ece-docs

IIASA ECE Scenario Services Team

Feb 26, 2026

CONTENTS

CHAPTER

ONE

THE IAMC DATA FORMAT

The *Integrated Assessment Modeling Consortium* (IAMC) developed a standardised tabular timeseries format to exchange scenario data related to energy systems modelling, land-use change, demand sectors, and economic indicators in the context of climate change mitigation pathways and the the Sustainable Development Goals (SDGs).

Read more about the *The IAMC timeseries data format*…

CHAPTER

TWO

SCENARIO DATABASES HOSTED BY IIASA

The *Energy, Climate and Environment* program (ECE) at IIASA is hosting many databases for the community to facilitate scenario analysis, model comparison and dissemination.

Read more about the *Scenario databases hosted by IIASA*…

CHAPTER THREE

STANDARDIZED NAMING CONVENTIONS AND GUIDELINES

Having agreed standards and guidelines on naming conventions is crucial for consistency and reusability in scenario analysis.

Read more about the *Standardized naming conventions*…

CHAPTER

FOUR

OPEN-SOURCE SOFTWARE PACKAGES

The is developing several open-source Python packages for analysis, processing and management of scenario data in the IAMC format.

Read more about the *Open-source software packages*…

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 The IAMC timeseries data format

5.1.1 Background

The *Integrated Assessment Modeling Consortium* (IAMC) developed a standardised tabular timeseries format to exchange scenario data related to energy systems modelling, land-use change, demand sectors, and economic indicators in the context of climate change mitigation pathways and the the Sustainable Development Goals (SDGs). Previous high-level use cases include reports by the *Intergovernmental Panel on Climate Change* (IPCC) and model comparison exercises within the *Energy Modeling Forum* (EMF) hosted by Stanford University.

Refer to https://data.ece.iiasa.ac.at/ for a selected list of projects using the IAMC data format.

5.1.2 An illustrative example

The table below shows a typical example of integrated-assessment scenario data following the IAMC format from the Horizon 2020 CD-LINKS project.

The scenario data is usually saved as *xlsx* or *csv* files in wide format (years as columns), but other types are also possible (e.g., via the pyam package).

	A	B	C	D	E	F	G	H
1	**Model**	**Scenario**	**Region**	**Variable**	**Unit**	**2005**	**2010**	**2015**
2	MESSAGE	CD-LINKS 400	World	Primary Energy	EJ/y	462.5	500.7	...

Fig. 1: Illustrative example of IAMC-format timeseries data from the IAMC 1.5°C Scenario Explorer

5.1.3 Naming conventions and standards

Read more about guidelines and naming conventions for *Variables in the IAMC format* and *Common and native regions*.

5.1.4 Project "templates"

Each project using the IAMC timeseries data format defines a list of "variables" and regions for comparison and scenario analysis, commonly known as a "variable template".

The IAM community is developing a shared resource of variable and region definitions. The aim is to provide a central location to facilitate reuse of definitions and mappings across projects.

Visit https://github.com/IAMconsortium/common-definitions for more information.

5.1.5 Related software packages

Read more about related *Open-source software packages*.

5.2 Scenario databases hosted by IIASA

The *Energy, Climate and Environment* program (ECE) at IIASA is hosting many databases for the community to facilitate scenario analysis, model comparison and dissemination.

5.2.1 Types of *Scenario Explorer*

Scenario Apps and ixmp4 instances (2024-)

New *Scenario Explorer* instances use the *ScSe Apps infrastructure* and the **ixmp4** package as a database backend, which were developed in 2024.

Legacy *Scenario Explorer* instances (2018-2024)

The previous generation of *Scenario Explorer* databases was developed in 2018 for the IPCC's *Special Report on Global Warming of 1.5°C* (SR15) and used for projects that started between 2018 and 2024.

Legacy *Web Apps* (2008-2018)

The first generation of web databases and user interfaces was developed and used for numerous projects including IPCC AR5 until 2018.

5.2.2 Access and permission management

There are public and project-internal (private) database instances or **platforms**. Access to project-internal platforms is managed via the Scenario Services Manager. To participate in a project, please create an account and send the username to the project managers by email.

You can see all public and private database instances to which you have access, including your view/submit/edit permissions, in the *Services* tab of the manager.

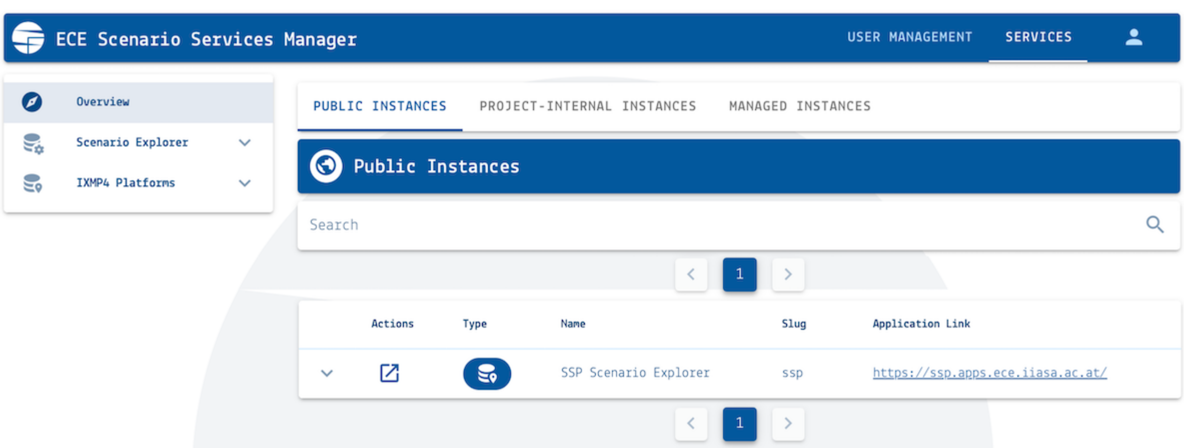


Fig. 2: Screenshot of the "Overview" page of the Scenario Services Manager

5.2.3 Model registration

To participate in a model comparison project using the IIASA scenario database infrastructure, you have to "register" your model. A model registration requires three specifications:

- A model name including a *version number*, preferably using semantic versioning
- A list or mapping of region names as they will be submitted (uploaded) to an IIASA database instance by the modeling team, and how the region names should appear in the processed scenario data
- A model mapping to perform region aggregation from *native_regions* to *common_regions* and renaming of model native regions (optional)

Note

When to change the model version number

The version number should (only) be changed if:

a) the regional resolution of the model changes (e.g., adding more countries/regions), or

b) new sectors or modules are added to the model

Minor updates or changes do not require a new version number.

Option 1) Registration using an Excel template

If you do not feel comfortable using GitHub, please use the Excel template and send it to the project managers by email.

Tip

Do you need to submit a model registration spreadsheet?

If your model uses "standard" regions like countries (see *A common list of country names*), NUTS regions (see *NUTS classification*), or US states, you typically do **not** need to submit a model registration spreadsheet.

A model registration is primarily needed for models with custom regional definitions or aggregations that require explicit mapping.

Option 2) Registration using a GitHub pull request

The preferred approach for model registration is starting a GitHub pull request. Please contact the administrators if permissions for the project repository are required.

Please follow the Model registration user guide.

5.2.4 Scenario version management

When submitting a scenario (a.k.a. "run") to an IIASA database instance with an already existing model-scenario combination, the database will save the new submission as a new version of that run. The **version number** is incremented automatically and the new version will be automatically set as **default version** for that model-scenario name.

To select other (non-default) versions, you can use the "Switch to Advanced View" button in the scenario-selection tab of an IIASA Scenario Explorer or you can use the `default_only=False` option of the function `pyam.read_iiasa()` or the **ixmp4** package (read the docs), see also the Section *Database API*.

5.2.5 Scenario processing

When submitting a scenario (a.k.a. "run") to an IIASA database instance, the server executes a scenario-processing workflow including *region-aggregation* and *scenario validation* prior to saving the scenario to the database. The processing uses the **nomenclature** package (read the docs).

The region-aggregation and validation is configured via a project-specific GitHub repository, usually named https://github.com/iiasa/<project>-workflow. Please contact the respective project managers or the Scenario Services team if you need access.

You can also run the project workflow locally (on your computer) before submission to an IIASA database instance, to make sure that the validation and processing works. See *Executing scenario processing locally* for more information.

The workflow for processing files uploaded via the IIASA Scenario Explorer is implemented in a modular fashion. It is possible to execute programs, code and tools developed by (non-IIASA) research partners as part of the processing workflow if the tool follows the *Requirements for processing modules*.

5.2.6 Database API

You can query scenario data from an IIASA database instance directly via Python or R. Refer to the User Guide for *Querying IIASA databases* for more information!

5.3 Standardized naming conventions

The IAMC community and the has developed standardized naming conventions, guidelines and tools to facilitate consistency and reusability in scenario analysis.

5.3.1 Variables in the IAMC format

The 'variable' column of the IAMC format describes the type of information represented in the specific timeseries. The variable name implements a "semi-hierarchical" structure using the `|` character (*pipe*, not l or i) to indicate the structure or "depth". Names (should) follow a structure like *Category|Subcategory|Specification*.

Semi-hierarchical means that a hierarchy can be imposed, e.g., one can enforce that the sum of *Emissions|CO2|Energy* and *Emissions|CO2|Other* must be equal to *Emissions|CO2* (if there are no other *Emissions|CO2|…* variables).

However, this is not mandatory, e.g., the sum of *Primary Energy|Coal*, *Primary Energy|Gas* and *Primary Energy|Fossil* should not be equal to *Primary Energy* because this would double-count fossil fuels.

Variable naming conventions

A variable name should adhere to the following conventions:

- A | (pipe) character indicates levels of hierarchy.
- Do not use spaces before and after the | character, but add a space between words

```
Primary Energy|Non-Biomass Renewables
```

- Do not use abbreviations (e.g, *PHEV*) unless strictly necessary.
- Do not use abbreviations of statistical operations (*min*, *max*, *avg*) but always spell out the term.
- All words must be capitalised (except for *and*, *w/*, *w/o*, etc.).
- If necessary to add a method or operation-identifier (e.g., Share, per capita) to a variable name, add it in square brackets, e.g.,

```
Population|Urban [Share]
```

Units

The **unit** associated with a variable should be compatible with the iam-units package. Alternatively, a variable can be *dimensionless*, i.e., not have a unit. An example of such a dimensionless variable are indices, e.g., the *Human Development Index*.

5.3.2 Common and native regions

Common regions

In model-comparison projects, it is useful to define **common regions** that can be computed consistently from original model results.

A widely used example are the R5, R9 and R10 regions.

Native model regions

In contrast to common regions used for comparison of scenarios across models, each model has a **native region** resolution.

Models with a coarse spatial resolution should add a model-specific identifier to the native model regions (e.g., *MESSAGEix-GLOBIOM 1.1|North America*) to avoid confusion when comparing results to other models with similar-but-different regions.

If a model has a country-level resolution (where disambiguation is not a concern), we recommend to *not add* a model identifier. Instead, use the naming convention following the common list of countries (see *A common list of country names*).

5.3.3 A common list of country names

Having a standardized list of country names is an important prerequisite for region definitions, scenario analysis and model comparison.

The `nomenclature` package (read the docs) builds on the `pycountry` package to provide a standardized list of country names based on the ISO 3166-1 standard.

For consistency with established conventions in the modelling community, several country names are shortened compared to ISO 3166-1, e.g. from "Bolivia, Plurinational State of" to "Bolivia". See the full list of changes on GitHub.

Also, "Kosovo" is added (with *alpha_3* code "KOS", following the IOC), even though it is not a universally recognized state and not oficially included in ISO 3166-1.

You can access the list of countries via the model-registration Excel template or via the Python API.

The `nomenclature` package also provides utility functions to work with the country-names and easily translate between country names and alpha-3/alpha-2 codes (also known as ISO3 and ISO2 codes). Here is an example:

```python
from nomenclature import countries

# list of country names
countries.names

# mappings between alpha_3 (ISO3), alpha_2 and country names
name = countries.get(alpha_3="...").name
alpha_3 = countries.get(name="...").alpha_3
alpha_2 = countries.get(name="...").alpha_2
```

5.3.4 NUTS classification

The `nomenclature` (read the docs) package makes use of the `pysquirrel` package (*more information here*) to provide a utility for region names based on the NUTS classification.

This feature allows users to easily import the NUTS regions which are territorial units with multiple levels of resolution, adding functionality to facilitate scenario analysis and model comparison.

The full list of NUTS regions is accessible in Eurostat's Excel file.

The geojson files for the NUTS 1, 2 and 3 regions are available in the [scse-geojson](https://github.com/iiasa/scse-geojson) repository.

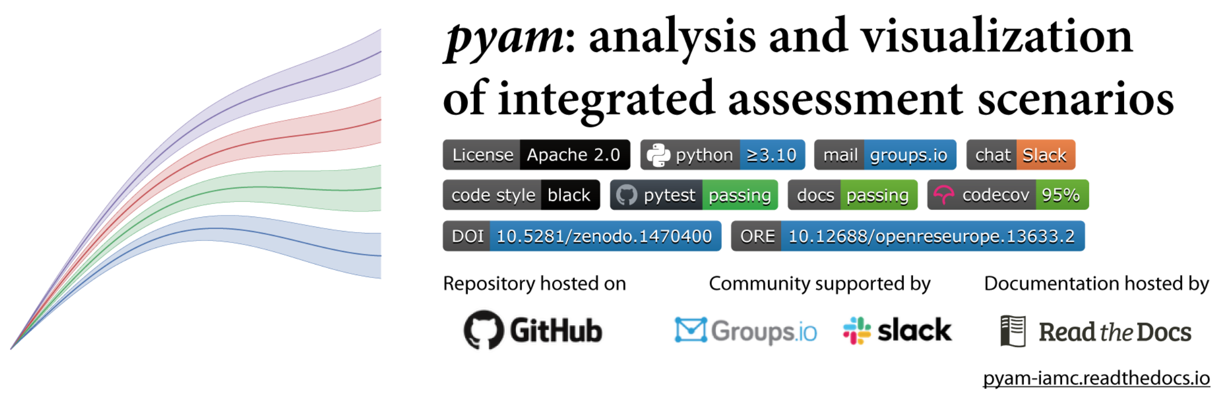
```python
from nomenclature import nuts

# list of NUTS region codes
nuts.codes

# list of NUTS region names
nuts.names
```

5.4 Open-source software packages

The is developing several open-source Python packages to work with scenario data in *The IAMC timeseries data format*.

5.4.1 Scenario analysis: pyam

Analysis and visualization of integrated-assessment & macro-energy scenarios



https://pyam-iamc.readthedocs.io

5.4.2 Scenario processing: nomenclature

Management of IAMC-format project definitions

https://nomenclature-iamc.readthedocs.io

5.4.3 Data warehouse: ixmp4

Data warehouse for high-powered scenario analysis in the domain of integrated assessment of climate change and energy systems modeling

https://docs.ece.iiasa.ac.at/projects/ixmp4

5.4.4 NUTS utility: pysquirrel

pysquirrel is a Python package designed to work with NUTS administrative divisions, searching the list of Eurostat territorial units.

For more information on how to install and use the package please refer to the README on GitHub.

https://github.com/iiasa/pysquirrel

5.5 User Guide & FAQ

5.5.1 Frequently Asked Questions

Why are there different *Scenario Explorers* for specific projects?

Each project brings together different consortium partners, focuses on a different research question, and uses aproject-specific list of variable and regions (see *Project "templates"*).

It is therefore easier to keep projects separate, with its own database platform, an independent *Scenario Explorer* and a related GitHub repository (see *Scenario processing*).

What is the level of spatial granularity of scenarios?

The spatial granularity of the scenario data depends on the scope of the project and the contributing modeling teams. The IAMC data format supports both aggregated data as well as more detailed regional data like country-level resolution (see *A common list of country names*) or the *NUTS classification* used by the European Union.

Which sectors are covered in the variable template?

The sectoral coverage of the data format is flexible and can be adapted to the needs of the project. Read more about the *Variables in the IAMC format*.

You can also look at the common-definitions repository for a list of variables and regions used in several projects to get an idea of typical use cases.

5.5.2 Detailed User Guides

Querying IIASA databases

You can query scenario data from an IIASA database instance directly via Python, R or a Rest API. The **pyam** package allows to directly query both the new *Scenario Apps* and the (legacy) *Scenario Explorer* databases (2018-2024).

```
import pyam
df = pyam.read_iiasa("<slug>", region="World", ...)
```

Here, the *slug* is the name of the database. To avoid large queries, you can also filter by model, scenario or variable.

Tip

Learn more about the Python package **pyam** on Read The Docs.

There is a tutorial to use **pyam** with **R** via the **reticulate** package.

The approach for more complex queries depends on the database backend:

- *Scenario Apps and ixmp4 instances*
- *Legacy Scenario Explorer instances*

Note

Access and permission management for project-internal databases

By default, your can connect to all public scenario database instances via the API. If you have permission to connect to a private, project-internal database instance, you can set your credentials by running the following command in a console:

```
ixmp4 login <username>
```

You will be prompted to enter your password.

Your username and password will be saved locally in plain-text for future use.

Scenario Apps and ixmp4 instances

New *Scenario Explorer* instances (set up since 2025) use the *ScSe Apps infrastructure* and the **ixmp4** package as a database backend. You can list all **ixmp4** platforms hosted by IIASA (and to which you have access) using `pyam.iiasa.platforms()`.

You can use `pyam.read_iiasa()` for simple queries or the **ixmp4** package for connecting to a platform and executing other requests.

```
import ixmp4

# connect to a database platform
platform = ixmp4.Platform("<slug>")

# get a table of all "scenario runs" in the database
platform.runs.tabulate()

# get a table of all IAMC variables
platform.iamc.variables.tabulate()
```

Legacy *Scenario Explorer* instances

You can use the **pyam** package to connect to a legacy *Scenario Explorer* instance developed by the Scenario Services and Scientific Software team from 2018 until 2024.

You can use `pyam.read_iiasa()` for simple queries or the `pyam.iiasa.Connection` class to get a list of available instances or for more elaborate queries.

```
import pyam

# get a list of all available legacy database instances
pyam.iiasa.Connection().valid_connections

# connect to a specific legacy database instance
database = pyam.iiasa.Connection("<slug>")

# get a table of all scenarios in this database
database.properties()
```

Executing scenario processing locally

A project workflow can be run on a scenario locally, before submission to an IIASA database instance. This can be useful to test region-aggregation and validation.

To run the project workflow locally, do the following:

1. Install Python and run `pip install nomenclature-iamc` to install the package
2. Git-clone the project repository (usually https://github.com/iiasa/<project>-workflow)

Then, run the workflow on a scenario data file using the terminal (from the folder where the project repository was cloned to).

```
nomenclature run-workflow <path-to-scenario-file>.xlsx
```

The output log in the terminal will show any validation errors. If the output is empty, the validation was successful.

> **Warning**
>
> Make sure to pull the latest project repository and update the **nomenclature** package when you repeat this step later to make sure that the IIASA infrastructure and your local check are in sync.

Read the User Guide of the **nomenclature** package for more information!

Requirements for processing modules

Any module (a.k.a. program, code or tool) must adhere to the following standards of best-practice software development. The aim of these guidelines is to ensure reliability of our services, minimize maintenance requirements, and guarantee reproducibility of results across platforms.

General requirements

- The program, code or tool must be implemented in Python (3.10) or R; compiled executables are not acceptable for security reasons
- Distribution of the source code - via an online version-controlled repository (preferably GitHub) to which the IIASA admin team has access; or - installation via a package manager (pip, conda, CRAN).
- The program must run on Debian (preferably Ubuntu)
- The dependencies must be clearly stated, e.g. as Dockerfile (describing execution environment, library dependencies etc.) Python package dependencies according to packaging user guide (e.g. as environment.yml, requirements.txt etc.) R dependencies
- The license must be clearly stated.
- The documentation of the program, code or tool must include: - Purpose of the program and individual top-level functions - Instructions how to run the program - Expected input (variables, region mappings) and standard output - Explanation of any settings and optional parameters

Application programming interface

Option 1:

The module is called via a command-line interface (CLI) and take the following arguments:

- `input`: path to an IAMC-formatted file (`xlsx` or `csv`)
- `output`: path where to write an output file (usually derived timeseries data) in the same format

- Any relevant settings and optional parameters must also be specified via the CLI

e.g. `"python process.py --input path-to-input-file.xlsx --output path-to-output-file.xlsx"`

Option 2 (applicable for packages/functions written in Python):

Importable Python functions that take and return `pandas.DataFrame` (with columns folllowing the IAMC format) or `pyam.IamDataFrame` objects can be called as part of the processing workflow. Any settings or optional parameters must be given as keyword arguments to the top-level function, preferably with the option to set them via a settings or configuration file.

CHAPTER SIX

LICENSE & SOURCE

The source code of this Sphinx project is available at https://github.com/iiasa/ece-docs under a Creative Commons CC0 1.0 License. The project is maintained by the **Scenario Services & Scientific Software** team in the *Energy, Climate, and Environment* program (ECE) at IIASA.

Visit https://software.ece.iiasa.ac.at for more information.