacific Asia
- R12_RCPA: MESSAGEix-GLOBIOM 1.1-R12|Rest Centrally Planned Asia
- R12_SAS: MESSAGEix-GLOBIOM 1.1-R12|South Asia
- R12_WEU: MESSAGEix-GLOBIOM 1.1-R12|Western Europe

common_regions:
- World:
  - R12_AFR
  - R12_CHN
  - R12_EEU
  - R12_FSU
  - R12_LAM
  - R12_MEA
  - R12_NAM
  - R12_PAO
  - R12_PAS
  - R12_RCPA
  - R12_SAS
  - R12_WEU
# R5 regions
- R5ASIA:
  - R12_CHN
  - R12_PAS
  - R12_RCPA
  - R12_SAS
- R5LAM:
  - R12_LAM
- R5MAF:
  - R12_AFR
  - R12_MEA
- R5OECD90+EU:
  - R12_EEU
```

(continues on next page)

(continued from previous page)

```
  - R12_NAM
  - R12_PAO
  - R12_WEU
- R5REF:
  - R12_FSU
# Individual countries
- CHN:
  - R12_CHN
- EU:
  - R12_EEU
  - R12_WEU
- IND:
  - R12_SAS
- USA:
  - R12_NAM
```

8.24.2 `def-regions.yaml`

This file would be placed at e.g. `definitions/region/MESSAGEix-GLOBIOM_1.1-R12.yaml` in a registration repository.

```
- MESSAGEix-GLOBIOM 1.1-R12:
  - MESSAGEix-GLOBIOM 1.1-R12|Sub-saharan Africa
  - MESSAGEix-GLOBIOM 1.1-R12|China
  - MESSAGEix-GLOBIOM 1.1-R12|Rest Centrally Planned Asia
  - MESSAGEix-GLOBIOM 1.1-R12|Eastern Europe
  - MESSAGEix-GLOBIOM 1.1-R12|Former Soviet Union
  - MESSAGEix-GLOBIOM 1.1-R12|Latin America and the Caribbean
  - MESSAGEix-GLOBIOM 1.1-R12|Middle East and North Africa
  - MESSAGEix-GLOBIOM 1.1-R12|North America
  - MESSAGEix-GLOBIOM 1.1-R12|Pacific OECD
  - MESSAGEix-GLOBIOM 1.1-R12|Other Pacific Asia
  - MESSAGEix-GLOBIOM 1.1-R12|South Asia
  - MESSAGEix-GLOBIOM 1.1-R12|Western Europe
```

8.25 Development practices

This page describes development practices for *`message_ix_models`* and `message_data` intended to help reproducibility, interoperability, and reusability.

In the following, the bold-face words **required**, **optional**, etc. have specific meanings as described in IETF RFC 2119.

On other pages:

- Contributing to development in the MESSAGEix docs. *All* of these apply to contributions to *`message_ix_models`* and `message_data`, including the Code style.
- *Data, metadata, and configuration*, for how to add and handle these.

On this page:

8.25.1 Advertise and check compatibility

There are multiple choices of the base structure for a model in the MESSAGEix-GLOBIOM family, e.g. different *Node code lists* and *Years or time periods (year/*.yaml)*.

Code that will only work with certain structures…

- **must** be documented, and include in its documentation any such limitation, e.g. "`example_func()` only produces data for R11 and year list B."
- **should** use *`check_support()`* in individual pieces of code to pre-emptively check and raise an exception. This prevents inadvertent use of the code where its data will be invalid:

```
def myfunc(context, *args):
    """A function that only works on R11 and years 'B'."""

    check_support(
        context,
        dict(regions=["R11"], years=["B"]),
        "Example data produced"
    )

    # … function code to execute if the check passes
```

Code **may** also check a *`Context`* instance and automatically adapt data from certain structures to others, e.g. by interpolating data for certain periods or areas. To help with validation, code that does this **should** log on the `logging.INFO` level to advertise these steps.

8.25.2 Upstream version policy

`message_ix_models` is developed to be compatible with the following versions of its upstream dependencies.

`ixmp` and `message_ix`

The most recent 4 minor versions, or all minor versions released in the past two (2) years—whichever is greater.

For example, as of 2024-04-08:

- The most recent release of `ixmp` and `message_ix` are versions 3.8.0 of each project. These are supported by *`message_ix_models`*.
- The previous 3 minor versions are 3.7.0, 3.6.0, and 3.5.0. All were released since 2022-04-08. All are supported by `message_ix_models`.
- `ixmp` and `message_ix` versions 3.4.0 were released 2022-01-24. These this is the fifth-most-recent minor version *and* was released more than 2 years before 2024-04-08, so it is not supported.

Python

All currently-maintained versions of Python.

The Python website displays a list of these versions (1, 2).

For example, as of 2024-04-08:

- Python 3.13 is in "prerelease" or "feature" development status, and is *not* supported by *`message_ix_models`*.
- Python 3.12 through 3.8 are in "bugfix" or "security" maintenance status, and are supported by *`message_ix_models`*.
- Python 3.7 and earlier are in "end-of-life" status, and are not supported by the Python community or by *`message_ix_models`*.

- Support for older versions of dependencies **may** be dropped as early as the first `message_ix_models` version released after changes in upstream versions.
  - Conversely, some parts of `message_ix_models` **may** continue to be compatible with older upstream versions, but this compatibility is not tested and may break at any time.
  - Users **should** upgrade their dependencies and other code to newer versions; we recommend the latest.
- Some newer code is marked with a `minimum_version()` decorator.
  - This indicates that the marked code relies on features only available in certain upstream versions (of one of the packages mentioned above, or another package), newer than those listed in pyproject.toml.
  - These minima **must** be mentioned in the `message_ix_models` documentation.
  - Users wishing to use this marked code **must** use compatible versions of those packages.

8.26 What's new

8.26.1 v2024.4.22

- Incorporate the *MESSAGEix-GLOBIOM global model* documentation (PR #107, PR #110). This documentation formerly lived at https://docs.messageix.org/global/ and in a separate repository at iiasa/message_doc.
- Improve tool for *Migrate individual modules using git filter-repo* (PR #174); expand documentation.
- New module for *Investment and fixed costs (tools.costs)* (PR #99).
- Update investment cost assumptions in *Investment and fixed costs (tools.costs)* for wind and solar technologies (PR #176).
  - Remove manually specified base year reference region costs for solar_pv_ppl and solar_pv_RC so that 2021 IEA WEO costs are used for these technologies.
  - Fix the manually specified base year reference region cost for wind_ppf.
  - Update cost reduction rates and scenarios for wind_ppf so that it follows the same narratives as wind_ppl.
- Convert Intratec data for *Investment and fixed costs (tools.costs)* from Excel to a simpler CSV format. (PR #167).
- Migrate *'Legacy' reporting (report.legacy)* to provide post-processing functionality for the *global model snapshot* (PR #159).
- Migrate and improve code for four sources of exogenous data (PR #162): `project.gea`, `project.shape`, `tools.gfei`, and `tools.iea.eei`.
- Add a *Quick-start guide to running a MESSAGEix-GLOBIOM baseline model* (PR #157).
- Expand *Data, metadata, and configuration* (PR #161).
- Add an explicit *Upstream version policy* (PR #162).

8.26.2 v2024.4.2

- The `SSPUpdate` data provider pulls data from the SSP 2024 "Release 3.0" data files, and handles both the earlier and current structures (PR #156).
- Improve `ExoDataSource` with `raise_on_extra_kw()` utility method, automatic copy of source keyword arguments (PR #156).
- Expose `node.nodes_ex_world()` for use as a genno (reporting) operator.
- Raise DeprecationWarning from `util.sdmx.eval_anno()`; remove internal usage of this deprecated method (PR #156).

- Reduce verbosity when using the **mix-models** CLI when `message_data` is not installed (GH #37, PR #156).
- Improve logging (PR #156).
  - Use multi-threaded logging for better performance. Logging to stdout and file is on a separate thread and does not block operations on the main thread.
  - Add automatic file logging. Log versions of packages to file when using *`workflow.make_click_command()`*.
  - New CLI command **mix-models last-log** to retrieve the location of the latest log file.
- Update *Command-line interface* (PR #156).
- Improve performance in *`disutility.data_conversion()`* (PR #156).
- Use `platformdirs.user_cache_path()` in more places; remove cache-path handling code (PR #156).
- Add *`util.datetime_now_with_tz()`* (PR #156).
- Add *`util.show_versions()`*, wrapping `ixmp.util.show_versions()` and returning its output as `str` (PR #156).
- *`util.private_data_path()`* returns an alternate, local data path if `message_data` is not installed (PR #156).
- Annotate `c="transport"` in *the commodity code list* with associated *IEA (E)WEB* flows (PR #153).

8.26.3 v2024.1.29

- Add *(Extended) World Energy Balances (tools.iea.web)* for handling data from the International Energy Agency (IEA) Extended World Energy Balances (GH #25, PR #75).
- Add *World Bank structures (tools.wb)* and *`assign_income_groups()`* to assign MESSAGE regions to World Bank income groups (PR #144).
- Adjust *`report.compat`* for genno version 1.22 (GH #141, PR #142).
- Raise informative exception from *`ScenarioInfo.io_units()`* (PR #151).

8.26.4 v2023.11.24

Migration notes

Update code that imports from the following modules:

- `message_ix_models.report.computations` → use `message_ix_models.report.operator`.

Code that imports from the old locations will continue to work, but will raise `DeprecationWarning`.

Data for *MESSAGEix-Nexus (model.water)* is no longer included in the PyPI distributions for *`message_ix_models`*. This reduces the package size from >20 MB to <5 MB. To automatically download and unpack these data into a local directory, use **mix-models fetch MESSAGEix-Nexus**.

All changes

- Improve *ExoDataSource* (PR #137):
  - New attributes *name*, *extra_dims*.
  - New method *transform()* that can be overridden by subclasses.
  - New arguments `archive_member`, `non_iso_3166` to *iamc_like_data_for_query()*.
- New provider for exogenous data from the *ADVANCE* project (PR #137). This module, *project.advance*, supersedes *tools.advance* and its idiosyncratic API, which are deprecated.
- New CLI commands (PR #137):
  - **mix-models testing fuzz-private-data**, superseding **mix-models ssp make-test-data**.
  - **mix-models fetch**, superseding **mix-models snapshot fetch**.
- New utility functions (PR #137).
  - *tools.iamc.describe()* to generate SDMX code lists that describe the structure of particular IAMC-format data (PR #137).
  - *workflow.make_click_command()* to generate `click` commands for any *Workflow*.
  - *util.minimum_version()* to ensure compatibility with upstream packages and aid test writing.
  - *util.iter_keys()* to generate keys for chains of `genno` computations.
- Add *message_ix_models.report.compat for emulating legacy reporting* (PR #134).
- Rename *message_ix_models.report.operator* (PR #137).
- Deprecate `iter_parameters()` in favour of `ixmp.Scenario.par_list()` with `indexed_by=...` argument from ixmp v3.8.0 (PR #137).

8.26.5 v2023.10.16

- New providers for exogenous data from the *SSPOriginal* and *SSPUpdate* (PR #125) sources.
- Improved *ScenarioInfo* (PR #125):
  - New attributes *model*, *scenario*, *version*, and (settable) *url*; class method *from_url()* to allow storing `Scenario` identifiers on ScenarioInfo objects.
  - New property *path*, giving a valid path name for scenario-specific file I/O.
- Improvements to *report* (PR #125):
  - New *report.Config* class collecting recognized settings for the module.
  - `context["report"]` always exists as an instance of *report.Config*.
  - New submodule *report.plot* with base class and 5 plots of time-series data stored on Scenarios.
  - Submodule *report.sim* provides *add_simulated_solution()* for testing reporting configuration.
  - New operator *filter_ts()*.
- New reusable command-line option **--urls-from-file** in *util.click* (PR #125).
- Add pyarrow to dependencies (PR #125).

8.26.6 v2023.9.12

All changes

- New module *project.ssp* (PR #122) to generate SDMX codelists for the 2017/original SSPs and the 2024 update, and provide these as Enum to other code.
- New module *tools.exo_data* to retrieve exogenous data for, among others, population and GDP (PR #122). This module has a general API that can be implemented by provider classes.
- New function *model.emissions.get_emission_factors()* and associated data file to provide data from this table in the MESSAGEix-GLOBIOM documentation (PR #122).
- New functions in *util.sdmx* (PR #122):
  - *read()*, *write()* to retrieve/store package data in SDMX-ML.
  - *make_enum()* to make pure-Python Enum (or subclass) data structures based on SDMX code lists.
- *same_node()* also fills "node_shares", "node_loc", and "node", as appropriate (PR #122).

Deprecations

- *eval_anno()* is deprecated; code should instead use sdmx.model.common.AnnotableArtefact.eval_annotation(), which provides the same functionality.

8.26.7 v2023.9.2

- New module *message_ix_models.report* for reporting (PR #116). Use of this module requires ixmp and message_ix version 3.6.0 or greater.
- Add documentation on *Migrate individual modules using git filter-repo* using **git filter-repo** and helper scripts (PR #89).

8.26.8 v2023.7.26

- Add code and CLI commands to *fetch and load MESSAGEix-GLOBIOM snapshots* (PR #102). Use of this module requires ixmp and message_ix version 3.5.0 or greater.
- Add *util.pooch.fetch()*, a thin wrapper for using Pooch (PR #102).
- New module *message_ix_models.model.macro* with utilities for calibrating message_ix.macro (PR #104).
- New method *Workflow.guess_target()* (PR #104).
- Change in behaviour of *Workflow.add_step()*: the method now returns the name of the newly-added workflow step, rather than the *WorkflowStep* object added to carry out the step (PR #104). The former is more frequently used in code that uses *Workflow*.
- Add the *17-region aggregation (R17)* node code list (PR #109).
- Add the *20-region aggregation (R20)* node code list (PR #109).

8.26.9 v2023.5.31

- Adjust `sdmx` usage for version 2.10.0 (PR #101).

8.26.10 v2023.5.13

- Adjust *`generate_product()`* for pandas 2.0.0 (PR #98).

8.26.11 2023.4.2

- Add *MESSAGEix-Nexus (model.water)* (PR #88, PR #91).
- New utility function *`replace_par_data()`* (PR #90).
- *`disutility.get_spec()`* preserves all `Annotations` attached to the `Code` object used as a template for usage technologies (PR #90).
- Add `CO2_Emission_Global_Total` to the *"A" relation codelist* (PR #90).
- *`Adapter`* and *`MappingAdapter`* can be imported from *`message_ix_models.util`* (PR #90).
- Bump `sdmx` requirement from v2.2.0 to v2.8.0 (PR #90).

8.26.12 2023.2.8

- Codelists for the `relation` MESSAGEix set (PR #85):
  - Add *three relation codelists*.
  - The *"bare" reference energy system* now includes relations from the codelist indicated by *`model.Config.relations`*; default "A".
- *Commodities (commodity.yaml)* (PR #85):
  - Add "biomass", "non-comm", "rc_spec", and "rc_therm".
  - Add "report" annotations for some items. These include string fragments to be used in variable names when reporting data in the IAMC data structure.
- *`generate_product()`* (and *`generate_set_elements()`*) can handle a regular expression to select a subset of codes for the Cartesian product (PR #85).
- New utility method *`Context.write_debug_archive()`* writes a ZIP archive containing files listed by *`Config.debug_paths`* (PR #85).
- *`WorkflowStep`* can store and apply keyword options for the optional `clone()` step at the start of the step execution (PR #85).
- Bugfix: *`WorkflowStep.__call__()`* ensures that *`Config.scenario_info`* on the *`Context`* instance passed to its callback matches the target scenario (PR #85).

8.26.13 2022.11.7

- Add the *Zambia (ZMB)* node code list (PR #83).
- Add the utility `same_time()`, to copy the set time in parameters (PR #83).
- New `Config` and `model.Config` `dataclasses` for clearer description/handling of recognized settings stored on `Context` (PR #82). `ConfigHelper` for convenience/utility functionality in `message_ix_models`-based code.
- New functions `generate_product()`, `generate_set_elements()`, `get_region_codes()` in `model.structure` (PR #82).
- Revise and improve the *Workflow API* (PR #82).
- Adjust for pandas 1.5.0 (PR #81).

8.26.14 2022.8.17

- Add `nodes_ex_world()` and use this in `disutility.data_conversion()` instead of expected a "World" node ID to be the first element in `ScenarioInfo.N` (PR #78).
- Add example files and documentation for *IIASA Scenario Explorer metadata* (PR #78).
- Expand ~ (i.e. `$HOME`) in the `"message local data"` *configuration setting* (PR #78).

8.26.15 2022.7.25

- Add `get_advance_data()`, and related tools for data from the ADVANCE project, including the *node codelist* for the data (PR #76).
- Add unit annotations to *Commodities (commodity.yaml)* (PR #76).
- New utility methods `ScenarioInfo.io_units()` to derive units for `input` and `output` parameters from `units_for()` commodity stocks and technology activities (PR #76).
- Transfer `add_tax_emission()` from `message_data`, improve, and add tests (PR #76).
- Unit annotations on commodity and technology codes are copied to child codes using `process_units_anno()` (PR #76).
- `make_matched_dfs()` accepts `pint.Quantity` to set both magnitude and units in generated data (PR #76).
- `strip_par_data()` also removes the set element for which data is being stripped (PR #76).
- The common CLI options **`--verbose`** and **`--dry-run`** are stored on `Context` automatically (PR #76).
- New utility method `Context.set_scenario()` (PR #76).
- `iam_units.registry` is the default unit registry even when `message_data` is not installed (PR #76).
- Expand `broadcast()` to allow `DataFrame` with multiple dimensions as input (PR #74).

8.26.16 2022.5.6

- Bump minimum required version of `message_ix` to v3.4.0 from v3.2.0 (PR #71).
- Add a documentation page on *Distributed computing* (PR #59).
- Add `testing.not_ci()` for marking tests not to be run on continuous integration services; improve `session_context()` (PR #62).
- `apply_spec()` also adds elements of the "node" set using `ixmp.Platform.add_region()` (PR #62).
- Add new logo the documentation (PR #68).
- Add `Workflow`; see *Multi-scenario workflows (workflow)* (PR #60).

8.26.17 2022.3.30

- Add `adapt_R11_R12`, a function for adapting data from the *11-region aggregation (R11)* to the *12-region aggregation (R12)* node lists (PR #56).
- Work around iiasa/ixmp#425 in `disutility.data_conversion()` (*docs*, PR #55).

8.26.18 2022.3.3

- Change the node name in R12.yaml from R12_CPA to R12_RCPA (PR #49).
- Register "message local data" ixmp configuration file setting and use to set the `.Context.local_path` when provided. See *(4) Other, system-specific ("local") directories* (PR #47)

8.26.19 2022.1.26

- New `Spec` class for easier handling of specifications of model (or model variant) structure (PR #39)
- New utility function `util.local_data_path()` (PR #39).
- `repr()` of `Context` no longer prints a (potentially very long) list of all keys and settings (PR #39).
- `as_codes()` accepts a `dict` with `Code` values (PR #39).

8.26.20 Earlier releases

2021.11.24

- Add **`--years`** and **`--nodes`** to `common_params()` (PR #35).
- New utility function `structure.codelists()` (PR #35).

2021.7.27

- Improve caching using mod:*genno* v1.8.0 (PR #29).

2021.7.22

- Migrate utilities *cached()*, *check_support()*, *convert_units()*, *maybe_query()*, *series_of_pint_quantity()* (PR #27)
- Add *testing.NIE*.
- Add the `--jvmargs` option to **pytest** (see *pytest_addoption()*).
- Remove `.Context.get_config_file()`, `.get_path()`, `.load_config()`, and `.units`, all deprecated since 2021-02-28.

2021.7.6

- Add *identify_nodes()*, a function for identifying a *Node code lists* based on a `Scenario` (PR #24).
- Add *adapt_R11_R14*, a function for adapting data from the *11-region aggregation (R11)* to the *14-region aggregation (R14)* node lists (PR #24).
- Add *export_test_data()* and **mix-models export-test-data** command (PR #16). See *Prepare data for testing*.
- Allow use of pytest's persistent cache across test sessions (PR #23). See *Reproducibility*.
- Add the *12-region aggregation (R12)* node code list (PR #14).

2021.4.7

- Add *model.disutility*, code for setting up structure and data for generalized consumer disutility (PR #13)

2021.3.24

- Add *Years or time periods (year/*.yaml)*, YAML data files, *ScenarioInfo.year_from_codes()* and associated tests (GH #11, PR #12)

2021.3.22

- Migrate *model.bare*, *model.build*, `model.cli`, and associated documentation (PR #9)
- Migrate utilities: *ScenarioInfo*, *add_par_data()*, *eval_anno()*, `iter_parameters()`, and *strip_par_data()*.

2021.3.3

- Migrate *util.click*, *.util.logging*; expand documentation (PR #8:).
- *Context.clone_to_dest()* method replaces `clone_to_dest()` function.
- Build PDF documentation on ReadTheDocs.
- Allow CLI commands from both *message_ix_models* and `message_data` via **mix-models**.
- Migrate **mix-models techs** CLI command.

2021.2.28

- Migrate *Context* class and *testing* module from message_data (PR #5:).
- Add *load_private_data()*, *package_data_path()*, *private_data_path()*.
- Document: *Data, metadata, and configuration* and *Command-line interface*.
- Update *node codelists* to ensure they contain both current and former ISO 3166 codes for countries that have changed status (PR #6:). For instance, ANT dissolved into BES, CUW, and SXM in 2010; all four are included in R11_LAM so this list can be used to handle data from either before or after 2010.

2021.2.26

- Add *get_codes()* and related code lists (PR #2:).
- Add *MessageDataFinder* and document *Migrating from message_data* (PR #3:).

2021.2.23

Initial release.

8.27 Migrating from message_data

message_ix_models coexists with the private repository/package currently named message_data. The latter is the location for code related to new research that has not yet been completed and published, data that must remain closed-source permanently, etc.

Over time:

- All other code will be migrated from message_data to *message_ix_models*.
- Code and data for individual projects will be moved from message_data to *message_ix_models* at a suitable point during the process of publication. (This point may vary from project to project.)
- message_data may be renamed.

8.27.1 Using both packages together

This section gives some practices and tips for using the two packages together.

Always import via *message_ix_models*

The package installs *MessageDataFinder* into Python's import system (importlib), which changes its default behaviour as follows: if

1. A module message_ix_models.model.*model_name* or message_ix_models.project.*project_name* is imported, and
2. This module does not actually exist in *message_ix_models*,

3. Then the code will instead file the respective modules `message_data.model.model_name` or `message_data.project.project_name`.

Even when using code that currently or temporarily lives in `message_data`, access it like this:

```
# Code in message_data/model/mymodelvariant.py
from message_ix_models.model import mymodelvariant

mymodelvariant.build(...)
```

This code is *future-proof*: it will not need adjustment if/when "mymodelvariant" is eventually moved from `message_data` to *message_ix_models*.

Use the `mix-models` command-line interface (CLI)

All CLI commands and subcommands defined in `message_data` are also made available through the *message_ix_models* CLI, the executable **`mix-models`**.

Use this program in documentation examples and in scripts. In a similar manner to the point above, these documents and scripts will remain correct if/when code is moved.

Don't import from `message_data` in *message_ix_models*

The open-source code **should** not depend on any private code. If this appears necessary, the code in `message_data` can probably be moved to *message_ix_models*.

Use *`message_ix_models.tools`* and *`util`* in `message_data`

The former have stricter quality standards and are more transparent, which is better for reproducibility.

At some points, similar code may appear in both packages as it is being migrated. In such cases, always import and use the code in *message_ix_models*, making any adjustments that are necessary.

8.27.2 Migrate individual modules using `git filter-repo`

This section describes a general process for migrating (sub)modules of `message_data` or other repositories, private or public, to *message_ix_models*. Using this process preserves the commit and development history of code and data. This is useful for future development, and can contain important methodological and research insights that would be lost with a simple copy.

The process:

- Uses the code in `message_ix_models/util/migrate.py`. This is an entirely stand-alone Python script.
- Has been tested on Ubuntu Linux.
- May need modification depending on the code to which it is applied.

Requirements:

- Install **`git lfs`**.
- Install the `migrate` optional dependencies for *message_ix_models*:

```
$ pip install message-ix-models[migrate]
```

Read through all the steps before starting.

0. Create a temporary directory:

```
$ mkdir tmp
$ cd tmp
```

This directory will hold *new clones* of both repositories. We use new clones to avoid interacting with local settings, uncommitted (ignored) files, or history from other work, and so we can safely delete an incomplete migration and start again.

1. In the temporary directory, run:

```
$ python -m message_ix_models.util.migrate step-1
```

This copies the `migrate.py` module into the temporary directory from (0).

Edit the file, particularly the variables `SOURCE`, `TARGET`, and `BATCH`. Use the section "Using **`git filter-repo`**," below, and comments in the file as a guide to the necessary changes.

2. Run:

```
$ python migrate.py step-2
```

This step:

- Clones the source and target repositories into directories with names like `source-a1b` and `target-2c3`.
- Fetches all available Git LFS objects associated with any commit in the source repository. These are needed as the history is replayed in the next step.

  If the source repository is `message_data`, this will download up to 6 GB of data from GitHub, so it can be slow. The `source-*` directory is not modified during the rest of the process, so if you do not modify it, this step will not need repeating.
- Creates symlinks pointing from `target-*/.git/lfs/objects/…` to `source-*/.git/lfs/objects/…`. This makes it appear as if the LFS objects are locally stored and available to the target repo.

3. Run:

```
$ python migrate.py step-3
```

This step:

- Connects the two repos together, with the target repo seeing the source repo as a Git remote. (Note that in Git terminology, 'remote' does not necessarily mean "on another machine". In this case, the remote is just located in a different directory.)
- Fetches the source branch.
- Rewrites the source branch history according to the rules in `BATCH`.
- Writes a file `rebase-todo.in` to be used in the next step.

4. Prepare for **`git rebase`**.

   Make a copy of the file `rebase-todo.in`—for instance, `rebase-todo.txt`—and open the copy. This file contains a list of commands for the rebase. You can edit this list before using it in step (5); if needed, restore the list by making a fresh copy of the original.

   To help with this, `duplicate-messages.txt` contains a list of identical commit messages that appear more than once in the history. These commits *may*—not necessarily—be indication of a *non-linear history*. This can occur when branches with similar commit names but different contents are merged together (despite our best efforts, this sometimes happens on `message_data`).

   Some changes you can make to `rebase-todo.txt`:

   - Remove lines for duplicated commits, per `duplicate-messages.txt`. This avoids commanding **`git`** to apply the same changes more than once, which can lead to conflicts. You could:
     - Keep only the *first* of two or more occurrences of duplicate commits.
     - Keep only the *last* of two or more occurrences of duplicate commits.
     - Use any other strategy to minimize conflicts.
   - Remove lines for merge commits. These are ignored by **`git rebase`** and **`git filter-repo`**, but you may need to manually skip them if you do not remove them at this step.
   - Add blank lines and comments to help yourself read the history.

5. Perform the rebase. Run the following; choose any name you like instead of `migrate-example`

```
$ git checkout -b migrate-example source-branch
$ git rebase --interactive --empty=drop main
```

   Replace the to-do list for the rebase with the one prepared in step (4).

   - One way to do this:
     - In the editor that opens, delete *everything*.
     - Paste in the contents of `rebase-todo.txt`.
     - Save the file and exit.
   - Another way:
     - Insert a single line with the text `break` at the top of the existing TODO list.
     - Save the file and exit. The rebase will begin, but stop before picking the first commit.
     - Open the file `.git/rebase-merge/git-rebase-todo` in a different editor; replace its contents with `rebase-todo.txt`, and save.
     - Run **`git rebase --continue`**.

   The interactive rebase begins.

   - Resolve any conflicts that arise in the usual way. After resolving, perhaps run:

```
$ git add --update && git status
$ git rebase --continue
```

   - If you see a message like the following:

```
error: commit 47db89c0128e6edf19ebb9ffbcea1d5da4d25176 is a merge but no -
↪m option was given.
hint: Could not execute the todo command
hint:
hint:     pick 47db89c0128e6edf19ebb9ffbcea1d5da4d25176 Merge pull request
↪#123 from iiasa/foo/bar
hint:
hint: It has been rescheduled; To edit the command before continuing,␣
↪please
hint: edit the todo list first:
hint:
hint:     git rebase --edit-todo
hint:     git rebase --continue
```

     …follow these instructions:

     1. Give **`git rebase --edit-todo`**.
     2. Delete the line/command related to the merge commit.
     3. Save and exit.
     4. Give **`git rebase --continue`**.

   - If many conflicts occur, you may run:

```
$ git rebase --abort
```

     Then, return to step (4) to adjust the list of commands, considering the history and apparent conflicts.

6. Push to `iiasa/message-ix-models`:

```
$ git push --set-upstream=origin migrate-example
```

…and open a pull request.

This can be initially a "draft" state, until you complete step (7). The pull request is partly to help you diagnose whether the above steps produced a reasonable result. The branch can also be inspected by others, for instance to compare it to the source repository.

7. Clean up.

   This *may* be done directly on the branch from (6). However, a better option to create a secondary branch from the head of (6), named like `migrate-example-tidy`, and make clean-up commits to this branch. Create a second pull request to merge this manual clean-up branch into the branch from (6). This way, if steps (1–6) need to be repeated, a new history can be force-pushed to `migrate-example`, and then the manual clean-up branch can be rebased on the newly updated `migrate-example` branch, with little disturbance.

   Push further changes to the clean-up branch to:

   - Modify imports and references.

     For example, when migrating `message_data.model.foo`, statements like:

     ```
     from message_data.model.foo.bar import baz
     ```

     …must be modified to:

     ```
     from message_ix_models.model.foo.bar import baz
     ```

     Similar changes must be made to intersphinx references in the documentation.

   - Adjust data handling.

     For example, usage of *`private_data_path()`* to locate data files must be modified to *`package_data_path()`* if the data files were moved during the migration. Tests can help to ensure that these changes are effective.

   - Address CI checks. For example:
     - Add tests, or exclude files from test coverage.
     - Lint files, or exclude files from linting.

   It is important to avoid *scope creep*: do not try to include large modifications, improvements, or refactoring of code in this step. This will greatly increase the complexity of the task and make it harder to complete. Instead, do these things either *after* or *before* migrating the code.

8. Invite review of your PR(s).
9. Merge the clean-up branch from (7) into (6), and then (6) into `main`.

To restart at any time, run **`python migrate.py reset`** from your temporary directory to delete the clone of *`message_ix_models`* and all other changes from steps (3–5). Then begin from step (2).

`git filter-repo` features and options

`git-filter-repo` (docs) is a powerful tool for rewriting **`git`** history. It has many command-line options and features.

`migrate.py` and `BATCH` use these features to, in particular:

- Move code. For example, all commits pertaining to a file like `message_data/model/foo/bar.py` are preserved, except they now appear to describe changes to `message_ix_models/model/foo/bar.py`.
- Move data. Data is moved from the unpackaged, private, top-level `data/` directory in `message_data`, to the packageable `message_ix_models/data/` directory. There are further considerations; see *Data, metadata, and configuration* and below.
- Discard everything else relating to `message_data` (or the source repo), especially other code and data that are *not* going to be migrated, according to your settings in step (1).

- Partly clean up commit messages that do not match the code style, for instance by ensuring they start with a capital letter.

These commands are **batched** when they cannot be given simultaneously in a single call to **`git filter-repo`**.

Below are some examples:

Listing 8.1: `migrate.py` config section, used in PR #107

```
S = SOURCE = RepoInfo(
    url="git@github.com:iiasa/message_doc.git",
    branch="main",
)

T = TARGET = RepoInfo(
    url="git@github.com:iiasa/message-ix-models.git",
    branch="main",
)

BATCH = (
    dict(
        args=[
            "--path-rename=:doc/global/",
            "--path-rename=doc/global/_static/:doc/_static/",
            "--replace-message=../replacements.txt",
        ],
        message_callback=message_callback,
    ),
    dict(
        args=["--invert-paths", "--path=doc/_static/combined-logo-white.png"],
    ),
)
```

Listing 8.2: `requirements.txt`, used in PR #107

```
regex:^(Add|Correct|Edit|Insert|Switch|Try)(ed|ing)==>\1
regex:^(Chang|Integrat|Remov|Renam|Updat)(ed|ing)==>\1e
regex:^Citation$==>Edit citation
Formatted==>Format
```

Listing 8.3: `migrate.py` config section, used in PR #88

```
S = SOURCE = RepoInfo(
    url="git@github.com:iiasa/message_data.git",
    branch="dev",
)

T = TARGET = RepoInfo(
    url="git@github.com:iiasa/message-ix-models.git",
    branch="main",
)

# Path fragment for using in BATCH
MOD = "water"

BATCH = (
    # Use --path-rename to rename several paths and files under them:
    # Use --message-callback to rewrite some commit messages, capitalizing the␣
↪first letter.
    dict(
        args=[
            # Add or remove lines here as necessary; not all modules have all the␣
```

(continues on next page)

(continued from previous page)

```
↪following
            # pieces, and some modules have additional pieces.
            #
            # Module data.
            f"--path-rename=data/{MOD}/:{T.base}/data/{MOD}/",
            # Module code. The "/model/" path fragment could also be "/project/",␣
↪or removed
            # entirely.
            f"--path-rename={S.base}/model/{MOD}/:{T.base}/model/{MOD}/",
            # Module tests.
            f"--path-rename={S.base}/tests/model/{MOD}/:{T.base}/tests/model/{MOD}/
↪",
        ],
        message_callback=message_callback
    ),
    #
    # Use --path to keep only a subset of files and directories.
    #
    # This has the effect of discarding the top-level message_data and data␣
↪directories,
    # keeping only message_ix_models. This operates on the paths renamed by the␣
↪previous
    # command. It would be possible to combine in a single command, but we would␣
↪then
    # need to specify the *original* paths to keep.
    dict(
        args=[
            f"--path={T.base}",
            #
            # Can add lines to keep other files, for instance:
            # f"--path=doc/{MOD}/",
        ],
    ),
    #
    # Use --invert-paths to *remove* some specific files, e.g. non-reporting test␣
↪data.
    dict(
        args=[
            "--invert-paths",
            f"--path-regex=^{T.base}/tests/data/[^r].*$",
        ],
    ),
)
```

After migrating

Some follow-up actions that **may** or **should** take place after the migration is complete:

- Discuss with the *message_ix_models* maintainers about releasing a new version of the package, so that the code is available in a released version.
- Open (an) additional issue(s) or PR(s) to record or immediately address missing items—for example, documentation, tests, or small enhancements for reusability—that were identified during the migration.
- Open a PR to *remove* the migrated code from `message_data`. This is important because future development should target the code in its new home in *message_ix_models*; other projects, workflows, and colleagues should be discouraged to depend on the old code in `message_data`, where it may not receive updates.

  The simplest way to do this is to delete the code entirely and adjust any other code that imports it to import from the new location in *message_ix_models*. For temporary compatibility, it is also possible to use `message_data.tools.migrated()`.

References

git and **git filter-repo** are both flexible programs with plenty of power and flexibility. The above is one suggested way of using them to achieve a clean, history-preserving migration, but there are alternate options.

- **git filter-repo** README, user manual, and discussions
- **git rebase** documentation, and in Chapter 3.6 of the Git Book.
- The description of PR #86 describes an alternate process.
- PRs that used this process include:
  - PR #88 + PR #91, plus this comment showing the manual edits to `rebase-todo.txt`.
  - PR #107 + PR #110.

8.28 Releasing

8.28.1 Version numbers

message_ix_models uses date-based version numbers like `Y.M.D`. Thus version `2021.2.23` is released on 23 February 2021. This is to establish a more direct correspondence between outputs of the code and the version(s) used to produce it.

8.28.2 Procedure

Before releasing, check:

- https://github.com/iiasa/message-ix-models/actions?query=branch:main to ensure that the push and scheduled builds are passing.
- https://readthedocs.com/projects/iiasa-energy-program-message-ix-models/builds/ to ensure that the docs build is passing.

Address any failures before releasing.

1. Edit `doc/whatsnew.rst`. Comment the heading "Next release", then insert another heading below it, at the same level, with the version number and date. Make a commit with a message like "Mark vX.Y.Z in doc/whatsnew".
2. Tag the release candidate version, i.e. with a `rcN` suffix, and push:

   ```
   $ git tag v1.2.3rc1
   $ git push --tags origin main
   ```

3. Check:
   - at https://github.com/iiasa/message-ix-models/actions?query=workflow:publish that the workflow completes: the package builds successfully and is published to TestPyPI.
   - at https://test.pypi.org/project/message-ix-models/ that:
     - The package can be downloaded, installed and run.
     - The README is rendered correctly.

   Address any warnings or errors that appear. If needed, make a new commit and go back to step (2), incrementing the rc number.
4. (optional) Tag the release itself and push:

   ```
   $ git tag v1.2.3
   $ git push --tags origin main
   ```

This step (but *not* step (2)) can also be performed directly on GitHub; see (5), next.

5. Visit https://github.com/iiasa/message-ix-models/releases and mark the new release: either using the pushed tag from (4), or by creating the tag and release simultaneously.
6. Check at https://github.com/iiasa/message-ix-models/actions?query=workflow:publish and https://pypi.org/project/message-ix-models/ that the distributions are published.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] Environmental Protection Agency (EPA). Global Mitigation of Non-CO2 Greenhouse Gases: 2010-2030. 2013. URL: https://www3.epa.gov/climatechange/Downloads/EPAactivities/MAC_Report_2013.pdf.

[2] International Energy Agency. Energy Balances. Technical Report, International Energy Agency, 2012.

[3] International Energy Agency. World Energy Outlook 2014. Technical Report, International Energy Agency, 2014. URL: http://www.worldenergyoutlook.org/weo2014/.

[4] Nikos Alexandratos and Jelle Bruinsma. World agriculture towards 2030/2050: the 2012 revision. Report 12-03, FAO, June 2012.

[5] Markus Amann, Imrich Bertok, Jens Borken-Kleefeld, Janusz Cofala, Chris Heyes, Lena Hoglund-Isaksson, Zbigniew Klimont, Binh Nguyen, Maximilian Posch, Peter Rafaj, Robert Sandler, Wolfgang Schopp, Fabian Wagner, and Wilfried Winiwarter. Cost-effective control of air quality and greenhouse gases in Europe: Modeling and policy applications. *Environmental Modelling & Software*, 26(12):1489–1501, 12 2011. doi:10.1016/j.envsoft.2011.07.012.

[6] Markus Amann, Rafal Cabala, Janusz Cofala, Chris Heyes, Zbigniew Klimont, Wolfgang Schopp, Leonor Tarrason, David Simpson, Peter Wind, and Jan-Eiof Jonson. "Current Legislation" and the "Maximum Technically Feasible Reduction" cases for the CAFE baseline emission projections. *IIASA, Vienna*, 2004. URL: https://www.researchgate.net/profile/Zbigniew_Klimont/publication/230709494_The_Current_Legislation_and_the_Maximum_Technically_Feasible_Reduction_cases_for_the_CAFE_baseline_emission_projections._CAFE_Report__2/links/0deec53cd2d778aafb000000.pdf (visited on 2016-03-24).

[7] Markus Amann, Zbigniew Klimont, and Fabian Wagner. Regional and global emissions of air pollutants: Recent trends and future scenarios. *Annual Review of Environment and Resources*, 38:31–55, 2013.

[8] Goran Berndes, Monique Hoogwijk, and Richard van den Broek. The contribution of biomass in the future global energy supply: a review of 17 studies. *Biomass and Bioenergy*, 25(1):1–28, 7 2003. doi:10.1016/S0961-9534(02)00185-X.

[9] A.F. Bouwman, K.W. Van der Hoek, B. Eickhout, and I. Soenario. Exploring changes in world ruminant production systems. *Agricultural Systems*, 84(2):121 – 153, 2005. doi:10.1016/j.agsy.2004.05.006.

[10] Stefan Bringezu, Helmut Schutz, Meghan O'Brien, Lea Kauppi, Robert W Howarth, and Jeff McNeely. *Assessing biofuels: towards sustainable production and use of resources*. United Nations Environment Programme, 2009. ISBN 92-807-3052-5.

[11] A. E. Carpentieri, E. D. Larson, and J. Woods. Future biomass-based electricity supply in northeast brazil. *Biomass and Bioenergy*, 4(3):149–173, 1993. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0027382662&partnerID=40&rel=R8.2.0.

[12] Lena Christiansson. Diffusion and learning curves of renewable-energy technologies. *IIASA Report*, 1995.

[13] Janusz Cofala, Markus Amann, Zbigniew Klimont, Kaarle Kupiainen, and Lena Hoglund-Isaksson. Scenarios of global anthropogenic emissions of air pollutants and methane until 2030. *Atmospheric Environment*, 41(38):8486–8499, 2007.

[14] Richard T. Conant and Keith Paustian. Grassland management activity data: current sources and future needs. *Environmental Management*, 33(4):467–473, 2004. doi:10.1007/s00267-003-9104-7.

[15] Rob Dellink, Jean Chateau, Elisa Lanzi, and Bertrand Magne. Long-term economic growth projections in the Shared Socioeconomic Pathways. *Global Environmental Change*, 2015. URL: http://pure.iiasa.ac.at/13280/.

[16] UN Population Division. World Population Projection. Technical Report, UN, 2010.

[17] Veronika Dornburg, APC Faaij, PA Verweij, Martin Banse, Kees van Diepen, Herman van Keulen, Hans Langeveld, Marieke Meeusen, Gerrie van de Ven, and Flip Wester. Biomass assessment: assessment of global biomass potentials and their links to food, water, biodiversity, energy demand and economy: inventory and analysis of existing studies: supporting document. *Report/WAB*, 2008.

[18] Bas Eickhout, Gert Jan van den Born, Jos Notenboom, M van Oorschot, JPM Ros, DP Van Vuuren, and HJ Westhoek. Local and global consequences of the EU renewable directive for biofuels: Testing the sustainability criteria. *Local and global consequences of the EU renewable directive for biofuels: testing the sustainability criteria*, 2008.

[19] Tommi Ekholm, Volker Krey, Shonali Pachauri, and Keywan Riahi. Determinants of household energy consumption in India. *Energy Policy*, 38(10):5696–5707, 2010.

[20] EPA. Us environmental protection agency global emissions database. Report, US Environmental Protection Agency, 2012. URL: http://www.epa.gov/climatechange/ghgemissions/global.html.

[21] K. Eurek, P. Sullivan, M. Gleason, D. Hettinger, D.M. Heimiller, and A. Lopez. An improved global wind resource estimate for integrated assessment models. *Energy Economics*, 64:552–567, 2017.

[22] FAO. Global forest resources assessment 2005. progress towards sustainable forest management. Report, Food and Agriculture Organization of the United Nations, 2006.

[23] FAO. Global forest resources assessment. Report, Food and Agriculture Organization of the United Nations, 2010. URL: http://www.fao.org/forestry/fra/fra2010/en/.

[24] RA Fischer, Derek Byerlee, and Gregory O Edmeades. Can technology deliver on the yield challenge to 2050? 2009. URL: http://www.fao.org/3/a-ak542e/ak542e12a.pdf.

[25] Chris E Forest, Peter H Stone, Andrei P Sokolov, Myles R Allen, and Mort D Webster. Quantifying uncertainties in climate system properties with the use of recent climate observations. *Science*, 295(5552):113–117, 2002.

[26] FPP. Holzernte in der durchforstung; leistungszahlen kosten - oebf seiltabelle sortimentverfahren (skm-tab). Report, Kooperationsabkommen Forst-Platte-Papier, 1999.

[27] Oliver Fricko, Petr Havlik, Joeri Rogelj, Zbigniew Klimont, Mykola Gusti, Nils Johnson, Peter Kolp, Manfred Strubegger, Hugo Valin, Markus Amann, Tatiana Ermolieva, Nicklas Forsell, Mario Herrero, Chris Heyes, Georg Kindermann, Volker Krey, David L. McCollum, Michael Obersteiner, Shonali Pachauri, Shilpa Rao, Erwin Schmid, Wolfgang Schoepp, and Keywan Riahi. The marker quantification of the shared socioeconomic pathway 2: a middle-of-the-road scenario for the 21st century. *Global Environmental Change*, 42:251–267, 2017.

[28] Oliver Fricko, Simon C Parkinson, Nils Johnson, Manfred Strubegger, Michelle TH van Vliet, and Keywan Riahi. Energy sector water use implications of a 2 °C climate policy. *Environmental Research Letters*, 11(3):034011, 2016.

[29] Steffen Fritz, Linda See, Ian McCallum, Christian Schill, Michael Obersteiner, Marijn van der Velde, Hannes Boettcher, Petr Havlik, and Frederic Achard. Highlighting continued uncertainty in global land cover maps for the user community. *Environmental Research Letters*, 6(4):044005, 2011. URL: http://stacks.iop.org/1748-9326/6/i=4/a=044005.

[30] Claire Granier, Bertrand Bessagnet, Tami Bond, Ariela D'Angiola, Hugo Denier van Der Gon, Gregory J Frost, Angelika Heil, Johannes W Kaiser, Stefan Kinne, and Zbigniew Klimont. Evolution of anthropogenic and biomass burning emissions of air pollutants at global and regional scales during the 1980–2010 period. *Climatic Change*, 109(1-2):163–190, 2011.

[31] Biomass Technology Group. *Handbook Biomass Gasification*. H.A.M. Knoef. ISBN: 90-810068-1-9, 2005.

[32] A. Grubler, C. Wilson, N. Bento, B. Boza-Kiss, V. Krey, D.L. McCollum, N.D. Rao, K. Riahi, J. Rogelj, S. De Stercke, J. Cullen, S. Frank, O. Fricko, F. Guo, M. Gidden, P. Havlík, D. Huppmann, G. Kiesewetter, P. Rafaj, W. Schoepp, and H. Valin. A low energy demand scenario for meeting the 1.5 °c target and sustainable development goals without negative emission technologies. *Nature Energy*, 3(6):515–527, 2018. doi:10.1038/s41560-018-0172-6.

[33] MI Gusti. An algorithm for simulation of forest management decisions in the global forest model. *Штучний інтелект*, 2010.

[34] C.N. Hamelinck and A.P.C. Faaij. Future prospects for production of methanol and hydrogen from biomass. Report, Utrecht University, Copernicus Institute, Science Technology and Society, 2001.

[35] B. R. Hartsough, X. Zhang, and R. D. Fight. Harvesting cost model for small trees in natural stands in the interior northwest. *Forest Products Journal*, 51(4):54–61, 2001. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0035306334&partnerID=40&rel=R8.2.0.

[36] Petr Havlik, Uwe A Schneider, Erwin Schmid, Hannes Bottcher, Steffen Fritz, Rastislav Skalsky, Kentaro Aoki, Stephane De Cara, Georg Kindermann, and Florian Kraxner. Global land-use implications of first and second generation biofuel targets. *Energy Policy*, 39(10):5690–5702, 2011.

[37] Petr Havlik, Hugo Valin, Mario Herrero, Michael Obersteiner, Erwin Schmid, Mariana C Rufino, Aline Mosnier, Philip K Thornton, Hannes Bottcher, and Richard T Conant. Climate change mitigation through livestock system transitions. *Proceedings of the National Academy of Sciences*, 111(10):3709–3714, 2014.

[38] M. Herrero, P. Havlik, H. Valin, M.C. Rufino, A.M.O. Notenbaert, P.K. Thornton, M. Blummel, F. Weiss, and M. Obertsteiner. Global livestock systems: biomass use, production, feed efficiencies and greenhouse gas emissions. *Proceedings of the National Academy of Sciences*, 110(52):20888–20893, 2013.

[39] M. Herrero, P.K. Thornton, R. Kruska, and R.S. Reid. Systems dynamics and the spatial distribution of methane emissions from african domestic ruminants to 2030. *Agriculture, Ecosystems & Environment*, 126(1-2):122 – 137, 2008. URL: http://www.sciencedirect.com/science/article/pii/S0167880908000121.

[40] GmbH Herzogbaum. Forstpflanzen-preisliste 2008. herzog.baum samen & pflanzen gmbh. koaserbauerstr. 10, a - 4810 gmunden. austria (also available at www.energiehoelzer.at). 2008.

[41] A. Heston, R. Summers, and B. Aten. Penn world table version 6.2. Report, Center for International Comparisons of Production, Income and Prices at the University of Pennsylvania. September 2006. http://pwt.econ.upenn.edu/php_site/pwt62/pwt62_form.php, 2006.

[42] Monique Hoogwijk and Wina Graus. Global potential of renewable energy sources: a literature assessment. *Background report prepared by order of REN21. Ecofys, PECSNL072975*, 2008.

[43] Monique Maria Hoogwijk. *On the global and regional potential of renewable energy sources*. PhD, Department of Science, Technology and Society. Utrecht University, 2004.

[44] Daniel Huppmann, Matthew Gidden, Oliver Fricko, Peter Kolp, Clara Orthofer, Michael Pimmer, Nikolay Kushin, Adriano Vinca, Alessio Mastrucci, Keywan Riahi, and Volker Krey. The messageix integrated assessment model and the ix modeling platform (ixmp): an open framework for integrated and cross-cutting analysis of energy, climate, the environment, and sustainable development. *Environmental Modelling & Software*, 112:143–156, 2019. doi:10.1016/j.envsoft.2018.11.012.

[45] IEA. World energy model - investment costs. Report, International Energy Agency (IEA), 2014. URL: http://www.worldenergyoutlook.org/media/weowebsite/2014/weio/WEIO2014PGAssumptions.xlsx.

[46] ILO. Occupational wages and hours of work and retail food prices, statistics from the ilo october inquiry. Report, International Labor Organisation, 2007.

[47] IPCC. *Revised 1996 IPCC Guidelines for National Greenhouse Gas Inventories: The Workbook (Volume 2)*. IPCC, Geneva, Switzerland, 1996. URL: http://www.ipcc-nggip.iges.or.jp/public/gl/invs5a.html.

[48] IPCC. *Climate Change 2007: Synthesis Report. Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, Geneva, Switzerland, 2007. URL: http://www.ipcc.ch/pdf/assessment-report/ar4/syr/ar4_syr_full_report.pdf.

[49] R. C. Izaurralde, J. R. Williams, W. B. McGill, N. J. Rosenberg, and M. C. Q. Jakas. Simulating soil c dynamics with epic: model description and testing against long-term data. *Ecological Modelling*, 192(3-4):362–384,

2006. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-31944437556&partnerID=40&rel=R8.2.0.

[50] Jessica Jewell, David McCollum, Johannes Emmerling, Christoph Bertram, David E. H. J. Gernaat, Volker Krey, Leonidas Paroussos, Loic Berger, Kostas Fragkiadakis, Ilkka Keppo, Nawfal Saadi, Massimo Tavoni, Detlef van Vuuren, Vadim Vinichenko, and Keywan Riahi. Limited emission reductions from fuel subsidy removal except in energy exporting regions. *Nature*, 554(10):229, 2018.

[51] R. Jiroušek, R. Klvač, and A. Skoupý. Productivity and costs of the mechanised cut-to-length wood harvesting system in clear-felling operations. *Journal of Forest Science*, 53(10):476–482, 2007. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-35448931938&partnerID=40&rel=R8.2.0.

[52] Nils Johnson, Manfred Strubegger, Madleine McPherson, Simon Parkinson, Volker Krey, and Patrick Sullivan. A reduced-form approach for representing the impacts of wind and solar pv deployment on the structure and operation of the electricity system. *Energy Economics*, 2016.

[53] Global Emissions Joint Research Centre. Emission Database for Global Atmospheric Research EDGAR v4.2. 11 2011. URL: http://edgar.jrc.ec.europa.eu/overview.php?v=42.

[54] Mike Jurvelius. Labor-intensive harvesting of tree plantations in the southern philippines. forest harvesting case -study 9. rap publication: 1997/41. Report, Food and Agriculture Organization of the United Nations, 1997.

[55] Samir KC and Wolfgang Lutz. The human core of the shared socioeconomic pathways: Population scenarios by age, sex and level of education for all countries to 2100. *Global Environmental Change*, 2014.

[56] Ilkka Keppo, Brian C O'Neill, and Keywan Riahi. Probabilistic temperature change projections and energy system implications of greenhouse gas emission scenarios. *Technological Forecasting and Social Change*, 74(7):936–961, 2007.

[57] Ilkka Keppo and Manfred Strubegger. Short term decisions for long term problems–The effect of foresight on model based energy systems analysis. *Energy*, 35(5):2033–2042, 2010.

[58] M.A. Keyzer, M.D. Merbis, I.F.P.W. Pavel, and C.F.A. van Wesenbeeck. Diet shifts towards meat and the effects on cereal use: can we feed the animals in 2030? *Ecological Economics*, 55(2):187–202, 2005. URL: http://www.sciencedirect.com/science/article/pii/S0921800904004100.

[59] G. Kindermann, M. Obersteiner, B. Sohngen, J. Sathaye, K. Andrasko, E. Rametsteiner, B. Schlamadinger, S. Wunder, and R. Beach. Global cost estimates of reducing carbon emissions through avoided deforestation. *Proceedings of the National Academy of Sciences*, 105(30):10302, 2008.

[60] G. E. Kindermann, I. McCallum, S. Fritz, and M. Obersteiner. A global forest growing stock, biomass and carbon map based on fao statistics. *Silva Fennica*, 42(3):387–396, 2008. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-46249088682&partnerID=40&rel=R8.2.0.

[61] Georg E Kindermann, Michael Obersteiner, Ewald Rametsteiner, and Ian McCallum. Predicting the deforestation-trend under different carbon-prices. *Carbon Balance and management*, 1(1):15, 2006.

[62] Volker Krey and Keywan Riahi. Implications of delayed participation and technology failure for the feasibility, costs, and likelihood of staying below temperature targets—Greenhouse gas mitigation scenarios for the 21st century. *Energy Economics*, 31:S94–S106, 2009.

[63] Eric D. Larson, Zheng Li, and Robert H. Williams. Chapter 12 - Fossil Energy. In *Global Energy Assessment - Toward a Sustainable Future*, pages 901–992. Cambridge University Press, Cambridge, UK and New York, NY, USA and the International Institute for Applied Systems Analysis, Laxenburg, Austria, 2012. URL: www.globalenergyassessment.org.

[64] S. Leduc, D. Schwab, E. Dotzauer, E. Schmid, and M. Obersteiner. Optimal location of wood gasification plants for methanol production with heat recovery. *International Journal of Energy Research*, 32:1080–1091 [2008], 2008.

[65] Benjamin D. Leibowicz. Growth and competition in renewable energy industries: insights from an integrated assessment model with strategic firms. *Energy Economics*, 52, Part A:13 – 25, 2015. doi:10.1016/j.eneco.2015.09.010.

[66] Aviva Loew, Paulina Jaramillo, and Haibo Zhai. Marginal costs of water savings from cooling system retrofits: A case study for Texas power plants. *Environmental Research Letters*, 11(10):104004, 2016.

[67] Richard Loulou, Gary Goldstein, and Ken Noble. *Documentation for the MARKAL Family of Models - Part II: MARKAL-MACRO*. IEA Energy Technology Systems Analysis Programme (ETSAP), October 2004. URL: https://www.iea-etsap.org/MrklDoc-II_MARKALMACRO.pdf.

[68] Alan Sussmann Manne and Richard G Richels. *Buying greenhouse insurance: the economic costs of carbon dioxide emission limits*. MIT press, 1992. ISBN 0-262-13280-X.

[69] Bruce A. McCarl and Thomas H. Spreen. Price endogenous mathematical programming as a tool for sector analysis. *American Journal of Agricultural Economics*, 62(1):87–102, 1980. URL: http://www.jstor.org/stable/1239475.

[70] D.L. McCollum, W. Zhou, C. Bertram, H.-S. De Boer, V. Bosetti, S. Busch, J. Després, L. Drouet, J. Emmerling, M. Fay, O. Fricko, S. Fujimori, M. Gidden, M. Harmsen, D. Huppmann, G. Iyer, V. Krey, E. Kriegler, C. Nicolas, S. Pachauri, S. Parkinson, M. Poblete-Cazenave, P. Rafaj, N. Rao, J. Rozenberg, A. Schmitz, W. Schoepp, D. Van Vuuren, and K. Riahi. Energy investment needs for fulfilling the paris agreement and achieving the sustainable development goals. *Nature Energy*, 3(7):589–599, 2018. doi:10.1038/s41560-018-0179-z.

[71] David L. McCollum, Charlie Wilson, Hazel Pettifor, Kalai Ramea, Volker Krey, Keywan Riahi, Christoph Bertram, Zhenhong Lin, Oreane Y. Edelenbosch, and Sei Fujisawa. Improving the behavioral realism of global integrated assessment models: an application to consumers' vehicle choices. *Transportation Research Part D: Transport and Environment*, 2016.

[72] Haewon McJeon, Jae Edmonds, Nico Bauer, Leon Clarke, Brian Fisher, Brian P. Flannery, Jerome Hilaire, Volker Krey, Giacomo Marangoni, Raymond Mi, Keywan Riahi, Holger Rogner, and Massimo Tavoni. Limited impact on decadal-scale climate change from increased use of natural gas. *Nature*, 514(7523):482–485, 2014.

[73] Malte Meinshausen. What does a 2 C target mean for greenhouse gas concentrations? A brief analysis based on multi-gas emission pathways and several climate sensitivity uncertainty estimates. *Avoiding dangerous climate change*, 2006.

[74] Malte Meinshausen, Nicolai Meinshausen, William Hare, Sarah CB Raper, Katja Frieler, Reto Knutti, David J Frame, and Myles R Allen. Greenhouse-gas emission targets for limiting global warming to 2 C. *Nature*, 458(7242):1158–1162, 2009.

[75] Malte Meinshausen, SCB Raper, and TML Wigley. Emulating coupled atmosphere-ocean and carbon cycle models with a simpler model, MAGICC6–Part 1: Model description and calibration. *Atmospheric Chemistry and Physics*, 11(4):1417–1456, 2011.

[76] Malte Meinshausen, Steven J Smith, K Calvin, John S Daniel, MLT Kainuma, JF Lamarque, K Matsumoto, SA Montzka, SCB Raper, and K Riahi. The RCP greenhouse gas concentrations and their extensions from 1765 to 2300. *Climatic change*, 109(1-2):213–241, 2011.

[77] James Meldrum, Syndi Nettles-Anderson, Garvin Heath, and Jordan Macknick. Life cycle water use for electricity generation: A review and harmonization of literature estimates. *Environmental Research Letters*, 8(1):015031, 2013.

[78] Sabine Messner. Endogenized technological learning in an energy systems model. *Journal of Evolutionary Economics*, 7(3):291–313, 1997.

[79] Sabine Messner and Leo Schrattenholzer. MESSAGE–MACRO: linking an energy supply model with a macroeconomic module and solving it iteratively. *Energy*, 25(3):267–282, 2000.

[80] Sabine Messner and Manfred Strubegger. User's Guide for MESSAGE III. 1995. URL: http://pure.iiasa.ac.at/id/eprint/4527/1/WP-95-069.pdf.

[81] Timothy D. Mitchell and Philip D. Jones. An improved method of constructing a database of monthly climate observations and associated high-resolution grids. *International Journal of Climatology*, 25(6):693–712, 2005. doi:10.1002/joc.1181.

[82] A. Muhammad, J. Seale, B. Meade, and A. Regmi. International evidence on food consumption patterns: an update using 2005 international comparison program data. Report 1929, USDA-ERS, 2011.

[83] Sanderine Nonhebel. Energy from agricultural residues and consequences for land requirements for food production. *Agricultural Systems*, 94(2):586–592, 2007.

[84] B.C. O'Neill, T.R. Carter, K.L. Ebi, J. Edmonds, S. Hallegatte, E. Kemp-Benedict, E. Kriegler, L. Mearns, R. Moss, K. Riahi, B. van Ruijven, and D. van Vuuren. Meeting report of the workshop on the nature and use of new socioeconomic pathways for climate change research. Report, NCAR, November 2-4, 2011 2012. URL: http://www.isp.ucar.edu/socio-economic-pathways.

[85] Brian C O'Neill, Elmar Kriegler, Kristie L Ebi, Eric Kemp-Benedict, Keywan Riahi, Dale S Rothman, Bas J van Ruijven, Detlef P van Vuuren, Joern Birkmann, and Kasper Kok. The roads ahead: narratives for shared socioeconomic pathways describing world futures in the 21st century. *Global Environmental Change*, 2015.

[86] Brian C O'Neill, Elmar Kriegler, Keywan Riahi, Kristie L Ebi, Stephane Hallegatte, Timothy R Carter, Ritu Mathur, and Detlef P van Vuuren. A new scenario framework for climate change research: the concept of shared socioeconomic pathways. *Climatic Change*, 122(3):387–400, 2014.

[87] Brian C O'Neill, Keywan Riahi, and Ilkka Keppo. Mitigation implications of midcentury targets that preserve long-term climate policy options. *Proceedings of the National Academy of Sciences*, 107(3):1011–1016, 2010.

[88] S. Pachauri, B. J. Van Ruijven, Y. Nagai, K. Riahi, D. P. Van Vuuren, A. Brew-Hammond, and N. Nakicenovic. Pathways to achieve universal household access to modern energy by 2030. *Environmental Research Letters*, 2013. Cited By :77. URL: www.scopus.com.

[89] Shonali Pachauri, Bas J van Ruijven, Yu Nagai, Keywan Riahi, Detlef P van Vuuren, Abeeku Brew-Hammond, and Nebojsa Nakicenovic. Pathways to achieve universal household access to modern energy by 2030. *Environmental Research Letters*, 8(2):024015, 2013.

[90] Simon Parkinson, Volker Krey, Daniel Huppmann, Taher Kahil, David McCollum, Oliver Fricko, Edward Byers, Matthew J Gidden, Beatriz Mayor, Zarrar Khan, and others. Balancing clean water-climate change mitigation trade-offs. *Environmental Research Letters*, 14(1):014009, 2019.

[91] Simon Parkinson, Volker Krey, Daniel Huppmann, Taher Kahil, David McCollum, Oliver Fricko, Edward Byers, Matthew J Gidden, Beatriz Mayor, Zarrar Khan, and others. Balancing clean water-climate change mitigation trade-offs. *Environmental Research Letters*, 14(1):014009, 2019. doi:10.1088/1748-9326/aaf2a3.

[92] W. J. Parton, J. M. O. Scurlock, D. S. Ojima, T. G. Gilmanov, R. J. Scholes, D. S. Schimel, T. Kirchner, J. C. Menaut, T. Seastedt, E. G. Moya, A. Kamnalrut, and J. I. Kinyamario. Observations and modeling of biomass and soil organic-matter dynamics for the grassland biome worldwide. *Global Biogeochemical Cycles*, 7:785–809, 1993.

[93] WJ Parton, DS Schimel, DS Ojima, and CV Cole. Analysis of factors controlling soil organic matter levels in great plains grasslands. *Soil Science Society of America Journal*, 51(5):1173–1179, 1987.

[94] R. C. Pietzcker, D. Stetter, S. Manger, and G. Luderer. Using the sun to decarbonize the power sector: the economic potential of photovoltaics and concentrating solar power. *Applied Energy*, 135:704–720, 2014.

[95] Andrew J Plantinga, Thomas Mauldin, and Douglas J Miller. An econometric analysis of the costs of sequestering carbon in forests. *American Journal of Agricultural Economics*, 81(4):812–824, 1999.

[96] Miguel Poblete-Cazenave and Shonali Pachauri. A structural model of cooking fuel choices in developing countries. *Energy Economics*, 75:449–463, 2018. URL: https://www.sciencedirect.com/science/article/pii/S0140988318303712, doi:https://doi.org/10.1016/j.eneco.2018.09.003.

[97] Miguel Poblete-Cazenave and Shonali Pachauri. A model of energy poverty and access: estimating household electricity demand and appliance ownership. *Energy Economics*, 98:105266, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0140988321001717, doi:https://doi.org/10.1016/j.eneco.2021.105266.

[98] Alexander Popp, Katherine Calvin, Shinichiro Fujimori, Petr Havlik, Florian Humpenöder, Elke Stehfest, Benjamin Leon Bodirsky, Jan Philipp Dietrich, Jonathan C. Doelmann, Mykola Gusti, Tomoko Hasegawa, Page Kyle, Michael Obersteiner, Andrzej Tabeau, Kiyoshi Takahashi, Hugo Valin, Stephanie Waldhoff, Isabelle Weindl, Marshall Wise, Elmar Kriegler, Hermann Lotze-Campen, Oliver Fricko, Keywan Riahi, and Detlef P. van Vuuren. Land-use futures in the shared socio-economic pathways. *Global Environmental Change*, 42:331–345, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0959378016303399, doi:https://doi.org/10.1016/j.gloenvcha.2016.10.002.

[99] N. Ramankutty, A.T. Evan, C. Monfreda, and J.A. Foley. Farming the planet: 1. geographic distribution of global agricultural lands in the year 2000. *Global Biogeochemical Cycles*, 22(1):1–19, 2008.

[100] E Rametsteiner, S Nilsson, H Bottcher, P Havlik, F Kraxner, S Leduc, M Obersteiner, F Rydzak, U Schneider, D Schwab, and L Willmore. Study of the effects of globalization on the economic viability of eu forestry. final report of the agri tender project: agri-g4-2006-06 [2007]. ec contract number 30-ce-0097579/00-89. Report, EC/IIASA, 2007. URL: http://ec.europa.eu/agriculture/analysis/external/viability_forestry/index_en.htm.

[101] S. Rao, Z. Klimont, S.J. Smith, R. Van Dingenen, F. Dentener, L. Bouwman, K. Riahi, M. Amann, B.L. Bodirsky, D.P. van Vuuren, L. Aleluia Reis, K. Calvin, L. Drouet, O. Fricko, S. Fujimori, D. Gernaat, P. Havlik, M. Harmsen, T. Hasegawa, C. Heyes, J. Hilaire, G. Luderer, T. Masui, E. Stehfest, J. Strefler, S. van der Sluis, and M. Tavoni. Future air pollution in the shared socio-economic pathways. *Global Environmental Change*, 42:346–358, 2017. doi:10.1016/j.gloenvcha.2016.05.012.

[102] Shilpa Rao, Vadim Chirkov, Frank Dentener, Rita Van Dingenen, Shonali Pachauri, Pallav Purohit, Markus Amann, Chris Heyes, Patrick Kinney, and Peter Kolp. Environmental modeling and methods for estimation of the global health impacts of air pollution. *Environmental Modeling & Assessment*, 17(6):613–622, 2012.

[103] Shilpa Rao, Shonali Pachauri, Frank Dentener, Patrick Kinney, Zbigniew Klimont, Keywan Riahi, and Wolfgang Schoepp. Better air for better health: Forging synergies in policies for energy access, climate change and air pollution. *Global environmental change*, 23(5):1122–1130, 2013.

[104] Shilpa Rao and Keywan Riahi. The Role of Non-CO_3 Greenhouse Gases in Climate Change Mitigation: Long-term Scenarios for the 21st Century. *The Energy Journal*, pages 177–200, 2006.

[105] Catherine E. Raptis and Stephan Pfister. Global freshwater thermal emissions from steam-electric power plants with once-through cooling systems. *Energy*, 97:46–57, 2016.

[106] CA Reynolds, TJ Jackson, and WJ Rawls. Estimating soil water-holding capacities by linking the food and agriculture organization soil map of the world with global pedon databases and continuous pedotransfer functions. *Water Resources Research*, 36(12):3653–3662, 2000.

[107] Keywan Riahi, Frank Dentener, Dolf Gielen, Arnulf Grubler, Jessica Jewell, Zbigniew Klimont, Volker Krey, David McCollum, Shonali Pachauri, Shilpa Rao, Bas van Ruijven, Detlef P. van Vuuren, and Charlie Wilson. Chapter 17 - Energy Pathways for Sustainable Development. In *Global Energy Assessment - Toward a Sustainable Future*, pages 1203–1306. Cambridge University Press, Cambridge, UK and New York, NY, USA and the International Institute for Applied Systems Analysis, Laxenburg, Austria, 2012. URL: http://www.globalenergyassessment.org.

[108] Keywan Riahi, Arnulf Grubler, and Nebojsa Nakicenovic. Scenarios of long-term socio-economic and environmental development under climate stabilization. *Technological Forecasting and Social Change*, 74(7):887–935, 2007.

[109] Keywan Riahi, Shilpa Rao, Volker Krey, Cheolhung Cho, Vadim Chirkov, Guenther Fischer, Georg Kindermann, Nebojsa Nakicenovic, and Peter Rafaj. RCP 8.5—A scenario of comparatively high greenhouse gas emissions. *Climatic Change*, 109(1-2):33–57, 2011.

[110] Keywan Riahi and R Alexander Roehrl. Greenhouse gas emissions in a dynamics-as-usual scenario of economic and energy development. *Technological Forecasting and Social Change*, 63(2):175–205, 2000.

[111] Keywan Riahi, Edward S Rubin, and Leo Schrattenholzer. Prospects for carbon capture and sequestration technologies assuming their technological learning. *Energy*, 29(9):1309–1318, 2004.

[112] Keywan Riahi, Detlef P. van Vuuren, Elmar Kriegler, Jae Edmonds, Brian O'Neill, Shinichiro Fujimori, Nico Bauer, Katherine Calvin, Rob Dellink, Oliver Fricko, Wolfgang Lutz, Alexander Popp, Jesus Crespo Cuaresma, Samir KC, Marian Leimbach, Leiwen Jiang, Tom Kram, Shilpa Rao, Johannes Emmerling, Kristie Ebi, Tomoko Hasegawa, Petr Havlik, Florian Humpenoder, Lara Aleluia Da Silva, Steve Smith, Elke Stehfest, Valentina Bosetti, Jiyong Eom, David Gernaat, Toshihiko Masui, Joeri Rogelj, Jessica Strefler, Laurent Drouet, Volker Krey, Gunnar Luderer, Mathijs Harmsen, Kiyoshi Takahashi, Lavinia Baumstark, Jonathan Doelman, Mikiko Kainuma, Zbigniew Klimont, Giacomo Marangoni, Hermann Lotze-Campen, Michael Obersteiner, Andrzej Tabeau, and Massimo Tavoni. The Shared Socioeconomic Pathways and their Energy, Land Use, and Greenhouse Gas Emissions Implications. *Global Environmental Change*, 42:153–168, 2017. URL: http://pure.iiasa.ac.at/13280/, doi:10.1016/j.gloenvcha.2016.05.009.

[113] M. Roelfsema, H. L. van Soest, M. Harmsen, D. P. van Vuuren, C. Bertram, M. den Elzen, N. Höhne, G. Iacobuta, V. Krey, E. Kriegler, G. Luderer, K. Riahi, F. Ueckerdt, J. Després, L. Drouet, J. Emmerling, S. Frank, O. Fricko, M. Gidden, F. Humpenöder, D. Huppmann, S. Fujimori, K. Fragkiadakis, K. Gi, K. Keramidas, A. C. Köberle, L. Aleluia Reis, P. Rochedo, R. Schaeffer, K. Oshiro, Z. Vrontisi, W. Chen,

G. C. Iyer, J. Edmonds, M. Kannavou, K. Jiang, R. Mathur, G. Safonov, and S. S. Vishwanathan. Taking stock of national climate policies to evaluate implementation of the paris agreement. *Nature Communications*, 2020. doi:https://doi.org/10.1038/s41467-020-15414-6.

[114] Joeri Rogelj, Oliver Fricko, Malte Meinshausen, Volker Krey, Johanna Zilliacus, and Keywan Riahi. Understanding the origin of paris agreement emission uncertainties. *Nature Communication*, pages 15748, 2017.

[115] Joeri Rogelj, David L McCollum, Brian C O'Neill, and Keywan Riahi. 2020 emissions levels required to limit warming to below 2 [thinsp][deg] C. *Nature Climate Change*, 3(4):405–412, 2013.

[116] Joeri Rogelj, David L McCollum, Andy Reisinger, Malte Meinshausen, and Keywan Riahi. Probabilistic cost estimates for climate change mitigation. *Nature*, 493(7430):79–83, 2013.

[117] Joeri Rogelj, Andy Reisinger, David L McCollum, Reto Knutti, Keywan Riahi, and Malte Meinshausen. Mitigation choices impact carbon budget size compatible with low temperature goals. *Environmental Research Letters*, 10(7):075003, 2015.

[118] H Rogner, Roberto F Aguilera, Christina Archer, Ruggero Bertani, S Bhattacharya, M Dusseault, Luc Gagnon, H Harbel, Monique Hoogwijk, and Arthur Johnson. Chapter 7 - Energy resources and potentials. In *Global Energy Assessment - Toward a Sustainable Future*, pages 423–512. Cambridge University Press, Cambridge, UK and New York, NY, USA and the International Institute for Applied Systems Analysis, Laxenburg, Austria, 2012.

[119] Hans-Holger Rogner. An assessment of world hydrocarbon resources. *Annual review of energy and the environment*, 22(1):217–262, 1997.

[120] Dmitry Rokityanskiy, Pablo C Benitez, Florian Kraxner, Ian McCallum, Michael Obersteiner, Ewald Rametsteiner, and Yoshiki Yamagata. Geographically explicit global modeling of land-use change, carbon sequestration, and biomass supply. *Technological Forecasting and Social Change*, 74(7):1057–1082, 2007.

[121] Aaron Ruesch and Holly K. Gibbs. New ipcc tier-1 global biomass carbon map for the year 2000. Report, Oak Ridge National Laboratory, 2008. URL: http://cdiac.ornl.gov/epubs/ndp/global_carbon/carbon_documentation.html.

[122] P. Russ, T. Wiesenthal, D. van Regemorter, and J.C. Ciscar. Global climate policy scenarios for 2030 and beyond: analysis of greenhouse gas emission reduction pathway scenarios with the poles and geme3 models. *Institute for Prospective technological Studies, October*, 2007.

[123] Jayant Sathaye, Peter Chan, Larry Dale, Willy Makundi, and Ken Andrasko. A summary note estimating global forestry GHG mitigation potential and costs: A dynamic partial equilibrium approach. *working draft, August*, 10:448–457, 2003.

[124] Jayant Sathaye, Willy Makundi, Larry Dale, Peter Chan, and Kenneth Andrasko. GHG mitigation potential, costs and benefits in global forests: a dynamic partial equilibrium approach. *The Energy Journal*, pages 127–162, 2006.

[125] T. Sauer, P. Havlik, G. Kindermann, and U.A. . Schneider. Agriculture, population, land and water scarcity in a changing world - the role of irrigation. In *Congress of the European Association of Agricultural Economists*. 2008.

[126] Andreas Schafer. Structural change in energy use. *Energy Policy*, 33(4):429–437, 2005.

[127] A. L. Schloss, D. W. Kicklighter, J. Kaduk, U. Wittenberg, and The Participants of the Potsdam NPP Model Comparison. Comparing global models of terrestrial net primary productivity (npp): comparison of npp to climate and the normalized difference vegetation index (ndvi). *Global Change Biology*, 5(S1):25–34, 1999. doi:10.1046/j.1365-2486.1999.00004.x.

[128] Erich A Schneider and William C Sailor. Long-term uranium supply estimates. *Nuclear Technology*, 162(3):379–387, 2008.

[129] Uwe A. Schneider, Bruce A. McCarl, and Erwin Schmid. Agricultural sector analysis on greenhouse gas mitigation in us agriculture and forestry. *Agricultural Systems*, 94(2):128 – 140, 2007. URL: http://www.sciencedirect.com/science/article/pii/S0308521X06001028.

[130] James Seale, Anita Regmi, and Jason Bernstein. International evidence on food consumption patterns. Report 1904, USDA-ERS, October 2003. URL: http://www.ers.usda.gov/Data/InternationalFoodDemand/.

[131] Timothy Searchinger, Ralph Heimlich, Richard A Houghton, Fengxia Dong, Amani Elobeid, Jacinto Fabiosa, Simla Tokgoz, Dermot Hayes, and Tun-Hsiang Yu. Use of US croplands for biofuels increases greenhouse gases through emissions from land-use change. *Science*, 319(5867):1238–1240, 2008.

[132] C. Sere and H. Steinfeld. World livestock production systems: current status, issues and trends. Report 127, Food and Agriculture Organisation, 1996. URL: http://www.fao.org/WAIRDOCS/LEAD/X6101E/X6101E00.HTM.

[133] R. Skalsky, Z. Tarasovicova, J. Balkovic, E. Schmid, M. Fuchs, E. Moltchanova, G. Kindermann, and P. Scholtz. Geo-bene global database for bio-physical modeling v.1.0. concepts, methodologies and data.technical report. Report, IIASA, accessed 13.03.09 2008. URL: http://www.geo-bene.eu/?q=node/1734S.

[134] Edward MW Smeets, Andre PC Faaij, Iris M Lewandowski, and Wim C Turkenburg. A bottom-up assessment and review of global bio-energy potentials to 2050. *Progress in Energy and combustion science*, 33(1):56–106, 2007.

[135] Pete Smith, Peter J Gregory, Detlef Van Vuuren, Michael Obersteiner, Petr Havlik, Mark Rounsevell, Jeremy Woods, Elke Stehfest, and Jessica Bellarby. Competition for land. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 365(1554):2941–2957, 2010.

[136] A. L. Sorensen. Economies of scale in biomass gasification systems. Report Interim Report IR-05-030, IIASA, 2005.

[137] Robert N Stavins. The costs of carbon sequestration: a revealed-preference approach. *The American Economic Review*, 89(4):994–1009, 1999.

[138] Elke Stehfest, Lex Bouwman, Detlef P Van Vuuren, Michel GJ Den Elzen, Bas Eickhout, and Pavel Kabat. Climate benefits of changing diet. *Climatic change*, 95(1-2):83–102, 2009.

[139] B. J. Stokes, D. J. Frederick, and D. T. Curtin. Field trials of a short-rotation biomass feller buncher and selected harvesting systems. *Biomass*, 11(3):185–204, 1986. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0022984004&partnerID=40&rel=R8.2.0.

[140] Patrick Sullivan, Volker Krey, and Keywan Riahi. Impacts of considering electric sector variability and reliability in the message model. *Energy Strategy Reviews*, 1(3):157–163, 2013.

[141] T. Takayama and G.G. Judge. *Spatial and temporal price and allocation models*. North-Holland Amsterdam, 1971.

[142] Francesco N Tubiello and Gunther Fischer. Reducing climate change impacts on agriculture: Global and regional effects of mitigation, 2000–2080. *Technological Forecasting and Social Change*, 74(7):1030–1056, 2007.

[143] Francesco N Tubiello, Mirella Salvatore, Simone Rossi, Alessandro Ferrara, Nuala Fitton, and Pete Smith. The faostat database of greenhouse gas emissions from agriculture. *Environmental Research Letters*, 8(1):015009, 2013. URL: http://stacks.iop.org/1748-9326/8/i=1/a=015009.

[144] Jasper van Vliet, Maarten van den Berg, Michiel Schaeffer, Detlef P van Vuuren, Michel Den Elzen, Andries F Hof, Angelica Mendoza Beltran, and Malte Meinshausen. Copenhagen accord pledges imply higher costs for staying below 2 C warming. *Climatic Change*, 113(2):551–561, 2012.

[145] Detlef van Vuuren, Washington Ochola, Susan Riha, Mario Giampietro, Hector Ginzo, Thomas Henrichs, Sajidin Hussain Hussain, Kaspar Kok, Moraka Makhura Makhura, and Monirul Mirza. Outlook on agricultural changes and its drivers. In *Agriculture at a Crossroads-the Global Report of the International Assessment of Agricultural Knowledge, Science, and Technology*. Island Press, 2009.

[146] Detlef P Van Vuuren, Elie Bellevrat, Alban Kitous, and Morna Isaac. Bio-energy use and low stabilization scenarios. *The Energy Journal*, pages 193–221, 2010.

[147] Detlef P Van Vuuren, Jasper van Vliet, and Elke Stehfest. Future bio-energy potential under various natural constraints. *Energy Policy*, 37(11):4220–4230, 2009.

[148] J. Wang, C. Long, J. McNeel, and J. Baumgras. Productivity and cost of manual felling and cable skidding in central appalachian hardwood forests. *Forest Products Journal*, 54(12):45–51, 2004. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-11844274724&partnerID=40&rel=R8.2.0.

[149] Tom ML Wigley. MAGICC/SCENGEN 5.3: User manual (version 2). *NCAR, Boulder, CO*, 2008.

[150] J.R. Williams and VP Singh. The epic model. *Computer models of watershed hydrology*, pages 909–1000, 1995.

[151] W. Wint and T. Robinson. *Gridded livestock of the world 2007*. FAO, 2007.

[152] Liangzhi You and Stanley Wood. An entropy approach to spatial disaggregation of agricultural production. *Agricultural Systems*, 90(1-3):329 – 347, 2006. URL: http://www.sciencedirect.com/science/article/B6T3W-4JKYWM1-1/2/381253576eb09660fc9860c6c8bb8e1f.

[153] Haibo Zhai and Edward S Rubin. Performance and cost of wet and dry cooling systems for pulverized coal power plants with and without carbon capture and storage. *Energy Policy*, 38(10):5653–5660, 2010.

[154] Chao Zhang, Laura Diaz Anadon, Hongpin Mo, Zhongnan Zhao, and Zhu Liu. Water- carbon trade-off in China`s coal power industry. *Environmental science & technology*, 48(19):11082–11089, 2014.

[155] OECD and NEA. Uranium 2003: resources, production and demand. Report NEA-05291, OECD/NEA, June 2004. URL: https://www.oecd-nea.org/ndd/pubs/2004/5291-uranium-2003.pdf.

[156] World Bank Group. *World Development Indicators 2012*. World Bank Publications, 2012. ISBN 0-8213-8985-8.

[1] International Energy Agency. Energy Balances. Technical Report, International Energy Agency, 2012.

[2] International Energy Agency. World Energy Outlook 2014. Technical Report, International Energy Agency, 2014. URL: http://www.worldenergyoutlook.org/weo2014/.

[3] Markus Amann, Imrich Bertok, Jens Borken-Kleefeld, Janusz Cofala, Chris Heyes, Lena Hoglund-Isaksson, Zbigniew Klimont, Binh Nguyen, Maximilian Posch, Peter Rafaj, Robert Sandler, Wolfgang Schopp, Fabian Wagner, and Wilfried Winiwarter. Cost-effective control of air quality and greenhouse gases in Europe: Modeling and policy applications. *Environmental Modelling & Software*, 26(12):1489–1501, 12 2011. doi:10.1016/j.envsoft.2011.07.012.

[4] Markus Amann, Zbigniew Klimont, and Fabian Wagner. Regional and global emissions of air pollutants: Recent trends and future scenarios. *Annual Review of Environment and Resources*, 38:31–55, 2013.

[5] A. E. Carpentieri, E. D. Larson, and J. Woods. Future biomass-based electricity supply in northeast brazil. *Biomass and Bioenergy*, 4(3):149–173, 1993. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0027382662&partnerID=40&rel=R8.2.0.

[6] Lena Christiansson. Diffusion and learning curves of renewable-energy technologies. *IIASA Report*, 1995.

[7] Janusz Cofala, Markus Amann, Zbigniew Klimont, Kaarle Kupiainen, and Lena Hoglund-Isaksson. Scenarios of global anthropogenic emissions of air pollutants and methane until 2030. *Atmospheric Environment*, 41(38):8486–8499, 2007.

[8] Richard T. Conant and Keith Paustian. Grassland management activity data: current sources and future needs. *Environmental Management*, 33(4):467–473, 2004. doi:10.1007/s00267-003-9104-7.

[9] Rob Dellink, Jean Chateau, Elisa Lanzi, and Bertrand Magne. Long-term economic growth projections in the Shared Socioeconomic Pathways. *Global Environmental Change*, 2015. URL: http://pure.iiasa.ac.at/13280/.

[10] UN Population Division. World Population Projection. Technical Report, UN, 2010.

[11] Tommi Ekholm, Volker Krey, Shonali Pachauri, and Keywan Riahi. Determinants of household energy consumption in India. *Energy Policy*, 38(10):5696–5707, 2010.

[12] EPA. Us environmental protection agency global emissions database. Report, US Environmental Protection Agency, 2012. URL: http://www.epa.gov/climatechange/ghgemissions/global.html.

[13] K. Eurek, P. Sullivan, M. Gleason, D. Hettinger, D.M. Heimiller, and A. Lopez. An improved global wind resource estimate for integrated assessment models. *Energy Economics*, 64:552–567, 2017.

[14] FAO. Global forest resources assessment 2005. progress towards sustainable forest management. Report, Food and Agriculture Organization of the United Nations, 2006.

[15] FAO. Global forest resources assessment. Report, Food and Agriculture Organization of the United Nations, 2010. URL: http://www.fao.org/forestry/fra/fra2010/en/.

[16] FPP. Holzernte in der durchforstung; leistungszahlen kosten - oebf seiltabelle sortimentverfahren (skm-tab). Report, Kooperationsabkommen Forst-Platte-Papier, 1999.

[17] Oliver Fricko, Petr Havlik, Joeri Rogelj, Zbigniew Klimont, Mykola Gusti, Nils Johnson, Peter Kolp, Manfred Strubegger, Hugo Valin, Markus Amann, Tatiana Ermolieva, Nicklas Forsell, Mario Herrero, Chris Heyes, Georg Kindermann, Volker Krey, David L. McCollum, Michael Obersteiner, Shonali Pachauri, Shilpa Rao, Erwin Schmid, Wolfgang Schoepp, and Keywan Riahi. The marker quantification of the shared socioeconomic pathway 2: a middle-of-the-road scenario for the 21st century. *Global Environmental Change*, 42:251–267, 2017.

[18] Oliver Fricko, Simon C Parkinson, Nils Johnson, Manfred Strubegger, Michelle TH van Vliet, and Keywan Riahi. Energy sector water use implications of a 2 °C climate policy. *Environmental Research Letters*, 11(3):034011, 2016.

[19] Steffen Fritz, Linda See, Ian McCallum, Christian Schill, Michael Obersteiner, Marijn van der Velde, Hannes Boettcher, Petr Havlik, and Frederic Achard. Highlighting continued uncertainty in global land cover maps for the user community. *Environmental Research Letters*, 6(4):044005, 2011. URL: http://stacks.iop.org/1748-9326/6/i=4/a=044005.

[20] Biomass Technology Group. *Handbook Biomass Gasification*. H.A.M. Knoef. ISBN: 90-810068-1-9, 2005.

[21] A. Grubler, C. Wilson, N. Bento, B. Boza-Kiss, V. Krey, D.L. McCollum, N.D. Rao, K. Riahi, J. Rogelj, S. De Stercke, J. Cullen, S. Frank, O. Fricko, F. Guo, M. Gidden, P. Havlík, D. Huppmann, G. Kiesewetter, P. Rafaj, W. Schoepp, and H. Valin. A low energy demand scenario for meeting the 1.5 °c target and sustainable development goals without negative emission technologies. *Nature Energy*, 3(6):515–527, 2018. doi:10.1038/s41560-018-0172-6.

[22] MI Gusti. An algorithm for simulation of forest management decisions in the global forest model. *Штучний інтелект*, 2010.

[23] C.N. Hamelinck and A.P.C. Faaij. Future prospects for production of methanol and hydrogen from biomass. Report, Utrecht University, Copernicus Institute, Science Technology and Society, 2001.

[24] B. R. Hartsough, X. Zhang, and R. D. Fight. Harvesting cost model for small trees in natural stands in the interior northwest. *Forest Products Journal*, 51(4):54–61, 2001. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0035306334&partnerID=40&rel=R8.2.0.

[25] Petr Havlik, Uwe A Schneider, Erwin Schmid, Hannes Bottcher, Steffen Fritz, Rastislav Skalsky, Kentaro Aoki, Stephane De Cara, Georg Kindermann, and Florian Kraxner. Global land-use implications of first and second generation biofuel targets. *Energy Policy*, 39(10):5690–5702, 2011.

[26] Petr Havlik, Hugo Valin, Mario Herrero, Michael Obersteiner, Erwin Schmid, Mariana C Rufino, Aline Mosnier, Philip K Thornton, Hannes Bottcher, and Richard T Conant. Climate change mitigation through livestock system transitions. *Proceedings of the National Academy of Sciences*, 111(10):3709–3714, 2014.

[27] M. Herrero, P. Havlik, H. Valin, M.C. Rufino, A.M.O. Notenbaert, P.K. Thornton, M. Blummel, F. Weiss, and M. Obertsteiner. Global livestock systems: biomass use, production, feed efficiencies and greenhouse gas emissions. *Proceedings of the National Academy of Sciences*, 110(52):20888–20893, 2013.

[28] M. Herrero, P.K. Thornton, R. Kruska, and R.S. Reid. Systems dynamics and the spatial distribution of methane emissions from african domestic ruminants to 2030. *Agriculture, Ecosystems & Environment*, 126(1-2):122 – 137, 2008. URL: http://www.sciencedirect.com/science/article/pii/S0167880908000121.

[29] GmbH Herzogbaum. Forstpflanzen-preisliste 2008. herzog.baum samen & pflanzen gmbh. koaserbauerstr. 10, a - 4810 gmunden. austria (also available at www.energiehoelzer.at). 2008.

[30] A. Heston, R. Summers, and B. Aten. Penn world table version 6.2. Report, Center for International Comparisons of Production, Income and Prices at the University of Pennsylvania. September 2006. http://pwt.econ.upenn.edu/php_site/pwt62/pwt62_form.php, 2006.

[31] Daniel Huppmann, Matthew Gidden, Oliver Fricko, Peter Kolp, Clara Orthofer, Michael Pimmer, Nikolay Kushin, Adriano Vinca, Alessio Mastrucci, Keywan Riahi, and Volker Krey. The messageix integrated assessment model and the ix modeling platform (ixmp): an open framework for integrated and cross-cutting analysis of energy, climate, the environment, and sustainable development. *Environmental Modelling & Software*, 112:143–156, 2019. doi:10.1016/j.envsoft.2018.11.012.

[32] IEA. World energy model - investment costs. Report, International Energy Agency (IEA), 2014. URL: http://www.worldenergyoutlook.org/media/weowebsite/2014/weio/WEIO2014PGAssumptions.xlsx.

[33] ILO. Occupational wages and hours of work and retail food prices, statistics from the ilo october inquiry. Report, International Labor Organisation, 2007.

[34] IPCC. *Revised 1996 IPCC Guidelines for National Greenhouse Gas Inventories: The Workbook (Volume 2)*. IPCC, Geneva, Switzerland, 1996. URL: http://www.ipcc-nggip.iges.or.jp/public/gl/invs5a.html.

[35] R. C. Izaurralde, J. R. Williams, W. B. McGill, N. J. Rosenberg, and M. C. Q. Jakas. Simulating soil c dynamics with epic: model description and testing against long-term data. *Ecological Modelling*, 192(3-4):362–384, 2006. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-31944437556&partnerID=40&rel=R8.2.0.

[36] Jessica Jewell, David McCollum, Johannes Emmerling, Christoph Bertram, David E. H. J. Gernaat, Volker Krey, Leonidas Paroussos, Loic Berger, Kostas Fragkiadakis, Ilkka Keppo, Nawfal Saadi, Massimo Tavoni, Detlef van Vuuren, Vadim Vinichenko, and Keywan Riahi. Limited emission reductions from fuel subsidy removal except in energy exporting regions. *Nature*, 554(10):229, 2018.

[37] R. Jiroušek, R. Klvač, and A. Skoupý. Productivity and costs of the mechanised cut-to-length wood harvesting system in clear-felling operations. *Journal of Forest Science*, 53(10):476–482, 2007. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-35448931938&partnerID=40&rel=R8.2.0.

[38] Nils Johnson, Manfred Strubegger, Madleine McPherson, Simon Parkinson, Volker Krey, and Patrick Sullivan. A reduced-form approach for representing the impacts of wind and solar pv deployment on the structure and operation of the electricity system. *Energy Economics*, 2016.

[39] Mike Jurvelius. Labor-intensive harvesting of tree plantations in the southern philippines. forest harvesting case -study 9. rap publication: 1997/41. Report, Food and Agriculture Organization of the United Nations, 1997.

[40] Samir KC and Wolfgang Lutz. The human core of the shared socioeconomic pathways: Population scenarios by age, sex and level of education for all countries to 2100. *Global Environmental Change*, 2014.

[41] Ilkka Keppo and Manfred Strubegger. Short term decisions for long term problems–The effect of foresight on model based energy systems analysis. *Energy*, 35(5):2033–2042, 2010.

[42] M.A. Keyzer, M.D. Merbis, I.F.P.W. Pavel, and C.F.A. van Wesenbeeck. Diet shifts towards meat and the effects on cereal use: can we feed the animals in 2030? *Ecological Economics*, 55(2):187–202, 2005. URL: http://www.sciencedirect.com/science/article/pii/S0921800904004100.

[43] G. Kindermann, M. Obersteiner, B. Sohngen, J. Sathaye, K. Andrasko, E. Rametsteiner, B. Schlamadinger, S. Wunder, and R. Beach. Global cost estimates of reducing carbon emissions through avoided deforestation. *Proceedings of the National Academy of Sciences*, 105(30):10302, 2008.

[44] G. E. Kindermann, I. McCallum, S. Fritz, and M. Obersteiner. A global forest growing stock, biomass and carbon map based on fao statistics. *Silva Fennica*, 42(3):387–396, 2008. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-46249088682&partnerID=40&rel=R8.2.0.

[45] Georg E Kindermann, Michael Obersteiner, Ewald Rametsteiner, and Ian McCallum. Predicting the deforestation-trend under different carbon-prices. *Carbon Balance and management*, 1(1):15, 2006.

[46] Volker Krey and Keywan Riahi. Implications of delayed participation and technology failure for the feasibility, costs, and likelihood of staying below temperature targets—Greenhouse gas mitigation scenarios for the 21st century. *Energy Economics*, 31:S94–S106, 2009.

[47] Eric D. Larson, Zheng Li, and Robert H. Williams. Chapter 12 - Fossil Energy. In *Global Energy Assessment - Toward a Sustainable Future*, pages 901–992. Cambridge University Press, Cambridge, UK and New York, NY, USA and the International Institute for Applied Systems Analysis, Laxenburg, Austria, 2012. URL: www.globalenergyassessment.org.

[48] S. Leduc, D. Schwab, E. Dotzauer, E. Schmid, and M. Obersteiner. Optimal location of wood gasification plants for methanol production with heat recovery. *International Journal of Energy Research*, 32:1080–1091 [2008], 2008.

[49] Benjamin D. Leibowicz. Growth and competition in renewable energy industries: insights from an integrated assessment model with strategic firms. *Energy Economics*, 52, Part A:13 – 25, 2015. doi:10.1016/j.eneco.2015.09.010.

[50] Aviva Loew, Paulina Jaramillo, and Haibo Zhai. Marginal costs of water savings from cooling system retrofits: A case study for Texas power plants. *Environmental Research Letters*, 11(10):104004, 2016.

[51] Alan Sussmann Manne and Richard G Richels. *Buying greenhouse insurance: the economic costs of carbon dioxide emission limits*. MIT press, 1992. ISBN 0-262-13280-X.

[52] Bruce A. McCarl and Thomas H. Spreen. Price endogenous mathematical programming as a tool for sector analysis. *American Journal of Agricultural Economics*, 62(1):87–102, 1980. URL: http://www.jstor.org/stable/1239475.

[53] D.L. McCollum, W. Zhou, C. Bertram, H.-S. De Boer, V. Bosetti, S. Busch, J. Després, L. Drouet, J. Emmerling, M. Fay, O. Fricko, S. Fujimori, M. Gidden, M. Harmsen, D. Huppmann, G. Iyer, V. Krey, E. Kriegler, C. Nicolas, S. Pachauri, S. Parkinson, M. Poblete-Cazenave, P. Rafaj, N. Rao, J. Rozenberg, A. Schmitz, W. Schoepp, D. Van Vuuren, and K. Riahi. Energy investment needs for fulfilling the paris agreement and achieving the sustainable development goals. *Nature Energy*, 3(7):589–599, 2018. doi:10.1038/s41560-018-0179-z.

[54] David L. McCollum, Charlie Wilson, Hazel Pettifor, Kalai Ramea, Volker Krey, Keywan Riahi, Christoph Bertram, Zhenhong Lin, Oreane Y. Edelenbosch, and Sei Fujisawa. Improving the behavioral realism of global integrated assessment models: an application to consumers' vehicle choices. *Transportation Research Part D: Transport and Environment*, 2016.

[55] Haewon McJeon, Jae Edmonds, Nico Bauer, Leon Clarke, Brian Fisher, Brian P. Flannery, Jerome Hilaire, Volker Krey, Giacomo Marangoni, Raymond Mi, Keywan Riahi, Holger Rogner, and Massimo Tavoni. Limited impact on decadal-scale climate change from increased use of natural gas. *Nature*, 514(7523):482–485, 2014.

[56] Malte Meinshausen, Nicolai Meinshausen, William Hare, Sarah CB Raper, Katja Frieler, Reto Knutti, David J Frame, and Myles R Allen. Greenhouse-gas emission targets for limiting global warming to 2 C. *Nature*, 458(7242):1158–1162, 2009.

[57] Malte Meinshausen, SCB Raper, and TML Wigley. Emulating coupled atmosphere-ocean and carbon cycle models with a simpler model, MAGICC6–Part 1: Model description and calibration. *Atmospheric Chemistry and Physics*, 11(4):1417–1456, 2011.

[58] Malte Meinshausen, Steven J Smith, K Calvin, John S Daniel, MLT Kainuma, JF Lamarque, K Matsumoto, SA Montzka, SCB Raper, and K Riahi. The RCP greenhouse gas concentrations and their extensions from 1765 to 2300. *Climatic change*, 109(1-2):213–241, 2011.

[59] James Meldrum, Syndi Nettles-Anderson, Garvin Heath, and Jordan Macknick. Life cycle water use for electricity generation: A review and harmonization of literature estimates. *Environmental Research Letters*, 8(1):015031, 2013.

[60] Sabine Messner. Endogenized technological learning in an energy systems model. *Journal of Evolutionary Economics*, 7(3):291–313, 1997.

[61] Sabine Messner and Leo Schrattenholzer. MESSAGE–MACRO: linking an energy supply model with a macroeconomic module and solving it iteratively. *Energy*, 25(3):267–282, 2000.

[62] Sabine Messner and Manfred Strubegger. User's Guide for MESSAGE III. 1995. URL: http://pure.iiasa.ac.at/id/eprint/4527/1/WP-95-069.pdf.

[63] Timothy D. Mitchell and Philip D. Jones. An improved method of constructing a database of monthly climate observations and associated high-resolution grids. *International Journal of Climatology*, 25(6):693–712, 2005. doi:10.1002/joc.1181.

[64] A. Muhammad, J. Seale, B. Meade, and A. Regmi. International evidence on food consumption patterns: an update using 2005 international comparison program data. Report 1929, USDA-ERS, 2011.

[65] Brian C O'Neill, Elmar Kriegler, Kristie L Ebi, Eric Kemp-Benedict, Keywan Riahi, Dale S Rothman, Bas J van Ruijven, Detlef P van Vuuren, Joern Birkmann, and Kasper Kok. The roads ahead: narratives for shared socioeconomic pathways describing world futures in the 21st century. *Global Environmental Change*, 2015.

[66] Brian C O'Neill, Elmar Kriegler, Keywan Riahi, Kristie L Ebi, Stephane Hallegatte, Timothy R Carter, Ritu Mathur, and Detlef P van Vuuren. A new scenario framework for climate change research: the concept of shared socioeconomic pathways. *Climatic Change*, 122(3):387–400, 2014.

[67] Brian C O'Neill, Keywan Riahi, and Ilkka Keppo. Mitigation implications of midcentury targets that preserve long-term climate policy options. *Proceedings of the National Academy of Sciences*, 107(3):1011–1016, 2010.

[68] S. Pachauri, B. J. Van Ruijven, Y. Nagai, K. Riahi, D. P. Van Vuuren, A. Brew-Hammond, and N. Nakicenovic. Pathways to achieve universal household access to modern energy by 2030. *Environmental Research Letters*, 2013. Cited By :77. URL: www.scopus.com.

[69] Shonali Pachauri, Bas J van Ruijven, Yu Nagai, Keywan Riahi, Detlef P van Vuuren, Abeeku Brew-Hammond, and Nebojsa Nakicenovic. Pathways to achieve universal household access to modern energy by 2030. *Environmental Research Letters*, 8(2):024015, 2013.

[70] Simon Parkinson, Volker Krey, Daniel Huppmann, Taher Kahil, David McCollum, Oliver Fricko, Edward Byers, Matthew J Gidden, Beatriz Mayor, Zarrar Khan, and others. Balancing clean water-climate change mitigation trade-offs. *Environmental Research Letters*, 14(1):014009, 2019.

[71] Simon Parkinson, Volker Krey, Daniel Huppmann, Taher Kahil, David McCollum, Oliver Fricko, Edward Byers, Matthew J Gidden, Beatriz Mayor, Zarrar Khan, and others. Balancing clean water-climate change mitigation trade-offs. *Environmental Research Letters*, 14(1):014009, 2019. doi:10.1088/1748-9326/aaf2a3.

[72] W. J. Parton, J. M. O. Scurlock, D. S. Ojima, T. G. Gilmanov, R. J. Scholes, D. S. Schimel, T. Kirchner, J. C. Menaut, T. Seastedt, E. G. Moya, A. Kamnalrut, and J. I. Kinyamario. Observations and modeling of biomass and soil organic-matter dynamics for the grassland biome worldwide. *Global Biogeochemical Cycles*, 7:785–809, 1993.

[73] WJ Parton, DS Schimel, DS Ojima, and CV Cole. Analysis of factors controlling soil organic matter levels in great plains grasslands. *Soil Science Society of America Journal*, 51(5):1173–1179, 1987.

[74] R. C. Pietzcker, D. Stetter, S. Manger, and G. Luderer. Using the sun to decarbonize the power sector: the economic potential of photovoltaics and concentrating solar power. *Applied Energy*, 135:704–720, 2014.

[75] Miguel Poblete-Cazenave and Shonali Pachauri. A structural model of cooking fuel choices in developing countries. *Energy Economics*, 75:449–463, 2018. URL: https://www.sciencedirect.com/science/article/pii/S0140988318303712, doi:https://doi.org/10.1016/j.eneco.2018.09.003.

[76] Miguel Poblete-Cazenave and Shonali Pachauri. A model of energy poverty and access: estimating household electricity demand and appliance ownership. *Energy Economics*, 98:105266, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0140988321001717, doi:https://doi.org/10.1016/j.eneco.2021.105266.

[77] Alexander Popp, Katherine Calvin, Shinichiro Fujimori, Petr Havlik, Florian Humpenöder, Elke Stehfest, Benjamin Leon Bodirsky, Jan Philipp Dietrich, Jonathan C. Doelmann, Mykola Gusti, Tomoko Hasegawa, Page Kyle, Michael Obersteiner, Andrzej Tabeau, Kiyoshi Takahashi, Hugo Valin, Stephanie Waldhoff, Isabelle Weindl, Marshall Wise, Elmar Kriegler, Hermann Lotze-Campen, Oliver Fricko, Keywan Riahi, and Detlef P. van Vuuren. Land-use futures in the shared socio-economic pathways. *Global Environmental Change*, 42:331–345, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0959378016303399, doi:https://doi.org/10.1016/j.gloenvcha.2016.10.002.

[78] N. Ramankutty, A.T. Evan, C. Monfreda, and J.A. Foley. Farming the planet: 1. geographic distribution of global agricultural lands in the year 2000. *Global Biogeochemical Cycles*, 22(1):1–19, 2008.

[79] E Rametsteiner, S Nilsson, H Bottcher, P Havlik, F Kraxner, S Leduc, M Obersteiner, F Rydzak, U Schneider, D Schwab, and L Willmore. Study of the effects of globalization on the economic viability of eu forestry. final report of the agri tender project: agri-g4-2006-06 [2007]. ec contract number 30-ce-0097579/00-89. Report, EC/IIASA, 2007. URL: http://ec.europa.eu/agriculture/analysis/external/viability_forestry/index_en.htm.

[80] S. Rao, Z. Klimont, S.J. Smith, R. Van Dingenen, F. Dentener, L. Bouwman, K. Riahi, M. Amann, B.L. Bodirsky, D.P. van Vuuren, L. Aleluia Reis, K. Calvin, L. Drouet, O. Fricko, S. Fujimori, D. Gernaat, P. Havlik, M. Harmsen, T. Hasegawa, C. Heyes, J. Hilaire, G. Luderer, T. Masui, E. Stehfest, J. Strefler, S. van der Sluis, and M. Tavoni. Future air pollution in the shared socio-economic pathways. *Global Environmental Change*, 42:346–358, 2017. doi:10.1016/j.gloenvcha.2016.05.012.

[81] Shilpa Rao, Shonali Pachauri, Frank Dentener, Patrick Kinney, Zbigniew Klimont, Keywan Riahi, and Wolfgang Schoepp. Better air for better health: Forging synergies in policies for energy access, climate change and air pollution. *Global environmental change*, 23(5):1122–1130, 2013.

[82] Shilpa Rao and Keywan Riahi. The Role of Non-CO_3 Greenhouse Gases in Climate Change Mitigation: Long-term Scenarios for the 21st Century. *The Energy Journal*, pages 177–200, 2006.

[83] Catherine E. Raptis and Stephan Pfister. Global freshwater thermal emissions from steam-electric power plants with once-through cooling systems. *Energy*, 97:46–57, 2016.

[84] CA Reynolds, TJ Jackson, and WJ Rawls. Estimating soil water-holding capacities by linking the food and agriculture organization soil map of the world with global pedon databases and continuous pedotransfer functions. *Water Resources Research*, 36(12):3653–3662, 2000.

[85] Keywan Riahi, Frank Dentener, Dolf Gielen, Arnulf Grubler, Jessica Jewell, Zbigniew Klimont, Volker Krey, David McCollum, Shonali Pachauri, Shilpa Rao, Bas van Ruijven, Detlef P. van Vuuren, and Charlie Wilson. Chapter 17 - Energy Pathways for Sustainable Development. In *Global Energy Assessment - Toward a Sustainable Future*, pages 1203–1306. Cambridge University Press, Cambridge, UK and New York, NY, USA and the International Institute for Applied Systems Analysis, Laxenburg, Austria, 2012. URL: http://www.globalenergyassessment.org.

[86] Keywan Riahi, Shilpa Rao, Volker Krey, Cheolhung Cho, Vadim Chirkov, Guenther Fischer, Georg Kindermann, Nebojsa Nakicenovic, and Peter Rafaj. RCP 8.5—A scenario of comparatively high greenhouse gas emissions. *Climatic Change*, 109(1-2):33–57, 2011.

[87] Keywan Riahi and R Alexander Roehrl. Greenhouse gas emissions in a dynamics-as-usual scenario of economic and energy development. *Technological Forecasting and Social Change*, 63(2):175–205, 2000.

[88] Keywan Riahi, Edward S Rubin, and Leo Schrattenholzer. Prospects for carbon capture and sequestration technologies assuming their technological learning. *Energy*, 29(9):1309–1318, 2004.

[89] Keywan Riahi, Detlef P. van Vuuren, Elmar Kriegler, Jae Edmonds, Brian O'Neill, Shinichiro Fujimori, Nico Bauer, Katherine Calvin, Rob Dellink, Oliver Fricko, Wolfgang Lutz, Alexander Popp, Jesus Crespo Cuaresma, Samir KC, Marian Leimbach, Leiwen Jiang, Tom Kram, Shilpa Rao, Johannes Emmerling, Kristie Ebi, Tomoko Hasegawa, Petr Havlik, Florian Humpenoder, Lara Aleluia Da Silva, Steve Smith, Elke Stehfest, Valentina Bosetti, Jiyong Eom, David Gernaat, Toshihiko Masui, Joeri Rogelj, Jessica Strefler, Laurent Drouet, Volker Krey, Gunnar Luderer, Mathijs Harmsen, Kiyoshi Takahashi, Lavinia Baumstark, Jonathan Doelman, Mikiko Kainuma, Zbigniew Klimont, Giacomo Marangoni, Hermann Lotze-Campen, Michael Obersteiner, Andrzej Tabeau, and Massimo Tavoni. The Shared Socioeconomic Pathways and their Energy, Land Use, and Greenhouse Gas Emissions Implications. *Global Environmental Change*, 42:153–168, 2017. URL: http://pure.iiasa.ac.at/13280/, doi:10.1016/j.gloenvcha.2016.05.009.

[90] M. Roelfsema, H. L. van Soest, M. Harmsen, D. P. van Vuuren, C. Bertram, M. den Elzen, N. Höhne, G. Iacobuta, V. Krey, E. Kriegler, G. Luderer, K. Riahi, F. Ueckerdt, J. Després, L. Drouet, J. Emmerling, S. Frank, O. Fricko, M. Gidden, F. Humpenöder, D. Huppmann, S. Fujimori, K. Fragkiadakis, K. Gi, K. Keramidas, A. C. Köberle, L. Aleluia Reis, P. Rochedo, R. Schaeffer, K. Oshiro, Z. Vrontisi, W. Chen, G. C. Iyer, J. Edmonds, M. Kannavou, K. Jiang, R. Mathur, G. Safonov, and S. S. Vishwanathan. Taking stock of national climate policies to evaluate implementation of the paris agreement. *Nature Communications*, 2020. doi:https://doi.org/10.1038/s41467-020-15414-6.

[91] Joeri Rogelj, Oliver Fricko, Malte Meinshausen, Volker Krey, Johanna Zilliacus, and Keywan Riahi. Understanding the origin of paris agreement emission uncertainties. *Nature Communication*, pages 15748, 2017.

[92] Joeri Rogelj, David L McCollum, Brian C O'Neill, and Keywan Riahi. 2020 emissions levels required to limit warming to below 2 [thinsp][deg] C. *Nature Climate Change*, 3(4):405–412, 2013.

[93] Joeri Rogelj, David L McCollum, Andy Reisinger, Malte Meinshausen, and Keywan Riahi. Probabilistic cost estimates for climate change mitigation. *Nature*, 493(7430):79–83, 2013.

[94] Joeri Rogelj, Andy Reisinger, David L McCollum, Reto Knutti, Keywan Riahi, and Malte Meinshausen. Mitigation choices impact carbon budget size compatible with low temperature goals. *Environmental Research Letters*, 10(7):075003, 2015.

[95] H Rogner, Roberto F Aguilera, Christina Archer, Ruggero Bertani, S Bhattacharya, M Dusseault, Luc Gagnon, H Harbel, Monique Hoogwijk, and Arthur Johnson. Chapter 7 - Energy resources and potentials. In *Global*

Energy Assessment - Toward a Sustainable Future, pages 423–512. Cambridge University Press, Cambridge, UK and New York, NY, USA and the International Institute for Applied Systems Analysis, Laxenburg, Austria, 2012.

[96] Hans-Holger Rogner. An assessment of world hydrocarbon resources. *Annual review of energy and the environment*, 22(1):217–262, 1997.

[97] Aaron Ruesch and Holly K. Gibbs. New ipcc tier-1 global biomass carbon map for the year 2000. Report, Oak Ridge National Laboratory, 2008. URL: http://cdiac.ornl.gov/epubs/ndp/global_carbon/carbon_documentation.html.

[98] T. Sauer, P. Havlik, G. Kindermann, and U.A. . Schneider. Agriculture, population, land and water scarcity in a changing world - the role of irrigation. In *Congress of the European Association of Agricultural Economists*. 2008.

[99] Andreas Schafer. Structural change in energy use. *Energy Policy*, 33(4):429–437, 2005.

[100] A. L. Schloss, D. W. Kicklighter, J. Kaduk, U. Wittenberg, and The Participants of the Potsdam NPP Model Comparison. Comparing global models of terrestrial net primary productivity (npp): comparison of npp to climate and the normalized difference vegetation index (ndvi). *Global Change Biology*, 5(S1):25–34, 1999. doi:10.1046/j.1365-2486.1999.00004.x.

[101] Erich A Schneider and William C Sailor. Long-term uranium supply estimates. *Nuclear Technology*, 162(3):379–387, 2008.

[102] Uwe A. Schneider, Bruce A. McCarl, and Erwin Schmid. Agricultural sector analysis on greenhouse gas mitigation in us agriculture and forestry. *Agricultural Systems*, 94(2):128 – 140, 2007. URL: http://www.sciencedirect.com/science/article/pii/S0308521X06001028.

[103] James Seale, Anita Regmi, and Jason Bernstein. International evidence on food consumption patterns. Report 1904, USDA-ERS, October 2003. URL: http://www.ers.usda.gov/Data/InternationalFoodDemand/.

[104] C. Sere and H. Steinfeld. World livestock production systems: current status, issues and trends. Report 127, Food and Agriculture Organisation, 1996. URL: http://www.fao.org/WAIRDOCS/LEAD/X6101E/X6101E00.HTM.

[105] R. Skalsky, Z. Tarasovicova, J. Balkovic, E. Schmid, M. Fuchs, E. Moltchanova, G. Kindermann, and P. Scholtz. Geo-bene global database for bio-physical modeling v.1.0. concepts, methodologies and data.technical report. Report, IIASA, accessed 13.03.09 2008. URL: http://www.geo-bene.eu/?q=node/1734S.

[106] A. L. Sorensen. Economies of scale in biomass gasification systems. Report Interim Report IR-05-030, IIASA, 2005.

[107] B. J. Stokes, D. J. Frederick, and D. T. Curtin. Field trials of a short-rotation biomass feller buncher and selected harvesting systems. *Biomass*, 11(3):185–204, 1986. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0022984004&partnerID=40&rel=R8.2.0.

[108] Patrick Sullivan, Volker Krey, and Keywan Riahi. Impacts of considering electric sector variability and reliability in the message model. *Energy Strategy Reviews*, 1(3):157–163, 2013.

[109] T. Takayama and G.G. Judge. *Spatial and temporal price and allocation models*. North-Holland Amsterdam, 1971.

[110] Francesco N Tubiello, Mirella Salvatore, Simone Rossi, Alessandro Ferrara, Nuala Fitton, and Pete Smith. The faostat database of greenhouse gas emissions from agriculture. *Environmental Research Letters*, 8(1):015009, 2013. URL: http://stacks.iop.org/1748-9326/8/i=1/a=015009.

[111] J. Wang, C. Long, J. McNeel, and J. Baumgras. Productivity and cost of manual felling and cable skidding in central appalachian hardwood forests. *Forest Products Journal*, 54(12):45–51, 2004. URL: http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-11844274724&partnerID=40&rel=R8.2.0.

[112] J.R. Williams and VP Singh. The epic model. *Computer models of watershed hydrology*, pages 909–1000, 1995.

[113] W. Wint and T. Robinson. *Gridded livestock of the world 2007*. FAO, 2007.

[114] Liangzhi You and Stanley Wood. An entropy approach to spatial disaggregation of agricultural production. *Agricultural Systems*, 90(1-3):329 – 347, 2006. URL: http://www.sciencedirect.com/science/article/B6T3W-4JKYWM1-1/2/381253576eb09660fc9860c6c8bb8e1f.

[115] Haibo Zhai and Edward S Rubin. Performance and cost of wet and dry cooling systems for pulverized coal power plants with and without carbon capture and storage. *Energy Policy*, 38(10):5653–5660, 2010.

[116] Chao Zhang, Laura Diaz Anadon, Hongpin Mo, Zhongnan Zhao, and Zhu Liu. Water- carbon trade-off in China`s coal power industry. *Environmental science & technology*, 48(19):11082–11089, 2014.

[117] OECD and NEA. Uranium 2003: resources, production and demand. Report NEA-05291, OECD/NEA, June 2004. URL: https://www.oecd-nea.org/ndd/pubs/2004/5291-uranium-2003.pdf.

[118] World Bank Group. *World Development Indicators 2012*. World Bank Publications, 2012. ISBN 0-8213-8985-8.

PYTHON MODULE INDEX

m

INDEX

Symbols

A

B

C

D

E

F

G

H

I

N

O

P

Q

R

S

T

U

V

W

Y