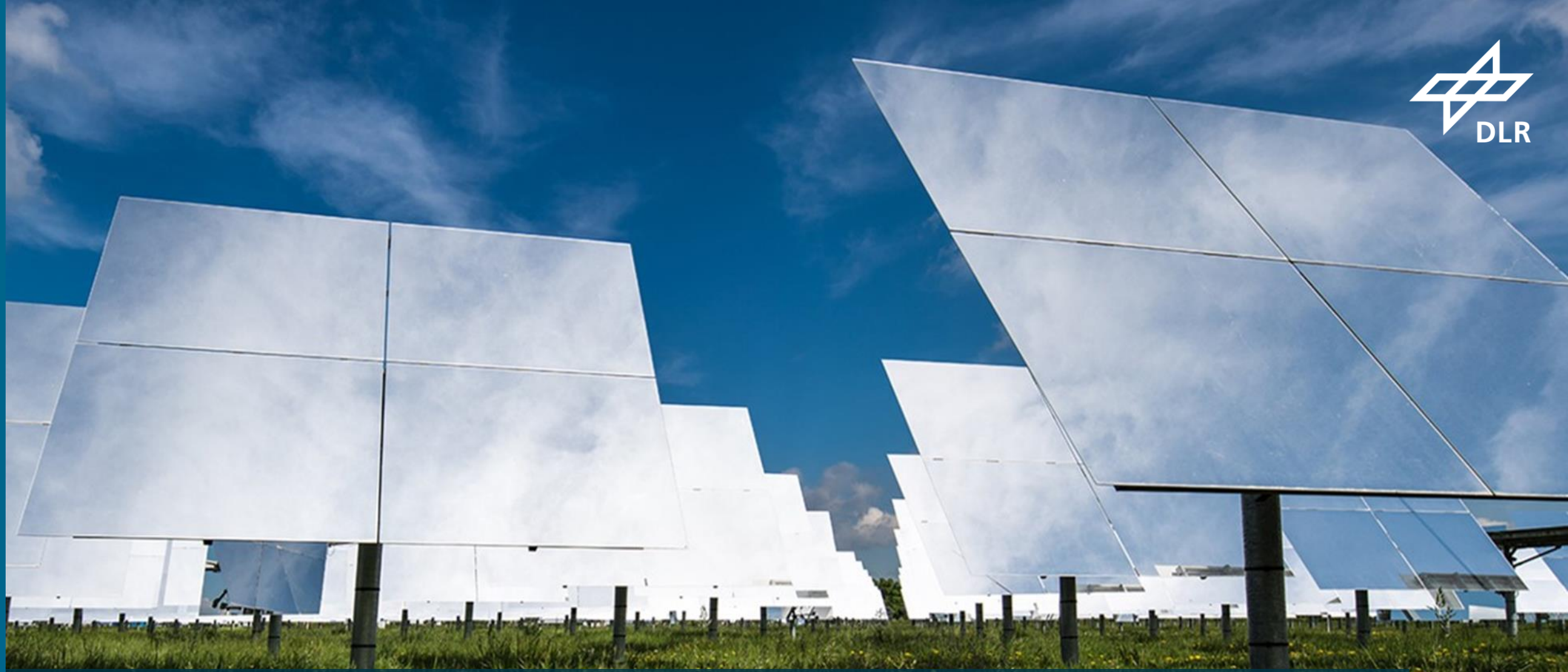
HANDS-ON: USING AMIRIS

open **A**gent-based **M**arket Model for the
Investigation of **R**enewable and **I**ntegrated Energy **S**ystems

Christoph Schimeczek, Institute of Networked Energy Systems



WELCOME

Christoph Schimeczek, Institute of Networked Energy Systems

Motivation
Market Modelling with AMIRIS

Transformation to **renewable-dominated** energy system

> Rising shares of fluctuating renewable energies
> Alignment of supply and demand challenging
> Electricity prices / Refinancing uncertain

Energy systems are **complex systems**

> Market actors' behaviour under uncertainty
> Interdependencies of actors
> Emergent and non-linear effects

Aim

- Understand **market effects** of integrating renewables and flexibilities
- Consider **actors' behaviour**, uncertainty and market distortions caused by regulatory framework
- Study **policy instruments** to incite system-friendly investment and operational decisions

Motivation

Market Modelling with AMIRIS

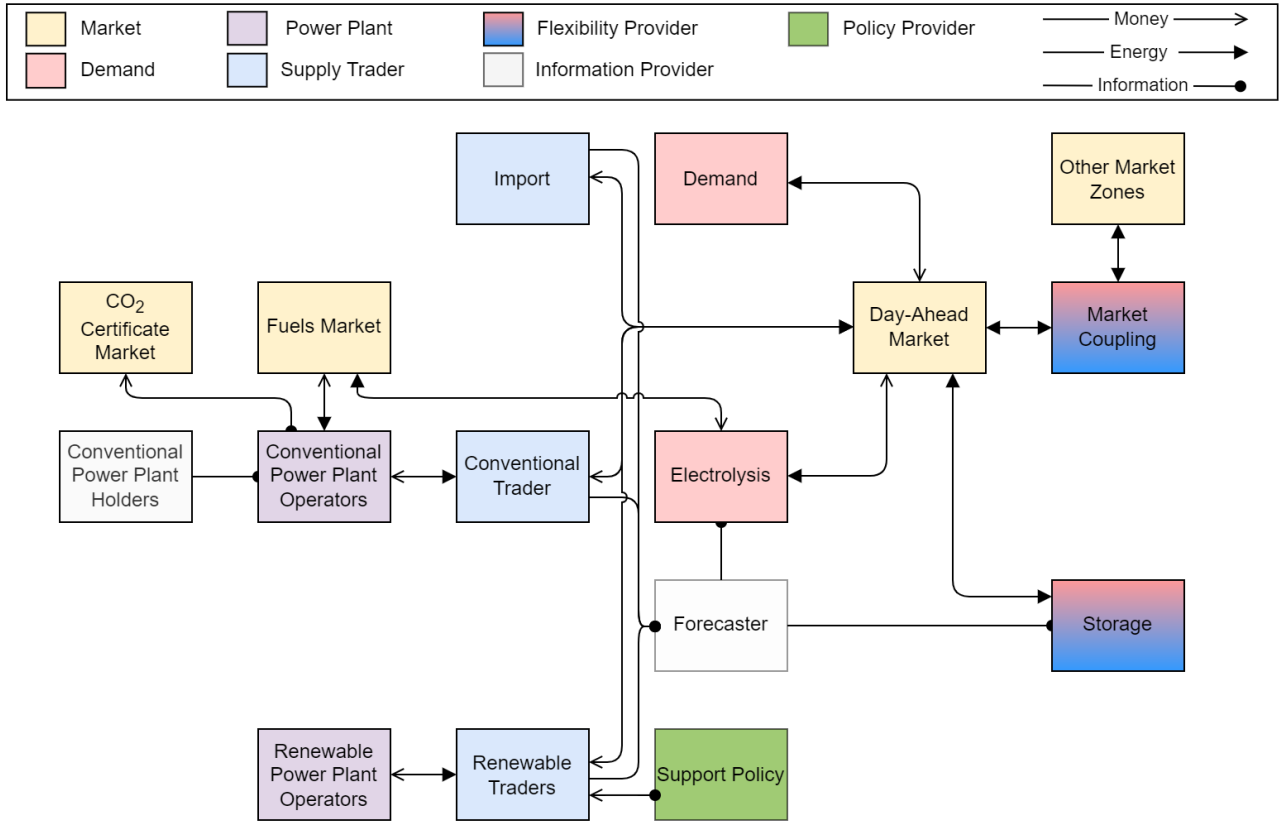
- Simulate trading and operation of power generation plants and flexibility options
- Model business-oriented behaviour under uncertainty
- Temporal resolution: ≤ hourly
- Spatial resolution: market zone(s)

Input

- Power plant park
- Efficiencies
- Availabilities
- Feed in potential
- Demand
- Fuel prices
- CO_2 prices

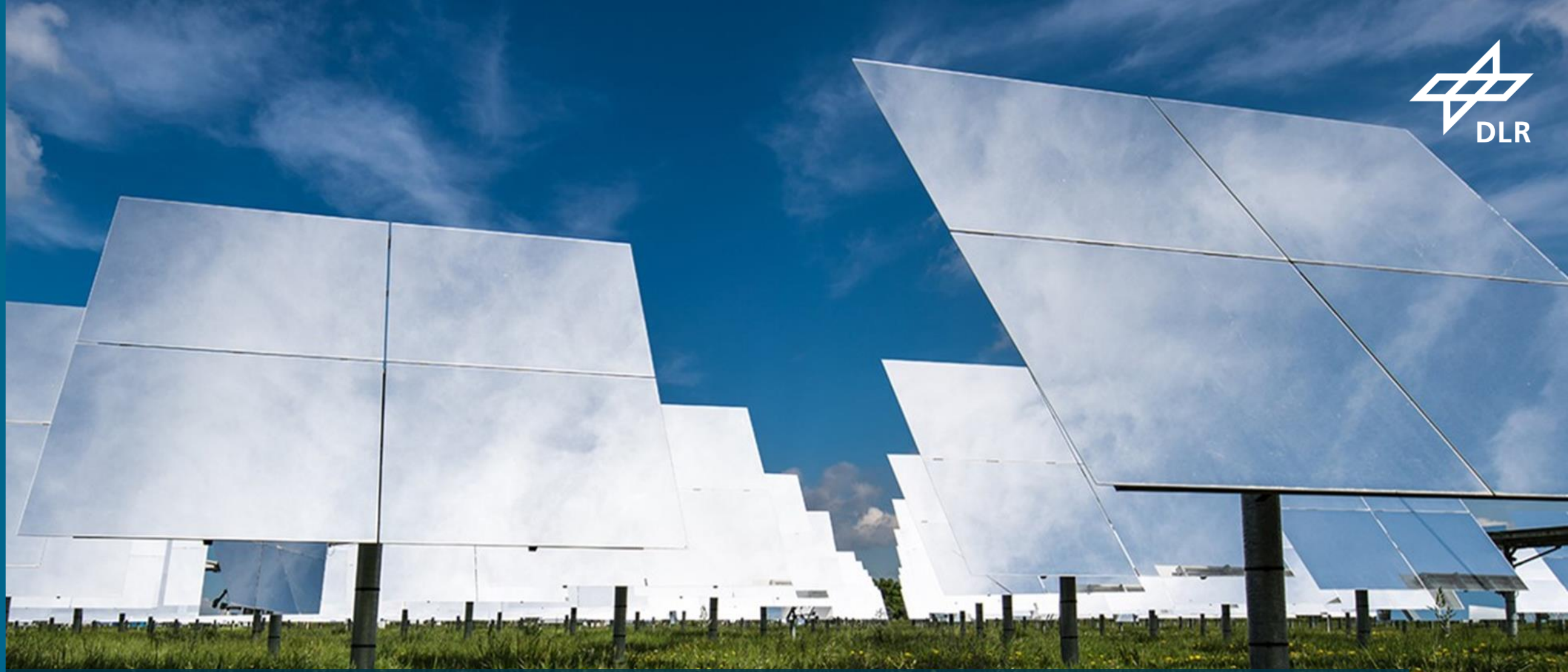
Output

- Electricity prices
- CO_2 emissions
- System costs
- Costs for support instruments
- Plant dispatch
- Market values



Federal Ministry for Economic Affairs and Climate Action
Federal Ministry of Education and Research
Bundesministerium für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 864276

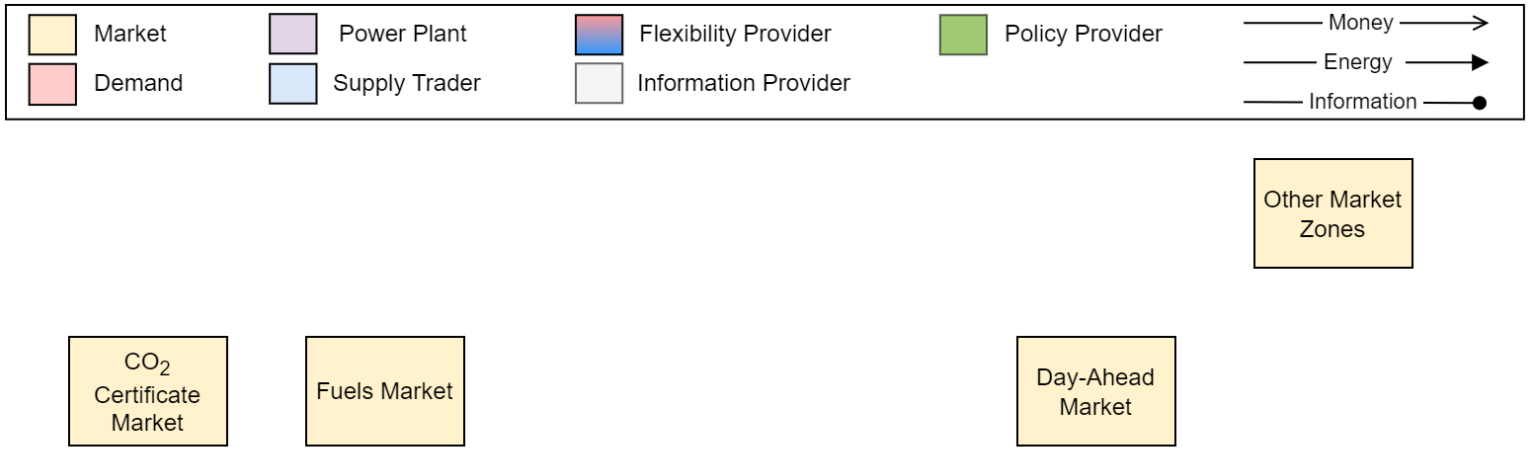


AMIRIS AGENTS 101

Christoph Schimeczek, Institute of Networked Energy Systems

AMIRIS
Agent Types

Markets
- Determine prices

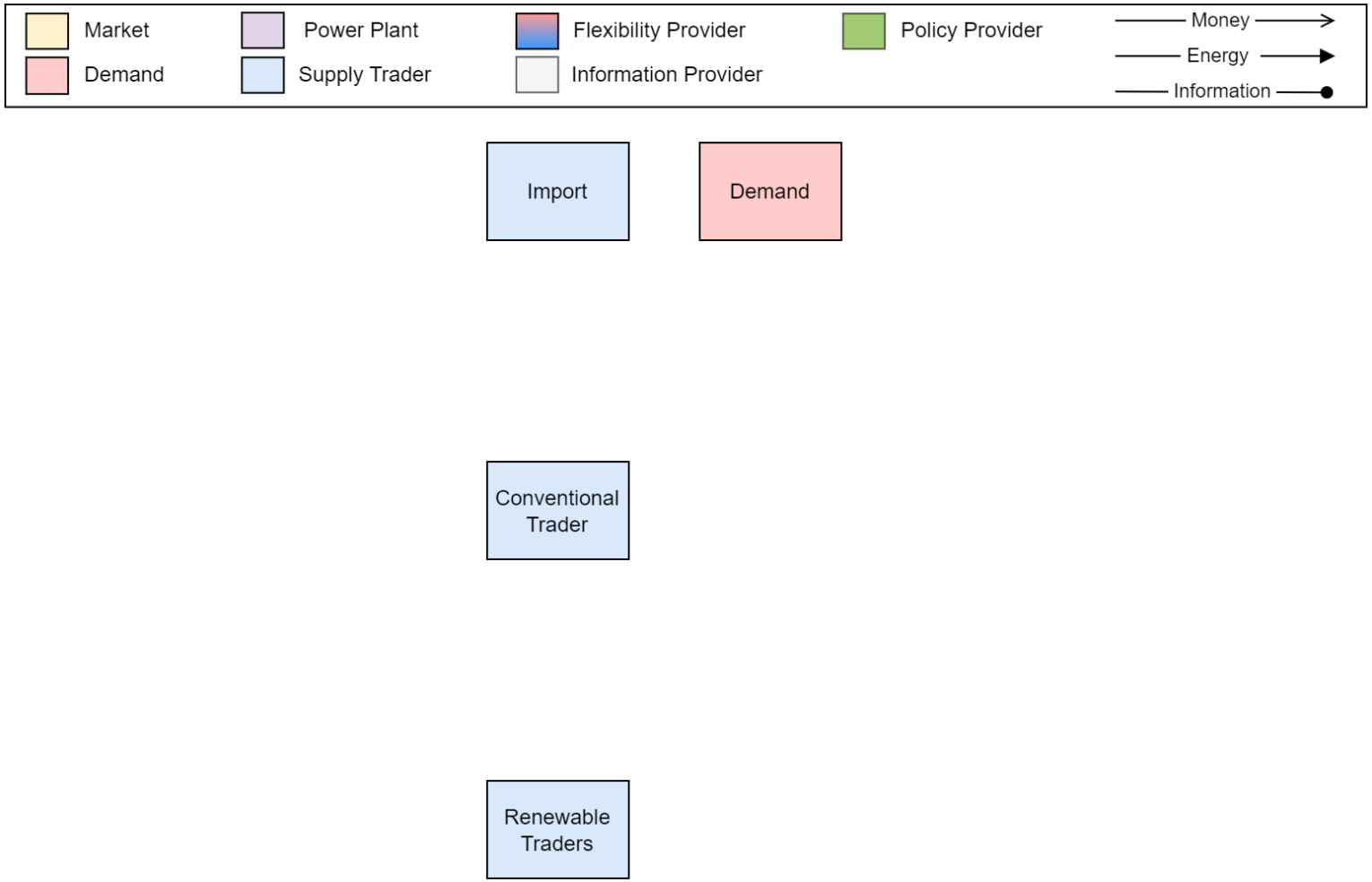


Christoph Schimeczek, Institute of Networked Energy Systems

AMIRIS
Agent Types

Markets
- Determine prices

Traders
- Fulfil marketing strategies

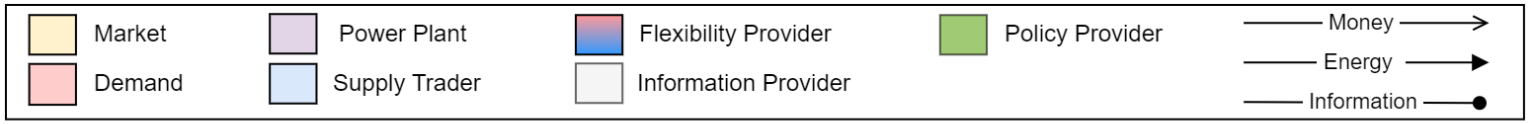


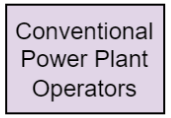
AMIRIS
Agent Types

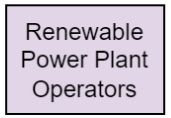
Markets
- Determine prices

Traders
- Fulfil marketing strategies

Plant operators
- Control power plants







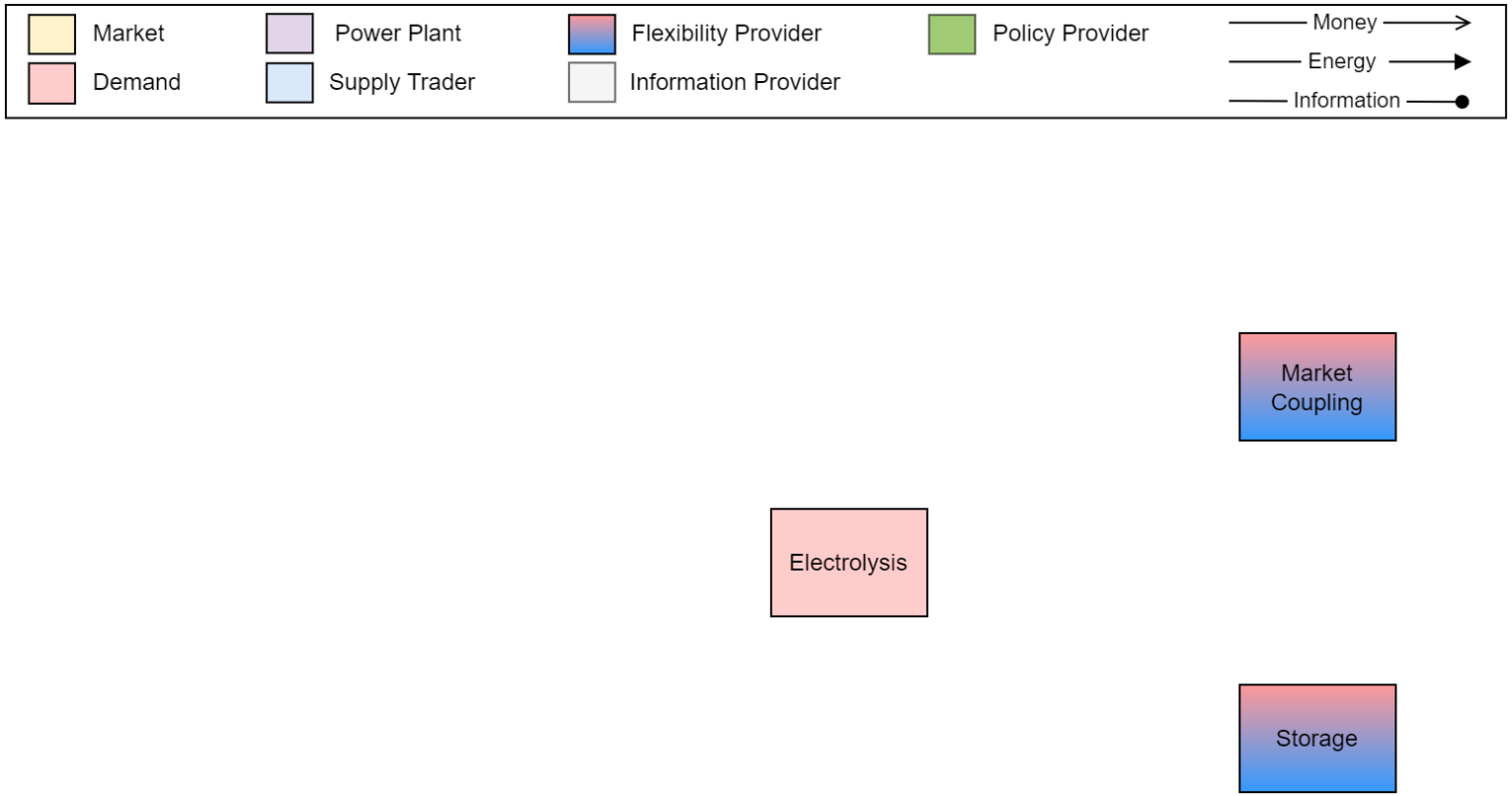
AMIRIS
Agent Types

Markets
- Determine prices

Traders
- Fulfil marketing strategies

Plant operators
- Control power plants

Flexibility providers
- Optimise dispatch



AMIRIS
Agent Types

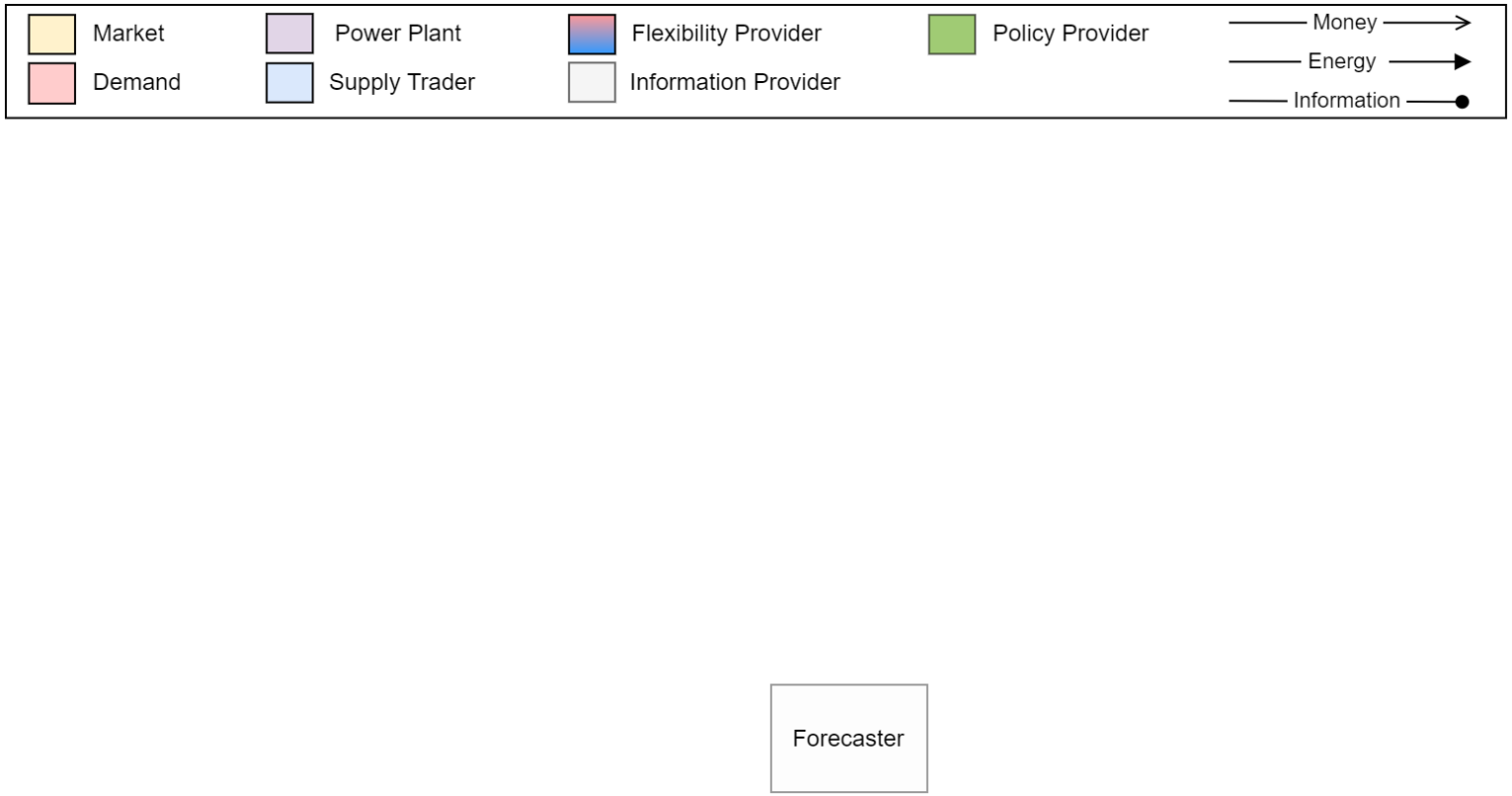
Markets
- Determine prices

Traders
- Fulfil marketing strategies

Plant operators
- Control power plants

Flexibility providers
- Optimise dispatch

Information provider
- Create forecasts



AMIRIS
Agent Types

Markets
- Determine prices

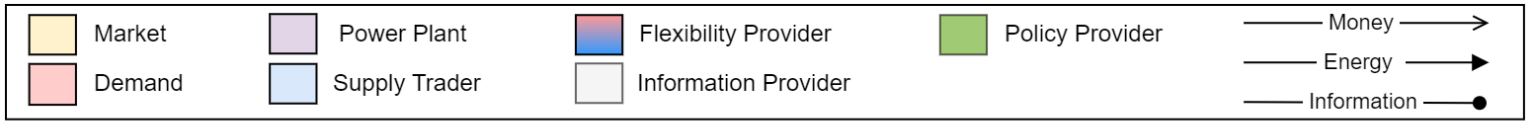
Traders
- Fulfil marketing strategies

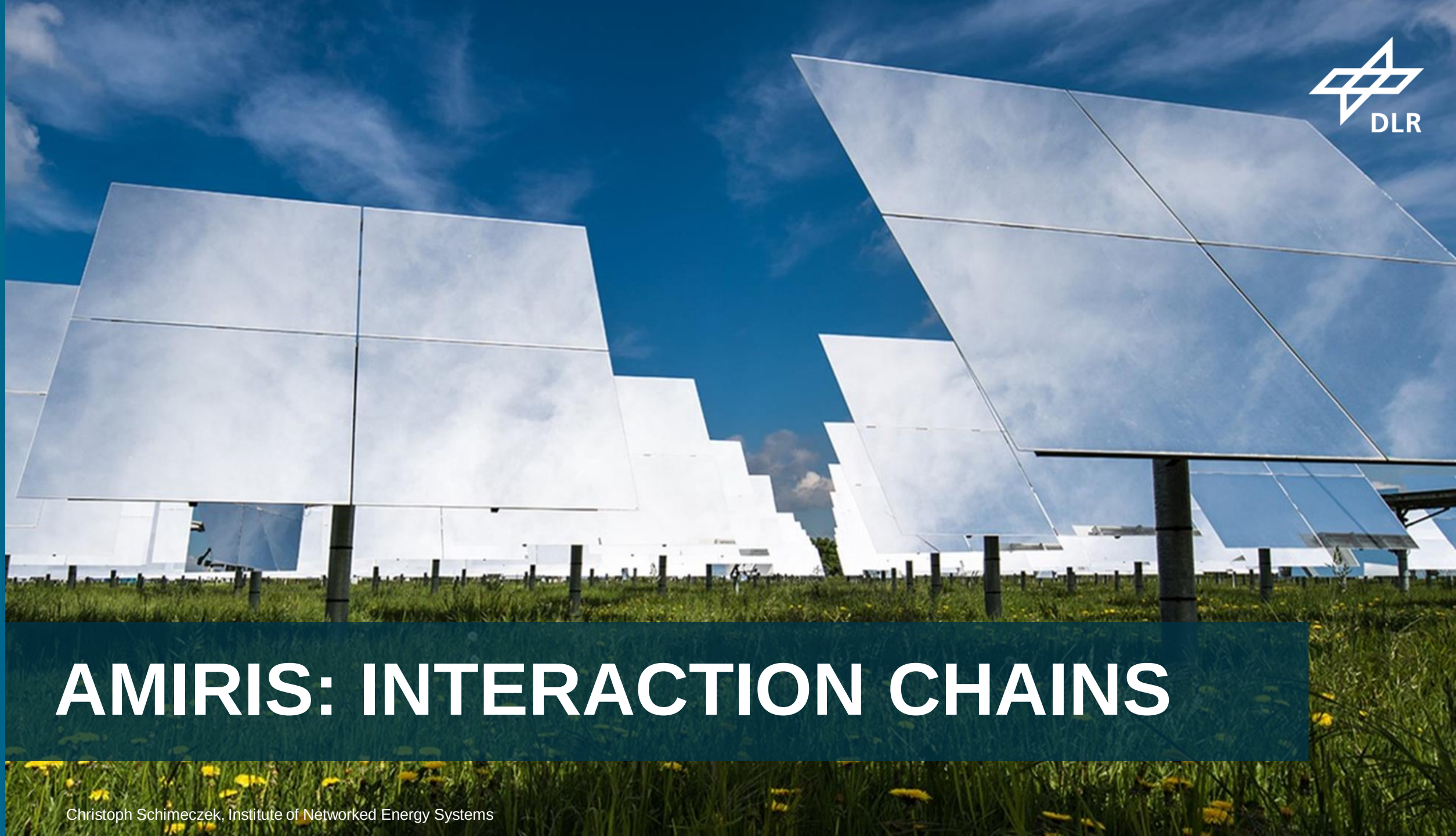
Plant operators
- Control power plants

Flexibility providers
- Optimise dispatch

Information provider
- Create forecasts

Policy
- Provide support





AMIRIS: INTERACTION CHAINS

Christoph Schimeczek, Institute of Networked Energy Systems

AMIRIS Interactions

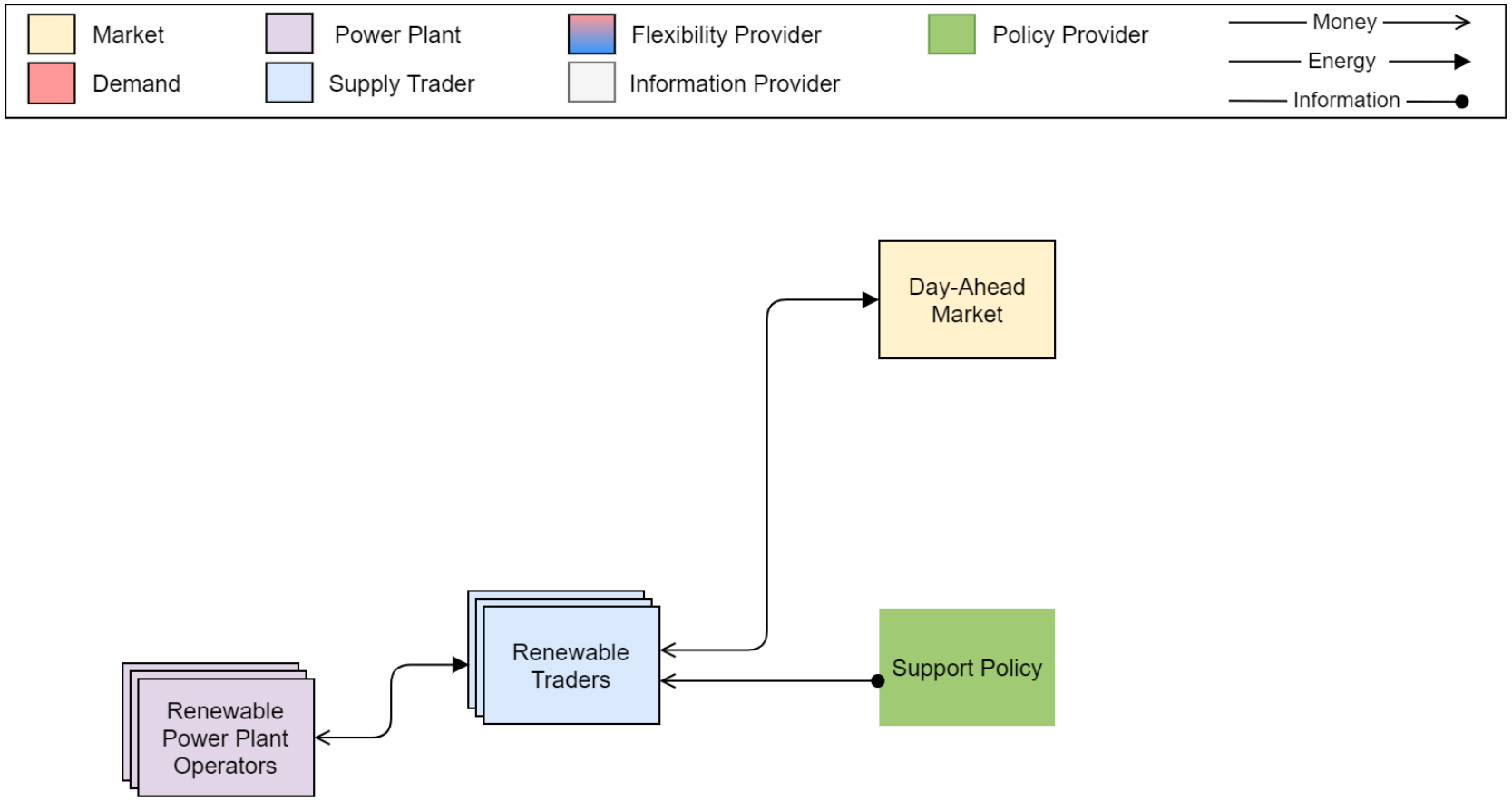
Renewables

Power Plant Operator
- Calculate marginal cost
- Dispatch power plants

Renewable Trader
- Create bid
- Request support

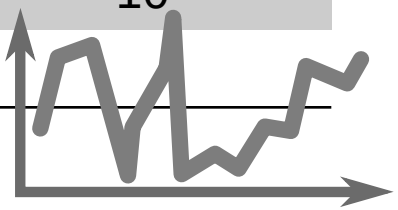
Support Policy
- Calculate support tariffs
- Provide support funding

Day-Ahead Market
- Clears Market



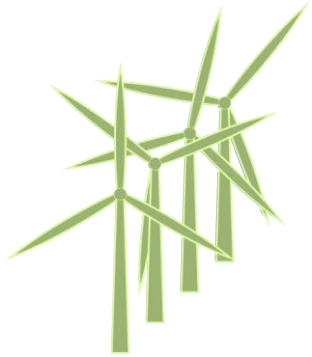
Renewables
Power Plant Operator

Input parameter	Value
EnergyCarrier	WindOn
InstalledPowerInMW	1000
OpexVarInEURperMWH	10
YieldProfile	



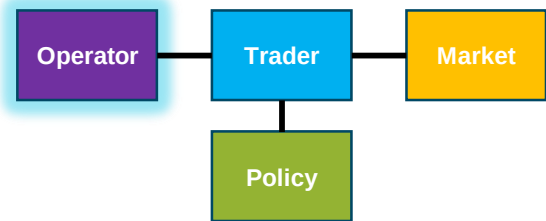
Actions

1) *Calculate power potential*
2) *Calculate marginal costs*
3) *Send marginals to Trader*
4) *Receive assignment*
5) *Dispatch plants*



MW	€/MWh
497	10

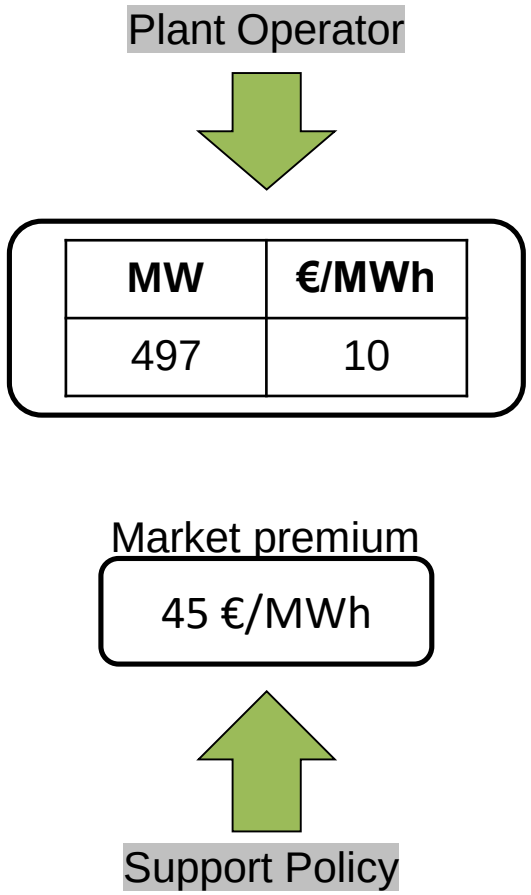
497 MW

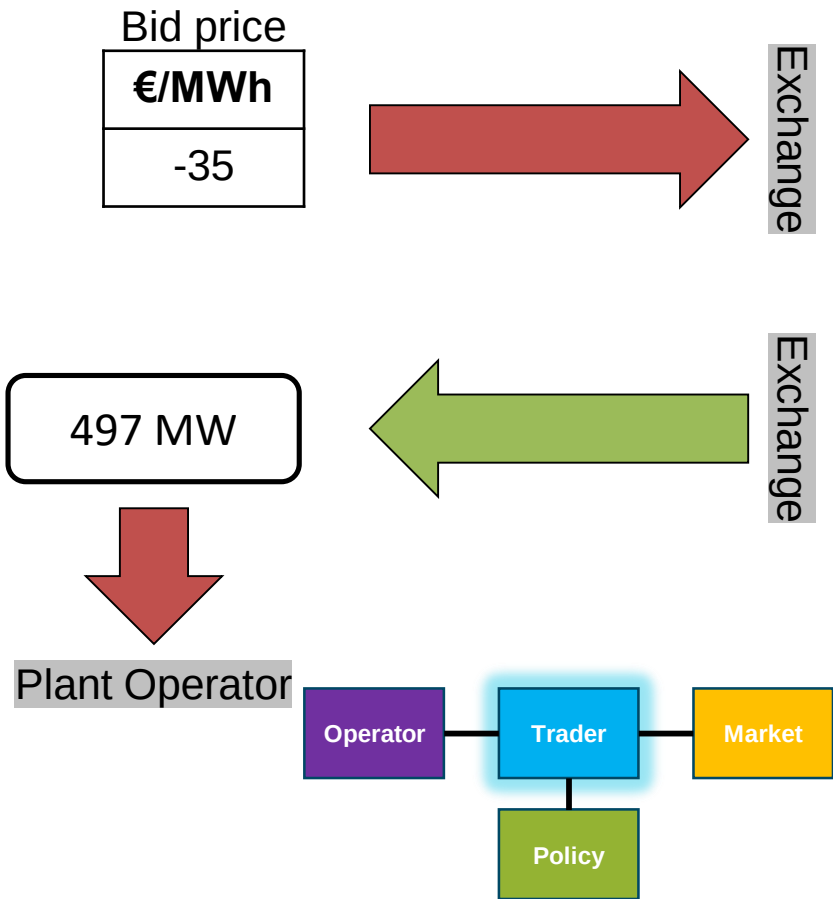


Renewables
Trader

Actions

1) *Receive marginal costs*
2) *Check support instrument*
3) *Derive bid*
4) *Send bids to Exchange*
5) *Receive awards*
6) *Forward power to operator*





Renewables
Support Policy

Actions

1) *Register clients*
2) *Track power prices*
3) *Track feed-in potentials*
4) *Calculate variable tariffs*
5) *Provide support*

Trader

Carrier	Instrument
PV	Feed-in tariff
WindOn	Market premium

Exchange

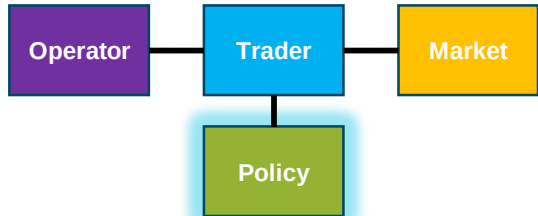
12 €/MWh

Plant Operator

Carrier	Feed-in
PV	0 MW
WindOn	497 MW

Carrier	Instrument	Support
PV	Feed-in tariff	90 €/MWh
WindOn	Market premium	45 €/MWh

Trader



Demand

Trader

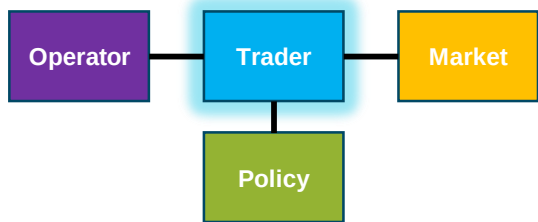
Input parameter	Value
ValueOfLostLoad	3000
DemandSeries	

Actions

1) Create bid

2) Send bid(s) to Exchange

MW	€/MWh
1017	3000



Day-Ahead Market

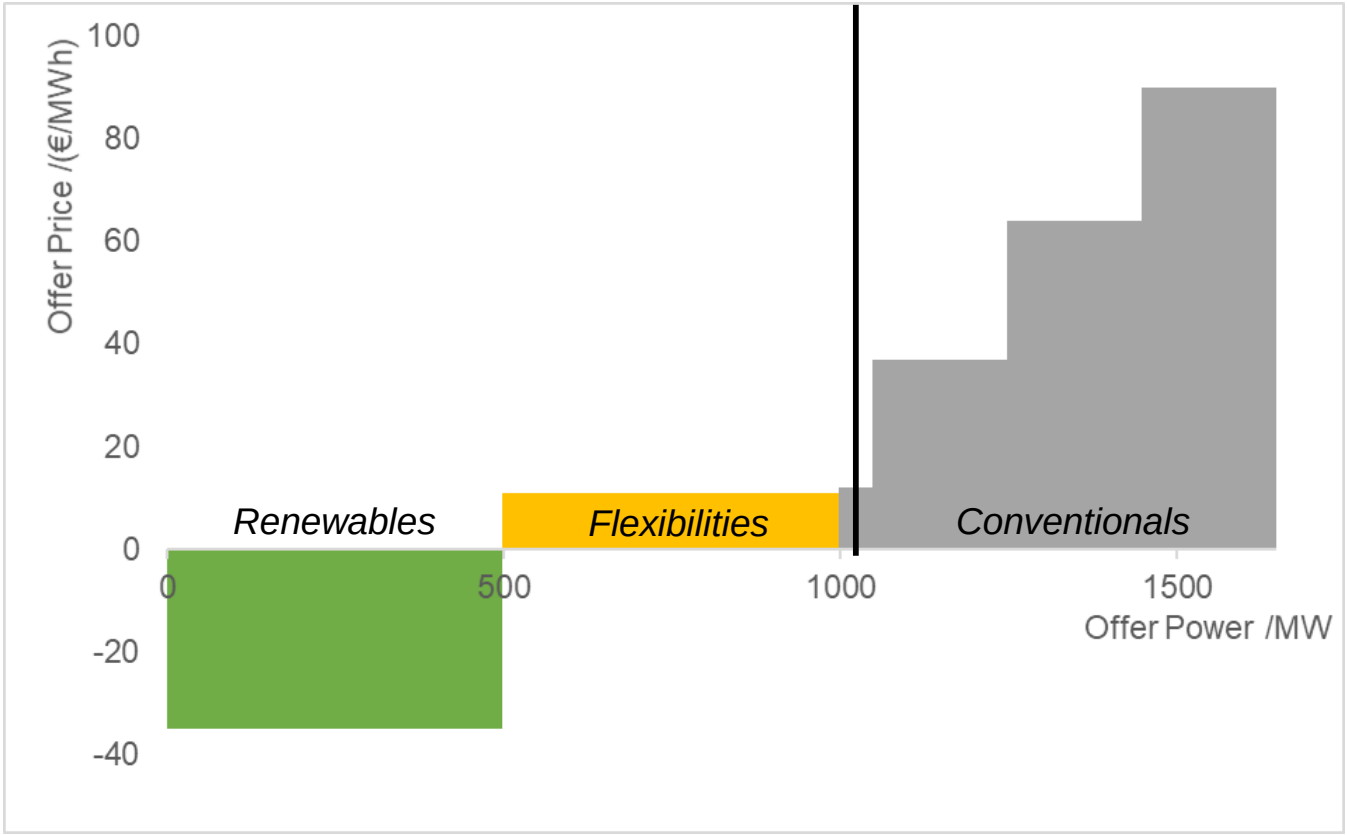
Market Clearing

Actions

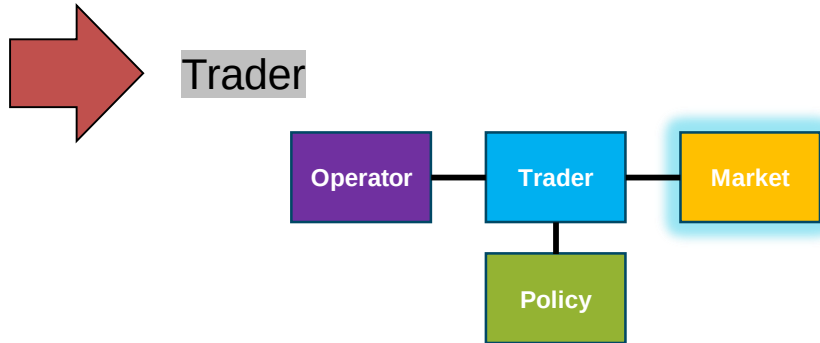
1) Receive bids

2) Clear market

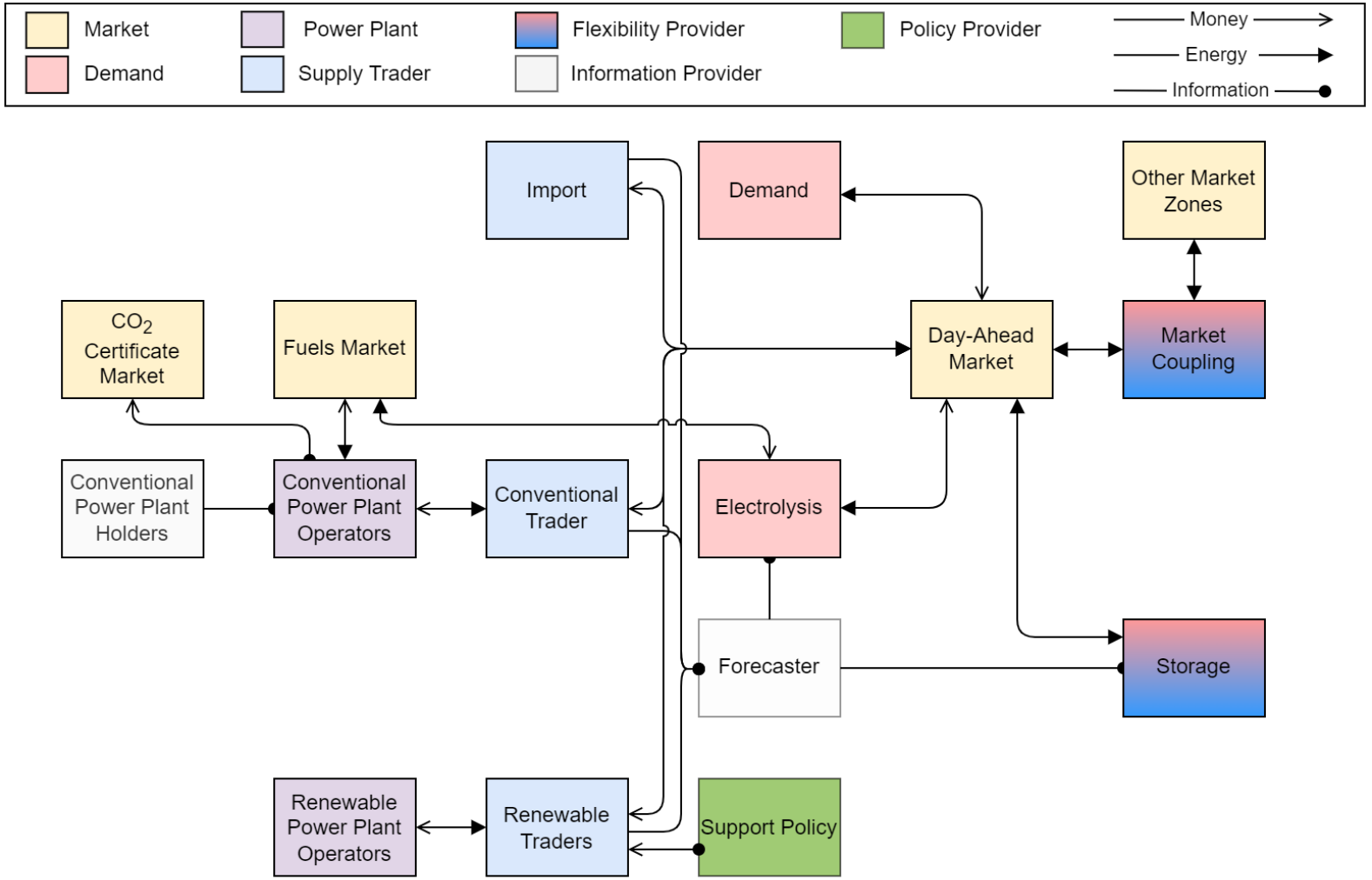
3) Send awards



MW	€/MWh
497	12
500	12
20	12

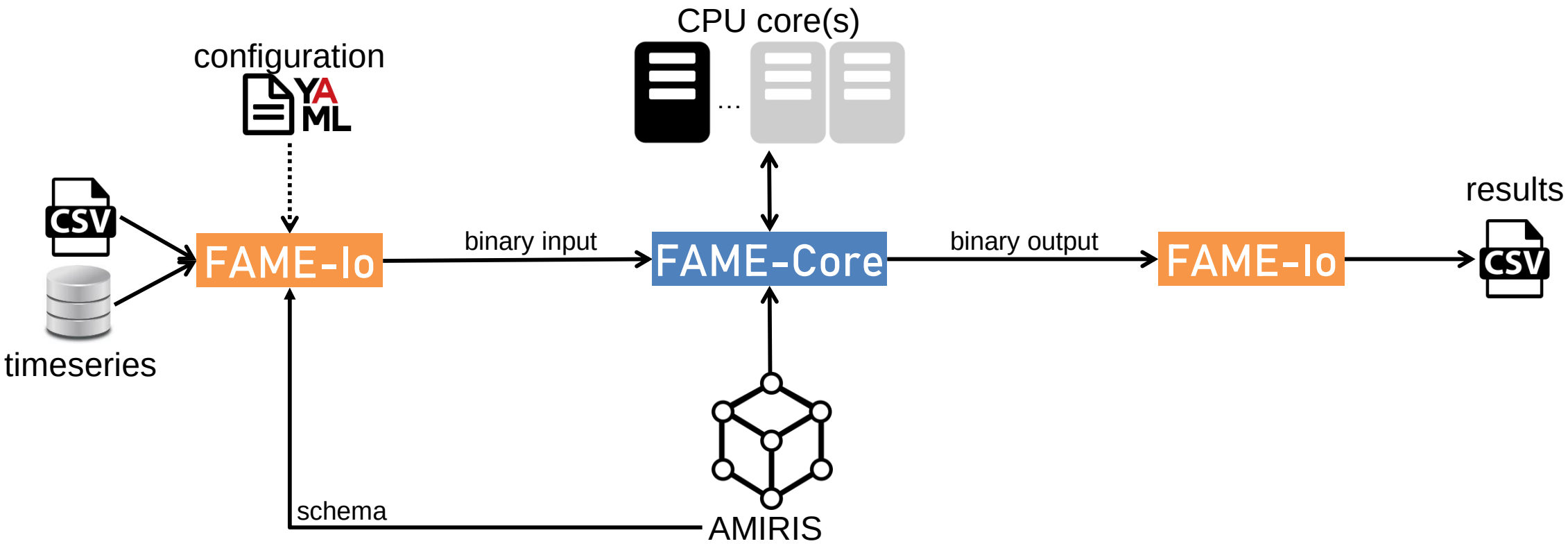


AMIRIS Agents
Overview



AMIRIS is based on FAME

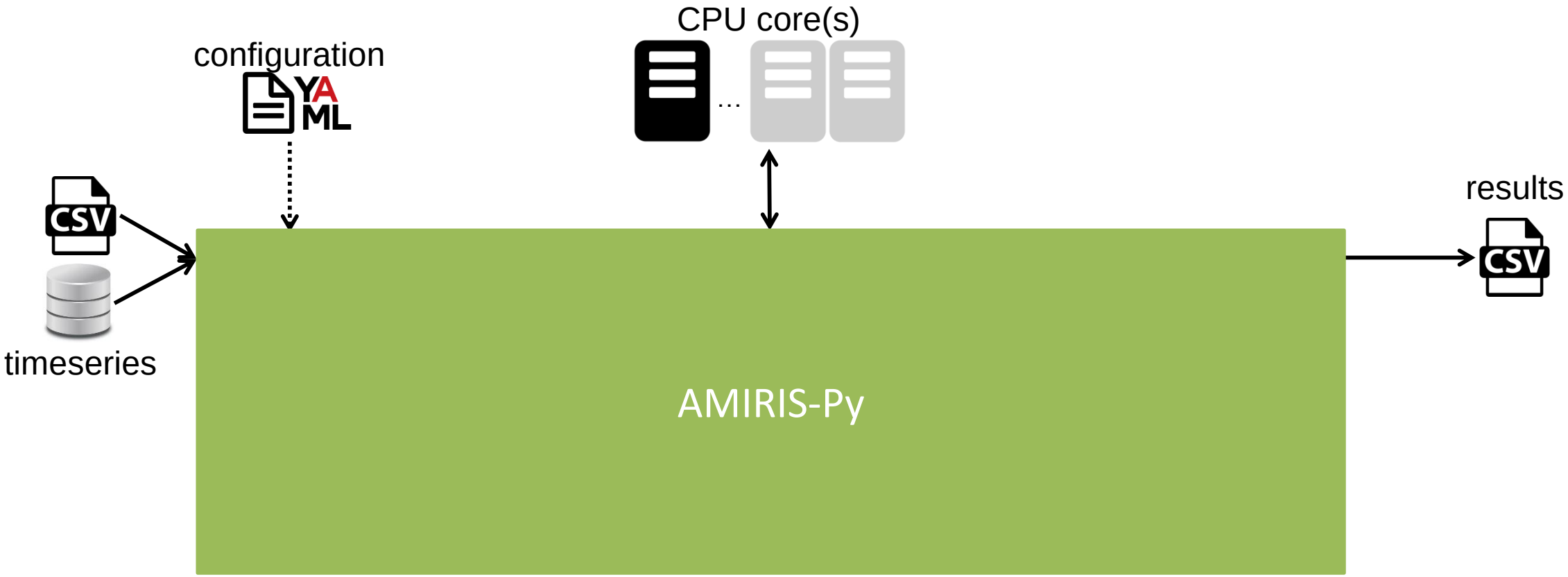
*open **F**ramework for distributed **A**gent-based **M**odelling of **E**nergy systems*



FAME-Core: https://joss.theoj.org/papers/10.21105/joss.05087
FAME-Io: https://joss.theoj.org/papers/10.21105/joss.04958

AMIRIS is based on FAME

*open **F**ramework for distributed **A**gent-based **M**odelling of **E**nergy systems*



FAME-Core: https://joss.theoj.org/papers/10.21105/joss.05087
FAME-Io: https://joss.theoj.org/papers/10.21105/joss.04958

AMIRIS: INSTALL & RUN

Christoph Schimeczek, Institute of Networked Energy Systems

Setup
Requirements

- Java JDK 11

```
(base) PS C:\> java --version
openjdk 11.0.9.1 2020-11-04
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.9.1+1)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.9.1+1, mixed mode)
```

  - Obtain from, e.g., https://adoptium.net/

- Python 3.9

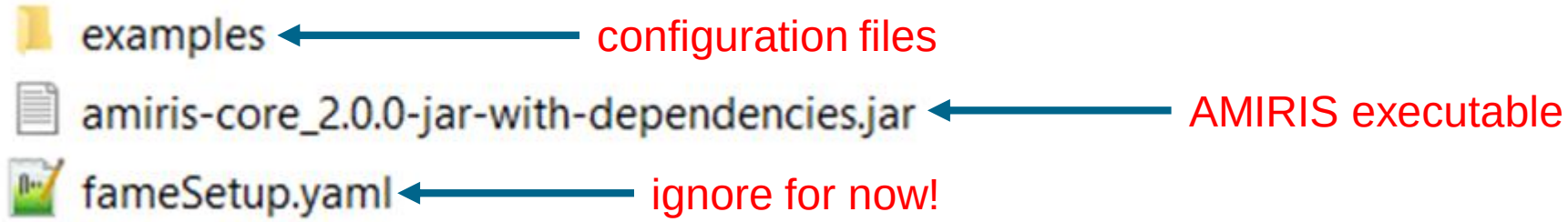
```
(base) PS C:\> python --version
Python 3.9.7
```

  - Obtain from, e.g., https://github.com/conda-forge/miniforge#mambaforge

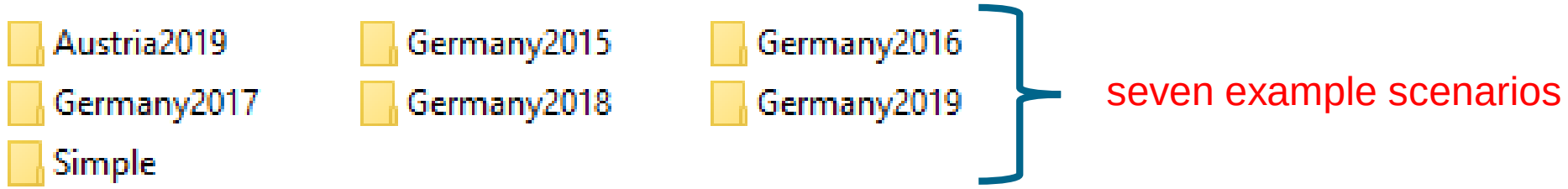
Setup
Python environment

- Create environment
```
(base) PS C:\> conda create -n AmirisEnv python=3.9
```
- Activate environment
```
(base) PS C:\> conda activate AmirisEnv
```
- Install *amirispy*
```
(AmirisEnv) PS C:\> pip install amirispy
```
- Create folder:
```
(AmirisEnv) PS C:\> mkdir amiris; cd amiris
```
- Install *AMIRIS*:
```
(AmirisEnv) PS C:\amiris> amiris install
```

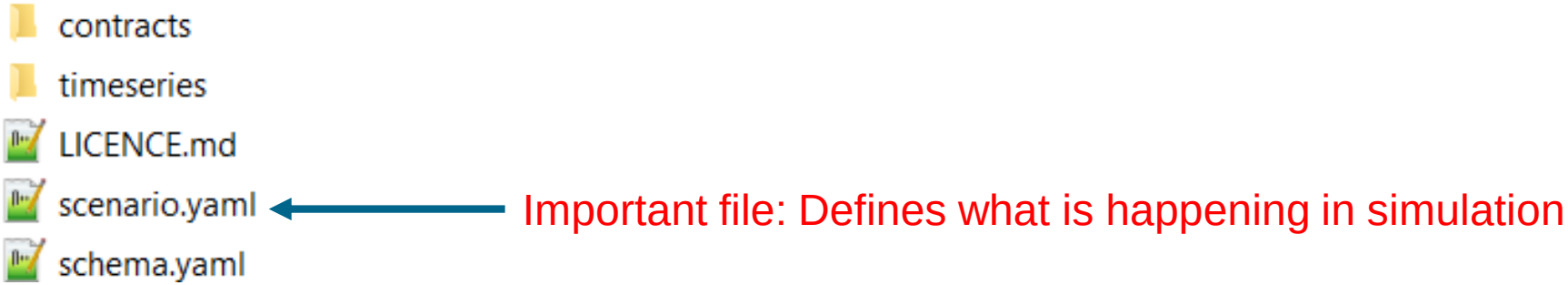
Setup
Files



examples/



Examples/Simple/



Setup
Run AMIRIS

```
(AmirisEnv) PS C:\amiris> amiris run
usage: amiris run [-h] --jar JAR --scenario SCENARIO
                  [--output OUTPUT]
amiris run: error: the following arguments are required: --jar/-j, --scenario/-s
```

Required arguments

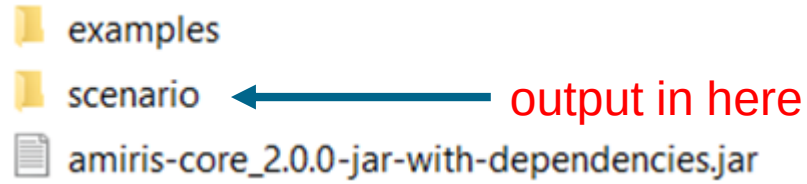
- -j AMIRIS executable
- -s Scenario file

```
(AmirisEnv) PS C:\amiris> amiris run -j .\amiris-core_2.0.0-jar-with-dependencies.jar
 -s .\examples\Simple\scenario.yaml
```

Use tab-completion for jar-file and scenario

Console output

```
14:18:38 – PRINT – Start running AMIRIS
Starting up 1 processes.
Warm-up completed after 1 ticks.
04.10.2023 14:18:39:: Simulation completed! Ran 219 ticks in 258 ms
14:18:40 – PRINT – Successfully executed AMIRIS. See your results in '.'
```



Setup
Redirect output

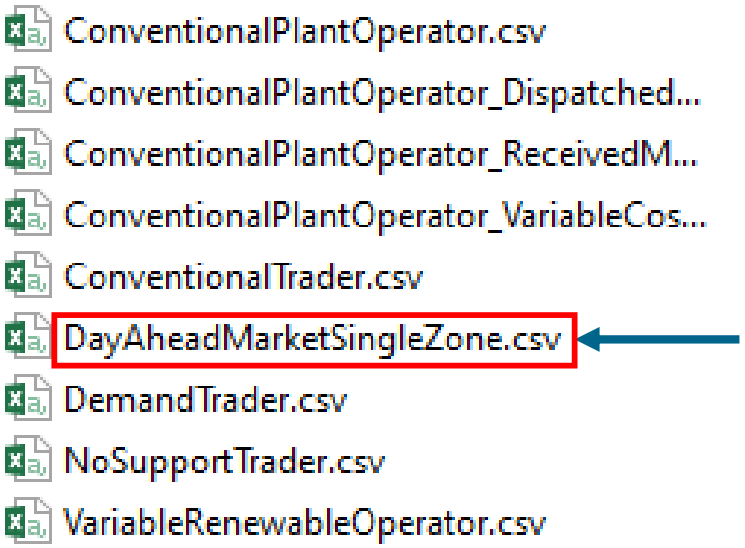
```
(AmirisEnv) PS C:\amiris> amiris run -h
usage: amiris run [-h] --jar JAR --scenario SCENARIO [--output OUTPUT]

optional arguments:
  -h, --help            show this help message and exit
  --jar JAR, -j JAR     Path to 'amiris-core_<version>-jar-with-dependencies.jar'
  --scenario SCENARIO, -s SCENARIO
                        Path to a scenario yaml-file
  --output OUTPUT, -o OUTPUT
                        Directory to write output to
```

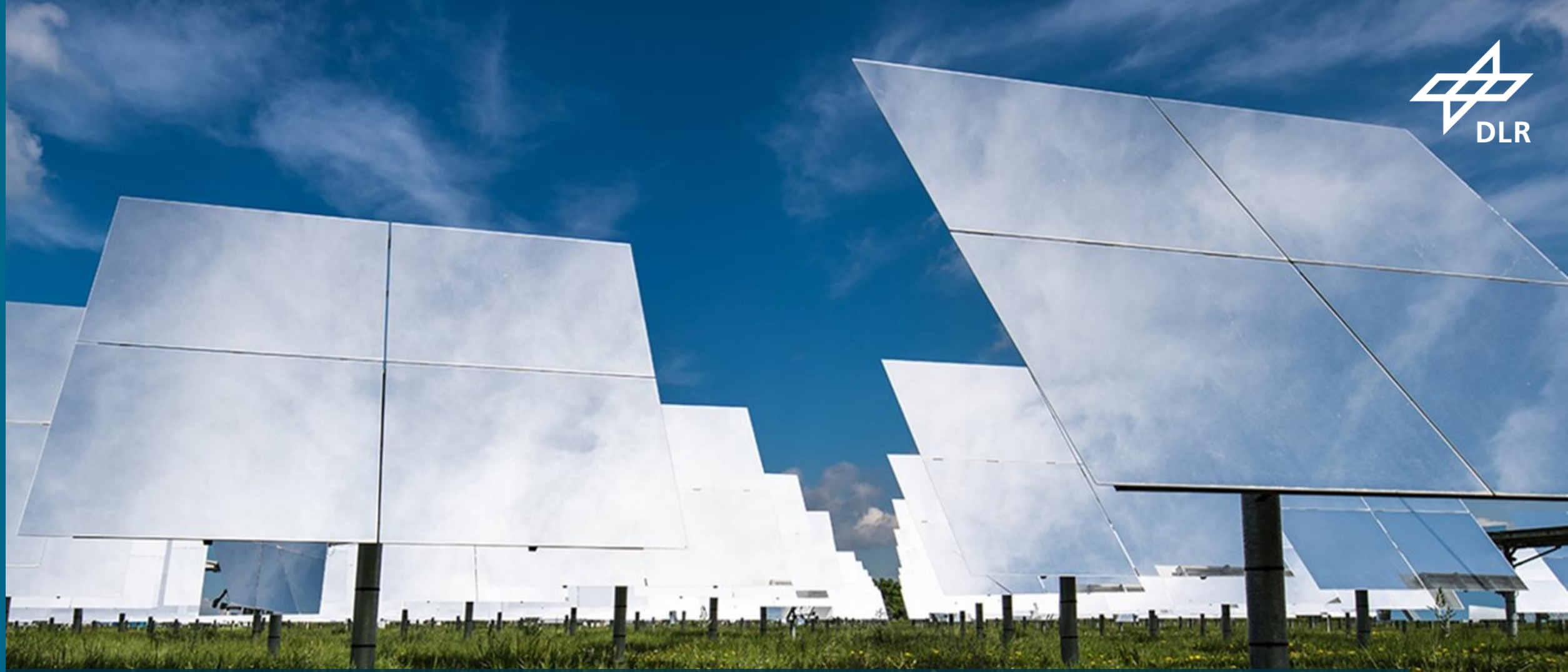
use this

```
(AmirisEnv) PS C:\amiris> amiris run -j .\amiris-core_2.0.0-jar-with-dependencies.jar
 -s .\examples\Simple\scenario.yaml -o simple
```

Setup
Results



AgentId	TimeStep	AwardedEnergyInMWH	ElectricityPriceInEURperMWH
1	01.01.2021 00:00	12431	267.4721054
1	01.01.2021 01:00	11416	262.9066734
1	01.01.2021 02:00	11163	260.8119727
1	01.01.2021 03:00	11036	257.4786831
1	01.01.2021 04:00	11192	256.4702082
1	01.01.2021 05:00	12177	256.2193284
1	01.01.2021 06:00	12685	256.2193284
1	01.01.2021 07:00	15222	259.7771467
1	01.01.2021 08:00	16491	260.2935264
1	01.01.2021 09:00	17125	257.9859146
1	01.01.2021 10:00	17378	255.7190453
1	01.01.2021 11:00	16997	255.4696391
1	01.01.2021 12:00	16237	257.2258181
1	01.01.2021 13:00	15476	256.4702082



PARAMETERISATION

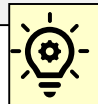
Parameterisation

Scenario: Main config file to bundle all simulation properties

Open: examples/Germany2019/scenario.yaml

scenario.yaml

schema	*definition of valid agent and contract structures*
general properties	*simulation start/end time, random seed*
variables	*YAML anchors, optional*
agents	*which agents have what parameters*
contracts	*how and when agents interact*



*You can split the scenario.yaml into separate files, e.g. schema, contracts, etc., and join them using **!include**, see https://gitlab.com/fame-framework/fame-io#split-and-join-multiple-yaml-files*

Parameterisation
General Properties

- Define
  - start and end of simulation
  - which random seed to use

```
GeneralProperties:
  RunId: 1
  Simulation:
    StartTime: 2018-12-31_23:58:00
    StopTime: 2019-12-31_23:58:00
    RandomSeed: 1
  Output:
```

ignore

ignore

*FAME's time definition **always** uses 365 days / 8760 hours per year, see also https://gitlab.com/fame-framework/wiki/-/wikis/architecture/decisions/TimeStamp*

YAML is indentation-based (2 spaces)

Parameterisation

Agents

- Define
  - agents
  - their type, ID, and attributes.
- Supported data types:
  - integer, floating point, enums, timeseries
- Supported structures
  - Any combination of blocks and lists
- Structure of attributes
  - depends on type of agent
  - is defined in schema

In YAML, dash is used to denote lists

```yaml
Agents:
  - Type: EnergyExchange
    Id: 1
    Attributes:
      DistributionMethod: SAME_SHARES
      GateClosureInfoOffsetInSeconds: 11

  - Type: CarbonMarket
    Id: 3
    Attributes:
      OperationMode: FIXED
      Co2Prices: "./timeseries/co2_price.csv"

  - Type: FuelsMarket
    Id: 4
    Attributes:
      FuelPrices:
        - FuelType: LIGHNITE
          Price: 5.00
          ConversionFactor: 1.0
        - FuelType: NATURAL_GAS
          Price: "./timeseries/natural_gas_cost.csv"
          ConversionFactor: 1.0
```

Time series attributes also support a single value.

*Every agent **must** have a unique ID within the simulation. This is how agents address each other.*

GERMANY 2019: BASE SCENARIO

Parameterisation

Timeseries

Open: examples/Germany2019/timeseries/co2_price.csv

- Use case: time-dependent input

- File Format
  - Timestamp – semicolon – (dot-separated) floating point value

- Timestamps Format
  - YYYY-MM-DD_hh:mm:ss
  - Refer to *simulation time* **!not UTC**!
  - Idea: Easily reapply timeseries from one year to another

- Missing datapoints: AMIRIS will interpolate

timestamp value

```
2019-01-07_00:00:00;23.01
2019-01-08_00:00:00;22.4
2019-01-10_00:00:00;21.4
2019-01-14_00:00:00;21.95
2019-01-15_00:00:00;22.55
2019-01-16_00:00:00;22.72
2019-01-17_00:00:00;23.55
2019-01-21_00:00:00;24.22
2019-01-22_00:00:00;24.42
2019-01-24_00:00:00;24.6
2019-01-28 00:00:00;23.01
```

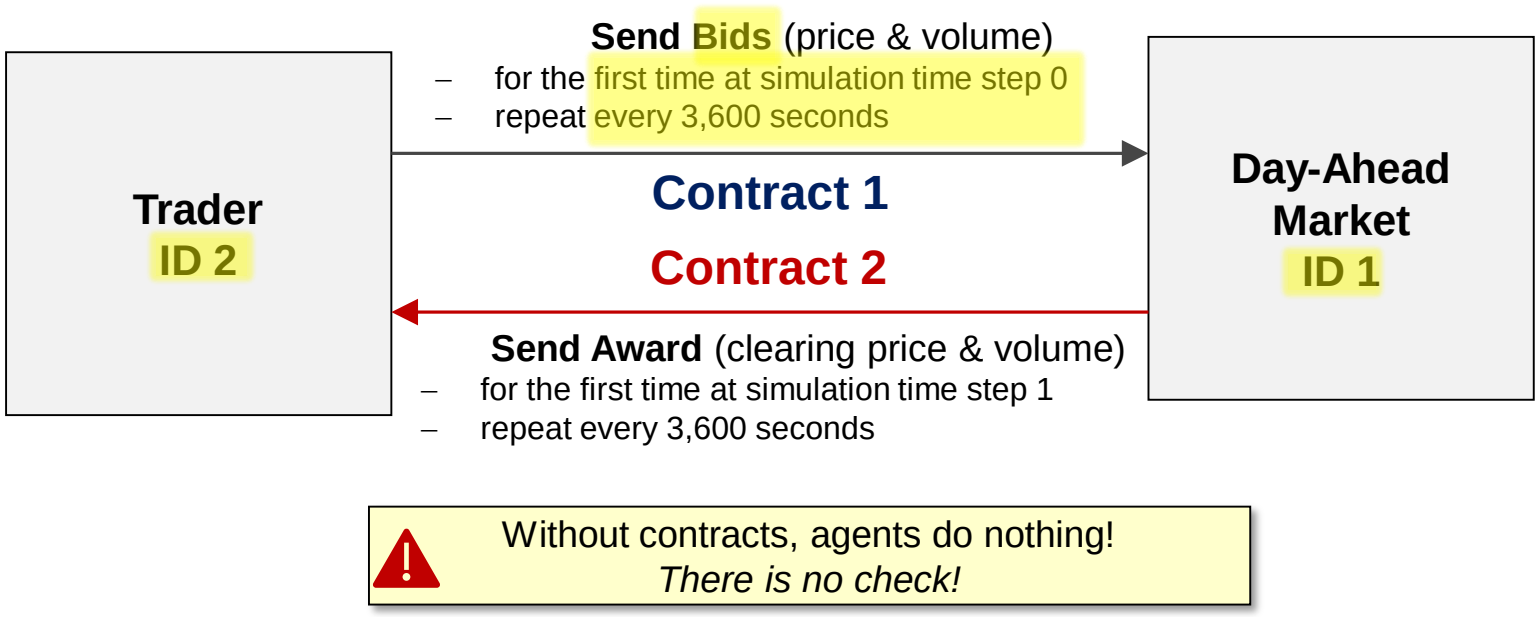
Additional columns (even empty) are not allowed and need to be removed.

Lines with empty time or value are not allowed and need to be removed.

GERMANY 2019: HIGH GAS PRICE

Parameterisation
Contracts

Define **when** agents send **what** data to **which** other agents



Without contracts, agents do nothing!
There is no check!

Parameterisation

Contracts: Advanced

Open: examples/Germany2019/contracts/conventionals.yaml

- Simulations often require *many* contracts!
- Contracts are often *similar*!
- Short notations available:
  - 1:N → one sender to multiple receivers
  - N:1 → one receiver from multiple senders
  - M:M → m senders, each to **one** of m receivers

```yaml
- SenderId: 1
  ReceiverId: [2,5]
  ProductName: Award
  FirstDeliveryTime: 5
  DeliveryIntervalInSteps: 3600

- SenderId: [2,5]
  ReceiverId: 1
  ProductName: Bid
  FirstDeliveryTime: 5
  DeliveryIntervalInSteps: 3600

- SenderId: [1,2,3]
  ReceiverId: [11,12,13]
  ProductName: Award
  FirstDeliveryTime: 5
  DeliveryIntervalInSteps: 3600
```

Parameterisation

Contracts: Advanced

Open: examples/Germany2019/contracts/conventionals.yaml

- Simulations often require *many* contracts!
- Contracts are often *similar*!
- Short notations available:
  - 1:N → one sender to multiple receivers
  - N:1 → one receiver from multiple senders
  - M:M → m senders, each to **one** of m receivers

- Sender / receiver lists *repeat* often!
- Use YAML anchors to replace similar lists
  - Define: &anchorName <something>
  - Reference: *anchorName

```yaml
AgentGroups:
  - &builders  [2000, 2001, 2002, 2003, 2004, 2005]
  - &traders   [1000, 1001, 1002, 1003, 1004, 1005]
  - &operators [500, 501, 502, 503, 504, 505]
  - &exchange 1
  - &carbonMarket 3
  - &fuelsMarket 4
  - &forecaster 6

Contracts:
################################
# -- PlantBuildingManagement -- #
################################
  - SenderId: *builders
    ReceiverId: *operators
    ProductName: PowerPlantPortfolio
    FirstDeliveryTime: -60
    DeliveryIntervalInSteps: 31536000

##############################
# -- Forecast Preparation -- #
##############################
  - SenderId: *forecaster
    ReceiverId: *traders
    ProductName: ForecastRequest
    FirstDeliveryTime: -26
    DeliveryIntervalInSteps: 3600
```

anchors

comment

reference

GERMANY 2019: EXTRA PV

Christoph Schimeczek, Institute of Networked Energy Systems

Schema

All parametrisation options

Open: examples/Germany2019/schema.yaml

- Defines
  - types of agents
  - their contract products
  - their attributes
  - attribute types
  - if attributes are mandatory / lists
  - attribute value restrictions

```
AgentTypes:
  CarbonMarket:
    Attributes:
      Co2Prices:
        AttributeType: time_series
        Mandatory: false
        List: false
      OperationMode:
        AttributeType: enum
        Mandatory: true
        List: false
        Values: [ 'FIXED', 'DYNAMIC' ]
Products: [ 'Co2PriceForecast', 'Co2Price', 'CertificateBill' ]
```

scenario.yaml

```
Type: CarbonMarket
Id: 3
Attributes:
  OperationMode: FIXED
  Co2Prices: "./timeseries/co2_price.csv"
```

Use schema to look up agents & attributes,
see also https://gitlab.com/dlr-ve/esy/amiris/amiris/-/wikis/Classes/Classes

Do not tamper with the schema!
Derived from Java code; Validates scenarios

Christoph Schimeczek, Institute of Networked Energy Systems

FINAL REMARKS

AMIRIS

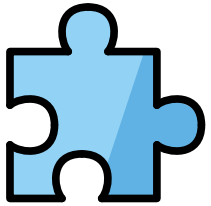
Following FAIR4RS Principles

Findable

- DOI
- Wikipedia
- COMSES
- HECI
- OEP
- openmod
- Website

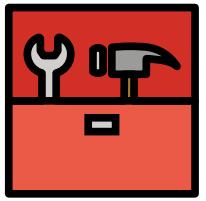
Accessible

- GitLab
- PyPI
- Zenodo



Interoperable

- API
- Workflow tools
- CSV
- YAML



(Re-)usable

- Apache 2.0
- REUSE
- Wiki
- Javadoc
- Win/Mac/Linux
- Scalable (H)PC

will upload presentation here

Key Indicators

Users

- 16 confirmed external user
- 6 bugs reported

PhD candidates

- 5 internal
- 5 external

Visibility

- 19k views on Wikipedia
- 10k views on openmod

Software

- 21 releases
- 11k downloads

Christoph Schimeczek, Institute of Networked Energy Systems

Barker et al. (2022). 10.1038/s41597-022-01710-x

AMIRIS

Release v2 – see also Poster

New Mechanics

Coupling of market zones,
Easy coupling with other models

New Agents

Electrolysis, Import

New Strategies

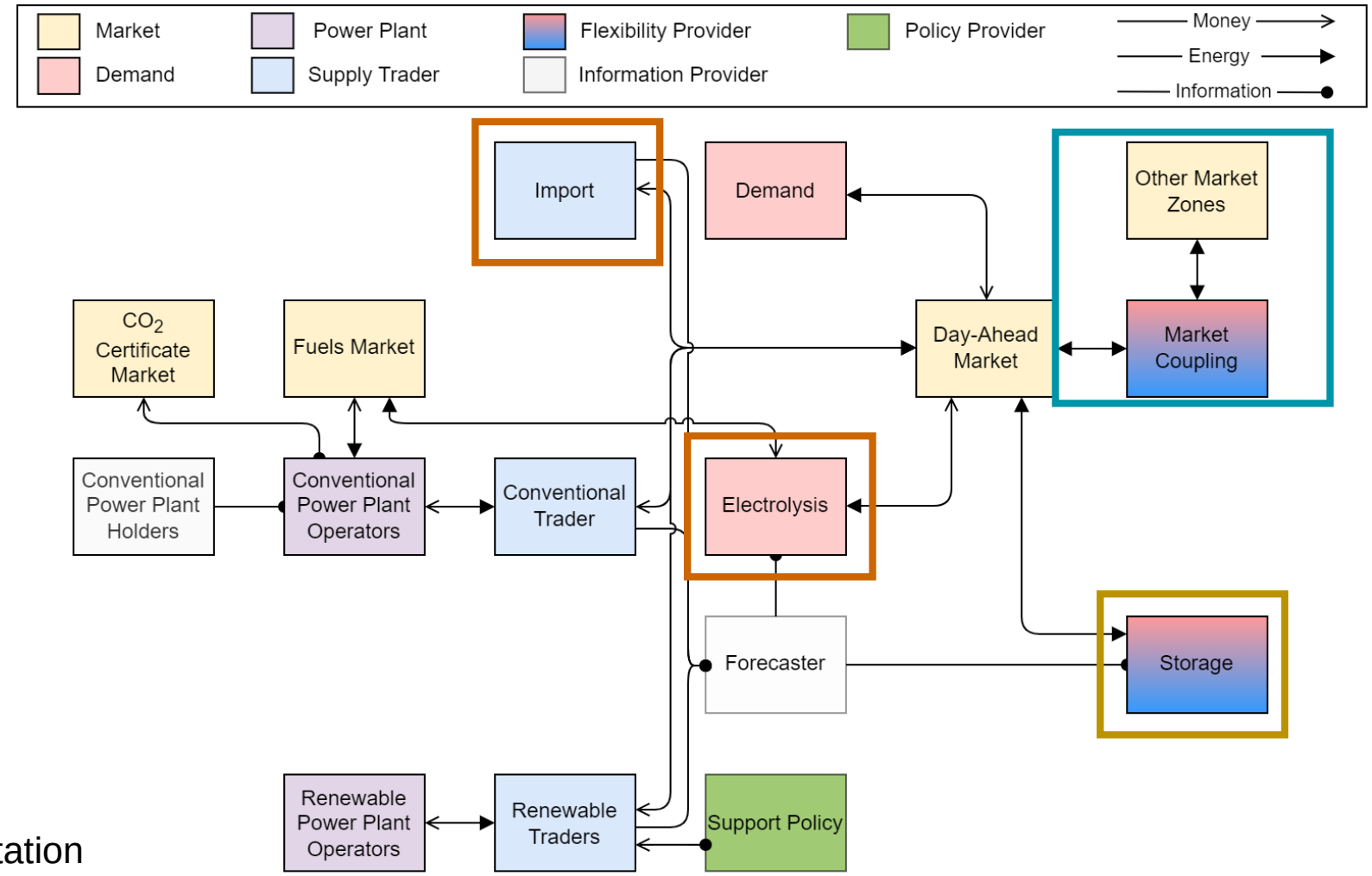
Storage dispatch (profit maximising, file)

New Data

Germany 2015-2018

Plus

Bug fixes, harmonisations, better documentation



Use AMIRIS

Use AMIRIS
- Report difficulties
- Ask questions in forum
- Create / publish scenarios
- Cite AMIRIS at JOSS

Make us enhance AMIRIS
- Report issues / bugs
- Post ideas in forum
- Make feature requests

Enhance AMIRIS yourself
- Improve / modify agents
- Sign Contributor License Agreement
- Make pull requests

Get in contact: amiris@dlr.de

*Ask us **questions**!*

*Join forces with us in a **project**!*

*Discuss modelling **ideas**!*

*Get **insights** on latest projects!*

***Collaborate** with us on extensions!*

Visit our website

Imprint

Topic:	Hands-on Workshop: Using AMIRIS
Date:	March 22nd 2024
Author:	Christoph Schimeczek
Institute:	Institute of Networked Energy Systems
Images:	DLR (CC BY-NC-ND 3.0)