**COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME**

**Call FP7-ICT-2011-7**

Project Acronym: **Cassandra**

Project Number: **288429**

Project Type: **COLLABORATIVE PROJECT: Small or medium scale focused research project (STREP)**

Project Full Title: **A multivariate platform for assessing the impact of strategic decisions in electrical power systems**

Work programme topic addressed: **ICT-2011.6.2 ICT systems for energy efficiency**

**D2.5 Cassandra software library**

|  |  |
| --- | --- |
| **Nature:** | **P** |
| **Dissemination Level:** | **PU** |
| **Version #:** | **1.0** |
| **Delivery Date:** | **30 July, 2014** |
| **Deliverable Leader:** | **CERTH** |
| **Author(s):** | **Fani Tzima** |
| **Status:** | **Draft** |
| **Reviewed on** | **21 July, 2014** |
| **Reviewed by:** | **Konstantinos Vavliakis** |

**Abstract**

This document contains the manual of the Cassandra stand-alone software library. The source code of the library, along with the up-to-date version of this document, are available online at https://github.com/cassandra-project/cassandra-stand-alone.

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version[[1]](#footnote-1)** | **Issue Date** | **Stage[[2]](#footnote-2)** | **Content and changes** |
| #0.1 | 10th July 2014 | Draft | Initial version of the document (CERTH) |
| #0.5 | 15th July 2014 | Draft | Added instructions for setting up dynamic scenarios (CERTH) |
| #0.7 | 18th July 2014 | Draft | Added the javadoc appendix (CERTH) |
| #0.9 | 22th July 2014 | Draft | Revised based on initial review round (CERTH) |
| #1.0 | 25th July 2014 | Final | Revised based on second review round (CERTH) |

List of participants:

|  |  |  |  |
| --- | --- | --- | --- |
| **Participant no. \*** | **Participant organisation name** | **Part. short name** | **Country** |
| 1  (Coordinator) | Centre for Research and Technology Hellas/ Informatics and Telematics Institute | CERTH-ITI | Greece |
| 2 | Aristotle University of Thessaloniki | AUTH | Greece |
| 3 | VaasaETT Global Energy Think Tank | VAASA | Finland |
| 4 | Politecnico di Milano | POLIMI | Italy |
| 5 | Coventry University Enterprises Ltd | COVUNI | UK |
| 6 | Luleå University of Technology | LTU-CDT | Sweden |
| 7 | Cleopa GmbH | CLEOPA | Germany |
| 8 | Erasmus research Institute of Management | ERASMUS | Netherlands |
| 9 | DRAXIS S.A | DRAXIS | Greece |

Table of contents

Document History 2

Table of contents 3

List of listings 4

1 Introduction 5

1.1 The Cassandra general-purpose, stand-alone software library 5

1.2 Purpose and overview of the document 5

2 Scenario modelling in CASSANDRA 7

3 Using the CASSANDRA software library 13

3.1 Requirements 13

3.2 Running a simulation 13

3.3 Printed output and output files 14

3.4 Setting up a scenario using an input file 14

3.5 Model libraries 19

4 Development using the CASSANDRA software library 20

5 Getting Help / Reporting Bugs / Contributing 21

6 Conclusions 22

References 23

A. Annex A – Title 24

List of listings

Listing 1 – The general properties section of the scenario input file. 19

Listing 2 – The scenario section of the scenario input file. 19

Listing 5 – The installation section of the scenario input file. 19

Listing 6 – The appliance section of the scenario input file. 19

Listing 7 – The consumption\_model section of the scenario input file. 20

Listing 8 – The person section of the scenario input file. 20

Listing 9 – The activity section of the scenario input file. 20

Listing 10 – The activity\_model section of the scenario input file. 21

Listing 9 – The simulation section of the scenario input file. 21

Listing 10 – The pricing\_policy section of the scenario input file. 22

Listing 11 – The demographics section of the scenario input file. 22

# Introduction

CASSANDRAis a software platform that provides end users with the ability to dynamically model and investigate segments of the energy market, according to their specific needs. Itis essentially a tool for modeling all market stakeholders, as well as the essential grid components, in such a manner, so that modeling of aggregate structures (such as a suburb or a city, a consumer coalition or a group of commercial buildings) is possible. Through CASSANDRAone may specify consumption habits and patterns, consumer types and behaviors, intermediaries and policy regulations, so that these are diffused in the simulated community, in order to identify behavior drivers or policy makers. Apart from that, CASSANDRAalso strives to upgrade the role and market power of small-scale electricity consumersutilizing the concept of consumer social networks (CSN).

## The Cassandra general-purpose, stand-alone software library

CASSANDRA is implemented as a web-service platform, with the back-end providing decision support functionalities related to energy efficiency and demand response through the CASSANDRA front-end web application.

However, in order to widen the applicability of the project’s results and in line with one of CASSANDRA’s main objectives, namely to “promote the proposed consumer behavior and load modeling approach, and disseminate the energy and environmental benefit/impact to all actors”, the appropriate modifications in the code and architecture were made to also provide **a general-purpose, stand-alone software library** based on the CASSANDRA modules.

The CASSANDRA software library is an additional important tool that will allow developers to directly use the CASSANDRA functionality for consumer activity modelling, consumption analysis, consumer grouping and demand response. Thus, it will greatly facilitate the exploitation of the platform since it will be easier:

1. for different organisations and businesses to integrate the functionalities of CASSANDRA into their own products;
2. for interested users to explore CASSANDRA, especially in academic and/or research environments; and
3. for the open source community to uptake and further develop the platform in the future.

## Purpose and overview of the document

The aim of the present deliverable is to offer a manual to accompany the Cassandra stand-alone software library. The up-to-date version of this document, along with the source code of the library, is located at: https://github.com/cassandra-project/ cassandra-stand-alone.

The current document is public and is intended to be read, not only by the members of the Consortium, but mainly by readers that are interested to use the library in order to model electrical installations and consumers and run the corresponding simulation scenarios.

The report corresponds to the user manual of the platform (Section 2), with each subsection corresponding to each entity modelled through the platform. Section 3 contains the manual of the training module, an external application that can be used to train the simulation agents‟ activity models form data based on electric measurements from the modelled installations. The document concludes with installation instructions for the Cassandra platform (Section 4).

# Scenario modelling in CASSANDRA

In what follows, we briefly describe the process of modeling a scenario to be simulated in the CASSANDRA platform. This description, that also includes details of all components comprising the scenario to be simulated (installations, persons, activities, appliances etc.), provides the basis (a) for presenting the process of setting up a scenario (Section 3.4) to be simulated using the CASSANDRA stand-alone software library, as well as (b) for the short guide to developing Java applications exploiting the functionalities offered by the CASSANDRA software library (Section 4).

When modeling and simulating a scenario in the CASSANDRA platform, the main usage workflow is as follows:

1. The user creates a scenario and sets up all the necessary entities (Installations, Persons, Appliances, Activities, Activity Models etc.) along with their properties.
2. The user specifies the simulation parameters.
3. The user runs the scenario.
4. The user inspects the results of the simulation.

At this point, it is worth noting that it is possible to automatically compute model parameters based on measurements obtained from actual consumer installations, through CASSANDRA’s Training Module (<https://github.com/cassandra-project/training>). However, in what follows we will describe the properties required to manually set up all entities entailed in a simulation scenario.

A **scenario** in CASSANDRA has the following user-defined properties:

* Name: the name of the scenario.
* Description: a short description of the scenario.
* Setup: a selection between “Static” and “Dynamic” scenario setups.

In the “Static” scenario case, the user builds the scenario step-by-step, defining all installations to be simulated, along with the appliances they contain, the people residing in them and their energy consuming activity patterns. This is probably the best choice for small scenarios or for larger scenarios where measurements are available and can be fed to the Training Module for analysis and automatic computation of the involved entities’ parameters.

In the “Dynamic” scenario case, the user defines “Appliances” and “Person” entities (along with their “Activities”) and then enters “Demographic Data”, based on which the platform populates the simulation’s installations. A dynamic scenario can also include complete installations (like the ones defined for static scenarios) to be included “as is” in the simulation. More details on how the probabilities of each entity’s participation in a dynamic scenario are calculated can be found in the last part of this Section, where **demographic data** are described.

In any case (static or dynamic), a scenario should contain (a) at least one complete **installation** and (b) a set of **simulation parameters**, so that it can be simulated.

An **installation** has the following properties:

* Name: the name of the installation.
* Type: the type of the installation.
* Description: a description of the installation.
* Transformer ID: a label indicating the ID of the transformer under which the installation is located.
* Location: the location of the installation.
* Lat: the latitude of the installation's geolocation.
* Long: the longitude of the installation's geolocation.

Each installation should include at least one complete **person** and one **appliance**.

An **appliance** is part of an installationandhas the following properties:

* Name: the name of the appliance.
* Type: the type of the appliance.
* Description: a short description of the appliance.
* Energy Class: the energy class of the appliance.
* Stand By: the stand-by consumption power.
* Base: whether the appliance is a base load.
* Shiftable: whether the appliance is shiftable.
* Controllable: whether the appliance is controllable.

Additionally, every appliance should have a **consumption model**, with the following properties:

* Name: the name of the consumption model.
* Description: a short description of the consumption model.
* P-Expression: the expression that provides the active power curve.
* Q-Expression: the expression that provides the re-active power curve.

P- and Q-Expressions have the following form, respectively:

{m {n1 [p1,d1,s1] [p2,d2,s2]}, {n2 [p3,d3,s3]}, ...} (1)

and

{m {n1 [q1,d1,s1] [q2,d2,s2]}, {n2 [q3,d3,s3]}, ...} (2)

where *p* is the active power, *q* is the reactive power, *d* is the duration in minutes and *s* is the slope. Tuples *p1* and *p2* in Eq. (1) will be executed for *n1* timesteps and then tuple *p3* for *n2* timesteps. Those *n1\*2+n2* timesteps (since there will be *n1* steps for *p1* and *n1* steps for *p2*) will be executed *m* times. Of course, there can be *n3*, *n4* etc. For loops, we set either *m* or *n* to 0.

A **person** resides in an installationand has the following properties:

* Name: the name of the person.
* Type: the type of the person.
* Description: a short description of the person.
* Awareness & sensitivity: values defining how a person “responds” in demand response scenarios.

Every person should have at least one **activity** with the following properties:

* Name: the name of the activity.
* Type: the type of the activity.
* Description: a short description of the activity.

Each activity should have at least on **activity model** defining the probabilistic behavior of the person using the appliances participating in the activity. Thus, an **activity model** has the following properties:

* Name: the name of the activity model.
* Type: the type of the activity model.
* Description: a short description of the activity model.
* Day type: the day type the activity model corresponds to. Acceptable values include any combination of the following: “any”, “weekdays”, “weekends”, abbreviations of specific weekdays (i.e. “[Mon, Tue, Sat]”), specific days (formatted as “1/12, 31/10”).
* Shiftable: whether the activity model can be considered shiftable or not.
* Exclusive: whether the activity can be performed concurrently with other activities or not.
* Participating appliances: appliance(s) that participate(s) in the activity model.

It should also be noted, that while multiple activity models can be defined per activity, only one activity model is fired for every activity and every time tick. For example, if for a specific activity, the user has specified one activity model of day type “any” and one of day type “weekdays” and the current simulation day is Monday, then the more specific of the two will be fired, i.e. the activity model with day type “weekdays”.

To complete the activity model the user must also provide the properties and parameters of three **distributions** that essentially specify the behavior of the activity model. These are the duration, start-time, and number of times per day distributions, which are the probabilistic models of how long the activity takes, at what time of the day it starts and how many times per day it is executed, respectively.

For the first two **distributions** (duration and start-time) the user can select between several types of distributions, such as “Uniform Distribution”, “Histogram”, “Normal Distribution”, and “Gaussian Mixture Models”, while for the third (number of times per day), the appropriate distribution is that of type “Histogram”.

For “Histogram”, one just needs to provide a comma-separated list of values: [1, 2, 3, 4...].

For the “Uniform Distribution”, one needs to provide an expression, defining the start and end values: [{"start":100, "end":200}].

For the “Normal Distribution”, one needs to provide an expression, defining the mean and standard deviation values: [{"mean":45, "std":10}]

Finally, for the “Gaussian Mixture Models”, one needs to provide tuples of weight, mean and standard deviation values: [{"w":0.5, "mean":45, "std":10}, {"w":0.5 , "mean":100, "std":10}], with all weights *w* summing up to 1.

Although the above process of defining an activity model seems copious, it should be pointed out that, under normal system operation, the three required distributions would be automatically computed via measurements obtained from the actual installation being modelled (through the Training Module) or would be the result of slightly altering an existing “Activity Model”, found in the system library.

Having defined a complete installation (containing at least one appliance and at least one person, with at least one activity employing the appliance), the **simulation parameters** that need to be entered by the user are the following:

* Name: the name of the parameters set.
* Location: the location of the simulation.
* Monte Carlo Runs: how many times the simulation will run.
* Date Started: the starting date of the simulation.
* Date Ends: the ending date of the simulation.
* Response Type: choice between “None”, “Optimal”, “Normal” and “Discrete”. The last three values correspond to specific time shifting models (as a result of change in price) for demand response scenarios, detailed in the theoretical models deliverables.
* Description: notes on the simulation parameters set.
* CO2: the CO2 factor for the simulation.
* Pricing Scheme: the **pricing scheme** under which the energy consumption of the installations will be billed.
* Baseline Pricing Scheme: the baseline **pricing scheme** for demand response scenarios.

In order to perform a demand response simulation, “Response Type” should be other than “None” and both “Pricing Scheme” and “Baseline Pricing Scheme” should be defined.

The CASSANDRA platform supports five **pricing schemes**. All of them, irrespective of type, include the following properties:

* Name: the name of the scheme.
* Type: Choice among “ScalarEnergyPricing”, “ScalarEnergyPricingTimeZones”, “EnergyPowerPricing”, “AllInclusivePricing” and “TOUPricing”.
* Description: a short description of the scheme
* Billing Cycle: billing cycle duration in days.
* Fixed Charge: fixed charge for every billing cycle.

Depending on its type, each pricing scheme may include additional properties. More specifically, **Scalar Energy Pricing** includes the following additional properties:

* Levels: pairs of price and energy levels.

In this case, the cost for a “Billing cycle” of 4 months, a “Fixed charge” of 15, four pricing levels:

* Price Level 1: [500 Kwh, 0.06]
* Price Level 2: [400 Kwh, 0.07]
* Price Level 3: [400 Kwh, 0.08]
* Price Level 4: [0.1]

and a measured consumption of 1500 kWh, is calculated as follows:

cost = 0.06 \* 500 + 0.07 \* 400 + 0.08 \* 400 + 0.01 \* 200 + 15 = 125.

One should always include a pricing level with 0 Kwh as a final level (see the last item on the bulleted list above) in order to calculate the remainder energy, if energy consumption is greater than the sum of all levels.

**Scalar Energy Pricing Time Zones** includes the following additional properties:

* Offpeak Price: the price during offpeak hours.
* Levels: pairs of price and energy levels.
* Offpeak: offpeak hours of pricing.

In this case, the cost for a “Billing cycle” of 4 months, a “Fixed charge” of 15, an “Offpeak price” of 0.05, four pricing levels:

* Price Level 1: [500 Kwh, 0.06]
* Price Level 2: [400 Kwh, 0.07]
* Price Level 3: [400 Kwh, 0.08]
* Price Level 4: [0.1]

and a measured consumption of 1000 kWh in peak and 500 Kwh in off-peak, is calculated as follows:

cost = 0.06 \* 500 + 0.07 \* 400 + 0.08 \* 100 + 0.05 \* 500 + 15 = 105.

**Combined Energy and Power Pricing** includes the following additional properties:

* Contracted Capacity: the contracted power capacity.
* Energy Price: the price of energy consumed.
* Power Price: the power pricing of the contracted capacity.

In this case, the cost for a “Billing cycle” of 1 month, a “Fixed charge” of 2, an “Energy price” of 0.08, a “Power price” of 2.5 and a measured consumption of 350 kWh, with a “Contracted capacity” of 10kW, is calculated as follows:

Cost = 0.08 \* 350 + 2.5 \* 10 + 2 = 55.

**All-inclusive Pricing** includes the following additional properties:

* Contracted Energy: the contracted energy.
* Fixed cost: the price of contracted energy.
* Additional cost: the price of additional energy.

In this case, the cost for a “Billing cycle” of 1 month, a “Fixed charge” of 0, a “Contracted Energy” of 200, a “Fixed cost (for contracted consumption)” of 20, an “Additional cost (for additional consumption above the contracted)” of 0.25 and a measured consumption of 300kWh, is calculated as follows:

Cost = 20 + 0.25 \* 100 + 0 = 45.

**TOU (Time Of Use) Pricing** includes the following additional properties:

* Timezones: the price of energy in specific timezones.

In order to set the timezones right, one should end one time zone with an hour (for example 18:00) and start the next timezone with this ending time (18:00 in the example). So, a valid scheme would be: 00:00 - 17:00, 17:00 - 23:00, 23:00 - 00:00.

As already mentioned, in the case of a “dynamic” scenario setup, the user also has to define a set of **demographic data** with the following properties:

* Name: the name of the demographics parameters.
* Type: the type of the demographics parameters.
* Description: a description of the demographics parameters.
* Number of entities: the number of entities (installations) to be created.
* Probabilities of participation: probabilities of participation for defined installations, persons and appliances.

A dynamic scenario must always include an installation construct named “Collection”, under which the modeled person types, along with their activities and the corresponding activity models and appliances, are defined. The scenario can also (optionally) include whole installations (as defined in the case of static scenarios) to be included in the simulation “as is”. Given these definitions of person types, appliances and installations, the provided demographic data are used to dynamically instantiate installations in the simulation. More specifically, the system instantiates “Number of entities” installations, selecting among dynamic (“Collection”) and static (“as is”) installations, based on the probabilities of participation defined by the user for installations. For dynamically instantiated installations, persons and appliances are also included stochastically, based on the corresponding participation probabilities, defined by the user. Note that (a) the probabilities of assignment of person types to installations and (b) the probabilities of selection among installation types (“Collection” and “as is” installations) should be set to correspond to mutually exclusive events. This is not true for appliances.

# Using the CASSANDRA software library

The CASSANDRA stand-alone software library can be downloaded from: <https://github.com/cassandra-project/cassandra-stand-alone>. This document is written for version 1.0. It assumes that you have downloaded and extracted “cassandra-stand-alone-master.zip” and that “cassandra-stand-alone-master”, found within, is your current working directory.

## Requirements

CASSANDRA requires Java version 1.7 or above. It comes bundled with other packages that implement the data storage functionality needed by CASSANDRA, namely MongoDB’s “mongo-java-driver-2.12.2.jar” and “derby-10.10.1.1.jar”. These files are found in the lib directory.

If you have MongoDB installed on your system (and provided that it can be reached on localhost by the mongo-java-driver), you can use it to store simulation output data. This can be achieved by setting the useDerby property to “false”, when setting up your scenario (see Section 3.4). If you do not have MongoDB installed (or don’t want to use it), you should set the useDerby property to “true”, so that simulation output data are automatically stored in an embedded Apache Derby database. For more information on Apache Derby see <http://db.apache.org/derby/>. For instructions on installing and using Derby (you do not need to follow these steps, unless you want to inspect the actual data stored in the relational tables) see <http://db.apache.org/derby/quick_start.html>.

## Running a simulation

CASSANDRA can be used very easily from the command line, provided that a scenario has been defined in a properly formatted input file (see Section 3.4). For example, to run a simulation for the scenario described in file “SimpleStaticScenario.txt” and store the output in the “output” directory (both under the working directory “cassandra-stand-alone-master”) type:

java -cp "./lib/\*" eu.cassandra.sim.StandAloneSimulation "SimpleStaticScenario.txt" "./output"

If you are on a Microsoft Windows system, you need to use back slashes instead of forward slashes (.**\**lib**\**\*). If you add the jar files to the system's CLASSPATH, you do not need to supply the -cp option at runtime. For the remainder of examples in this tutorial we will assume that this is the case.

## Printed output and output files

Running a simulation for the scenario described in file “SimpleStaticScenario.txt”, will output the following:

Classifier\_name : meka.classifiers.multilabel.BR

Classifier\_ops : [-W, weka.classifiers.bayes.NaiveBayes, --, , , ]

Classifier\_info :

Dataset\_name : Music

Type : ML

Threshold : 0.9974578524138343

Verbosity : 1

1. : csv + data in a document (mongodb) or relational (derbydb) database.

## Setting up a scenario using an input file

As already mentioned, the user can set up a scenario using a properly formatted input file. Having defined the scenario, the corresponding file is used as input to the following command that runs a simulation:

java -cp "./lib/\*" eu.cassandra.sim.StandAloneSimulation <your\_input\_file> <your\_output\_dir>

In what follows, we will use the file “SimpleStaticScenario.txt” (that can be found under the working directory “cassandra-stand-alone-master”) as an example to explain how the input file for setting up a scenario should be formatted.

As one can easily observe, the input file contains a **section** for each entity to be modeled (installation, person etc.). Each section’s start is marked by the characters “-->” followed by the proper string from the set

{ general\_properties, scenario, installation, appliance, consumption\_model, person, activity, activity\_model, simulation,   
pricing\_policy, pricing\_policy\_baseline, demographics }.

Sections general\_properties, pricing\_policy and pricing\_policy\_baseline are optional. The demographics section is only required for dynamic scenarios. The rest of the sections are required (for both types of scenarios – static and dynamic), since a scenario to be simulated must have exactly one set of simulation parameters and at least one installation; an installation must have at least one person and at least one appliance; an appliance must have exactly one consumption\_model; a person must have at least one activity and an activity must have at least one activity\_model.

The general\_properties section includes, three optional parameters: (a) the random seed for the simulation, with default value 0, that if > 0 renders the simulation results repeatable; (b) the choice between using MongoDB or Apache Derby for storing simulation output data, which defaults to Apache Derby and (c) the verbosity level of the output, with the default false value for printKPIs suppressing the printing of KPIs.

--> general\_properties

seed = 171181

useDerby = false

printKPIs = true

Listing 1 – The general properties section of the scenario input file.

The scenario section includes all the properties of the scenario to be simulated, as defined in Section 2. The name and setup parameters are required, while description is optional.

--> scenario

name = Scenario1

description = A simple static scenario (1 installation, 1 person, 1 activity & 2 appliances).

setup = static

Listing 2 – The scenario section of the scenario input file.

The installation section is used to define an installation. Most of the parameters are optional, with id and name being the only required ones.

--> installation

id = inst1

name = Fani’s house

type = Flat

description = Sample installation

trans\_id = 12345678

location = Thessaloniki

lat = 0

long = 0

Listing 3 – The installation section of the scenario input file.

The appliance section is used to define an appliance within an installation. Most of the parameters are optional, with id, name, installation and consumption\_model being the only required ones. The installation parameter must “point” to the id of the installation that the appliance belongs to (in this case the only installation defined in the scenario input file with id=inst1 – see Listing 3). The consumption\_model parameter must “point” to the id of the consumption model for the appliance (see Listing 5).

--> appliance

id = appl1

name = Cleaning Washing Machine

type = Cleaning

description = Description of Cleaning Washing Machine

energy\_class = Class A

standy\_consumption = 0

base = false

shiftable = false

controllable = false

installation = inst1

consumption\_model = consModel1

Listing 4 – The appliance section of the scenario input file.

The consumption\_model section is used to define an appliance consumption model. All parameters, except description, are required. For an explanation of the active and re-active power curves’ expressions (pmodel and qmodel, respectively), see Section 2.

--> consumption\_model

id = consModel1

name = Cleaning Washing Machine Consumption Model

description = P and Q Consumption Model for Cleaning Washing Machine

pmodel = { "n" : 0 , "params" : [ { "n" : 1 ,   
 "values" : [ { "p" : 107.74000000000001 , "d" : 10 , "s" : 0}]}]}

qmodel = { "n" : 0 , "params" : [ { "n" : 1 ,   
 "values" : [ { "q" : 107.74000000000001 , "d" : 10 , "s" : 0}]}]}

Listing 5 – The consumption\_model section of the scenario input file.

The person section is used to define a person within an installation. Most of the parameters are optional, with id, name and installation being the only required ones.

--> person

id = person1

name = Fani

type = Girl

description = Single person

installation = inst1

awareness = 0.8

sensitivity = 0.3

Listing 6 – The person section of the scenario input file.

The activity section is used to define a person’s activity. The id, name and person parameters are required. The person parameter must “point” to the id of the person that performs the activity (in this case the only person defined in the scenario input file with id=person1 – see Listing 6).

--> activity

id = act1

name = Person Cleaning Activity

type = Cleaning

description = Person Cleaning Activity

person = person1

Listing 7 – The activity section of the scenario input file.

The activity\_model section is used to define an activity model, bound to a specific activity. All parameters, except type, description, shiftable (default false), and exclusive (default true) are required. Valid input values/expressions for the day\_type parameter are explained in Section 2. The activity parameter must “point” to the id of the activity that the activity model refers to (in this case the only activity defined in the scenario input file with id=act1 – see Listing 7). The containsAppliances parameter must be a comma-separated list that “points” to the ids of the appliances that the activity model entails (in this case the two appliances defined in the scenario input file with id=app1 – see Listing 4; and id=app2 – definition not shown here for brevity). As already mentioned in Section 2, to complete the activity model the user must also provide the properties and parameters of three distributions. These distributions are marked by the prefixes duration\_, start\_ and repetitions\_ in the parameter list, within the activity\_model section. For each of them, the user must define the distrType parameter and, depending on the distribution type, one of the values or parameters parameters. For more details per distribution type, see Section 2.

--> activity\_model

id = actModel1

name = Person Cleaning Activity Model

type = Cleaning

description = Person Cleaning Activity Model

day\_type = any

shiftable = false

exclusive = true

activity = act1

containsAppliances = appl1, appl2

duration\_distrType = Normal Distribution

duration\_values = [ ]

duration\_parameters = [ { "mean" : 1 , "std" : 1}]

start\_distrType = Histogram

start\_values = [0,0,0,0,0,0,0,0,…,0,  
 0.00476,0.00476,0.00476,0.00476, … ,0,0,0,0,0]

start\_parameters = [ ]

repetitions\_distrType = Histogram

repetitions\_values = [0.25,0.375,0.25,0,0,0,0,0.125]

repetitions\_parameters = [ ]

Listing 8 – The activity\_model section of the scenario input file.

The simulation section includes all the properties comprising the simulation parameters set, as defined in Section 2, except the pricing schemes that, if used, are defined in separate sections (see Listing 10). Most of the parameters are optional, with name and response\_type being the only required ones. Valid input values for the response\_type parameter are explained in Section 2.

--> simulation

name = Sample simulation parameters

locationInfo = Thessaloniki

mcruns = 5

start\_dayOfMonth = 5

start\_month = 4

start\_year = 2014

numberOfDays = 3

response\_type = None

description = A simple static scenario in Thessaloniki to be simulated for 3 days, starting on 05/04/2014

co2 = 2

Listing 9 – The simulation section of the scenario input file.

The (optional) pricing\_policy section includes all the properties of the pricing scheme under which the energy consumption of the installations will be billed. The name, type, billingCycle and fixedCharge parameters are required for all pricing schemes, while additional parameters may be required, depending on the pricing scheme type. In the example of Listing 10, where the pricing scheme type is “AllInclusivePricing”, those additional (required) parameters are fixedCost, additionalCost and contractedEnergy. For the additional parameters required for defining pricing schemes of other types, see Section 2. Examples for all types of pricing schemes may also be found in the “model\_library” folder (under the working directory “cassandra-stand-alone-master”).

--> pricing\_policy

name = TestAIP

type = AllInclusivePricing

billingCycle = 120

fixedCharge = 15

fixedCost = 100

additionalCost = 50

contractedEnergy = 100

Listing 10 – The pricing\_policy section of the scenario input file.

The pricing\_policy\_baseline section includes all the properties of the baseline pricing scheme for demand response scenarios. It is formatted exactly like the pricing\_policy section and is ignored if the response\_type parameter of the simulation section has the value “None”. In any other case (response\_type parameter value different than “None”), it is a required section, along with the pricing\_policy one, defining the pricing schemes to be used for the demand response scenario.

Finally, the demographics section includes the properties, based on which the system instantiates installations in the case of dynamic scenarios. Given that our example file so far (“SimpleStaticScenario.txt”) models a static scenario, it includes no demographics section. The “SimpleDynamicScenario.txt” file, however, modelling a simple dynamic scenario with two person types (person1 and person2), five appliances (app1 - app5) and an “as is” installation (inst1), includes the (required for dynamic scenarios) demographics section presented in Listing 11. Therein, it is defined that the system should instantiate 10 (number\_of\_entities) installations, selecting equiprobably between copying the “as is” installation and dynamically creating an installation, based on the defined person types and appliances. For the dynamic creation of installations, person1 is selected for inclusion with probability 0.7, while person2 is selected with probability 0.3 (notice that the probabilities for persons, as well as installations, are defined so as to correspond to mutually exclusive events). As far as appliances are concerned, they are each included in a dynamically created installation (and, thus, the corresponding person’s activities) with probability 0.75.

--> demographics

name = demographic data

type = mixed

description = scenario with dynamic and as-is installations

number\_of\_entities = 10

participation\_probs\_installations = [ col1:0.5 , inst1:0.5 ]

participation\_probs\_persons = [ person1:0.7 , person2:0.3 ]

participation\_probs\_apps = [ app1:0.75 , app2:0.75, … , app5:0.75 ]

Listing 11 – The demographics section of the scenario input file.

## Model libraries

Apart from the scenario setup files for static (“SimpleStaticScenario.txt”) and dynamic (“SimpleDynamicScenario.txt”) scenarios that can be found in the working directory “cassandra-stand-alone-master”, the CASSANDRA stand-alone library also offers sample definitions of entities in the “model\_library” folder. Each of the files therein, contains the definition of one or more entities and is named accordingly. The files essentially contain the corresponding definition sections for each entity (as explained in Section 3.4), ready to be copied and reused in a scenario setup file, provided, of course, that the “id” properties and the properties “pointing” to other entities are properly set.

For example the file “WashingActivity\_2ActModels”,

# Development using the CASSANDRA software library

The user has to write a new class extending “eu.cassandra.sim.Simulation” and overriding the setup method that includes all of the scenario’s setup.

Then they do

For our example we will try to implement a class that defi

# Getting Help / Reporting Bugs / Contributing

If you need help with the CASSANDRA stand-alone software library, you can mail your problem to [fani.tzima@iti.gr](mailto:fani.tzima@iti.gr).

The Developers Quick-Start Guide contains the initial steps to get up to speed with the development environment used in the CASSANDRA project: <https://github.com/cassandra-project/platform/wiki/Developers-quickstart-guide>.

If you have found a bug with the CASSANDRA stand-alone software library, you can report it by opening an issue on: <https://github.com/cassandra-project/cassandra-stand-alone/issues>.

If you would like to contribute to the CASSANDRA stand-alone software library, by adding for example new KPIs or additional output options, please get in touch with the developers or send your code, along with a list of your additions) to [fani.tzima@iti.gr](mailto:fani.tzima@iti.gr).

More information (such as contact information) can be found at the CASSANDRA website: <http://www.cassandra-fp7.eu>.

# Conclusions

You may add or erase as many sections as you need but the final section must have the title Conclusions and the conclusions of the deliverable stated clearly.

References

1. References are marked using a numbered list style referred to as References. This style has the following specifications: Arial, 11pt., 1.5 line spacing, 6 pt. space from the previous paragraph.
2. Additional references.
3. Annex A – Title

Text goes here

* 1. Subsection 1 of Annex A

Text

* + 1. Subsection 2of Annex A

Text

1. Please use a new number for each new version of the deliverable. Add the date when this version was issued and list the items that have been added or changed. The ‘what’s new’ column will help the reader in identifying the relevant changes. Don’t forget to update the version number and date on the front page and the header. [↑](#footnote-ref-1)
2. A deliverable can be in either of these stages: “draft” or “final”. For each stage, several versions of a document can be issued. *Draft:* Work is being done on the contents. *Final*: All chapters have been completed. [↑](#footnote-ref-2)