Package 'aemo'

November 1, 2026

Title Download Australian Energy Market Operator Data

Version 0.4.3

Description Fetch Australian Energy Market Operator (AEMO) public data from 'NEMweb' <http://nemweb.com.au> and the Market Management System Data Model (MMSDM) historical archive. Provides tidy access to 5-minute and 30-minute wholesale electricity prices, regional demand, dispatch-unit output, interconnector flows, rooftop photovoltaic generation, generator bids, predispatch forecasts, frequency control ancillary services markets, and gas market data across the National Electricity Market (NEM) regions. Data is published by AEMO under its Copyright Permissions Notice <https://www.aemo.com.au/privacy-and-legal-notices/copyright-permissions>.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Imports cli (>= 3.6.0), httr2 (>= 1.0.0), openssl (>= 2.0.0), tools, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://charlescoverdale.github.io/aemo/, https://github.com/charlescoverdale/aemo

BugReports https://github.com/charlescoverdale/aemo/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Charles Coverdale [aut, cre]

Maintainer Charles Coverdale <charlesfcoverdale@gmail.com>

Repository CRAN

Date/Publication 2026-11-01 15:10:30 UTC

Contents

aemo_bids . . . 2
aemo_cache_info . . . 4
aemo_clear_cache . . . 4
aemo_constraints . . . 5
aemo_demand . . . 6
aemo_dispatch_units . . . 8
aemo_dlf . . . 9
aemo_fcas . . . 10
aemo_fcas_enablement . . . 11
aemo_gas . . . 12
aemo_gencon . . . 13
aemo_interconnector . . . 14
aemo_interconnectors . . . 15
aemo_market_notices . . . 16
aemo_mlf . . . 17
aemo_nemweb_download . . . 18
aemo_nemweb_ls . . . 19
aemo_outages . . . 20
aemo_participants . . . 21
aemo_pasa . . . 22
aemo_predispatch . . . 23
aemo_price . . . 24
aemo_price_caps . . . 26
aemo_regions . . . 27
aemo_rooftop_pv . . . 28
aemo_settlement . . . 29
aemo_snapshot . . . 30
aemo_spd_constraints . . . 31
aemo_throttle . . . 32
aemo_units . . . 33
print.aemo_tbl . . . 34

Index **35**

`aemo_bids` *Generator bid stack*

Description

Returns `BIDDAYOFFER_D` (daily bid summary: 10 price bands, MaxAvail, fixed load, locked at 12:30 D-1), `BIDPEROFFER_D` (per-interval availability and rebids), or the two joined on `(duid, settlementdate, bidtype)`.

Usage

```
aemo_bids(
  duid,
  start,
  end,
  resolution = c("day", "period", "joined"),
  allow_large = FALSE
)
```

Arguments

duid	Character vector of DUIDs. Required.
start, end	Window.
resolution	One of "day" (default, BIDDAYOFFER_D only), "period" (BIDPEROFFER_D only), or "joined" (BIDDAYOFFER_D inner-joined to BIDPEROFFER_D on (duid, settlementdate, bidtype)).
allow_large	Logical. Default FALSE.

Details

Parent / child structure. BIDDAYOFFER_D carries the **price bands** (priceband1..priceband10), which are locked at 12:30 on the day ahead and cannot be rebid. BIDPEROFFER_D carries the **per-interval availability bands** (bandavail1..bandavail10), which can be rebid intraday. Serious bidding analysis (Goncalves & Menezes 2022 Energy Economics 113 106398; Nelson et al. 2024 AJARE 68(4)) needs both joined.

Size warning. BIDPEROFFER_D monthly archives are multi-gigabyte. By default aemo_bids() refuses spans longer than 30 days; pass allow_large = TRUE to override.

Upstream gap. AEMO has a documented gap in BIDPEROFFER_D between March 2021 and July 2024. Rows in that range may be missing.

Value

An aemo_tbl.

See Also

Other dispatch: aemo_constraints(), aemo_dispatch_units(), aemo_fcas_enablement(), aemo_gencon(), aemo_interconnector(), aemo_market_notices(), aemo_outages(), aemo_rooftop_pv(), aemo_spd_constraints()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  # Daily bid summary (price bands)
  b <- aemo_bids(duid = "BW01",
                 start = now - 86400, end = now)
```

```
  # Joined: price bands + per-interval volumes
  bj <- aemo_bids(duid = "BW01", start = now - 86400, end = now,
                  resolution = "joined")
})
options(op)
```

`aemo_cache_info` *Inspect the local aemo cache*

Description

Inspect the local aemo cache

Usage

```
aemo_cache_info()
```

Value

A list with `dir`, `n_files`, `size_bytes`, `size_human`, `files`.

See Also

Other configuration: `aemo_clear_cache()`, `aemo_throttle()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
aemo_cache_info()
options(op)
```

`aemo_clear_cache` *Clear the aemo cache*

Description

Clear the aemo cache

Usage

```
aemo_clear_cache()
```

Value

Invisibly returns `NULL`.

See Also

Other configuration: aemo_cache_info(), aemo_throttle()

Examples

```
op <- options(aemo.cache_dir = tempdir())
aemo_clear_cache()
options(op)
```

---

aemo_constraints *Binding transmission and system constraints*

Description

Returns the DISPATCHCONSTRAINT table from NEMweb: one row per binding (or near-binding) constraint per 5-minute dispatch interval. Each row records the constraint ID, the left-hand side (LHS) and right-hand side (RHS) values, the marginal value (shadow price on the constraint in AUD/MWh), and the violation degree if any.

Usage

```
aemo_constraints(
  start,
  end,
  constraint_id = NULL,
  intervention = FALSE,
  min_marginal_value = 0.01
)
```

Arguments

start, end	Window (inclusive), character or POSIXct.
constraint_id	Optional character vector of constraint IDs (e.g. "N>>N-NIL_1", "V::S_NIL_TBSE"). NULL returns all binding constraints in the window.
intervention	Logical. Default FALSE.
min_marginal_value	Numeric. Only return constraints with marginal value at or above this threshold (AUD/MWh). 0 returns every row; the default of 0.01 filters out near-zero shadow prices that are typically noise.

Details

This is the table that answers the question "why was the RRP so high at 17:35?": the sum of marginal values across binding constraints at the Regional Reference Node equals the regional price component attributable to network limits.

Constraint equations and RHS terms (`GENCONDATA`, `GENCONSETINVOKE`) are published through MMSDM on a separate cadence and are not exposed directly by this function; use `aemo_nemweb_download()` on an MMSDM URL for those.

Value

An `aemo_tbl` with columns `settlementdate`, `constraintid`, `rhs`, `marginalvalue`, `violationdegree`, `lhs`, plus the intervention flag.

Source

AEMO NEMweb DISPATCHIS_Reports, DISPATCHCONSTRAINT table.

See Also

Other dispatch: `aemo_bids()`, `aemo_dispatch_units()`, `aemo_fcas_enablement()`, `aemo_gencon()`, `aemo_interconnector()`, `aemo_market_notices()`, `aemo_outages()`, `aemo_rooftop_pv()`, `aemo_spd_constraints()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  c <- aemo_constraints(start = now - 3600, end = now)
  head(c)
})
options(op)
```

---

aemo_demand *Regional electricity demand*

Description

Returns 5-minute regional demand from `DISPATCHREGIONSUM`. Three demand measures are supported, aligned with AEMO's Demand Terms taxonomy:

Usage

```
aemo_demand(
  region,
  start,
  end,
  measure = c("operational", "operational_less_snsg", "native"),
```

```
  intervention = FALSE
)
```

Arguments

region	NEM region code. Vector accepted.
start, end	Window (inclusive), character or POSIXct.
measure	One of "operational" (default), "operational_less_snsg", or "native".
intervention	Logical. Default FALSE filters to market pricing runs.

Details

- "operational" (default): TOTALDEMAND, the grid-measured demand met by scheduled and semi-scheduled generation plus net interchange. This is the quantity AEMO dispatches.
- "operational_less_snsg": TOTALDEMAND minus small non-scheduled generation (SS_SOLAR_UIGF + SS_WIND_UIGF where present).
- "native": TOTALDEMAND plus estimated rooftop PV generation. Closest to end-use consumption. If the rooftop PV component is not available in DISPATCHREGIONSUM the function warns and returns TOTALDEMAND; users should join with aemo_rooftop_pv() for a full native-demand estimate.

Timestamps are AEST (UTC+10, no DST).

Value

An aemo_tbl with columns settlementdate, regionid, demand_mw (the requested measure), plus the underlying DISPATCHREGIONSUM columns used in the derivation.

Source

AEMO NEMweb, AEMO Copyright Permissions Notice.

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  d <- aemo_demand("VIC1", now - 3600, now)
  head(d)
})
options(op)
```

`aemo_dispatch_units` *Per-DUID dispatch output*

Description

Returns 5-minute generator output for one or more DUIDs. Three measures are available:

Usage

```
aemo_dispatch_units(
  duid = NULL,
  start,
  end,
  measure = c("scada_mw", "target_mw", "both")
)
```

Arguments

`duid`	Optional character vector of DUIDs. `NULL` returns all generators.
`start, end`	Window (inclusive).
`measure`	One of `"scada_mw"` (default), `"target_mw"`, or `"both"`.

Details

- `"scada_mw"` (default): actual metered output from `DISPATCH_UNIT_SCADA` (`SCADAVALUE`).
- `"target_mw"`: dispatch target from `DISPATCHLOAD` (`TOTALCLEARED`). This is the MW AEMO *asked* the unit to produce at the end of the interval.
- `"both"`: returns `SCADAVALUE`, `INITIALMW` (SCADA at the start of the interval) and `TOTALCLEARED` (target at the end) in a single row per DUID per interval. Use this for ramp-trajectory research: the ramp applied during the interval is the straight line from `INITIALMW` to `TOTALCLEARED`.

Timestamps are AEST (UTC+10, no DST).

Value

An `aemo_tbl` with columns `settlementdate`, `duid`, and the requested measure(s).

See Also

Other dispatch: `aemo_bids()`, `aemo_constraints()`, `aemo_fcas_enablement()`, `aemo_gencon()`, `aemo_interconnector()`, `aemo_market_notices()`, `aemo_outages()`, `aemo_rooftop_pv()`, `aemo_spd_constraints()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  # SCADA actual
  d <- aemo_dispatch_units(duid = "BW01", start = now - 3600,
                           end = now)

  # Paired: INITIALMW, TOTALCLEARED, SCADAVALUE (ramp research)
  d_both <- aemo_dispatch_units(duid = "BW01",
                                start = now - 3600, end = now,
                                measure = "both")
})
options(op)
```

`aemo_dlf` *Distribution Loss Factors (DLF) by connection point*

Description

Returns the Distribution Loss Factor for NEM connection points from the MMSDM `LOSSFACTORMODEL` table. DLFs measure the average energy loss on the distribution network between a connection point and the transmission network boundary; a DLF of 1.02 means the generator must produce 2% more energy than it delivers to the transmission system.

Usage

```
aemo_dlf(year = NULL, connection_point_id = NULL)
```

Arguments

`year`	Financial year(s) as `"YYYY-YY"`. `NULL` returns all available years.
`connection_point_id`	Optional character vector of connection point IDs.

Details

DLFs are published annually by AEMO (NER 3.6.3) and combined with MLFs in settlement to give the total loss factor (TLF = MLF x DLF) used in energy payments.

Value

An `aemo_tbl` with at minimum `financial_year`, `connectionpointid`, and `dlf`.

Source

AEMO MMSDM archive, LOSSFACTORMODEL table. AEMO Copyright Permissions Notice.

See Also

Other reference: aemo_interconnectors(), aemo_mlf(), aemo_participants(), aemo_price_caps(), aemo_regions(), aemo_settlement(), aemo_snapshot(), aemo_units()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  dlf <- aemo_dlf(year = "2024-25")
  head(dlf)
})
options(op)
```

---

aemo_fcas *Frequency control ancillary services (FCAS) prices*

Description

Returns regional FCAS market prices across the eight contingency services plus the two regulation services. Ten services are live in the NEM since R1/L1 Very Fast commenced on 9 October 2023.

Usage

```
aemo_fcas(region, start, end, service = NULL, intervention = FALSE)
```

Arguments

region	NEM region code.
start, end	Window (inclusive).
service	Optional character vector of service names (e.g. "RAISE6SEC", "LOWER60SEC", "RAISE1SEC").
intervention	Logical. See aemo_price().

Details

Thin wrapper over aemo_price(..., market = "fcas").

Value

An aemo_tbl with one row per interval and columns for each requested FCAS RRP (AUD/MW).

See Also

Other price: aemo_price()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  f <- aemo_fcas("NSW1", now - 3600, now)
  head(f)
})
options(op)
```

`aemo_fcas_enablement` *FCAS enablement volumes by generator unit*

Description

Returns the MW enabled for each of the ten Frequency Control Ancillary Services (FCAS) per DUID per 5-minute dispatch interval from `DISPATCHLOAD`. This is the quantity side of the FCAS market: how much each unit was enabled (dispatched) to provide each service, as distinct from the price returned by `aemo_fcas()`.

Usage

```
aemo_fcas_enablement(
  duid = NULL,
  start,
  end,
  service = NULL,
  intervention = FALSE
)
```

Arguments

`duid`	Optional character vector of DUIDs. `NULL` returns all units. See `aemo_units()` for the full DUID registry.
`start, end`	Window (inclusive), character or POSIXct.
`service`	Optional character vector of service names, e.g. `c("raise6sec", "lowerreg")`. Case-insensitive. `NULL` returns all ten services.
`intervention`	Logical. Default `FALSE`.

Details

Ten services are active since 9 October 2023 when Very Fast (`RAISE1SEC` / `LOWER1SEC`) commenced. Before that date only eight services are present; the `raise1secmw` / `lower1secmw` columns will be `NA` for intervals before that date.

Value

An aemo_tbl with columns settlementdate, duid, and one column per FCAS service (raise1secmw, lower1secmw, raise6secmw, lower6secmw, raise60secmw, lower60secmw, raise5minmw, lower5minmw, raiseregmw, lowerregmw). Values are MW; zero means the unit was not enabled for that service.

Source

AEMO NEMweb DISPATCHIS_Reports, DISPATCHLOAD table.

See Also

Other dispatch: aemo_bids(), aemo_constraints(), aemo_dispatch_units(), aemo_gencon(), aemo_interconnector(), aemo_market_notices(), aemo_outages(), aemo_rooftop_pv(), aemo_spd_constraints()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  e <- aemo_fcas_enablement(start = now - 3600, end = now)
  head(e)
})
options(op)
```

---

aemo_gas *Gas market data (STTM, DWGM)*

Description

Returns Short Term Trading Market (STTM, Adelaide, Brisbane, Sydney hubs) or Declared Wholesale Gas Market (DWGM, Victoria) prices and volumes.

Usage

```
aemo_gas(market = c("sttm", "dwgm"), hub = NULL, start, end)
```

Arguments

market	One of "sttm" (default) or "dwgm".
hub	Optional STTM hub: "adelaide", "brisbane", or "sydney". Ignored for DWGM.
start, end	Window.

Value

An aemo_tbl.

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  g <- aemo_gas(market = "sttm", hub = "sydney",
                start = now - 7 * 86400, end = now)
  head(g)
})
options(op)
```

---

aemo_gencon *Generic constraint equations and RHS terms (GENCONDATA)*

Description

Downloads the GENCONDATA table from the most recent MMSDM monthly archive. GENCONDATA contains the equation definitions for every generic constraint active in the NEM: the constraint ID, the type (equality/inequality), a description of what the constraint models (thermal limit, voltage stability, system strength, etc.), and the default RHS value.

Usage

```
aemo_gencon(constraint_id = NULL, type = NULL)
```

Arguments

constraint_id	Optional character vector of constraint IDs to filter on. NULL returns all equations.
type	Optional constraint type filter (e.g. "LE" for <=, "GE" for >=, "EQ" for =). NULL returns all types.

Details

Pair this with aemo_constraints() to go from a binding dispatch interval to the underlying network equation. The workflow is: aemo_constraints() tells you *which* constraint bound and *how hard* (marginalvalue = shadow price); aemo_gencon() tells you *what the constraint is* (which elements and why the RHS was set at that level).

Value

An aemo_tbl with columns including genconid (constraint ID), constrainttype, description, genericconstraintrhs (default RHS value). Additional columns from GENCONDATA (effective dates, generic constraint equation weights) will be present when available.

Source

AEMO NEMweb MMSDM archive, GENCONDATA table. AEMO Copyright Permissions Notice.

See Also

aemo_constraints() for the real-time binding constraint shadow prices.

Other dispatch: aemo_bids(), aemo_constraints(), aemo_dispatch_units(), aemo_fcas_enablement(), aemo_interconnector(), aemo_market_notices(), aemo_outages(), aemo_rooftop_pv(), aemo_spd_constraints()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  # Find equations for the Heywood interconnector thermal limits
  g <- aemo_gencon(constraint_id = c("V::S_NIL_TBSE", "V::S_NIL_FCSPS"))
  head(g)
})
options(op)
```

---

aemo_interconnector *NEM interconnector flows*

Description

Returns MW flows, losses, and limits for one or more NEM interconnectors from DISPATCHINTERCONNECTORRES.

Usage

```
aemo_interconnector(flow = NULL, start, end, intervention = FALSE)
```

Arguments

flow	Optional character vector of interconnector IDs.
start, end	Window.
intervention	Logical. Default FALSE.

Details

AEMO's METEREDMWFLOW is positive when power flows from REGIONFROM to REGIONTO. For per-interconnector direction conventions see aemo_interconnectors().

Value

An aemo_tbl.

See Also

Other dispatch: aemo_bids(), aemo_constraints(), aemo_dispatch_units(), aemo_fcas_enablement(), aemo_gencon(), aemo_market_notices(), aemo_outages(), aemo_rooftop_pv(), aemo_spd_constraints()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  i <- aemo_interconnector(flow = "V-SA",
                           start = now - 3600, end = now)
  head(i)
})
options(op)
```

`aemo_interconnectors` *NEM interconnectors*

Description

Returns a static table of the seven NEM interconnectors. The seventh entry is Project EnergyConnect (PEC), whose Stage 1 was energised on 30 April 2025 with full ~800 MW capability expected mid-2026.

Usage

```
aemo_interconnectors()
```

Value

An `aemo_tbl` with columns `interconnector_id`, `from_region`, `to_region`, `name`, `energised`.

See Also

Other reference: `aemo_dlf()`, `aemo_mlf()`, `aemo_participants()`, `aemo_price_caps()`, `aemo_regions()`, `aemo_settlement()`, `aemo_snapshot()`, `aemo_units()`

Examples

```
aemo_interconnectors()
```

`aemo_market_notices` *NEM market notices*

Description

Returns the `MARKETNOTICEDATA` feed from MMSDM. Market notices are AEMO's free-text log of market-relevant events: Lack of Reserve (LOR1/2/3) declarations, Reliability and Emergency Reserve Trader (RERT) activations, market suspensions, market directions, unit withdrawals, system security events, administered price declarations, and operator advisories.

Usage

```
aemo_market_notices(start, end, notice_type = NULL, region = NULL)
```

Arguments

`start`, `end`	Window (inclusive) applied to `EFFECTIVEDATE`.
`notice_type`	Optional character vector (e.g. `"LOR1"`, `"RERT"`, `"PRICES SUBJECT TO REVIEW"`). Case-insensitive substring match.
`region`	Optional NEM region code.

Details

Pair with `aemo_constraints()` and `aemo_price()` to sequence the causal chain of a price event. Rangarajan, Svec, Foley and Trück (2025, *Energy Economics* 141) use `MARKETNOTICEDATA` to order the intervention messages that mark entry to and exit from the June 2022 NEM suspension.

Value

An `aemo_tbl` with columns from MARKETNOTICEDATA: `noticeid`, `effectivedate`, `typeid` (notice category), `originator`, `priority`, `reason` (the notice text), `externalreference`, and where present `regionid`.

Source

AEMO NEMweb MMSDM archive, MARKETNOTICEDATA table. AEMO Copyright Permissions Notice.

See Also

Other dispatch: `aemo_bids()`, `aemo_constraints()`, `aemo_dispatch_units()`, `aemo_fcas_enablement()`, `aemo_gencon()`, `aemo_interconnector()`, `aemo_outages()`, `aemo_rooftop_pv()`, `aemo_spd_constraints()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  # LOR declarations during the June 2022 NEM suspension
  n <- aemo_market_notices(
    start = "2022-06-13",
    end   = "2022-06-14",
    notice_type = "LOR"
  )
  head(n)
})
options(op)
```

`aemo_mlf` *Marginal Loss Factors (MLF) by DUID and financial year*

Description

Returns the Marginal Loss Factor applicable to each DUID for the requested financial year(s). MLFs measure the incremental network loss at a connection point relative to the Regional Reference Node (RRN); a DUID with MLF = 0.97 receives 97% of the regional RRP per MWh generated.

Usage

```
aemo_mlf(year = NULL, duid = NULL)
```

Arguments

`year`	Financial year(s) as `"YYYY-YY"` strings (e.g. `"2025-26"`). `NULL` (the default) returns the current financial year. Multiple years accepted; each needs up to 13 small downloads.
`duid`	Optional character vector of DUIDs to filter on. MMSDM publishes MLFs by connection point, so DUIDs are mapped to connection points through the unit registry (`aemo_units()`), which adds a `duid` column. `NULL` returns every connection point.

Details

MLFs are published annually by AEMO under NER 3.6.2 and are used in settlement calculations and in PPA revenue reconstruction. Each year's determination is read from the `TRANSMISSIONLOSSFACTOR` table of the June MMSDM archive that precedes it, together with the revisions in the monthly archives after it. There is no bundled fallback: if MMSDM cannot be reached the function errors rather than return unsourced values.

Value

An aemo_tbl with one row per connection point and effective date: financial_year, connectionpointid, regionid, mlf, effectivedate, versionno, secondary_tlf and lastchanged, plus duid when duid is given. A factor revised during the year appears once per effective date.

Source

AEMO MMSDM archive, TRANSMISSIONLOSSFACTOR table. AEMO Copyright Permissions Notice.

See Also

aemo_units() for the DUID registry, aemo_price() for regional RRPs.

Other reference: aemo_dlf(), aemo_interconnectors(), aemo_participants(), aemo_price_caps(), aemo_regions(), aemo_settlement(), aemo_snapshot(), aemo_units()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  mlf <- aemo_mlf(year = "2024-25")
  head(mlf)
})
options(op)
```

---

aemo_nemweb_download *Download a NEMweb zipped CSV to the local cache*

Description

Thin wrapper over the internal cache-aware downloader.

Usage

```
aemo_nemweb_download(url, cache = TRUE)
```

Arguments

url	A fully-qualified NEMweb URL (zipped CSV).
cache	Logical. Reuse cached file if present.

Value

Path to the cached file.

See Also

Other low-level: aemo_nemweb_ls()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  files <- aemo_nemweb_ls("/Reports/Current/DispatchIS_Reports/")
  if (nrow(files) > 0) {
    f <- aemo_nemweb_download(files$url[1])
    file.exists(f)
  }
})
options(op)
```

---

aemo_nemweb_ls *List files in a NEMweb directory*

Description

Returns a data frame of files in a NEMweb path, parsed from the Apache directory-listing HTML.

Usage

```
aemo_nemweb_ls(path)
```

Arguments

path	NEMweb subpath (e.g. "/Reports/Current/DispatchIS_Reports/"). Leading and trailing slashes are optional.

Value

A data frame with name, modified, size, url.

Source

AEMO NEMweb https://nemweb.com.au, published under the AEMO Copyright Permissions Notice.

See Also

Other low-level: aemo_nemweb_download()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  files <- aemo_nemweb_ls("/Reports/Current/DispatchIS_Reports/")
  head(files)
})
options(op)
```

`aemo_outages` *Planned and forced network outages*

Description

Returns the `NETWORK_OUTAGEDETAIL` table from MMSDM, which records every planned and forced outage on NEM transmission and distribution network elements. Outages are a primary driver of binding constraints and price spikes: when a line or transformer is out of service the network is more constrained, reducing the thermal limits that appear as RHS values in DISPATCHCONSTRAINT.

Usage

```
aemo_outages(start, end, element_id = NULL, outage_type = NULL, region = NULL)
```

Arguments

`start`, `end`	Outage window (inclusive). Filters on `starttime` and `endtime`: any outage active during the window is returned. Character or POSIXct.
`element_id`	Optional character vector of network element IDs. `NULL` returns all elements.
`outage_type`	Optional character. One of `"PLANNED"`, `"FORCED"`, or `NULL` (both). Case-insensitive.
`region`	Optional NEM region code. Filters on the region column where available.

Details

Use case. Pair with `aemo_constraints()` to explain a price spike: `aemo_outages()` tells you which elements were off-service at the time; `aemo_constraints()` tells you which constraints bound; together they answer "why was SA spot price AUD 15,000 at 17:35?".

Value

An `aemo_tbl` with columns from NETWORK_OUTAGEDETAIL including `outageid`, `starttime`, `endtime`, `substationid`, `equipmenttype`, `equipmentid`, `outagetype`, `regionid`, and `restarttimeunknown`. Exact column set depends on the MMSDM version.

Source

AEMO NEMweb MMSDM archive, NETWORK_OUTAGEDETAIL table. AEMO Copyright Permissions Notice.

See Also

aemo_constraints() for the binding constraint shadow prices that these outages drive.

Other dispatch: aemo_bids(), aemo_constraints(), aemo_dispatch_units(), aemo_fcas_enablement(), aemo_gencon(), aemo_interconnector(), aemo_market_notices(), aemo_rooftop_pv(), aemo_spd_constraints()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  # Forced outages during a recent high-price period in SA
  o <- aemo_outages(
    start = "2024-03-01",
    end   = "2024-03-02",
    outage_type = "FORCED"
  )
  head(o)
})
options(op)
```

---

aemo_participants *NEM market participants and their registered DUIDs*

Description

Returns a mapping of NEM market participants (companies) to their registered Dispatchable Unit Identifiers (DUIDs), joined from the MMSDM PARTICIPANT and DUDETAILSUMMARY tables. Use this for corporate ownership analysis: rolling up generator output or bids from DUID-level data to the company level.

Usage

```
aemo_participants()
```

Value

An aemo_tbl with columns participantid, participantclassid (e.g. GENERATOR, LOAD, TRADER), name (company name), duid, stationid, regionid, dispatchtype, schedule_type. Rows are one per participant-DUID combination. If MMSDM is unreachable, returns an empty table with a warning.

Source

AEMO NEMweb MMSDM archive, PARTICIPANT and DUDETAILSUMMARY tables. AEMO Copyright Permissions Notice.

See Also

aemo_units() for DUID-level registry without participant mapping.

Other reference: aemo_dlf(), aemo_interconnectors(), aemo_mlf(), aemo_price_caps(), aemo_regions(), aemo_settlement(), aemo_snapshot(), aemo_units()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  pt <- aemo_participants()
  # DUIDs owned by AGL
  pt[grepl("AGL", pt$name, ignore.case = TRUE), ]
})
options(op)
```

---

aemo_pasa *Projected Assessment of System Adequacy (PASA)*

Description

Returns short-term (STPASA, 1-7 day) or medium-term (MTPASA, 2-year) system adequacy projections.

Usage

```
aemo_pasa(
  horizon = c("short", "medium"),
  region = NULL,
  start = NULL,
  end = NULL
)
```

Arguments

horizon	One of "short" (default) or "medium".
region	Optional NEM region code.
start, end	Optional window of run-times. Defaults to the last 24 hours.

Value

An aemo_tbl.

See Also

Other forecast: aemo_predispatch()

Examples

```
op <- options(aemo.cache_dir = tempdir())

# STPASA is republished every half hour, so the 24-hour default window
# pulls dozens of files. Pass an explicit narrow window to keep the
# example to a couple of downloads.
try({
  p <- aemo_pasa(horizon = "short", region = "NSW1",
                 start = Sys.time() - as.difftime(1, units = "hours"),
                 end   = Sys.time())
})
options(op)
```

`aemo_predispatch` *Price and demand forecasts (P5MIN, PREDISPATCH)*

Description

Returns AEMO's forecast prices and demand for a NEM region. Two horizons:

- `"p5min"`: 5-minute-ahead forecast, 12 intervals ahead, published every 5 minutes.
- `"predispatch"`: 40-hour-ahead predispatch at 30-minute resolution, published every 30 minutes.

Usage

```
aemo_predispatch(
  region,
  start,
  end,
  horizon = c("predispatch", "p5min"),
  run_datetime = NULL
)
```

Arguments

`region`	NEM region code.
`start`, `end`	Window of forecast run-times.
`horizon`	One of `"predispatch"` (default) or `"p5min"`.
`run_datetime`	Optional character or `POSIXct`. A specific `RUN_DATETIME` to pin to. `NULL` (default) returns every vintage issued in `[start, end]`.

Details

The 7-day predispatch publication was retired when 5-minute settlement commenced; for longer horizons use `aemo_pasa()`.

Vintages. PREDISPATCH and P5MIN forecasts are re-issued every 30 or 5 minutes respectively. Every vintage is archived by its `RUN_DATETIME`. By default (`run_datetime = NULL`) this function returns all vintages whose *file* timestamp falls in `[start, end]`, i.e. all the forecasts *issued* during the window. The `periodid` / `datetime` columns on the returned rows give each forecast's *target* time.

For forecast-error research (comparing a forecast vintage against the realised dispatch) pass `run_datetime` to pin a specific vintage:

```
# The PREDISPATCH run issued at 15:00 on 1 June 2024 --
# 80 half-hour-ahead rows covering 15:30 out to 31:30 hours.
v <- aemo_predispatch("NSW1", start = "2024-06-01", end = "2024-06-02",
                      run_datetime = "2024-06-01 15:00")
```

This is the vintage-aware pattern used in Prakash (2023) *NEMSEER* (JOSS 8(92) 5883).

Value

An `aemo_tbl`.

See Also

Other forecast: `aemo_pasa()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  p <- aemo_predispatch("NSW1", start = now - 3600, end = now)
  head(p)
})
options(op)
```

---

aemo_price *Wholesale electricity prices*

Description

Returns 5-minute dispatch prices or 30-minute trading prices for a NEM region over a specified window. Filters intervention runs by default so the returned prices are the market clearing prices used in settlement.

Usage

```
aemo_price(
  region,
  start,
  end,
  interval = c("5min", "30min"),
  market = c("energy", "fcas"),
  intervention = FALSE
)
```

Arguments

region	One of "NSW1", "QLD1", "SA1", "TAS1", "VIC1". Accepts a vector.
start, end	Start and end times (inclusive). Character (parsed as AEST) or POSIXct.
interval	One of "5min" (default) or "30min".
market	One of "energy" (default, returns RRP) or "fcas" (returns the FCAS service RRPs).
intervention	Logical. FALSE (default) returns only the market pricing run; TRUE returns both market and physical runs, with the intervention column preserved.

Details

Timestamps are AEST (UTC+10, no daylight savings) to match AEMO's market clock. See the package-level documentation for the period-ending timestamp convention (a row stamped 00:05 is the 5-minute period ending at 00:05).

Intervention. DISPATCHPRICE contains both market pricing runs (INTERVENTION = 0) and physical / intervention runs (INTERVENTION = 1). The default filters to market runs. Pass intervention = TRUE to get both.

30-minute settlement and the 5MS transition. Before 1 October 2021 the NEM settled on 30-minute trading prices from TRADINGPRICE (TRADINGIS). On 1 October 2021 five-minute settlement (5MS) commenced and settlement moved to native 5-minute prices. When interval = "30min":

- For the pre-5MS period (start < 2021-10-01): prices are read from TRADINGIS (TRADINGPRICE).
- For the post-5MS period: prices are derived by taking the arithmetic mean of the six 5-minute dispatch prices within each 30-minute trading interval, consistent with how AEMO calculates the TRADINGPRICE column in TradingIS post-5MS.

Data availability. NEMweb holds the 5-minute dispatch files in its Current directory for only about two days, and daily (dispatch) or weekly (trading) rollups in its Archive for about 13 months. Each day of a query is read from an Archive rollup when one exists and from Current files otherwise. For queries older than the Archive window, use aemo_nemweb_download() with an MMSDM URL directly.

Value

An aemo_tbl. Key columns include settlementdate (POSIXct AEST), regionid, rrp (AUD/MWh, energy) or the FCAS service RRPs (AUD/MW), and intervention.

Source

AEMO NEMweb https://nemweb.com.au, AEMO Copyright Permissions Notice.

See Also

Other price: aemo_fcas()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  p <- aemo_price("NSW1", now - 3600, now)
  head(p)
})
options(op)
```

---

aemo_price_caps *NEM market price caps, price floor, and CPT by financial year*

Description

Returns a static reference table of the **Market Price Cap (MPC)**, **Market Price Floor (MPF)**, **Cumulative Price Threshold (CPT)**, and **Administered Price Cap (APC)** that apply in each NEM financial year (1 July to 30 June). These are set by the AEMC under NER 3.9.4 and indexed annually to CPI.

Usage

```
aemo_price_caps()
```

Details

Use this table to interpret spot-price extremes: any RRP hitting the MPC (typically AUD 15,000 to 17,500 per MWh depending on year) indicates a price cap event; when the rolling-seven-day cumulative price in a region exceeds the CPT (~AUD 1.5 million per MWh-equivalent), AEMO imposes the APC (AUD 300/MWh) until the CPT falls back below the threshold.

The table covers 2015-16 onwards and is updated on each package release. For the authoritative current values see https://www.aemc.gov.au/regulation/energy-rules/national-electricity-rules and the AEMO Reliability Settings publications.

Value

An aemo_tbl with columns year (financial year, "YYYY-YY"), market_price_cap_aud_per_mwh, market_price_floor_aud_per_mwh, cumulative_price_threshold_aud, administered_price_cap_aud_per_mwh, and source.

See Also

Other reference: aemo_dlf(), aemo_interconnectors(), aemo_mlf(), aemo_participants(), aemo_regions(), aemo_settlement(), aemo_snapshot(), aemo_units()

Examples

```
caps <- aemo_price_caps()
head(caps)
```

---

aemo_regions *NEM regions*

Description

Returns a static table of the five NEM regions with metadata. The market_timezone column is the AEMO market clock (AEST, UTC+10, no DST, year-round) that applies to every timestamp in NEMweb files; wall_timezone is the local civil time zone consumers experience (observes DST in NSW/VIC/TAS/SA).

Usage

```
aemo_regions()
```

Value

An aemo_tbl with columns region, name, state, wall_timezone, market_timezone, commenced.

See Also

Other reference: aemo_dlf(), aemo_interconnectors(), aemo_mlf(), aemo_participants(), aemo_price_caps(), aemo_settlement(), aemo_snapshot(), aemo_units()

Examples

```
aemo_regions()
```

`aemo_rooftop_pv` *Rooftop PV actuals and forecasts*

Description

Returns AEMO's region-level estimate of rooftop PV generation, either actuals or forecasts. Published at 30-minute resolution.

Usage

```
aemo_rooftop_pv(region, start, end, type = c("actual", "forecast"))
```

Arguments

`region`	NEM region code.
`start`, `end`	Window.
`type`	One of `"actual"` (default) or `"forecast"`.

Details

The "actual" figure is an AEMO estimate derived from the APVI sampling model and weather data, not metered SCADA output. It is the best available public measure of aggregate rooftop PV generation but is subject to revision.

Value

An `aemo_tbl`.

See Also

Other dispatch: `aemo_bids()`, `aemo_constraints()`, `aemo_dispatch_units()`, `aemo_fcas_enablement()`, `aemo_gencon()`, `aemo_interconnector()`, `aemo_market_notices()`, `aemo_outages()`, `aemo_spd_constraints()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  now <- Sys.time()
  r <- aemo_rooftop_pv("NSW1",
                       start = now - 3600, end = now)
  head(r)
})
options(op)
```

`aemo_settlement` *Settlement cash-flow and residue tables*

Description

Returns settlement reconciliation tables from MMSDM. Three views are exposed, corresponding to the three tables gentailer hedging workflows most commonly need:

Usage

```
aemo_settlement(table = c("cashflow", "fcas_recovery", "residues"), start, end)
```

Arguments

`table`	One of `"cashflow"`, `"fcas_recovery"`, or `"residues"`.
`start, end`	Window (inclusive) applied to `SETTLEMENTDATE` / `PAYMENTDATE`.

Details

- `"cashflow"` (default): `SETCFM` (NEMDE-derived energy settlement amounts per participant per trading interval).
- `"fcas_recovery"`: `SETFCASREGIONRECOVERY` (the recovery allocation of FCAS costs to customer load per region per trading interval).
- `"residues"`: `SETRESIDUECONTRACTPAYMENT` (settlement residue auction (SRA) contract payments against interconnector settlement residues).

Access. These tables are in the MMSDM monthly archive (not the NEMweb Current retention). Expect two-month latency between trading date and availability.

Value

An `aemo_tbl` with the raw MMSDM columns for the requested table.

Source

AEMO NEMweb MMSDM archive, SETCFM / SETFCASREGIONRECOVERY / SETRESIDUECONTRACTPAYMENT. AEMO Copyright Permissions Notice.

See Also

`aemo_mlf()` for the transmission loss factors that scale settlement amounts; `aemo_interconnector()` for the flow side of the residue calculation.

Other reference: `aemo_dlf()`, `aemo_interconnectors()`, `aemo_mlf()`, `aemo_participants()`, `aemo_price_caps()`, `aemo_regions()`, `aemo_snapshot()`, `aemo_units()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  s <- aemo_settlement(table = "cashflow",
                       start = "2024-06-01",
                       end   = "2024-06-02")
  head(s)
})
options(op)
```

`aemo_snapshot` *Snapshot provenance for an aemo_tbl*

Description

Returns a one-row-per-source provenance record for an `aemo_tbl`, suitable for inclusion in a paper appendix, a Zenodo deposit, or a CRAN-style data manifest. The snapshot captures:

Usage

```
aemo_snapshot(x)
```

Arguments

`x`	An `aemo_tbl` (or a list of them).

Details

- `title`: the human-readable title of the table;
- `source`: the NEMweb / MMSDM URL family the table was drawn from;
- `licence`: the AEMO Copyright Permissions Notice (always);
- `retrieved`: the `POSIXct` timestamp at which the table was constructed;
- `rows`, `cols`: observed dimensions;
- `sha256`: a SHA-256 digest of the table's printed body, stable across R versions and platforms.

The `sha256` column is what makes the snapshot *pinnable*: if the same query returns a different hash, the underlying data has changed (or the row-order has, which is also worth knowing). Pair with a git commit of the analysis script to give a reader a closed reproducibility loop.

Value

A data frame with one row per table.

See Also

Other reference: `aemo_dlf()`, `aemo_interconnectors()`, `aemo_mlf()`, `aemo_participants()`, `aemo_price_caps()`, `aemo_regions()`, `aemo_settlement()`, `aemo_units()`

Examples

```
x <- structure(
  data.frame(settlementdate = Sys.time(), region = "NSW1", rrp = 80),
  aemo_title = "Demo",
  aemo_source = "https://nemweb.com.au",
  aemo_licence = "AEMO Copyright Permissions Notice",
  aemo_retrieved = Sys.time(),
  class = c("aemo_tbl", "data.frame")
)
aemo_snapshot(x)
```

---

aemo_spd_constraints *SPD constraint tables (regions, interconnectors, connection points)*

Description

Returns the SPD (Security and Projected Dispatch) constraint coefficient tables from MMSDM. Where GENCONDATA gives the high-level equation definition, the SPD tables give the per-term *coefficients* used in the NEMDE dispatch optimisation:

Usage

```
aemo_spd_constraints(
  table = c("region", "interconnector", "connection_point"),
  constraint_id = NULL
)
```

Arguments

table	One of "region" (default), "interconnector", or "connection_point".
constraint_id	Optional character vector of GENCONID/CONSTRAINTID values.

Details

- "region": SPDREGIONCONSTRAINT (regional-demand and regional-generation terms).
- "interconnector": SPDINTERCONNECTORCONSTRAINT (interconnector-flow terms).
- "connection_point": SPDCONNECTIONPOINTCONSTRAINT (per-DUID connection-point terms).

These tables are required for NEM dispatch replication (the nempy Python package, Gorman, Bruce & MacGill 2022, *JOSS* 7(70) 3596, doi:10.21105/joss.03596, uses all three when reproducing NEMDE solves).

Value

An aemo_tbl with columns from the requested SPD table. GENCONID and FACTOR (the coefficient) are always present; other columns vary by table.

Source

AEMO NEMweb MMSDM archive, SPDREGIONCONSTRAINT / SPDINTERCONNECTORCONSTRAINT / SPDCONNECTIONPOINTCONSTRAINT.

See Also

aemo_gencon() for the equation-level metadata, aemo_constraints() for the 5-min binding-constraint feed with shadow prices.

Other dispatch: aemo_bids(), aemo_constraints(), aemo_dispatch_units(), aemo_fcas_enablement(), aemo_gencon(), aemo_interconnector(), aemo_market_notices(), aemo_outages(), aemo_rooftop_pv()

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  s <- aemo_spd_constraints(table = "interconnector")
  head(s)
})
options(op)
```

---

aemo_throttle *Configure request throttling*

Description

Controls the delay between successive NEMweb requests. Defaults to 1 second (half a request per second).

Usage

```
aemo_throttle(delay = 1)
```

Arguments

delay	Numeric. Minimum delay between requests in seconds. Internally this becomes `httr2::req_throttle(rate = 1/delay)`.

Value

Invisibly returns the previous value.

See Also

Other configuration: aemo_cache_info(), aemo_clear_cache()

Examples

```
old <- aemo_throttle(0.5)  # 2 requests per second
aemo_throttle(old)         # restore
```

`aemo_units` *NEM generators (DUID registry)*

Description

Downloads the `DUDETAILSUMMARY` table from the most recent MMSDM monthly archive and returns one row per registered DUID (Dispatchable Unit Identifier) with station, region, dispatch type, classification, and schedule type. Typical output is 500+ DUIDs covering scheduled, semi-scheduled, and non-scheduled generators, bidirectional storage (BESS), and loads.

Usage

```
aemo_units(as_of = NULL)
```

Arguments

`as_of`	Optional `Date` or `POSIXct`. Returns the DUID registry as it was on that date. `NULL` (default) returns the current registry.

Details

Effective-date filtering. `DUDETAILSUMMARY` is effective-dated (`START_DATE`, `END_DATE` per row with multiple `VERSIONNO` vintages per DUID over time). The default (`as_of = NULL`) returns rows whose `[START_DATE, END_DATE]` interval covers the date the MMSDM archive was published: i.e. the *current* registry. Pass an `as_of` date to get the registry as it was on that date (essential for historical analysis: Liddell's four DUIDs were retired in April 2023; pre-2023 queries need them).

This matches the as-of-join pattern documented in Gorman et al. (2018) *NEMOSIS* (APSRC) for correct historical joins.

Value

An `aemo_tbl` keyed by `duid`. Columns include (from DUDETAILSUMMARY): `duid`, `stationid`, `regionid`, `dispatchtype` (GENERATOR / LOAD), `connectionpointid`, `schedule_type` (SCHEDULED / SEMI-SCHEDULED / NON-SCHEDULED), `start_date`, `end_date`.

Source

AEMO NEMweb MMSDM archive, DUDETAILSUMMARY table, AEMO Copyright Permissions Notice.

See Also

Other reference: `aemo_dlf()`, `aemo_interconnectors()`, `aemo_mlf()`, `aemo_participants()`, `aemo_price_caps()`, `aemo_regions()`, `aemo_settlement()`, `aemo_snapshot()`

Examples

```
op <- options(aemo.cache_dir = tempdir())
try({
  # Current DUID registry
  u <- aemo_units()

  # DUIDs as they were on 1 March 2023 (pre-Liddell retirement)
  u_2023 <- aemo_units(as_of = "2023-03-01")
})
options(op)
```

`print.aemo_tbl` *Print an aemo_tbl*

Description

Print an aemo_tbl

Usage

```
## S3 method for class 'aemo_tbl'
print(x, ...)
```

Arguments

`x`	An `aemo_tbl`.
`...`	Passed to the next print method.

Value

Invisibly returns `x`.

Examples

```
x <- data.frame(settlementdate = Sys.time(), region = "NSW1", rrp = 80)
x <- structure(x, aemo_title = "Demo", aemo_source = "https://nemweb.com.au",
               aemo_licence = "AEMO Copyright Permissions Notice",
               aemo_retrieved = Sys.time(),
               class = c("aemo_tbl", "data.frame"))
print(x)
```

Index

∗ **configuration**
  aemo_cache_info, 4
  aemo_clear_cache, 4
  aemo_throttle, 32
∗ **demand**
  aemo_demand, 6
∗ **dispatch**
  aemo_bids, 2
  aemo_constraints, 5
  aemo_dispatch_units, 8
  aemo_fcas_enablement, 11
  aemo_gencon, 13
  aemo_interconnector, 14
  aemo_market_notices, 16
  aemo_outages, 20
  aemo_rooftop_pv, 28
  aemo_spd_constraints, 31
∗ **forecast**
  aemo_pasa, 22
  aemo_predispatch, 23
∗ **gas**
  aemo_gas, 12
∗ **low-level**
  aemo_nemweb_download, 18
  aemo_nemweb_ls, 19
∗ **price**
  aemo_fcas, 10
  aemo_price, 24
∗ **reference**
  aemo_dlf, 9
  aemo_interconnectors, 15
  aemo_mlf, 17
  aemo_participants, 21
  aemo_price_caps, 26
  aemo_regions, 27
  aemo_settlement, 29
  aemo_snapshot, 30
  aemo_units, 33

aemo_bids, 2
aemo_bids(), *6*, *8*, *12*, *14*, *16*, *21*, *28*, *32*
aemo_cache_info, 4
aemo_cache_info(), *5*, *32*
aemo_clear_cache, 4
aemo_clear_cache(), *4*, *32*
aemo_constraints, 5
aemo_constraints(), *3*, *8*, *12–14*, *16*, *20, 21*, *28*, *32*
aemo_demand, 6
aemo_dispatch_units, 8
aemo_dispatch_units(), *3*, *6*, *12*, *14*, *16*, *21*, *28*, *32*
aemo_dlf, 9
aemo_dlf(), *15*, *18*, *22*, *27*, *29, 30*, *33*
aemo_fcas, 10
aemo_fcas(), *11*, *26*
aemo_fcas_enablement, 11
aemo_fcas_enablement(), *3*, *6*, *8*, *14*, *16*, *21*, *28*, *32*
aemo_gas, 12
aemo_gencon, 13
aemo_gencon(), *3*, *6*, *8*, *12*, *14*, *16*, *21*, *28*, *32*
aemo_interconnector, 14
aemo_interconnector(), *3*, *6*, *8*, *12*, *14*, *16*, *21*, *28, 29*, *32*
aemo_interconnectors, 15
aemo_interconnectors(), *10*, *14*, *18*, *22*, *27*, *29, 30*, *33*
aemo_market_notices, 16
aemo_market_notices(), *3*, *6*, *8*, *12*, *14*, *21*, *28*, *32*
aemo_mlf, 17
aemo_mlf(), *10*, *15*, *22*, *27*, *29, 30*, *33*
aemo_nemweb_download, 18
aemo_nemweb_download(), *6*, *19*, *25*
aemo_nemweb_ls, 19
aemo_nemweb_ls(), *19*
aemo_outages, 20
aemo_outages(), *3*, *6*, *8*, *12*, *14*, *16*, *28*, *32*

aemo_participants, 21
aemo_participants(), *10*, *15*, *18*, *27*, *29, 30*, *33*
aemo_pasa, 22
aemo_pasa(), *24*
aemo_predispatch, 23
aemo_predispatch(), *22*
aemo_price, 24
aemo_price(), *10*, *16*, *18*
aemo_price_caps, 26
aemo_price_caps(), *10*, *15*, *18*, *22*, *27*, *29, 30*, *33*
aemo_regions, 27
aemo_regions(), *10*, *15*, *18*, *22*, *27*, *29, 30*, *33*
aemo_rooftop_pv, 28
aemo_rooftop_pv(), *3*, *6*, *8*, *12*, *14*, *16*, *21*, *32*
aemo_settlement, 29
aemo_settlement(), *10*, *15*, *18*, *22*, *27*, *30*, *33*
aemo_snapshot, 30
aemo_snapshot(), *10*, *15*, *18*, *22*, *27*, *29*, *33*
aemo_spd_constraints, 31
aemo_spd_constraints(), *3*, *6*, *8*, *12*, *14*, *16*, *21*, *28*
aemo_throttle, 32
aemo_throttle(), *4, 5*
aemo_units, 33
aemo_units(), *10, 11*, *15*, *17, 18*, *22*, *27*, *29, 30*

print.aemo_tbl, 34