Day-Ahead Optimization API Specification

Version: 1.0 **Last Updated:** 2025-11-05 **Status:** Draft

1. Overview

1.1 Purpose

This API provides optimization services for energy site management, focusing on: 1. **Optimal Bidding**: Generate multi-step bid curves for day-ahead (DA) spot market and ancillary services (aFRR, mFRR) 2. **Device Planning**: Create operational schedules for site devices given market commitments and locked ancillary service reservations

1.2 Technology Stack

- **Framework**: FastAPI (Python)
- **Pattern**: Asynchronous job-based execution
- **Compression**: gzip (automatic, transparent to clients)
- **Format**: JSON (REST)
- **Base URL**: `/api/v1/`

1.3 Key Features

- 15-minute or 1-hour resolution (96 or 24 intervals per day)
- **Operational clients**: Up to 296 intervals (~3 days at 15-min resolution)
- **Investment clients**: Up to 100,000 intervals (~11 years at 1-hour resolution)
- Multi-step bid curves with optimal price-quantity discretization
- Device-specific ancillary service capabilities (operational clients only)
- Flexible constraint modeling (can_run, must_run, power ranges)
- Expected profit maximization
- **Multi-site optimization**: Multiple sites can be optimized together in a single request

1.4 Authentication

To be specified - Placeholder for API key, OAuth2, or other authentication mechanism.

1.5 Client Types and Capabilities

The API supports two client types with different capabilities based on their use cases:

Feature	Operational Client	Investment Client
Max Intervals	296 (3 days @ 15-min)	100,000 (~11 years @ 1-hour)
Resolution	15-min or 1-hour	1-hour only
Endpoints	`/optimal-bidding`, `/device-planning`	`/device-planning` only*
ANS Optimization	✅ Yes	❌ No
Binary Variables	✅ Yes (CHP on/off)	⚠ Continuous (relaxed)
Timeout	300 seconds (5 min)	3600 seconds (1 hour)
Use Case	Day-ahead bidding, short-term dispatch	Capacity planning, investment ROI

*Investment clients receive `403 Forbidden` when attempting to access `/optimal-bidding` endpoint.

1.5.1 Authentication and Client Identification

Client type is identified via API key prefix: - **Operational clients**: API key starts with `op_` (e.g., `op_1234567890abcdef`) - **Investment clients**: API key starts with `inv_` (e.g., `inv_9876543210fedcba`)

Example authentication header:

```
Authorization: Bearer op_1234567890abcdef
```

1.5.2 Feature Restrictions by Client Type

Operational Clients: - Can optimize ancillary services (aFRR, mFRR) - Support both 15-minute and 1-hour resolution - Maximum 296 intervals per optimization - Access to all endpoints

Investment Clients: - Cannot optimize ancillary services (field must be omitted) - Only 1-hour resolution supported - Maximum 100,000 intervals per optimization - Access to `/device-planning` endpoint only - Binary variables automatically relaxed to continuous

2. Core Data Models

2.1 Time Series Representation

All time-varying data uses explicit period boundaries:

```
{
  "period_start": "2025-11-06T00:00:00+01:00",
  "period_end": "2025-11-06T00:15:00+01:00",
  "resolution": "15min",
  "values": [30.0, 32.5, 28.0, ...]
}
```

Notes: - Timestamps use ISO 8601 format with timezone (must be Europe/Prague) - `resolution` is for check (derived from start/end and array length) - Array length determines actual resolution

2.2 Device Schema (Nested Structure)

All devices follow this base structure:

```
{
  "name": "string",            // Unique device identifier
  "type": "string",            // Device type: "battery", "chp", "heat_accumulator", etc.
  "properties": {},            // Type-specific properties (see device types below)
  "schedule": {},              // Operational schedule constraints (optional)
  "ancillary_services": {}     // Ancillary service capabilities (optional)
}
```

2.2.1 Schedule Constraints

The `schedule` object defines when and how a flexible device can operate:

```
{
  "schedule": {
    // Runtime constraints (all optional)
    "min_continuous_run_hours": 2.0,      // Minimum runtime once started
    "max_continuous_run_hours": 12.0,     // Maximum continuous operation
    "max_hours_per_day": 18.5,            // Total hours per day (can be fractional)
    "max_starts_per_day": 3,              // Maximum number of startups
    "min_downtime_hours": 1.0,            // Minimum off time between runs

    // Binary availability arrays (96 values for 15-min resolution)
    "can_run": [0, 0, 0, 1, 1, 1, ...],   // 0=cannot run, 1=can run (optional, null=always available)
    "must_run": [0, 0, 0, 0, 1, 1, ...],  // 1=must run (optional, null=no forced operation)

    // Power ranges when must_run=1 (96 values, optional)
    "min_power": [0, 0, 0, 0, 2.0, 2.5, ...],  // Minimum power when must_run=1 (MW)
    "max_power": [0, 0, 0, 0, 5.0, 5.0, ...]   // Maximum power when must_run=1 (MW)
  }
}
```

Notes: - All time-based arrays have 96 values (15-minute resolution) or 24 values (1-hour resolution) - `can_run` and `must_run` are binary (0 or 1) - `can_run` and `must_run` must be consistent for all periods (i.e., must_run=1 implies can_run=1) - If `must_run=1` for a period, device must operate within `[min_power, max_power]` range - If both `can_run` and `must_run` are null, device is fully flexible

2.2.2 Ancillary Services

⚠ **OPERATIONAL CLIENTS ONLY** - Investment clients must omit this field.

The `ancillary_services` object defines capabilities for frequency regulation markets:

```
{
  "ancillary_services": {
    "afrr_plus": {
      "can_provide": [0, 0, 1, 1, 1, 0],         // Binary (0/1), 6 values (4-hour blocks)
      "expected_activation_profit": [50.0, 52.0, 48.0, ...] // EUR/MWh, 6 values (4-hour blocks)
    },
    "afrr_minus": {
      "can_provide": [1, 1, 1, 1, 1, 1],
      "expected_activation_profit": [45.0, 47.0, 43.0, ...]
    },
    "mfrr_plus": {
      "can_provide": [0, 0, 0, 0, 0, 0],
      "expected_activation_profit": null  // Can be null if can_provide is all zeros
    },
    "mfrr_minus": null  // Entire service can be null if not applicable
  }
}
```

Notes: - `can_provide`: 6 values for 4-hour blocks (00-04, 04-08, 08-12, 12-16, 16-20, 20-24) - `expected_activation_profit`: 6 values (4-hour blocks) (EUR/MW/h activation profit when reserved) - Services: `afrr_plus`, `afrr_minus`, `mfrr_plus`, `mfrr_minus` - Only devices capable of providing ancillary services need this object

Investment Client Restriction:

Investment clients will receive a validation error if this field is present:

```
{
  "error": {
    "code": "forbidden_feature",
    "message": "Ancillary services not available for investment clients",
    "field": "devices[0].ancillary_services"
  }
}
```

2.3 Device Types

2.3.1 Battery

```
{
  "name": "Battery1",
  "type": "battery",

  "properties": {
    "capacity": 10.0,           // Energy capacity (MWh)
    "max_power": 5.0,           // Power rating (MW) for charge/discharge
    "efficiency": 0.90,         // Round-trip efficiency (0-1)
    "initial_soc": 0.5          // Initial state of charge (0-1)
  },

  "schedule": {
    "can_run": null,            // Batteries typically always available
    "must_run": null
  },

  "ancillary_services": {
    "afrr_plus": {
      "can_provide": [1, 1, 1, 1, 1, 1],
      "expected_activation_profit": [80.0, 85.0, 88.0, 90.0, 87.0, 82.0]  // 6 values (4-hour blocks)
    },
    "afrr_minus": {
      "can_provide": [1, 1, 1, 1, 1, 1],
      "expected_activation_profit": [75.0, 80.0, 83.0, 85.0, 82.0, 78.0]  // 6 values (4-hour blocks)
    }
  }
}
```

2.3.2 CHP (Combined Heat and Power)

```
{
  "name": "CHP1",
  "type": "chp",

  "properties": {
    "gas_input": 8.0,       // Gas consumption at full load (MW)
    "el_output": 3.0,       // Electricity generation at full load (MW)
    "heat_output": 4.0,     // Heat generation at full load (MW)
    "is_binary": true,      // True=on/off only, False=modulating
    "min_power": 0.5        // (optional) if power modulation is limited to e.g. 50%-100% (0-1)
  },

  "schedule": {
    "min_continuous_run_hours": 2.0,
    "max_hours_per_day": 18.0,
    "max_starts_per_day": 3,
    "can_run": [0, 0, 0, 0, 0, 0, 1, 1, 1, ...],  // Can only run 6:00-22:00
    "must_run": null
  },

  "ancillary_services": {
    "afrr_plus": {
      "can_provide": [0, 0, 1, 1, 1, 0],  // Can provide during daytime blocks
      "expected_activation_profit": [0, 0, 52.0, 55.0, 53.0, 0]  // 6 values (4-hour blocks)
    },
    "mfrr_plus": {
      "can_provide": [0, 0, 1, 1, 1, 0],
      "expected_activation_profit": [0, 0, 58.0, 62.0, 60.0, 0]  // 6 values (4-hour blocks)
    }
  }
}
```

2.3.3 Heat Accumulator

```
{
  "name": "HeatAccumulator1",
  "type": "heat_accumulator",

  "properties": {
    "capacity": 5.0,           // Thermal energy capacity (MWh)
    "max_power": 2.0,          // Charge/discharge power (MW)
    "efficiency": 0.98,        // Storage efficiency (0-1)
    "initial_soc": 0.6,        // Initial state of charge (0-1)
    "loss_rate": 0.001         // Standing losses (fraction/hour, e.g. 0.001 = 0.1%/hour)
  }
}
```

2.3.4 Photovoltaic

```
{
  "name": "PV1",
  "type": "photovoltaic",

  "properties": {
    "peak_power_mw": 5.0,
    "location": {
      "latitude": 50.0751,
      "longitude": 14.4378
    },
    "tilt": 35,                // Panel tilt angle (degrees)
    "azimuth": 180,            // Azimuth angle (degrees, 180=south)
    "generation_profile": [0, 0, 0, 0, 0.1, 0.3, ...] // Optional: normalized profile (96 values)
  },

  "schedule": {
    "can_run": [0, 0, 0, 0, 0, 0.2, 0.5, 0.8, ...],  // Generation availability (can be fractional)
    "must_run": null
  },

  "ancillary_services": null
}
```

Note: For PV, `can_run` can contain fractional values (0.0 to 1.0) representing normalized generation capability.

2.3.5 Heat Demand

```
{
  "name": "HeatDemand1",
  "type": "heat_demand",

  "properties": {
    "min_demand_profile": [2.0, 1.8, 1.5, 1.2, ...],  // MW thermal (not MWh!), 96 values
    "max_demand_profile": [2.0, 1.8, 1.5, 1.2, ...]  // MW thermal (not MWh!), 96 values
  },
}
```

2.3.6 Electricity Demand

```
{
  "name": "ElectricityDemand1",
  "type": "electricity_demand",

  "properties": {
    "max_demand_profile": [3.0, 2.8, 2.5, ...],  // MW electric (not MWh!), 96 or 24 values
    "min_demand_profile": [3.0, 2.8, 2.5, ...]   // MW electric (not MWh!), 96 or 24 values
  }
}
```

2.3.7 Electricity Import (Market Interface)

```
{
  "name": "ElectricityImport",
  "type": "electricity_import",

  "properties": {
    "price": [30.0, 28.0, 26.0, ...],      // EUR/MWh, 96 or 24 values
    "max_import": 8.0,                      // Maximum import capacity (MW)
    "max_import_unit_cost": 144.0          // Optional: reserved capacity cost (EUR/MW/year)
  }
}
```

Note: Electricity import devices represent the grid connection for buying electricity. The `price` array represents day-ahead spot prices. The optional `max_import_unit_cost` enables optimizing the reserved grid capacity (e.g., distribution fees).

2.3.8 Electricity Export (Market Interface)

```
{
  "name": "ElectricityExport",
  "type": "electricity_export",

  "properties": {
    "price": [30.0, 28.0, 26.0, ...],      // EUR/MWh, 96 or 24 values
    "max_export": 5.0,                      // Maximum export capacity (MW)
    "max_export_unit_cost": 50.0           // Optional: grid export capacity cost (EUR/MW/year)
  }
}
```

Note: Electricity export devices represent the grid connection for selling electricity back to the grid.

2.3.9 Gas Import (Market Interface)

```
{
  "name": "GasImport",
  "type": "gas_import",

  "properties": {
    "price": [25.0, 25.0, 25.0, ...],      // EUR/MWh, 96 or 24 values
    "max_import": 10.0,                     // Maximum import capacity (MW)
    "max_import_unit_cost": 100.0          // Optional: reserved capacity cost (EUR/MW/year)
  }
}
```

Note: Gas import devices represent the gas supply connection. Gas is typically only imported (consumed), not exported.

2.3.10 Heat Export (Market Interface)

```
{
  "name": "HeatExport",
  "type": "heat_export",

  "properties": {
    "price": [40.0, 40.0, 40.0, ...],      // EUR/MWh, 96 or 24 values
    "max_export": 3.0,                      // Maximum export capacity (MW)
    "max_export_unit_cost": 75.0           // Optional: district heating connection cost (EUR/MW/year)
  }
}
```

Note: Heat export devices represent connections to district heating networks where excess heat can be sold.

2.4 Site Definition

Complete site specification with all devices including market interfaces:

```
{
  "site_id": "industrial_site_1",
  "description": "Industrial facility with CHP, battery, and solar",

  "devices": [
    {/* Physical devices */},
    {/* Battery device */},
    {/* CHP device */},
    {/* Heat accumulator */},
    {/* PV device */},
    {/* Heat demand */},
    {/* Electricity demand */},

    {/* Market interface devices (grid/network connections) */},
    {/* Electricity import device - represents grid import capacity and pricing */},
    {/* Electricity export device - represents grid export capacity and pricing */},
    {/* Gas import device - represents gas supply connection */}
  ]
}
```

Note: Grid connections are represented as market interface devices in the `devices` array. This provides more flexibility as each connection can have its own pricing, capacity limits, and optimization parameters.

Example with specific devices:

```
{
  "site_id": "industrial_site_1",
  "devices": [
    {
      "name": "Battery1",
      "type": "battery",
      "properties": {
        "capacity": 10.0,
        "max_power": 5.0,
        "efficiency": 0.90,
        "initial_soc": 0.5
      }
    },
    {
      "name": "GridImport",
      "type": "electricity_import",
      "properties": {
        "price": [30.0, 28.0, ...],  // 96 or 24 values
        "max_import": 8.0
      }
    },
    {
      "name": "GridExport",
      "type": "electricity_export",
      "properties": {
        "price": [30.0, 28.0, ...],  // 96 or 24 values
        "max_export": 5.0
      }
    },
    {
      "name": "GasSupply",
      "type": "gas_import",
      "properties": {
        "price": [25.0, 25.0, ...],  // 96 or 24 values
        "max_import": 10.0
      }
    }
  ]
}
```

2.5 Market Forecasts

2.5.1 Day-Ahead Market Prices

Note: Day-ahead electricity and gas prices are now specified directly in the market interface device properties (see sections 2.3.7-2.3.10). Each market device has its own `price` array with 96 or 24 values.

Example:

```
{
  "devices": [
    {
      "name": "GridImport",
      "type": "electricity_import",
      "properties": {
        "price": [30.0, 32.5, 28.0, ...],  // EUR/MWh, 96 or 24 values
        "max_import": 8.0
      }
    }
  ]
}
```

2.5.2 Ancillary Services Forecast

```
{
  "ancillary_services": {
    "afrr_plus": {  // must be included if ancillary services are selected at any device level
      "max_accepted_price_forecast": [15.0, 18.0, 17.0, 19.0, 18.5, 16.0],  // Expected auction price
      "period_duration_hours": 4
    },
    "afrr_minus": {
      "max_accepted_price_forecast": [12.0, 14.0, 13.5, 15.0, 14.0, 13.0],
      "period_duration_hours": 4
    },
    "mfrr_plus": {
      "max_accepted_price_forecast": [20.0, 23.0, 22.0, 24.0, 23.5, 21.0],
      "period_duration_hours": 4
    },
    "mfrr_minus": {
      "max_accepted_price_forecast": [18.0, 20.0, 19.5, 21.0, 20.5, 19.0],
      "period_duration_hours": 4
    }
  }
}
```

2.5.3 Opportunity Costs

Opportunity costs can be specified in multiple ways:

Option 1: Global constant

```
{
  "opportunity_costs": {
    "global": 5.0  // EUR/MWh for all times and devices
  }
}
```

Option 2: Time-varying (standard intervals)

```
{
  "opportunity_costs": {
    "by_interval": [3.0, 4.0, 5.0, 6.0, ...]  // EUR/MWh, 96 values (15-min) (or 24 values in 1-regime)
  }
}
```

Option 3: Device-specific with intervals

```
{
  "opportunity_costs": {
    "by_device": {
      "Battery1": [3.0, 4.0, 5.0, 6.0, ...],  // EUR/MWh, 96 values (15-min) (or 24 values in 1-regime)
      "CHP1": [3.0, 4.0, 5.0, 6.0, ...]      // EUR/MWh, 96 values (15-min) (or 24 values in 1-regime)
    }
  }
}
```

2.6 Output: Bid Curves

2.6.1 Day-Ahead Bids

```
{
  "da_bids": [
    {
      "period_start": "2025-11-06T00:00:00+01:00",
      "period_end": "2025-11-06T00:15:00+01:00",
      "bids": [
        {"price": 25.0, "quantity_mw": 2.0},   // Willing to export 2 MW at 25 EUR/MWh
        {"price": 30.0, "quantity_mw": 4.0},   // Or export 4 MW at 30 EUR/MWh
        {"price": 35.0, "quantity_mw": 5.0}    // Or export 5 MW at 35 EUR/MWh
      ]
    },
    {
      "period_start": "2025-11-06T00:15:00+01:00",
      "period_end": "2025-11-06T00:30:00+01:00",
      "bids": [
        {"price": 28.0, "quantity_mw": -3.0},  // Negative = import (buy 3 MW at 28 EUR/MWh)
        {"price": 26.0, "quantity_mw": -5.0}   // Or buy 5 MW at 26 EUR/MWh
      ]
    }
    // ... 96 periods total (15-min resolution) or 24 periods (1-hour resolution)
  ]
}
```

Note: Positive quantity = export (sell), negative quantity = import (buy)

2.6.2 Ancillary Services Bids

```
{
  "ancillary_bids": {
    "afrr_plus": [
      {
        "period_start": "2025-11-06T00:00:00+01:00",
        "period_end": "2025-11-06T04:00:00+01:00",
        "bids": [
          {"price": 15.0, "capacity_mw": 2.0},   // Offer 2 MW reserve at 15 EUR/MW
          {"price": 18.0, "capacity_mw": 3.0},   // Or 3 MW at 18 EUR/MW
          {"price": 22.0, "capacity_mw": 4.0}    // Or 4 MW at 22 EUR/MW
        ]
      },
      {
        "period_start": "2025-11-06T04:00:00+01:00",
        "period_end": "2025-11-06T08:00:00+01:00",
        "bids": [
          {"price": 16.0, "capacity_mw": 2.0},
          {"price": 19.0, "capacity_mw": 3.5},
          {"price": 23.0, "capacity_mw": 4.5}
        ]
      }
      // ... 6 periods total (4-hour blocks)
    ],
    "afrr_minus": [
      // Same structure
    ],
    "mfrr_plus": [
      // Same structure
    ],
    "mfrr_minus": [
      // Same structure
    ]
  }
}
```

2.7 Output: Device Schedules

Device schedules are organized by site. When multiple sites are optimized together, each site's devices and grid flows are reported separately under the site_id key.

```
{
  "sites": {
    "industrial_site_1": {
      "device_schedules": {
        "Battery1": {
          "flows": {
            "electricity": [2.0, -3.5, 0.0, ...],  // MW (not MWh!), 96 or 24 values (positive=discharge, negative=charge)
          },
          "soc": [0.5, 0.48, 0.51, ...],           // State of charge (0-1), 96 or 24 values
          "ancillary_reservations": {
            "afrr_plus": [0, 0, 2.0, 2.0, ...],    // Reserved capacity (MW), 96 or 24 values
            "afrr_minus": [1.5, 1.5, 0, 0, ...]
          }
        },
        "CHP1": {
          "flows": {
            "gas": [-8.0, -8.0, 0.0, ...],         // MW (not MWh!), negative = consumption, 96 or 24 values
            "electricity": [3.0, 3.0, 0.0, ...],   // MW (not MWh!), positive = generation, 96 or 24 values
            "heat": [4.0, 4.0, 0.0, ...],          // MW (not MWh!), 96 or 24 values
          },
          "binary_status": [1, 1, 0, ...],         // 96 or 24 values
          "ancillary_reservations": {
            "mfrr_plus": [0, 0, 1.5, 1.5, ...]     // MW, 96 or 24 values
          }
        }
        // ... all devices for this site
      },
      "grid_flows": {
        "import": [0, 3.5, 2.0, ...],              // MW imported from grid, 96 or 24 values
        "export": [2.0, 0, 0, ...]                 // MW exported to grid, 96 or 24 values
      }
    },
    "residential_site_1": {
      "device_schedules": {
        // ... devices for second site
      },
      "grid_flows": {
        // ... grid flows for second site
      }
    }
  },

  "summary": {
    "total_da_revenue": 12500.50,              // EUR (aggregated across all sites)
    "total_ancillary_revenue": 3200.00,        // EUR (capacity payments, aggregated)
    "total_cost": 8500.00,                     // EUR (fuel, grid fees, aggregated)
    "expected_profit": 7200.50,                // EUR (revenue - cost, aggregated)
    "solver_status": "optimal",
    "solve_time_seconds": 2.5,
    "sites_count": 2                           // Number of sites optimized
  }
}
```

2.8 Locked Reservations (Use Case 2)

For device planning with pre-committed ancillary services:

```
{
  "locked_reservations": {
    "afrr_plus": {
      "capacity": [0, 0, 3.0, 2.5, 0, 0],  // MW for each 4-hour block (00-04, 04-08, 08-12, 12-16, 16-20, 20-24)
      "devices": ["Battery1", "CHP1"]       // Devices providing this reservation
    },
    "afrr_minus": {
      "capacity": [0, 0, 0, 0, 0, 0],
      "devices": []
    },
    "mfrr_plus": {
      "capacity": [0, 0, 0, 0, 2.0, 0],     // 2 MW reserved in block 4 (16:00-20:00)
      "devices": ["CHP1"]
    },
    "mfrr_minus": {
      "capacity": [0, 0, 0, 0, 0, 0],
      "devices": []
    }
  }
}
```

Notes: - `capacity`: Array of 6 values (MW) for 4-hour blocks aligned with ancillary service periods - `devices`: List of device names that must provide this reserved capacity - Blocks: [00-04, 04-08, 08-12, 12-16, 16-20, 20-24]

3. API Endpoints

3.1 Use Case 1: Optimal Bidding

Generate optimal bid curves for DA market and ancillary services.

Client Type: ⚠ Operational Only

Endpoint: `POST /api/v1/jobs/optimal-bidding`

Access Control: - ✅ Operational clients (`op_*`): Full access - ❌ Investment clients (`inv_*`): 403 Forbidden

Request Body:

```
{
  "sites": [
    {
      /* Site definition (see 2.4) */
    },
    {
      /* Additional sites (optional) */
    }
  ],
  "timespan": {
    "period_start": "2025-11-06T00:00:00+01:00",
    "period_end": "2025-11-07T00:00:00+01:00",
    "resolution": "15min"
  },
  "market_forecasts": {
    "da_price_forecast": [30.0, 32.5, ...],
    "ancillary_services": {
      /* Ancillary services forecast (see 2.5.2) */
    }
  },
  "opportunity_costs": {
    /* Opportunity costs (see 2.5.3) */
  },
  "optimization_config": {
    "max_bid_steps": 10,                  // Maximum steps in bid curves
    "objective": "expected_profit",       // Optimization objective
    "time_limit_seconds": 300             // Solver timeout
  }
}
```

Response (202 Accepted):

```
{
  "job_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "pending",
  "created_at": "2025-11-06T10:30:00+01:00",
  "message": "Optimization job created successfully"
}
```

3.2 Use Case 2: Device Planning

Create operational schedule with locked ancillary reservations.

Client Type: ✅ Operational or Investment

Endpoint: `POST /api/v1/jobs/device-planning`

Access Control: - ✅ Operational clients (`op_*`): Max 296 intervals, ANS features available - ✅ Investment clients (`inv_*`): Max 100,000 intervals, NO ANS features

Validation Rules by Client Type:

Field	Operational	Investment
`timespan.resolution`	"15min" or "1h"	"1h" only
`timespan` interval count	≤ 296	≤ 100,000
`devices[].ancillary_services`	Allowed	**Forbidden** (must be null/omitted)
`devices[].schedule`	Allowed	Allowed
`locked_reservations`	Allowed	**Forbidden** (must be null/omitted)
`optimization_config.time_limit_seconds`	≤ 300	≤ 3600

Request Body:

```
{
  "sites": [
    {
      /* Site definition (see 2.4) */
    },
    {
      /* Additional sites (optional) */
    }
  ],
  "timespan": {
    "period_start": "2025-11-06T00:00:00+01:00",
    "period_end": "2025-11-07T00:00:00+01:00",
    "resolution": "15min"
  },
  "market_forecasts": {
    "da_price_forecast": [30.0, 32.5, ...]
  },
  "locked_reservations": {
    /* Pre-committed ancillary services (see 2.8) */
  },
  "optimization_config": {
    "objective": "maximize_da_revenue",
    "time_limit_seconds": 300
  }
}
```

Response (202 Accepted):

```
{
  "job_id": "660e8400-e29b-41d4-a716-446655440001",
  "status": "pending",
  "created_at": "2025-11-06T10:35:00+01:00",
  "message": "Planning job created successfully"
}
```

3.3 Get Job Status and Results

Endpoint: `GET /api/v1/jobs/{job_id}`

Response (Job Running - 200 OK):

```
{
  "job_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "running",
  "created_at": "2025-11-06T10:30:00+01:00",
  "started_at": "2025-11-06T10:30:05+01:00",
  "progress": 45,  // Percentage (optional)
  "message": "Optimization in progress"
}
```

Response (Job Completed - 200 OK):

For Use Case 1:

```
{
  "job_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "completed",
  "created_at": "2025-11-06T10:30:00+01:00",
  "started_at": "2025-11-06T10:30:05+01:00",
  "completed_at": "2025-11-06T10:30:15+01:00",
  "result": {
    "da_bids": [
      /* DA bid curves (see 2.6.1) */
    ],
    "ancillary_bids": {
      /* Ancillary bid curves (see 2.6.2) */
    },
    "sites": {
      /* Site-organized device schedules (see 2.7) */
    },
    "summary": {
      /* Optimization summary */
    }
  }
}
```

For Use Case 2:

```
{
  "job_id": "660e8400-e29b-41d4-a716-446655440001",
  "status": "completed",
  "result": {
    "sites": {
      /* Site-organized device schedules and grid flows (see 2.7) */
    },
    "summary": {
      /* Optimization summary */
    }
  }
}
```

Response (Job Failed - 200 OK):

```
{
  "job_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "failed",
  "created_at": "2025-11-06T10:30:00+01:00",
  "started_at": "2025-11-06T10:30:05+01:00",
  "failed_at": "2025-11-06T10:30:12+01:00",
  "error": {
    "code": "infeasible",
    "message": "Optimization problem is infeasible. Cannot satisfy all constraints.",
    "details": {
      "conflicting_constraints": [
        "HeatDemand1: must_run requirement conflicts with heat generation capacity",
        "Battery1: SOC constraints violated in periods 45-60"
      ]
    }
  }
}
```

Response (Job Not Found - 404 Not Found):

```
{
  "error": {
    "code": "job_not_found",
    "message": "Job with ID 550e8400-e29b-41d4-a716-446655440000 not found"
  }
}
```

3.4 Cancel/Delete Job

Endpoint: `DELETE /api/v1/jobs/{job_id}`

Response (200 OK):

```
{
  "job_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "cancelled",
  "message": "Job cancelled successfully"
}
```

Response (Job Already Completed - 409 Conflict):

```
{
  "error": {
    "code": "cannot_cancel",
    "message": "Cannot cancel job in status 'completed'"
  }
}
```

4. Error Handling

4.1 Validation Errors (400 Bad Request)

```
{
  "error": {
    "code": "validation_error",
    "message": "Request validation failed",
    "details": [
      {
        "field": "site.devices[0].properties.capacity_mwh",
        "message": "Must be a positive number"
      },
      {
        "field": "timespan.period_start",
        "message": "Must be a valid ISO 8601 datetime with Europe/Prague timezone"
      },
      {
        "field": "site.devices[0].schedule.can_run",
        "message": "Array length must be 96 (15-min) or 24 (1-hour) matching timespan resolution"
      },
      {
        "field": "site.devices[1].ancillary_services.afrr_plus.expected_activation_profit",
        "message": "Must have 6 values (4-hour blocks)"
      }
    ]
  }
}
```

4.2 Optimization Errors

Infeasible Problem:

```
{
  "error": {
    "code": "infeasible",
    "message": "No feasible solution exists for the given constraints",
    "details": {
      "conflicting_constraints": ["..."]
    }
  }
}
```

Solver Timeout:

```
{
  "error": {
    "code": "timeout",
    "message": "Solver exceeded time limit of 300 seconds",
    "details": {
      "best_solution_gap": 5.2  // Percentage gap from optimal
    }
  }
}
```

Unbounded Problem:

```
{
  "error": {
    "code": "unbounded",
    "message": "Optimization problem is unbounded (objective can increase indefinitely)",
    "details": {}
  }
}
```

4.3 Server Errors (500 Internal Server Error)

```
{
  "error": {
    "code": "internal_error",
    "message": "An unexpected error occurred",
    "request_id": "req_123456789"  // For support/debugging
  }
}
```

4.4 Access Control Errors (403 Forbidden)

4.4.1 Wrong Endpoint for Client Type

Investment clients attempting to access optimal bidding:

```
{
  "error": {
    "code": "forbidden_client_type",
    "message": "Investment clients cannot access optimal-bidding endpoint",
    "allowed_endpoints": ["/api/v1/jobs/device-planning"],
    "client_type": "investment"
  }
}
```

4.4.2 Forbidden Feature Usage

Investment clients using ancillary services:

```
{
  "error": {
    "code": "forbidden_feature",
    "message": "Ancillary services not available for investment clients",
    "field": "devices[0].ancillary_services",
    "client_type": "investment"
  }
}
```

Investment clients using locked reservations:

```
{
  "error": {
    "code": "forbidden_feature",
    "message": "Locked reservations not available for investment clients",
    "field": "locked_reservations",
    "client_type": "investment"
  }
}
```

4.4.3 Limit Exceeded

Operational client exceeding interval limit:

```
{
  "error": {
    "code": "limit_exceeded",
    "message": "Operational clients limited to 296 intervals",
    "requested": 8760,
    "max_allowed": 296,
    "suggestion": "Use investment client (inv_*) for long-term planning horizons"
  }
}
```

Investment client exceeding interval limit:

```
{
  "error": {
    "code": "limit_exceeded",
    "message": "Investment clients limited to 100,000 intervals",
    "requested": 150000,
    "max_allowed": 100000
  }
}
```

Investment client using 15-minute resolution:

```
{
  "error": {
    "code": "invalid_resolution",
    "message": "Investment clients only support 1-hour resolution",
    "requested": "15min",
    "allowed": ["1h"],
    "client_type": "investment"
  }
}
```

5. Complete Examples

5.1 Example: Optimal Bidding Request

```
{
  "sites": [
    {
      "site_id": "example_site_1",
      "devices": [
      {
        "name": "Battery1",
        "type": "battery",
        "properties": {
          "capacity": 10.0,
          "max_power": 5.0,
          "efficiency": 0.90,
          "initial_soc": 0.5
        },
        "schedule": null,
        "ancillary_services": {
          "afrr_plus": {
            "can_provide": [1, 1, 1, 1, 1, 1],
            "expected_activation_profit": [80, 85, 88, 90, 87, 82]
          },
          "afrr_minus": {
            "can_provide": [1, 1, 1, 1, 1, 1],
            "expected_activation_profit": [75, 80, 83, 85, 82, 78]
          }
        }
      },
      {
        "name": "CHP1",
        "type": "chp",
        "properties": {
          "gas_input": 8.0,
          "el_output": 3.0,
          "heat_output": 4.0,
          "is_binary": true,
          "min_power": 0.5
        },
        "schedule": {
          "min_continuous_run_hours": 2.0,
          "max_hours_per_day": 18.0,
          "max_starts_per_day": 3,
          "can_run": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
          "must_run": null
        },
        "ancillary_services": {
          "mfrr_plus": {
            "can_provide": [0, 1, 1, 1, 1, 0],
            "expected_activation_profit": [0, 58, 62, 60, 55, 0]
          }
        }
      },
      {
        "name": "HeatAccumulator1",
        "type": "heat_accumulator",
        "properties": {
          "capacity": 5.0,
          "max_power": 2.0,
          "efficiency": 0.98,
          "initial_soc": 0.6,
          "loss_rate": 0.001
        }
      },
      {
        "name": "HeatDemand1",
        "type": "heat_demand",
        "properties": {
          "min_demand_profile": [1.5,1.4,1.3,1.2,1.2,1.3,1.5,1.8,2.0,2.2,2.3,2.4,2.4,2.3,2.2,2.1,2.0,2.2,2.5,2.8,2.6,2.2,1.9,1.7,1.5,1
          "max_demand_profile": [1.5,1.4,1.3,1.2,1.2,1.3,1.5,1.8,2.0,2.2,2.3,2.4,2.4,2.3,2.2,2.1,2.0,2.2,2.5,2.8,2.6,2.2,1.9,1.7,1.5,1
        }
      },
      {
        "name": "GridImport",
        "type": "electricity_import",
        "properties": {
          "price": [30,28,26,24,22,23,25,28,32,38,42,45,48,50,52,50,48,45,42,55,65,70,68,60,55,50,45,40,35,33,32,35,38,42,46,50,53,55,
          "max_import": 8.0
        }
      },
      {
        "name": "GridExport",
        "type": "electricity_export",
        "properties": {
          "price": [30,28,26,24,22,23,25,28,32,38,42,45,48,50,52,50,48,45,42,55,65,70,68,60,55,50,45,40,35,33,32,35,38,42,46,50,53,55,
          "max_export": 5.0
        }
      },
      {
        "name": "GasSupply",
        "type": "gas_import",
        "properties": {
          "price": [25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0,
          "max_import": 10.0
        }
      }
    ]
    }
  ],
  "timespan": {
    "period_start": "2025-11-06T00:00:00+01:00",
    "period_end": "2025-11-07T00:00:00+01:00",
    "resolution": "15min"
  },
  "market_forecasts": {
    "ancillary_services": {
      "afrr_plus": {
        "max_accepted_price_forecast": [15.0, 18.0, 22.0, 25.0, 23.0, 16.0],
        "period_duration_hours": 4
      },
      "afrr_minus": {
        "max_accepted_price_forecast": [12.0, 15.0, 18.0, 20.0, 19.0, 13.0],
        "period_duration_hours": 4
      },
      "mfrr_plus": {
        "max_accepted_price_forecast": [20.0, 24.0, 28.0, 32.0, 30.0, 22.0],
        "period_duration_hours": 4
      },
      "mfrr_minus": {
        "max_accepted_price_forecast": [18.0, 22.0, 26.0, 29.0, 27.0, 20.0],
        "period_duration_hours": 4
      }
    }
  },
  "opportunity_costs": {
    "global": 5.0
  },
  "optimization_config": {
    "max_bid_steps": 10,
    "objective": "expected_profit",
    "time_limit_seconds": 300
  }
}
```

5.2 Example: Device Planning Request

Note: Day-ahead prices are now included in market device properties (see section 5.1 for full site definition with market devices).

```
{
  "sites": [
    {
      /* Same site definition as in section 5.1 - includes market devices with prices */
    }
  ],
  "timespan": {
    "period_start": "2025-11-06T00:00:00+01:00",
    "period_end": "2025-11-07T00:00:00+01:00",
    "resolution": "15min"
  },
  "locked_reservations": {
    "afrr_plus": {
      "capacity": [0, 0, 3.0, 2.5, 0, 0],
      "devices": ["Battery1"]
    },
    "mfrr_plus": {
      "capacity": [0, 0, 0, 0, 2.0, 0],
      "devices": ["CHP1"]
    }
  },
  "optimization_config": {
    "objective": "maximize_da_revenue",
    "time_limit_seconds": 300
  }
}
```

6. Implementation Notes

6.1 Compression

Enable gzip compression in FastAPI:

```
from fastapi import FastAPI
from fastapi.middleware.gzip import GZipMiddleware

app = FastAPI()
app.add_middleware(GZipMiddleware, minimum_size=1000)  # Compress responses > 1KB
```

6.2 Job Storage

- Use Redis for job queue and results
- TTL for completed jobs (e.g., 24 hours)
- Store large results in object storage (S3, MinIO) with references in job metadata

6.3 Validation

Use Pydantic models for request/response validation:

```python
from pydantic import BaseModel, Field, validator, root_validator
from typing import List, Optional
from zoneinfo import ZoneInfo

class DeviceSchedule(BaseModel):
    min_continuous_run_hours: Optional[float] = Field(None, ge=0)
    max_hours_per_day: Optional[float] = Field(None, ge=0, le=24)
    can_run: Optional[List[int]] = None
    must_run: Optional[List[int]] = None
    min_power: Optional[List[float]] = None
    max_power: Optional[List[float]] = None

    @validator('can_run', 'must_run')
    def validate_binary_array(cls, v):
        if v is not None:
            # Must be either 96 (15-min) or 24 (1-hour)
            if len(v) not in [24, 96]:
                raise ValueError('Array length must be 96 (15-min) or 24 (1-hour)')
            if not all(x in [0, 1] for x in v):
                raise ValueError('Arrays must contain only 0 or 1')
        return v

    @root_validator
    def validate_consistency(cls, values):
        can_run = values.get('can_run')
        must_run = values.get('must_run')
        min_power = values.get('min_power')
        max_power = values.get('max_power')

        # Check must_run implies can_run
        if can_run is not None and must_run is not None:
            for i, (c, m) in enumerate(zip(can_run, must_run)):
                if m == 1 and c == 0:
                    raise ValueError(f'must_run=1 requires can_run=1 at index {i}')

        # Check power arrays length matches must_run
        if must_run is not None:
            expected_len = len(must_run)
            if min_power is not None and len(min_power) != expected_len:
                raise ValueError(f'min_power length must match must_run length ({expected_len})')
            if max_power is not None and len(max_power) != expected_len:
                raise ValueError(f'max_power length must match must_run length ({expected_len})')

        return values

class AncillaryServiceCapability(BaseModel):
    can_provide: List[int] = Field(..., min_items=6, max_items=6)
    expected_activation_profit: List[float] = Field(..., min_items=6, max_items=6)

    @validator('can_provide')
    def validate_binary(cls, v):
        if not all(x in [0, 1] for x in v):
            raise ValueError('can_provide must contain only 0 or 1')
        return v

class TimeSpan(BaseModel):
    period_start: str
    period_end: str
    resolution: str

    @validator('period_start', 'period_end')
    def validate_timezone(cls, v):
        # Ensure timezone is Europe/Prague
        try:
            dt = datetime.fromisoformat(v)
            if dt.tzinfo is None or dt.tzinfo != ZoneInfo("Europe/Prague"):
                raise ValueError('Timezone must be Europe/Prague')
        except Exception as e:
            raise ValueError(f'Invalid datetime: {e}')
        return v
```

6.4 API Versioning

- Use URL path versioning: `/api/v1/`, `/api/v2/`
- Maintain backward compatibility within major version
- Document breaking changes clearly

6.5 Rate Limiting

Consider rate limiting for production:

```
from slowapi import Limiter
from slowapi.util import get_remote_address

limiter = Limiter(key_func=get_remote_address)
app.state.limiter = limiter

@app.post("/api/v1/jobs/optimal-bidding")
@limiter.limit("10/minute")
async def create_bidding_job(request: Request):
    ...
```

Document Status: Draft for review **Next Steps:** Review with stakeholders, refine schemas, begin implementation