AMES Design Notes

Sean L. Mooney, Dheepak Krishnamurthy

April 11, 2017

1 Load Profile Scenarios

Module to support providing multiple load profile scenarios for repeated batch runs of AMES (V4.0).

1.1 Load Profile Scenario Classes

1.1.1 Definitions

LoadProfile	A $m \times (n * 24)$ matrix, of m LSE's and n buses. A load profile repesents a full day. There is a set of entries for each hour for each bus/lse combination.
LoadScenario	A list of load profiles, indexed by day. There is one profile for each day from 0 to Day_{max} of a simulation.
Profile User	Any object which may use a LoadScenario in the AMES (V4.0) code base.
ScenarioProvider	Any class which is capable of providing one or more LoadScenarios for a AMES (V4.0) simulation.

1.1.2 Sample LoadProfile File

One way to respresent a single load profile is with a comma seperated value file(CSV) with headers:

hour, lse-id, bus-id, value

Note, this table does not literally match the defination for a LoadProfile, but it does represent the same information. The input file format is flexible, relative to the abstraction of a LoadProfile.

1.1.3 LoadScenario File Reader

Each entry in the load profile scenario is indexed by a year, month, day and hour. An individual load profile stores the data for a single day's worth of hourly load profile information for all of the LSE's. In order to interface cleanly with the existing AMES archetecture, which only has the concept of days, each entry (year, month, day) must be converted to an absolute day number, starting from the epoch (year = 1, month = 1, day = 1).

Converting to absolute days Suppose we have a year, y, month, m, and day, d. Then the formula to get the absolute day is:

$$dIndex = (y - 1) * 365 + (\texttt{daysInPrevMonth}(m)) + d$$

The formula assumes there are always 365 days per year. The `daysInPrevMonth` function computes or looks up the number of days in each month that have passed before month m. e.g.

$$\texttt{daysInPrevMonth}(1) = 0$$

,

$$\texttt{daysInPrevMonth}(2) = 31$$

,

$$\texttt{daysInPrevMonth}(6) = 151$$

For example, $y = 1, m = 2, d = 3$ corresponds to absolute day 34 and $y = 2, m = 1, d = 1$ corresponds to absolute day 366.

Data validation Validation conditions.

- $y > 0$
- $m > 0 \wedge m \leq 12$
- $d > 0 \wedge d \leq$ number of days for month m
- There is an absolute day number for each day between 1 the total number of days in the simulation
- Each absolute day number occures *exactly* once.

The data file reader is responsible for validating each scenario file.

1.2 File Specication

1.3 Load Scenario Classes

A `LoadScenarioProvider` is composed of multiple `LoadScenario`s. A `LoadScenario` is composed of multiple `LoadProfile`s. A single `LoadProfile` repesents the load profile for a single day.

The `LoadScenarioProvider` is an interface that enables the rest of the 'world' to interact with individual load scenarios.

1.3.1 LoadScenarioProvider Interface

Proposed interface for the LoadScenarioProvider.

```
package loadscenario;
public interface LoadScenarioProvider {
    /**
     * Max number of days in the current load profile.
     */
    int getDayMax();
    /**
     * Number of LSE's in the current load profile
     */
```

Load Case Control File Specification

Scenario File Specification

scen-file ::= cf-decl
sn-delc
scen-desc
load-profile*
cf-decl ::= Control File : file-name
sn-decl ::= Scenario Number : pos-int
scen-desc ::= Day Hour zone-decl*
load-profile ::= pos-int pos-intpos-int*
zone-decl ::= Zpos-int

```
    int getNumLSE();
    /**
     * Number of LoadProfileScenarios
     */
    int getNumScenarios();
    /**
     * Get the LoadProfile for day 'day' from the current scenario
     */
    LoadProfile getLoadProfile(int day);
    /**
     * Select a scenarios for use.
     */
    void selectScenario(int sNum);
    /**
     * Get the probablility for chosing each scenario
     * in the provider
     */
    double[] getScenarioProbs();
}
```

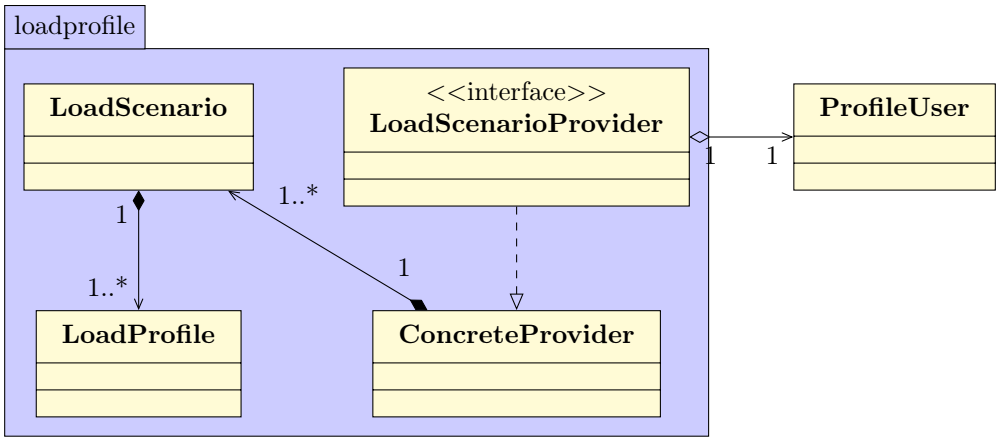

Figure 1: Load Profile Support Class Diagram

1.4 LoadScenarioProvider Usage

A load scenario represents should be used for an entire simulation, for Day_0 to Day_{max} or until some other stopping condition occurs. A scenario should only be selected when RePast's 'Start' hook method is called.

For example:

```
public void StartSimulation(){
    double[] loadProb = loadScenarioProvider.getScenarioProbs();
    loadScenarioProvider.selectScenario(
        chooseUniformily(loadProb)
    );
}
```

would pseudo-randomly chose a scenario number and tell the `loadScenarioProvider` to use that scenario for all subsequent actions (e.g. getting the load profile for a single day).

1.5 Accounting - Profits/Loss

Both LSE's and GenCo's shall compute profits and losses for both the day-ahead and real-time markets. Each class can compute this information using an instance of the `Ledger` class.

```
public class Ledger{
  private SimpleLedger rtLedger;
  private SimpleLedger daLedger;

  /*
   * Methods to interact with rtLedger
   * delegate to the common methods with the rtLedger object
   */
  /*
   * Methods to interact with daLedger
   * delegate to the common method with the daLedger
```

```
   */

  //Methods to interact with a single ledger

  private static SimpleLedger{
    //...keep a single ledger here...
  }
}
```

2 Market Operation

2.1 Sequence Diagrams

The sequence diagram in Figure 2 shows the sequence of actions for simulating market operation on day, d and hour, h. In this diagram, sections related to the day-ahead market have been shaded green and sections related to the real-time market have been shaded blue.

Figure 3 details the actions for solving the optimization problem at hour 12. The interactions involved in accessing data have been elided for clarity. In the full sorce code, the GenCo supply offers and LSE load profiles are accessed and stored in the approriate fields.

Figure 4 shows the seqeunce for operating the real time market.

Figure 5 show the sequence of methods for solving the optimization problem for the realtime market. As with Figure 3 the data access operations have been elided.

Calling the external scuc requires the following parameters:

- Save file name
- Location of model file
- Solver to user (e.g. cplex or glpk)
- Location of pyomo, runef
- Arbitrary options to pass through to the pyome/runef or even further through to the solver.
- name of python instance used to run the SCED script. (e.g. coopr_python)

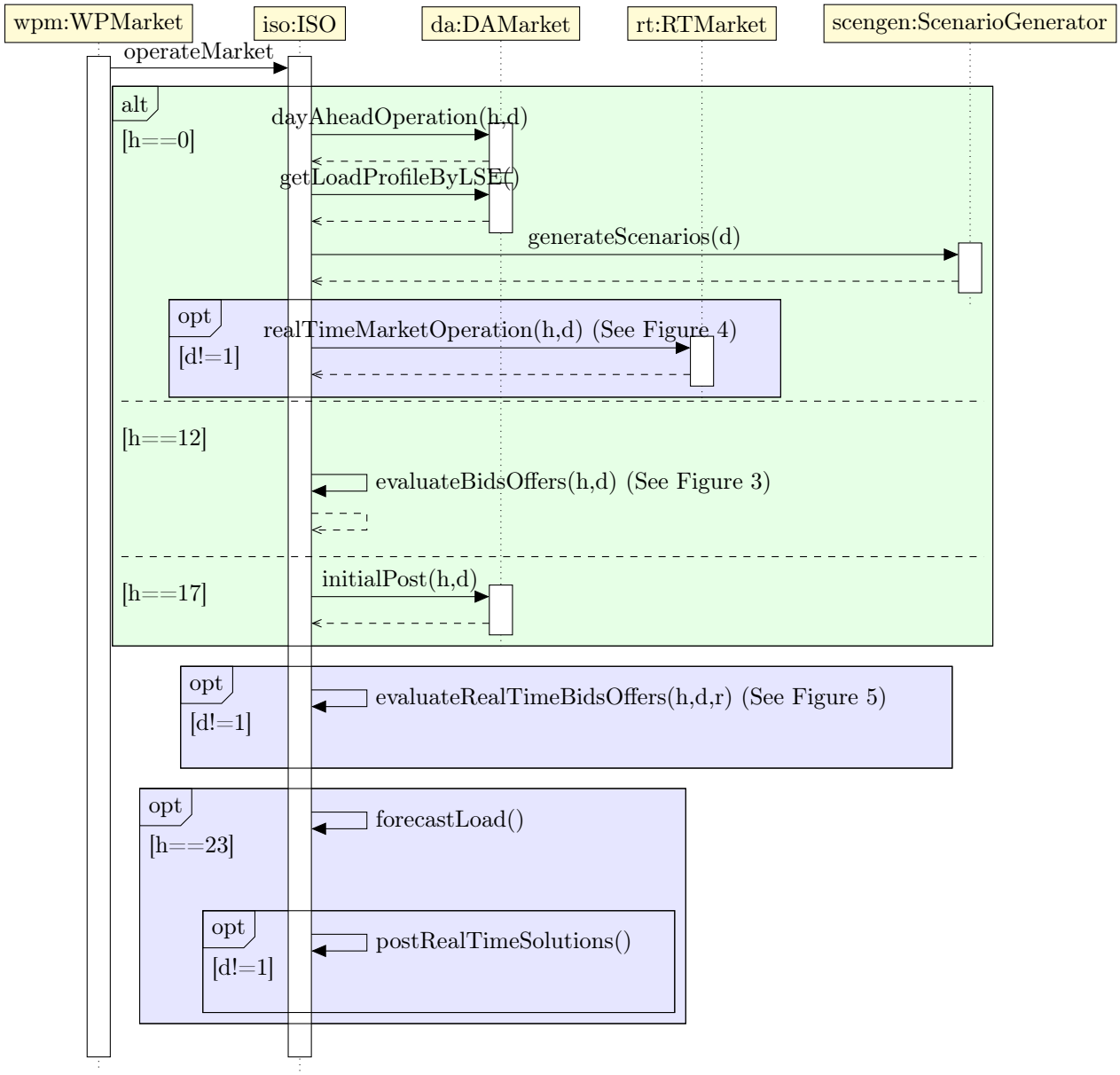

Figure 2: Sequence Diagram for day=d, hour=h

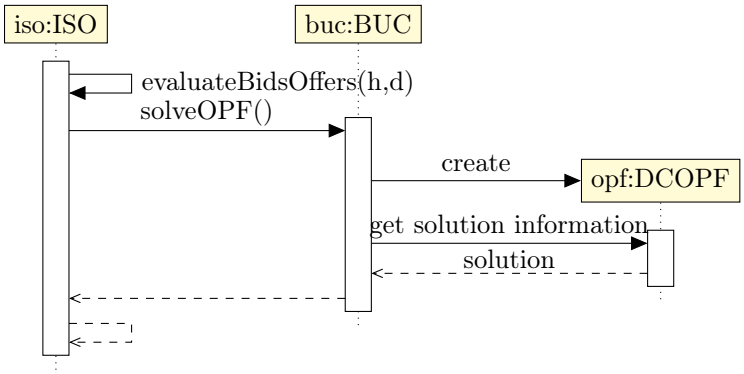

Figure 3: Sequence Diagram for day=d, hour=12

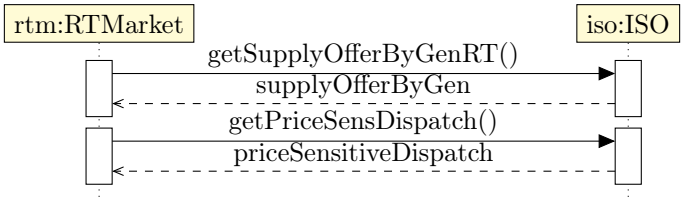

Figure 4: Operate Realtime Market

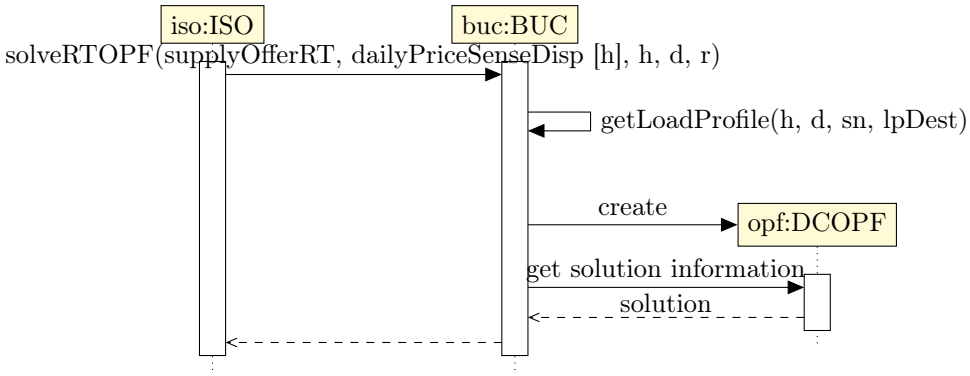

Figure 5: Evaluate Realtime Bids