"MESSAGEix models"

IIASA Energy, Climate, and Environment (ECE) Program

Mar 22, 2021

USER GUIDE

`message_ix_models` provides tools for research using the MESSAGEix-GLOBIOM family of models. These models are built in the MESSAGEix framework and on the ix modeling platform (ixmp).

CHAPTER

ONE

INSTALLATION

1.1 From PyPI

1. Run:

```
$ pip install message-ix-models
```

1.2 From source

1. Clone the code.
2. Run[1]:

```
$ pip install --no-use-pep517 --editable .
```

[1] See pypa/pip#7953.

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 whether they are **input** or **output** data.

This page describes how to store and handle such files in both `message_ix_models` and `message_data`.

2.1 Choose locations for data

These are listed in order of preference.

2.1.1 (1) `message_ix_models/data/`

- Files in this directory are **public**, and are packaged, published, and installable from PyPI with `message_ix_models`; in standard Python terms, these are "package data".
- This is the preferred location for:
  - General-purpose metadata.
  - Basic configuration, e.g. for reporting, not specific to any model variant or project.
  - Data for publicized model variants and completed/published projects.
- Data here can be loaded with *`load_package_data()`* or other, more specialized code.
- Documentation files like `doc/pkg-data/*.rst` describe the contents of these files. For example: *Node code lists*.

2.1.2 (2) `data/` directory in the `message_data` repo

- Files in this directory are **private** and not installable from PyPI (because `message_data` is not installable).
- This is the preferred location for:
  - Data for model variants and projects under current development.
  - Specific data files that cannot be made public, e.g. due to licensing issues.
- Data here can be loaded with *`load_private_data()`* or other, more specialized code.

2.1.3 (3) Other, system-specific ("local") directories

- These are the preferred location for:
  - Caches i.e. temporary data files used to speed up other code.
  - Output e.g. data or figure files generated by reporting.
- These kinds of data **should not** be committed to either `message_ix_models` or `message_data`.
- Each user **may** configure a location for these data, appropriate to their system.

  This setting can be made in multiple ways. In order of ascending precedence:

  1. The default location is the *current working directory*, i.e. whichever directory the *Command-line interface* is invoked in, or in which Python code is run that imports and uses `message_ix_models`.
  2. The `ixmp` configuration setting `message local data`.
  3. The `MESSAGE_LOCAL_DATA` environment variable.
  4. The `--local-data` CLI option and related options such as `--cache-path` or 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`. If not, use `.gitignore` files to hide these from Git.

2.2 General guidelines

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

Prefer text formats …such as e.g. CSV and YAML. 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 (1) or (2) above can be retrieved with the utility funtions mentioned, instead of hard-coded paths.

For system-specific paths (3) 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 Use (1) or (2), above, for the format, and (3) for the latter.

Use a consistent scheme for data locations For a submodule for a specific model variant or project named, e.g. `message_ix_models.model.[name]` or `message_ix_models.projects.[name]`, keep input data in a well-organized directory under `[base]/model/[name]/`, `[base]/project/[name]/`, or similar, where `[base]` is (1) or (2), 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 (3).

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. Instea, 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 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, e.g. in `message_data` or another location.

2.3.1 Fetch from a remote source

Use a configuration file in `message_ix_models` to store metadata, i.e. the Internet location and other information needed to retrieve the data. Then, write code that retrieves the data and caches it locally:

```
import requests

# Load some configuration
config = yaml.safe_load(load_package_data("big-data-source", "config.yaml"))

# Local paths for the cached raw files and extracted file(s)
cache_path = context.get_cache_path("big-data-source")
downloaded = cache_path / "downloaded_file.zip"
extracted = cache_path / "extracted_file.csv"

with open(downloaded) as f:
    remote_data = requests.get(config["url"])
    # Handle the data, writing to `f`

# Extract the data from `downloaded` to `extracted`
```

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

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 one-line text file with a hash of the full file, and the full file is stored separately. The IIASA GitHub account has up to 300 GB of space for 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 ordinary Git history. See the Git LFS documentation at the link above for more detail.

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, i.e. any original source 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 data-iea, below, 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

See model- and project-specific documentation for further context settings, e.g. `model.bare`.

Setting	Type	Description
cache_path	Path	Base path cache, e.g. as given by the `--cache-path` CLI option. Default *`local_data`*`/cache/`.
dry_run	bool	Whether an operation should be carried out, or only previewed.
local_data	Path	Base path for system-specific (3) data, e.g. as given by the `--local-data` CLI option.
platform_info	dict	Dictionary with keyword arguments for the `ixmp.Platform` constructor, from the `--platform` or `--url` CLI options.
scenario_info	dict	Dictionary with keys 'model' and 'scenario' as given by the `--model`/`--scenario` or `--url` CLI options.
url	dict	A scenario URL, e.g. as given by the `--url` CLI option.
units	pint.UnitRegistry	**Deprecated.** Use `from iam_units import registry`.

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 `ixmp.config`. List existing platforms:

```
$ ixmp platform list
```

You probably want to add one of the IIASA ENE databases such as the main or 'production' database. These are listed on the ENE wiki. Use the ixmp CLI:

```
$ ixmp platform add [PLATFORMNAME] jdbc oracle gp3.iiasa.ac.at:1521:GP2 [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` stores only one configuration value in `config.json`: `'message model dir'`, the path to the GAMS model files. MESSAGEix-GLOBIOM uses the GAMS model files from the current `message_ix` `master` branch, so you should not set this, or unset it when using `message_ix_models`.

`message_ix_models` will use the `config.json` value `"message_local_data"` for local data, if it is set and not overridden.

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-data` 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:

```
$ mix-models --help
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, e.g.:

          mix-models cd-links --help
          mix-models cd-links run --help

  The top-level options --platform, --model, and --scenario are used by
  commands that access specific message_ix scenarios; these can also be
  specified with --url.

  For more information, see
  https://docs.messageix.org/projects/models2/en/latest/cli.html

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

Commands:
  cd-links         CD-LINKS project.
  dl               Retrieve data from primary sources.
  engage           ENGAGE project.
  iiasapp          Import power plant capacity.
  material         Model with materials accounting.
  prep-submission  Prepare scenarios for submission to database.
  report           Postprocess results.
  res              MESSAGE-GLOBIOM reference energy system (RES).
  techs            Export data from data/technology.yaml to CSV.
  transport        MESSAGEix-Transport variant.
```

To explain further:

`--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`/`--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

REPRODUCIBILITY

4.1 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 indicator 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.

4.2 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 for.

tests	Test suite for `message_ix_models`.

4.2.1 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.test_import	
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_cli	

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.test_import

Functions

test_import()	Test that the package can be imported.

message_ix_models.tests.test_import.test_import

message_ix_models.tests.test_import.**test_import**()
Test that the package can be imported.

message_ix_models.tests.util

Tests of submodules of *message_ix_models.util*.

Modules

message_ix_models.tests.util.test_click	Basic tests of the command line.
message_ix_models.tests.util.test_logging	
message_ix_models.tests.util.test_scenarioinfo	

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_store_context()	Test *store_context()*.

message_ix_models.tests.util.test_click.test_default_path_cb

message_ix_models.tests.util.test_click.**test_default_path_cb**(*session_context*)
Test *default_path_cb()*.

message_ix_models.tests.util.test_click.test_store_context

message_ix_models.tests.util.test_click.**test_store_context**()
Test *store_context()*.

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*)

message_ix_models.tests.util.test_scenarioinfo

Classes

TestScenarioInfo()	

message_ix_models.tests.util.test_scenarioinfo.TestScenarioInfo

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

Bases: object

__init__()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__()	Initialize self.
test_empty()	ScenarioInfo created from scratch.
test_from_scenario(test_context)	ScenarioInfo initialized from an existing Scenario.

test_empty()

ScenarioInfo created from scratch.

test_from_scenario(*test_context*)

ScenarioInfo initialized from an existing Scenario.

4.3 Continuous testing

The test suite (*message_ix_models.tests*) is run using GitHub Actions for new commits on the `main` branch, or on any branch associated with a pull request.

CHAPTER

FIVE

MODELS AND MODEL VARIANTS

5.1 `model.structure`: Model structure information

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

Return codes for the 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>,
```

(continues on next page)

(continued from previous page)

```
 <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',
 '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>
```

CHAPTER

SIX

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.

6.1 Context settings

Setting	Type	Description
regions	str	The 'node' set (regional aggregation) to use; must be "R14" (default), "R11", "RCP" or "ISR".
period_start	int	Model start period; must be 2010.
period_end	int	Model end period; must be 2110.
time_step	int	The duration of periods, i.e. for 'year' and 'duration_period'. Either 10 (default) or 5.
res_with_dummies	bool	If `True`, create and include dummy technologies. See `get_dummy_data()`. Default `False`

See documentation for further context settings in *Top-level settings*.

6.2 Code reference

CHAPTER

SEVEN

BUILDING MODELS (`MODEL.BUILD`)

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

- A *scenario* to be used as a basis.
- A specification, or *spec*, which is a `dict` 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 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']` 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 `apply_par_data()`.

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

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

7.1 Code reference

CHAPTER

EIGHT

SPECIFIC RESEARCH PROJECTS

CHAPTER

NINE

GENERAL PURPOSE MODELING TOOLS

CHAPTER

TEN

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.
scenarioinfo	`ScenarioInfo` class.

Commonly used:

as_codes(data)	Convert *data* to a `list` of `Code` objects.
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.
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`.
Context(*args, **kwargs)	Context and settings for `message_ix_models` code.
ScenarioInfo([scenario])	Information about a `Scenario` object.

message_ix_models.util.**PACKAGE_DATA: Dict[str, Any] = {}**
Package data already loaded with *load_package_data()*.

message_ix_models.util.**PRIVATE_DATA: Dict[str, Any] = {}**
Data already loaded with *load_private_data()*.

message_ix_models.util.**add_par_data**(*scenario: message_ix.core.Scenario*, *data: Mapping[str, pandas.core.frame.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.**as_codes**(*data: Union[List[str], Dict[str, Dict]]*) → List[sdmx.model.Code]

Convert *data* to a `list` of `Code` objects.

Various inputs are accepted:

- `list` of `str`.
- `dict`, in which keys are `Code.id` and values are further `dict` with keys matching other `Code` attributes.

message_ix_models.util.**eval_anno**(*obj: sdmx.model.AnnotableArtefact*, *id: str*)

Retrieve the annotation *id* from *obj*, run `eval()` on its contents.

This can be used for unpacking Python values (e.g. `dict`) stored as an annotation on a `Code`.

Returns `None` if no attribute exists with the given *id*.

message_ix_models.util.**iter_parameters**(*set_name*)

Iterate over MESSAGEix parameters with *set_name* as a dimension.

Parameters **set_name** (*str*) – Name of a set.

Yields *str* – Names of parameters that have *set_name* indexing 1 dimension.

message_ix_models.util.**load_package_data**(**parts: str*, *suffix: Optional[str] = '.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.`**`package_data_path`**(**parts*) → pathlib.Path
Construct a path to a file under `message_ix_models/data/`.

`message_ix_models.util.`**`private_data_path`**(**parts*) → pathlib.Path
Construct a path to a file under `data/` in `message_data`.

`message_ix_models.util.`**`strip_par_data`**(*scenario*, *set_name*, *value*, *dry_run=False*, *dump: Optional[Dict[str, pandas.core.frame.DataFrame]] = None*)
Remove data from parameters of *scenario* where *value* in *set_name*.

Returns

Return type Total number of rows removed across all parameters.

See also:

add_par_data

10.1 `util.click`

Command-line utilities.

These are used for building CLIs using `click`.

`message_ix_models.util.click.`**`PARAMS = {'dest': <Option dest>, 'dry_run': <Option dry_run>`**
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.`**`store_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.

10.2 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"].

Context provides additional methods to do common tasks that depend on configurable settings:

clone_to_dest()	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.
get_local_path(*parts[, suffix])	Return a path under local_data.
get_platform([reload])	Return a ixmp.Platform from platform_info.
get_scenario()	Return a message_ix.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*.

The following Context methods and attribute are **deprecated**:

get_config_file(*parts[, ext])	Return a path under metadata_path.
get_path(*parts)	Return a path under message_data_path by joining *parts*.
load_config(*parts[, suffix])	Load configuration from message_ix_models.
units	Access the unit registry.

clone_to_dest() → Tuple[message_ix.core.Scenario, ixmp.core.Platform]

Return a scenario based on the --dest command-line option.

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")
```

(continues on next page)

(continued from previous page)

```
@click.pass_obj
def foo(context, dest):
    scenario, mp = context.clone_to_dest()
```

or, store the settings `dest_scenario` and `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*) → pathlib.Path
Return a path to a local cache file.

get_config_file(**parts*, *ext='yaml'*) → pathlib.Path
Return a path under `metadata_path`.

The suffix ".{ext}" is added; defaulting to ".yaml".

Deprecated since version 2021.2.28: Use *package_data_path()* instead. Will be removed on or after 2021-05-28.

classmethod get_instance(*index=0*) → *message_ix_models.util.context.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*, *suffix=None*)
Return a path under `local_data`.

get_path(**parts*) → pathlib.Path
Return a path under `message_data_path` by joining *parts*.

parts may include directory names, or a filename with extension.

Deprecated since version 2021.2.28: Use *private_data_path()* instead. Will be removed on or after 2021-05-28.

get_platform(*reload=False*) → ixmp.core.Platform
Return a `ixmp.Platform` from `platform_info`.

When used through the CLI, `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() → message_ix.core.Scenario
Return a `message_ix.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*, *_store_as=('platform_info', 'scenario_info')*)
Handle command-line arguments.

May update the `data_path`, `platform_info`, `scenario_info`, and/or `url` settings.

load_config(**parts*, *suffix=None*)
Load configuration from `message_ix_models`.

Deprecated since version 2021.2.28: Use *`load_package_data()`* instead. Will be removed on or after 2021-05-28.

classmethod only() → *message_ix_models.util.context.Context*
Return the only *`Context`* instance.

Raises **IndexError** – If there is more than one instance.

property units
Access the unit registry.

Deprecated since version 2021.2.28: Instead, use:

```
from iam_units import registry
```

Will be removed on or after 2021-05-28.

use_defaults(*settings*)
Update from *settings*.

10.3 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`.

10.4 util._logging

Logging utilities.

class message_ix_models.util._logging.**Formatter**(*colorama*)
Formatter for log records.

Parameters **colorama** (*module*) – If provided, `colorama` is used to colour log messages printed to stdout.

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.

message_ix_models.util._logging.**make_formatter**()
Return a *Formatter* instance for the message_ix_models logger.

See also:

setup

message_ix_models.util._logging.**setup**(*level='NOTSET'*, *console=True*)
Initialize logging.

Parameters

- **level** (*str, optional*) – Log level for message_ix_models and message_data.
- **console** (*bool, optional*) – If True, print all messages to console using a *Formatter*.

message_ix_models.util._logging.**silence_log**()
Context manager to temporarily silence log output.

Examples

```
>>> with silence_log():
>>>     log.warning("This message is not recorded.")
```

10.5 util.scenarioinfo

ScenarioInfo class.

class message_ix_models.util.scenarioinfo.**ScenarioInfo**(*scenario=None*)
Information about a Scenario object.

Code that prepares data for a target Scenario can accept a ScenarioInfo instance. This avoids any need to load a Scenario, which can be slow under some conditions. ScenarioInfo objects are also used by apply_spec() to describe the contents of a scenario before it is created.

ScenarioInfo objects have the following attributes:

set	Elements of ixmp/message_ix sets in the Scenario.
is_message_macro	True if a MESSAGE-MACRO scenario.
N	Elements of the set 'node'.
Y	Elements of the set 'year' that are >= the first model year.
y0	First model year, if set, else Y[0].
yv_ya	(year_vtg, year_act) for the entire model horizon.

property N
Elements of the set 'node'.

property Y
Elements of the set 'year' that are >= the first model year.

is_message_macro: bool = False
`True` if a MESSAGE-MACRO scenario.

set: Dict[str, List] = {}
Elements of `ixmp`/`message_ix` sets in the Scenario.

y0: int = -1
First model year, if set, else `Y[0]`.

property yv_ya
(year_vtg, year_act) for the entire model horizon.

CHAPTER

ELEVEN

TEST UTILITIES AND FIXTURES

CHAPTER

TWELVE

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.

12.1 32-region aggregation (R32)

```
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]

R32CAS:
  parent: World
  name: Central Asia
```

(continues on next page)

(continued from previous page)

```
  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
```

(continues on next page)

(continued from previous page)

```
  name: North Africa
  description: This region includes the countries of North Africa.
  child: [DZA, EGY, ESH, LBY, MAR, TUN]

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:
```

(continues on next page)

(continued from previous page)

```
  parent: World
  name: Turkey
  child: [TUR]

R32TWN:
  parent: World
  name: Taiwan
  child: [TWN]

R32USA:
  parent: World
  name: United States of America
  description: United States of America.
  child: [PRI, USA, VIR]
```

12.2 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
#   - 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,
```

(continues on next page)

(continued from previous page)

```
    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]

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
```

(continues on next page)

(continued from previous page)

```
  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]
```

12.3 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]

R11_CPA:
  parent: World
  name: Centrally Planned Asia
  child: [CHN, HKG, KHM, LAO, MNG, PRK, VNM]

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]

R11_FSU:
  parent: World
  name: Former Soviet Union
  child: [ARM, AZE, BLR, GEO, KAZ, KGZ, MDA, RUS, TJK, TKM, UKR, UZB]

R11_LAM:
  parent: World
  name: Latin America and The Caribbean
```

(continues on next page)

(continued from previous page)

```
  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]

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]

R11_NAM:
  parent: World
  name: North America
  child: [CAN, GUM, PRI, SPM, USA, VIR]

R11_PAO:
  parent: World
  name: Pacific OECD
  child: [AUS, JPN, NZL]

R11_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]

R11_SAS:
  parent: World
  name: South Asia
  child: [AFG, BGD, BTN, IND, LKA, MDV, NPL, PAK]

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]
```

12.4 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, 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".
```

(continues on next page)

(continued from previous page)

```
    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]
```

CHAPTER

THIRTEEN

OTHER CODE LISTS

13.1 Commodities (`commodity.yaml`)

Each of these codes has the following annotations:

`level` Level where this commodity typically (not exclusively) occurs.

`unit` Units typically associated with this commodity.

```yaml
coal:
  name: Coal
  unit: GWa

crudeoil:
  name: Crude oil
  description: >-
    For secondary energy, use 'fueloil', 'lightoil', etc.
  level: primary
  unit: GWa

d_heat:
  name: (?) District heat
  description: >-
    FIXME provide an unambiguous description of what this commodity represents.

electr:
  name: Electricity
  unit: GWa

ethanol:
  name: Ethanol
  unit: GWa

freshwater_supply:
  name: (?) Fresh water
  description: >-
```

(continues on next page)

(continued from previous page)

```
    FIXME provide an unambiguous description of what this commodity represents.

fueloil:
  name: Fuel oil
  description: Heavy fuel oil.
  level: secondary
  unit: GWa

gas:
  name: Natural Gas
  unit: GWa

hydrogen:
  name: Gaseous hydrogen
  unit: GWa

lh2:
  name: Liquid hydrogen
  unit: GWa

lightoil:
  name: Light oil
  description: Includes gasoline, diesel oil.
  level: secondary
  unit: GWa

methanol:
  name: Methanol
  unit: GWa

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
  unit: GWa

# 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
# 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
```

(continues on next page)

(continued from previous page)

```
# 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
# biomass
# 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
# crude_7
# crude_8
# Def_CO2_G4M
# Def_CO2_GLO
# dumagr
# dumfert
```

(continues on next page)

(continued from previous page)

```
# 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
# gas_nam
# gas_pao
```

(continues on next page)

(continued from previous page)

```
# 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
# lh2
# lignite
# LiquidTotal
# LNG
# LoggingResidues
# LU_CO2
# LU_GHG
# LucGrs_CO2
# LucOth_CO2
# NewFor_G4M
# NH3_LandUseChangeEM
# NH3_ManureEM
# NH3_RiceEM
# NH3_SavanBurnEM
# NH3_SoilEM
# non-comm
# 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
```

(continues on next page)

(continued from previous page)

```
# Price|Agriculture|Non-Energy Crops|Index
# Price|Primary Energy|Biomass
# pu
# puq
# puq2
# rc_spec
# rc_therm
# 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
```

13.2 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
```

(continues on next page)

(continued from previous page)

```
  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.

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
# import
# land_use_reporting
# land_use
# piped-gas
# resource
# stocks
# water_supply_constraint
```

13.3 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.
```

(continues on next page)

(continued from previous page)

```
#   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'.
#
#   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
```

(continues on next page)

(continued from previous page)

```
  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␣
↪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
```

(continues on next page)

(continued from previous page)

```
  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␣
↪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
```

(continues on next page)

(continued from previous page)

```
  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
  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]
```

(continues on next page)

(continued from previous page)

```yaml
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
  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
```

(continues on next page)

(continued from previous page)

```
  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]
  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]
```

(continues on next page)

(continued from previous page)

```
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]
  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
```

(continues on next page)

(continued from previous page)

```
  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
  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:
```

(continues on next page)

(continued from previous page)

```
  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]

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]
```

(continues on next page)

(continued from previous page)

```
  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:
  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
```

(continues on next page)

(continued from previous page)

```
  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
  sector: electricity
  input: [electr, secondary]
  output: [exports, secondary]

wind_curtailment1:
  name: wind_curtailment1
```

(continues on next page)

(continued from previous page)

```
  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:
  name: wind_ppl
  description: Wind power plant onshore (provides capacity only)
```

(continues on next page)

(continued from previous page)

```
  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
```

(continues on next page)

(continued from previous page)

```
  description: Wind offshore potential and generation 3
  type: secondary
  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
```

(continues on next page)

(continued from previous page)

```
  output: [electr, secondary]

csp_sm3_res5:
  name: csp_sm3_res5
  description: Concentrating solar power (CSP) with solar multiple of 3 potential and␣
↪generation 6
  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
```

(continues on next page)

(continued from previous page)

```
  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]

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]
```

(continues on next page)

(continued from previous page)

```
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]

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
```

(continues on next page)

(continued from previous page)

```
  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
  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]
```

(continues on next page)

(continued from previous page)

```yaml
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]
  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]
```

(continues on next page)

(continued from previous page)

```yaml
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:
  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]
```

(continues on next page)

(continued from previous page)

```
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
  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]
```

(continues on next page)

(continued from previous page)

```
  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:
    - [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
```

(continues on next page)

(continued from previous page)

```
  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: gas
  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
  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]
```

(continues on next page)

(continued from previous page)

```
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:
  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
```

(continues on next page)

(continued from previous page)

```
  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
  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:
```

(continues on next page)

(continued from previous page)

```
  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:
  name: h2_coal_ccs
  description: Hydrogen production via coal gasification with carbon capture and␣
↪storage
  type: secondary
  vintaged: TRUE
  sector: hydrogen
  input:
    - [coal, secondary]
```

(continues on next page)

(continued from previous page)

```
    - [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 technology)
  type: final
  vintaged: TRUE
  sector: hydrogen
  input: [hydrogen, secondary]
```

(continues on next page)

(continued from previous page)

```
  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
  type: final
```

(continues on next page)

(continued from previous page)

```
  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
```

(continues on next page)

(continued from previous page)

```
  input: [ethanol, final]
  output: [i_therm, useful]

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]
```

(continues on next page)

(continued from previous page)

```
loil_i:
  name: loil_i
  description: Lightoil for thermal uses in industry thermal
  type: useful
  vintaged: TRUE
  sector: industry
  input: [lightoil, final]
  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
```

(continues on next page)

(continued from previous page)

```
  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
  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:
```

(continues on next page)

(continued from previous page)

```
  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]
  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]
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
  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]

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:
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
  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]

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
```

(continues on next page)

(continued from previous page)

```
  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:
  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:
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
  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
  output: [dummy, primary]

landfill_oxdn:
  name: landfill_oxdn
  description: Landfill mitigation technology (oxidation) 1 for CH4 mitigation
  type: dummy
  vintaged: TRUE
```

(continues on next page)

(continued from previous page)

```
  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
  sector: non-co2
  output: [dummy, primary]
```

(continues on next page)

(continued from previous page)

```yaml
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
```

(continues on next page)

(continued from previous page)

```
  sector: other conversion
  output: [exports, secondary]

back_RC:
  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]
```

(continues on next page)

(continued from previous page)

```
h2_rc:
  name: h2_rc
  description: Hydrogen (gaseous) catalytic heating in
  type: secondary
  sector: residential/commercial
  input: [hydrogen, final]
  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]
```

(continues on next page)

(continued from previous page)

```
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
  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]
```

(continues on next page)

(continued from previous page)

```
RCtherm_4:
  name: RCtherm_4
  description: Conservation cost curve step for residential/commercial thermal demand
  type: useful
  sector: residential/commercial
  output: [rc_therm, useful]

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:
```

(continues on next page)

(continued from previous page)

```
  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
  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]
```

(continues on next page)

(continued from previous page)

```
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
  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]
```

(continues on next page)

(continued from previous page)

```
  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]

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:
```

(continues on next page)

(continued from previous page)

```
  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
  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:
```

(continues on next page)

(continued from previous page)

```
  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
  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]
```

(continues on next page)

(continued from previous page)

```
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]
```

CHAPTER

FOURTEEN

WHAT'S NEW

14.1 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()`*.

14.2 2021.3.3

- PR #8: Migrate *`util.click`*, `util.logging`; expand documentation.
- *`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.

14.3 2021.2.28

- PR #5: Migrate *`Context`* class and `testing` module from `message_data`.
- Add *`load_private_data()`*, *`package_data_path()`*, *`private_data_path()`*.
- Document: *Data, metadata, and configuration* and *Command-line interface*.
- PR #6: Update *node codelists* to ensure they contain both current and former ISO 3166 codes for countries that have changed status. 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.

14.4 2021.2.26

- PR #2: Add `get_codes()` and related code lists.
- PR #3: Add `MessageDataFinder` and document *Migrating from message_data*.

14.5 2021.2.23

Initial release.

CHAPTER

FIFTEEN

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.

This page 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.

CHAPTER

SIXTEEN

RELEASING

16.1 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.

16.2 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

SEVENTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

Symbols

A

C

D

E

F

G

H

I

L

M

N

O

P

S

T

U

Y