Melodic
Big data cloud

Multi-cloud Execution-ware for Large-scale Optimized Data-Intensive Computing

H2020-ICT-2016-2017
Leadership in Enabling and Industrial Technologies; Information and Communication Technologies

Grant Agreement No.:
731664

Duration:
1 December 2016
30 November 2019

www.melodic.cloud

Deliverable reference:
5.08

Date:
31 March 2018

Responsible partner:
7bulls

Editor(s):
Edyta Bańkowska

Author(s)
Edyta Bańkowska

Approved by:
Ernst Gunnar Gran

ISBN number:
N/A

Document URL:
http://www.melodic.cloud/deliverables/D5.08 Platform prototype release.pdf

Title:

Platform prototype release

Abstract

This document presents the process of testing Melodic Release 2.0. Software testing is an activity aimed at evaluating the quality of a program, and improve it by identifying any defects and potential problems. Melodic is a framework that supports automated deployment of both data and the applications processing the data, based on the constraints set by the organisation owning the data and the application.

This deliverable accompanies the second release of the integrated MELODIC framework, including the security and DLMS technical components. It incorporates the feedback from the testing of release 1.0 and release 1.5. In addition, the deliverable reports on integration and technical testing activities (regression, functionality, performance and security tests) for release 2.0, based on the test strategy and plan described in D5.06 “Test Strategy and Environment” [4].

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731664

Document	
Period Covered	M6-18
Deliverable No.	D5.08
Deliverable Title	Platform prototype release
Editor(s)	Edyta Bańkowska
Author(s)	Edyta Bańkowska
Reviewer(s)	Amir Taherkordi, Ernst Gunnar Gran
Work Package No.	5
Work Package Title	Integration and security
Lead Beneficiary	7bulls
Distribution	PU
Version	1.0
Draft/Final	Final
Total No. of Pages	34 + One Appendix

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Index of Tables

Index of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

1 Introduction

This document presents information about the testing process of Melodic release 2.0, and describes the corresponding new testing environments, the processes of testing, and new test cases for the new components in release 2.0. This work has been done during a six months period of the project and briefly shows the steps of the test process in the project.

New test cases include functional testing of new components and features, not used in previous releases, like: Cloudiator 2.0 with Spark, Data Life-cycle Management System (DLMS), Function-as-a-Service (FaaS) and dockerized applications. Apart from the functional requirements, non-functional features are also tested in order to verify proper implementation in the Melodic. The non-functional features tested in this time period is related to security. For more details on the security implementation of Melodic, we refer to "D5.03 Security requirements & design" [2].

This document starts with an overview of the Melodic software components, including the new features and components used in release 2.0. Next, the new configurations of our University of Oslo (UiO) and Ulm University (UULM) Openstack environments are described, followed by an installation guide, a testing guide and a description of executed Test Cases.

1.1 Release 2.0

For the Melodic project, as presented by the Melodic Project Proposal, three main releases were planned:

- Release 1 – planned for 30 November 2017
- Release 2 – planned for 30 November 2018
- Release 3 – planned for 30 November 2019

The aim of Release 1 was to integrate underlying frameworks without changing their features. To make it possible to fully evaluate the Melodic use case applications after the first release, an additional release 1.5 was introduced with a set of new features and improvements. After release 1.5, the scope of release 2.0 includes the following:

- CAMEL 2.0 models
- Spark deployment
- Component dockerized – Docker image built from repository
- Reconfiguration
- Cloudiator 2.0 (Upperware integration)
- Authorization service
- DLMS integration with Melodic platform
- EMS integration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

1.2 Structure of the document

This deliverable is structured into the following chapters:

- Chapter 2, '**Components – The Melodic Architecture**': A brief summary of the Melodic architecture along with the release 2.0 components.
- Chapter 3, '**Testing environments**': Information about the testing environments currently used in the project.
- Chapter 4 '**The Melodic 2.0 installation guide**': Requirements and steps on how to install release 2.0 of the Melodic platform.
- Chapter 5, '**Testing Guide**': Detailed instructions on how to execute the Test Cases.
- Chapter 6, '**Test Cases**': A comparison of all executed Test Cases in release 2.0, detailed in Appendix A, 'Test Cases Release 2.0'.
- Chapter 7, '**Bugs**': An overview of all reported bugs in release 2.0 and their status.
- Chapter 8, '**Summary**': The deliverable ends with a short summary.

2 Components – The Melodic Architecture

Figure 1 presents an overview of the Melodic architecture. The figure also shows the high-level interaction between the main Melodic components, while a detailed architecture is presented in "D2.2 Architecture and initial feature definitions" [1]. The main Melodic sub-systems are:

- **Upperware**: Applications and data models created through the modelling interfaces in the form of CAMEL, are provided as input to the Melodic Upperware. The job of the Upperware is to calculate optimal data placements and application deployments on dynamically acquired Cross-Cloud resources in accordance with the specified application and data models in CAMEL, as well as in consideration to the current Cloud performance, workload situation, and costs. An overview of the Upperware Components is shown in Figure 2, with further details in [1].
- **Executionware**: The Executionware is responsible for the actual deployment of the Cloud application and its monitoring infrastructure, as well as the corresponding publishing of measurement information to the Upperware. An overview of the Executionware Components is shown in Figure 3.
- **Integration layer**: The components of the Melodic platform are integrated through two separate integration layers: the Control Plane and the Monitoring Plane (blue boxes in Figure 1), each bringing its own set of unique requirements.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

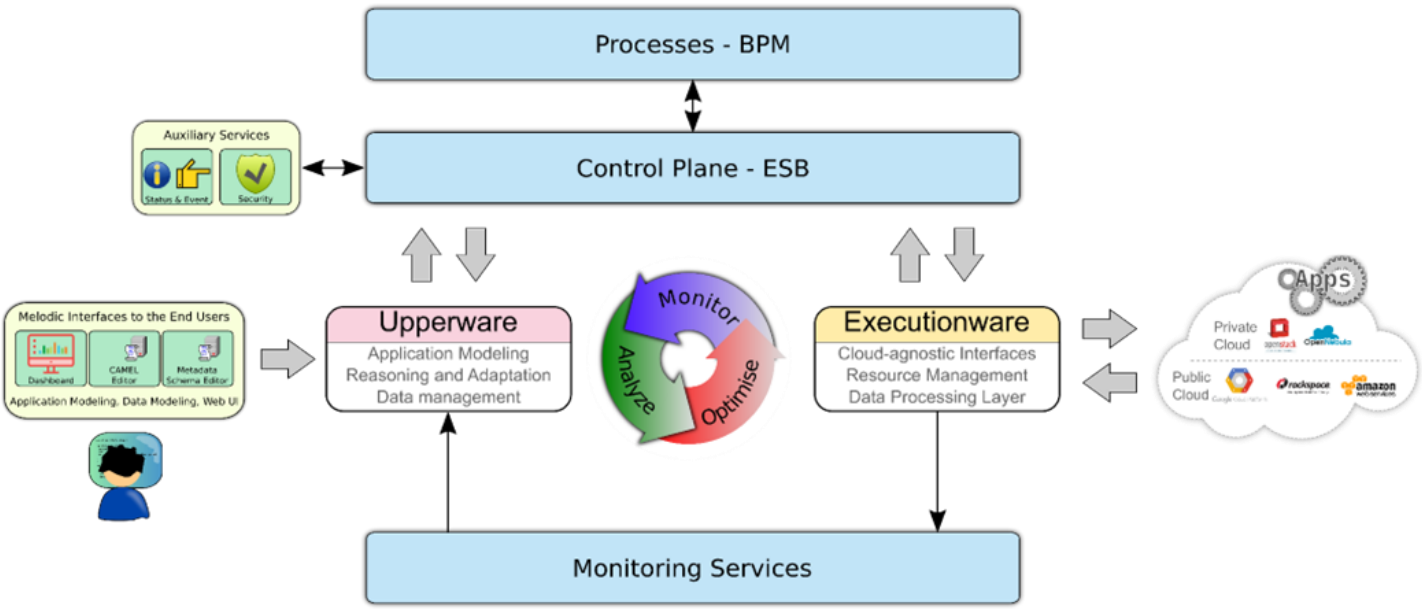


Figure 1: Overview of the Melodic architecture

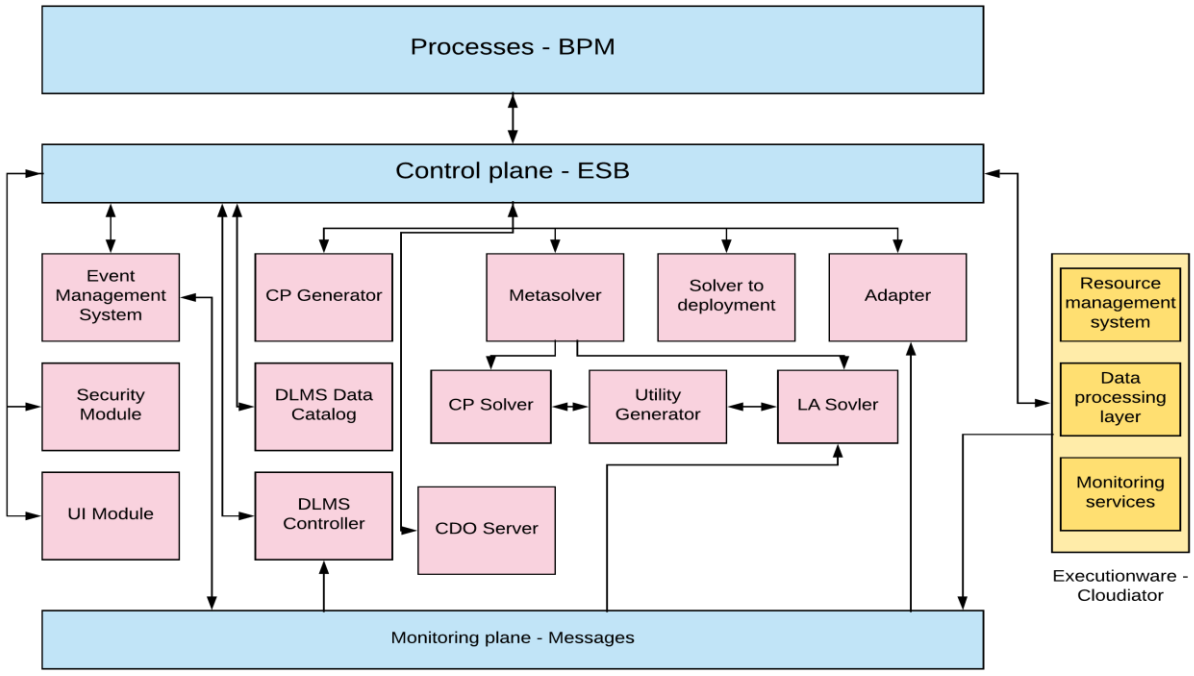


Figure 2: Overview of the Upperware Components

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

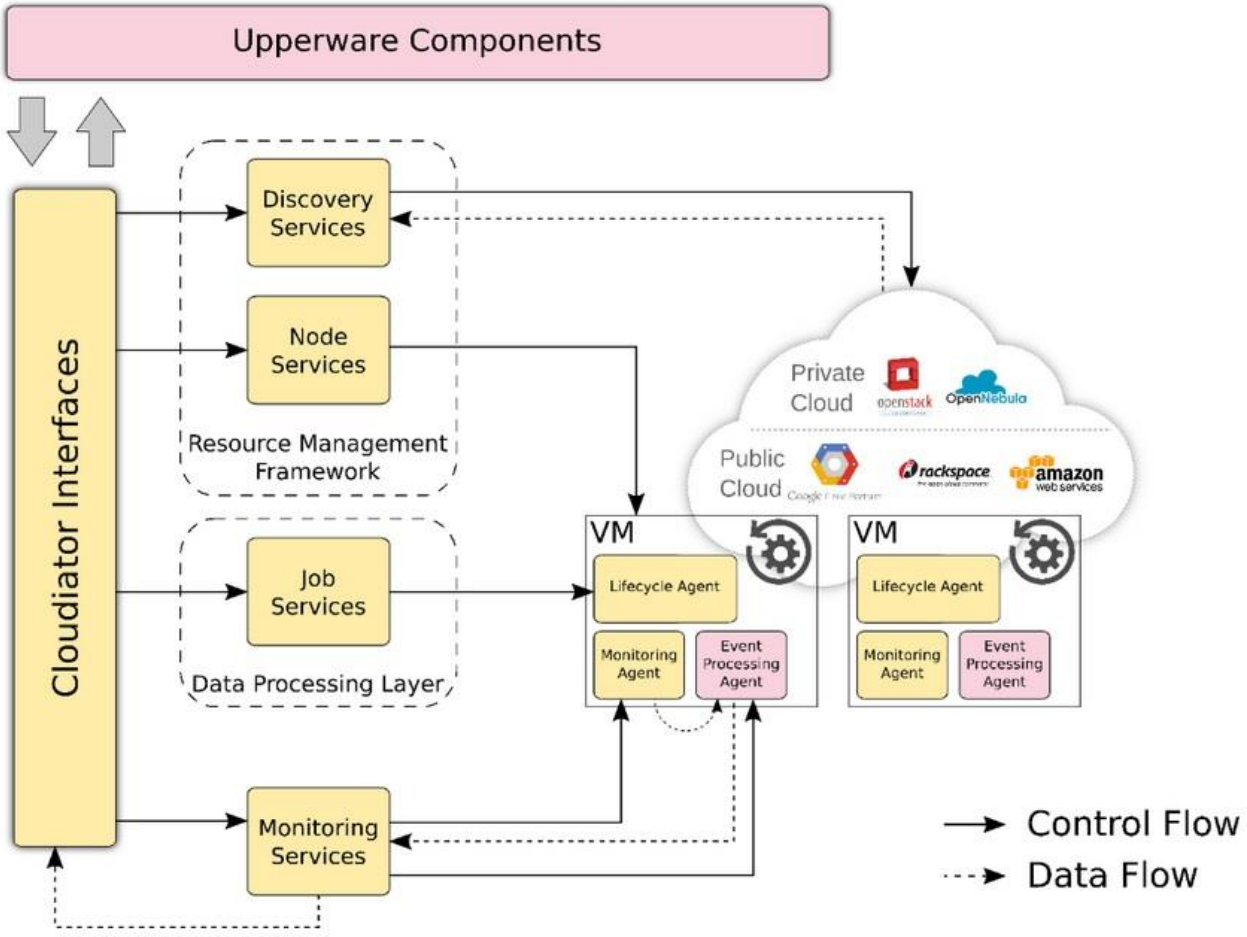


Figure 3: High-level Executionware Architecture

2.1 Software Components

Table 1 lists the key components in Melodic for release 2.0, including their key features and the sub-systems they belong to. Further details are provided in D2.2 [1].

Table 1: Main components used in the Melodic software, with particular relevance to testing

Component	Description, key features	Sub-system	Framework
CP Generator	Profiling of the application and preparation of the constraint programming (CP) model	Upperware	PaaSage
CP Solver	Solving all types of problems encoded in the CP model using a gradient descent approach.	Upperware	PaaSage
Solver-To-Deployment	Transforming CP models encompassing the solution produced by solvers to a provider-specific deployment model.	Upperware	PaaSage

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Adapter	Deployment control and adaptation of multi-cloud applications		Upperware	PaaSage
Cloudiator[1]	**Cloudiator (server part)**	Deploying an application and infrastructure to the Cloud Providers.	Executionware	Cloudiator is a result of the PaaSage project and was extended in the Cactos project.
	Cloudiator (VM part)	Components deployed on the created Virtual Machines (VMs)		
Camunda[2] **with status/event service**	Camunda is an open-source workflow engine written in Java that can execute business processes		Integration layer	Melodic
ESB	Ensure communication between all components (The exception is CDO [1])		Integration layer	Melodic
EMS	Gathering metrics		Upperware	Melodic
DLMS	Enables holistic management of the data lifecycle in Cross-Cloud environments		Upperware	Melodic
Utility Generator	An object acting on behalf of the application owner in the system, used to assign a utility value to a given deployment configuration proposed by the solver		Upperware	Melodic

3 Testing environments

A testing environment is a setup of software and hardware on which the testing team performs the testing of the software product. This setup consists of the physical setup which includes hardware, and a logical setup which may include a server operating system, a client operating system, a database server, a front end running environment, a browser (if we are referring to a web application), an IIS (version on server side) or any other software components required to run the software product. This testing setup is to be built on both ends, i.e. at the server side and at the client side.

We used a total of four different testing environments for release 2.0, located at UiO and UULM. The corresponding parameters of the testing environments are provided in Section 3, Table 2. Non-functional and functional tests were executed on the same set of environments.

[1] http://cloudiator.org/

[2] https://camunda.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3.1 Environments used during release 2.0

For Melodic release 2.0, we used eight virtual machines (VMs) for testing on which we installed the Melodic platform and the Cloudiator component. The specifications for the four different environments are provided in Table 2.

Table 2: Specification of the environments

Environment name	Installed platform/component	Machine name	IP	Details
qb1	Melodic	qb1	158.39.75.33	4CPU, 16GB of RAM, UiO Openstack
	Cloudiator	bravo	158.37.63.149	4CPU, 16GB of RAM, UiO Openstack
qb2	Melodic	qb2	134.60.64.238	4CPU, 16GB of RAM, UULM Openstack
	Cloudiator	delta	134.60.64.104	4CPU, 16GB of RAM, UULM Openstack
qb3	Melodic	qb3	158.39.75.236	4CPU, 16GB of RAM, UiO Openstack
	Cloudiator	charlie	134.60.64.125	4CPU, 16GB of RAM, UULM Openstack
mel1	Melodic	mel1	158.39.75.140	4CPU, 16GB of RAM, UiO Openstack
	Cloudiator	echo	158.37.63.167	4CPU, 16GB of RAM, UiO Openstack

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4 The Melodic 2.0 installation guide

This section details the hardware and operating systems requirements for Melodic release 2.0, and describes how to install the framework.

4.1 Requirements for Melodic's machine

Table 3 details the hardware and operating system requirements for the current Melodic software, while Table 4 specifies the port numbers being used by Melodic.

Table 3: Requirements for Melodic's machine

OS	Memory	Storage
Ubuntu 16.04	RAM: 32GB+	60GB+

Table 4: Open port numbers required by Melodic

Port Number	Protocol	Component	Purpose
22	TCP	SSH	Console
8080-8099	TCP	Components	REST endpoints of Melodic components (generator, cp-solver, solver2deployment, adapter, esb, process)
9001-10000	TCP		
2036, 3306	TCP	CDO	MySQL database
80	TCP	UI	Cloudiator's web interface
4001	TCP	Lance	etcd registry
9000	TCP	Cloudiator V2 components	Cloudiator's REST API
8080	TCP	Axe	Time-series database
33034	TCP	Lance	rmi registry
2222	TCP	VMS Discovery Server	
3308	TCP	DBAuth	
61616	TCP	Esper	sending metrics to JMS Queue
443, 6443	TCP	JWT Server	authentication of users
11211	TCP	Memcache	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

20000	TCP		
5005	TCP	components	connection from debugger
2037	TCP		
389, 636	TCP	ldap server	managing users

4.2 Melodic installation steps

A full installation of Melodic involves installing both the Upperware and the Executionware components of Melodic. The following sections details two alternative ways of installing the two components. The first alternative describes how to install both components on the same virtal machine, while the second alternative describes how to install the two components on separate virtual machines.

1. Installation of Upperware and Executionware components on the same virtual machine:

- Login to the created virtual machine, for example using: ssh *username*@<*VM's IP*>
- Run the following command to download the installation files from the bitbucket repository (set branch to release 2.0):

```
git clone https://bitbucket.7bulls.eu/scm/mel/utils.git
cd ~/utils
git checkout RC2.0
```

- Run Melodic's installation script:

```
sudo ~/utils/melodic_installation/installMelodic.sh install_melodic_with_cloudiator
```

- After installation, a new ".profile" is created in the home directory of the user. Load it by executing the following:

```
cd ~/
. .profile
```

- Now, the machine is ready to download and run the latest docker images from the Melodic and Cloudiator artefact repository. To download and start the components, run the following:

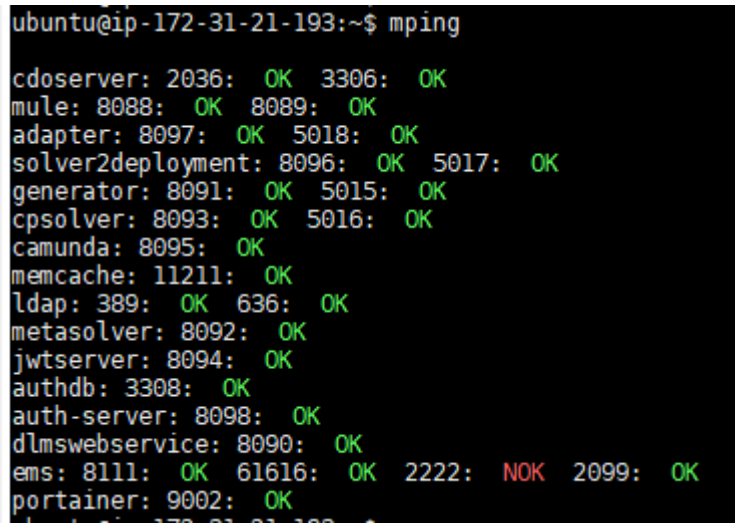
```
drestart
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Running this for the first time can take some time as Docker Swarm is being initialised. After the above command, components should be started. You can check the status by running the following two commands:

```
dps
mping
```

The screenshot below shows part of the output after executing the *dps* and *mping* commands:



In order to manage Melodic users, you will need to create a new Idap user by following the instructions at: https://confluence.7bulls.eu/display/MEL/Managing+of+users+in+Melodic.

The credentials of your new user will be used in deploy requests to Mule.

The process GUI should now be available at:

```
http://{PUBLIC_MELODIC_IP}:8095      (admin:admin)
```

2. Installation of Upperware and Executionware components on the separate virtual machines:

- For the Upperware installation, please follow the same installation guide as provided above, with just one change: please replace the installation script parameter with the following (the third step):

```
sudo ~/utils/melodic_installation/installMelodic.sh install_melodic
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- For the Executionware installation, please follow these instructions:

Installation of Cloudiator using Docker (docker-compose).

- install Docker: https://docs.docker.com/install/
- install docker-compose: https://docs.docker.com/compose/install/
- install git
- git clone the repository:
  git clone --recurse-submodules https://github.com/cloudiator/docker.git
- edit the env-template to e.g. use own API-Key-Token
- cp the env-template to .env: cp env-template .env
- run docker-compose up

Use:

- REST-Server automatically starts on port 9000.
- edit env-template

  if auth.mode=testmode:

  user 'testuser' will be generated in userDB and [auth.token] will be its valid Token

4.3 Useful aliases

Below you find some useful commands to manage Melodic components.

Commands:

dps	– displays running Docker containers (an alias for *sudo Docker images*)
mping	– tests connections to each of the components
drestart	– stops and then starts all the Melodic's components
dundeploy	– stops all components
ddeploy	– starts all components
~/logs$ tail -300f <component_name>.log	– displays the log of the selected component
sudo docker stop <component_ID>	– stops the selected component

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5 Testing Guide

This section describes the procedures for performing all activities within the testing processes. Each Test Case has an attached CAMEL model. Every created and included model specifies the testing application, including requirements, topology, metrics and credentials.

5.1 How to execute Test Cases with attached CAMEL Models

This section presents the Quality Assurance guide of testing Test Cases with attached CAMEL models.

1. Login to the machine with the Melodic platform and the Cloudiator component installed
2. Download the jar file from:
   https://s3.console.aws.amazon.com/s3/object/melodic.testing.data/cdo-uploader-1.0.1SNAPSHOT-jar-with-dependencies.jar
   Download *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar*
   a. In "/home/user/", create the hidden directory ".paasage" and put into this directory the configuration file for CDO named *eu.paasage.mddb.cdo.client.properties* which can be downloaded from:
      https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/?region=us-east-1&tab=overview
   b. In the *eu.paasage.mddb.cdo.client.properties* configuration file, change the IP address to the one of your virtual machine for the "host" property
   c. Run the jar file: *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar*
3. Create the directory "models" in "/home/user"
4. Download the respective CAMEL model needed for a test case from https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data (the required CAMEL Model is always attached in Test Case)
5. Move the downloaded CAMEL model (having a *.xmi* postfix) into the "/home/user/models" directory
6. By using the tools **SoapUI** or **Postman** execute the following steps:
   a. Create new REST project with URL
      http://<VM's IP>:8088/api/frontend/deploymentProcess and body:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Sample body

```
# In "CamelModelName" user can use for example "TwoComponentApp"
# Rest of the body doesn't need to be changed

{     "applicationId": "CamelModelName",          "watermark": {
            "user": "User-Test",
            "system": "UI",
            "date": "2016-02-28T16:41:41+0000",
            "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
    }
}
```

b. Using the POST method, start the process of deploying an application using the selected provider. For example, if we use the website: https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:sort=launchTime we can check the created instances.
c. For *OneComponentApp,* we copy the public URL of the created virtual machine and copy it to a new window in our browser. Then the AWS web page should be displayed. The AWS installation is included in the *AmazonEC2.xmi Provider* Model. We do not have to install it manually.
d. Check that *TwoComponentApp* is created and installed on one provider (i.e. AWS). One virtual machine is created for the database and a second virtual machine is created for an application. We can open the AWS web page (as before), copy the IP of the application VM, and enhance it as follows: http://Virtual_machine_IP:9999/demo/all. If we open this URL on another web browser tab, the application website should be displayed.
e. For the Test Cases where the user selects two different providers, one virtual machine is created on one provider and a second virtual machine on another provider. We can use for example AWS as the first provider, and Omistack as the second provider: https://omistack.e-technik.uni-ulm.de/dashboard/auth/login/. Then we can use the link: http://Virtual_machine_IP:9999/demo/all (as before) to deploy.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6 Test Cases

The test process starts at the very beginning of a project life cycle. During the Analysis and Design phase, the test team starts producing a Test Plan, as this should be prepared as early as possible. A Test Plan contains test cases, which are described in detail in the "D5.10 Quality Assurance Guide" deliverable [3].

A Test Case describes how to perform a specific test. The Test Case includes a set of test data, pre-conditions, expected results and post-conditions targeting a certain implementation, developed for a specific purpose or for the condition mapping to the test such as the execution of a program path, or to verify compliance with a specific requirement. Test Cases are created by the test team; either its members or the test leader.

The Test Plan should also explicate the dependencies between the Test Cases (if needed), by clarifying which Test Cases should be executed before others. This whole exercise has three main benefits:

- It allows the test team to understand the system to be developed
- It serves as a review of the system specifications and requirements
- It eases solving issues, as all parties (the test team and the development teams) have the same base data (the test data; input parameters for test cases, necessary to execute test cases and to reproduce bugs, if occurring during test case execution).

6.1 New Test Cases created during Release 2.0

In total, for release 2.0, 72 new Test Cases were created. Every Test Case was categorised into a particular non-functional or functional group according to the component's functionality, as shown in Table 5. The groups not specific to a use case, are further described below the table.

Table 5: Categories of new test cases in release 2.0

Functional Test Cases groups	Group ID	Test Cases created during release 2.0 (summary)
Cloudiator/Spark	CLDTR	20
DLMS	DLMS	11
CAS - use case application	CAS	9
Initial deployment	INIT	3
CE-Traffic – use case application	CET	8
Global reconfiguration	GLOB	6
Total		57

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Non-functional Test Cases groups		
Security – access control rules verification	SEC	15

- **Cloudiator/Spark** – This group contains all test cases related to Cloudiator V2 and Spark, including node candidates fetching, creation of VMs, Spark framework creation, Docker Container deployment, and BYO node creation.
- **DLMS** – Test cases in this group include DLMS metrics gathering, CAMEL-DLMS data registration, Data life-cycle events, unified namespace, and utility value calculation.
- **Security (Access control rules verification)** – Test cases referring to selective restrictions of access to a place or resource. Verification of authentication and authorization.
- **Initial deployment** – This group contains all test cases related to the initial deployment of an application in the Melodic platform.
- **Global reconfiguration** – Global reconfiguration refers to the reconfiguration of the application at a global scope, where a new solution is applied globally for the whole application, and not only on its specific parts (in contrast to local reconfiguration). Global reconfiguration test cases also verify the functionality of the EMS component. The role of EMS is to properly deliver events (produced according to Camel model specifications) to the Metasolver.

6.2 Status of Test Cases executed during Release 2.0

Table 6 lists the new executed Test Cases for Release 2.0, including a summary of each test case (where 'T' equals True/A Positive Test Case, and 'F' equals False/A Negative Test Case) in terms of its task corresponding identifier and name, as well as its execution status.

Table 6: List of all executed Test Cases in release 2.0

ID	Summary	Type	Group	Status
UC-CAS-8	[T] Deployment According To Security Rule/s	Non-Functional	CAS	Passed
UC-CAS-7	[T] Application Is Deployed Cross-Cloud	Functional	CAS	Passed
UC-CAS-6	[T] Application Is Deployed On 1&1 IONOS	Functional	CAS	Passed
UC-CAS-5	[T] Application Is Deployed On NordicStack	Functional	CAS	Moved to Release 2.5

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

UC-CAS-4	[T] Application Is Deployed On OMI	Functional	CAS	Passed
UC-CAS-3	[T] Application Is Deployed On AWS	Functional	CAS	Passed
UC-CAS-2	[T] Reconfiguration Happens Within Bounds Based On SLOs and UF	Functional	CAS	Passed
UC-CAS-1	[T] Multi-Component App Is Initially Correctly Deployed	Functional	CAS	Passed
UC-CET-8	[F] Deploy pySpark app on AWS and download packages via pip	Functional	CET	In progress
UC-CET-7	[F] Deploy pySpark app on AWS with additional compiled libraries for different platform	Functional	CET	In progress
UC-CET-6	[T] Deploy pySpark app on AWS with additional compiled libraries	Functional	CET	In progress
UC-CET-5	[T] Deploy pySpark app on AWS with additional libraries	Functional	CET	In progress
UC-CET-4	[T] Deploy Spark app on AWS	Functional	CET	In progress
UC-CET-3	[T] Deploy instance with GB RAM >= 8	Functional	CET	In progress
UC-CET-2	[T] Deploy app on AWS in United Kingdom	Functional	CET	In progress
UC-CET-1	[F] Request VM with exactly 3 cores at AWS	Functional	CET	In progress
BD5	[T] Execute Java based Spark Job	Functional	CLDTR	Passed
BD4	[T] Execute Python based Spark Job	Functional	CLDTR	Passed
BD3	[T] Use maximum resources available for Spark Job	Functional	CLDTR	Passed
BD2	[T] Execute Spark Job with concrete Spark Job requirements	Functional	CLDTR	Passed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

BD1	[T] Deploy Spark Cluster for 2 Spark Workers	Functional	CLDTR	Passed
VM3	[T] Create VM over two different provider's resources	Functional	CLDTR	Moved to Release 2.5
VM2	[T] Create VM on UULM OpenStack resource	Functional	CLDTR	Passed
VM1	[T] Create VM on ProfitBricks resource	Functional	CLDTR	Moved to Release 2.5
NC3	[T] Successfully retrieve Node Candidates for ProfitBricks	Functional	CLDTR	Moved to Release 2.5
NC2	[F] Assure that wrong cloud configuration forbids retrieval of Node Candidates	Functional	CLDTR	Passed
NC1	[T] Successfully retrieve Node Candidates for UULM OpenStack	Functional	CLDTR	Passed
AC-13	[F] At least one node of Deployment Instance Model (given to Adapter) has less than 16 GB RAM	Non-functional	SEC	Passed
AC-12	[T] The nodes of Deployment Instance Model (given to Adapter) have 16 GB RAM or more (each)	Non-functional	SEC	Passed
AC-11	[F] Deployment Instance Model (given to Adapter) has a total number of GB RAM (across all nodes) higher than 64	Non-functional	SEC	Passed
AC-10	[T] Deployment Instance Model (given to Adapter) has a total number of GB RAM (across all nodes) lower or equal than 64	Non-functional	SEC	Passed
AC-9	[F] Deployment Instance Model (given to Adapter) contains at least one node with NO location information	Non-functional	SEC	Passed
AC-8	[F] Deployment Instance Model (given to Adapter) contains at least one node NOT located in DE	Non-functional	SEC	Passed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

AC-7	[T] Deployment Instance Model (given to Adapter) contains node located in DE (all of them)	Non-functional	SEC	Passed
AC-6	[F] At least one node of Deployment Instance Model (given to Adapter) has exactly 1 core	Non-functional	SEC	Passed
AC-5	[T] The nodes of Deployment Instance Model (given to Adapter) have more than 1 core (each)	Non-functional	SEC	Passed
AC-4	[F] Deployment Instance Model (given to Adapter) has a total number of cores (across all nodes) higher than 4	Non-functional	SEC	Passed
AC-3	[T] Deployment Instance Model (given to Adapter) has a total number of cores (across all nodes) lower or equal than 4	Non-functional	SEC	Passed
AC-2	[F] Successfully connect to Authorization Server and get a Negative response	Non-functional	SEC	Passed
AC-1	[T] Successfully connect to Authorization Server and get a Positive response	Non-functional	SEC	Passed
T7.4	[F] New plan deployment authorisation	Non-functional	SEC	Passed
T7.3	[T] New plan deployment authorisation	Non-functional	SEC	Passed
T7.2	[F] ESB Authentication	Non-functional	SEC	Passed
T7.1	[T] ESB Authentication	Non-functional	SEC	Passed
T6.6	[T] Dynamic scalability (using a single public Cloud location)	Non-functional	Performance[3]	Passed

[3] Scenario group from JIRA, as used for Release 1.0/1.5

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T6.3	[T] Temporary unavailability of Cloud Provider	Non-functional	Fault handling[3]	Moved to Release 2.5
T1.5b	[T] Installation and deployment of FCR application, where one component is installed in a Docker container and another on a normal VM on two different Cloud Providers	Functional	INIT	Moved to Release 2.5
T1.5a	[T] Installation and deployment of FCR application in Docker containers on two different Cloud Providers	Functional	INIT	Moved to Release 2.5
T1.4	[T] Installation and deployment of FCR application in Docker container on one Cloud Provider	Functional	INIT	Moved to Release 2.5
T4.9	[T]Global reconfiguration - FCR, time > 3600s	Functional	GLOB	Passed
T4.8	[T]Global reconfiguration - FCR, time < 3600s	Functional	GLOB	Passed
T4.7	[T]Global reconfiguration - GenomWithSpark, time < 3600s	Functional	GLOB	Passed
T4.6	[T]Global reconfiguration - GenomWithSpark, time > 3600s	Functional	GLOB	Passed
T4.5	[T]Global reconfiguration - GenomWithSpark deployed on OpenStack Cloud Provider	Functional	GLOB	Passed
T4.4	[T] Global reconfiguration - GenomWithSpark deployed on AWS Cloud Provider	Functional	GLOB	Passed
U.5	[T] Successfully retrieve utility value at utility generator based on cost algorithm	Functional	DLMS	In progress
U.4	[T] Successfully retrieve utility value at utility generator based on total utility	Functional	DLMS	In progress
U.3	[T] Successfully retrieve utility value at utility generator based on data-aware algorithm	Functional	DLMS	In progress

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

U.2	[T] Successfully retrieve utility value at utility generator based on dc-aware algorithm	Functional	DLMS	In progress
U.1	[T] Successfully retrieve utility value at utility generator based on affinity-aware algorithm	Functional	DLMS	In progress
N.1	[T] Successfully mount data sources under alluxio cluster	Functional	DLMS	Passed
M.3	[T] Successfully read DataModel from CAMEL and register data sources in the DLMS	Functional	DLMS	Passed
M.2[	[T] Successfully retrieve data metrics from configured mysql data source in the DLMS algorithms	Functional	DLMS	In progress
M.1	[T] Successfully retrieve data metrics from configured alluxio data sources in the DLMS algorithms	Functional	DLMS	In progress
DLMS1	[T] Deploy working DLMS Agent	Functional	DLMS	In progress
DL.1	[T] Successfully execute data life-cycle event using DLMS Agent on the nodes	Functional	DLMS	In progress
DEP4	[T] Deploy dockerized application from private registry	Functional	CLDTR	Passed
DEP3	[T] Deploy application with native and dockerized component	Functional	CLDTR	Passed
DEP2	[T] Deploy dockerized application	Functional	CLDTR	Passed
DEP1	[T] Deploy native application	Functional	CLDTR	Passed

The whole content of the executed Test Cases during release 2.0 can be found in Appendix A, 'Test Cases Release 2.0'.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6.3 Regression Tests

Regression testing is the process of testing changes to computer programs to make sure that previously developed and tested parts of the program still works after the newly added changes. Regression testing is a normal part of the program development process and, in larger companies, it is done by code testing specialists. Test department coders develop code test scenarios and exercises that will test new units of code after they have been written. These test cases form what becomes the *test bucket*. Before a new version of a software product is released, the old test cases are run against the new version to make sure that all the old features still work.

Regression tests for Melodic release 2.0 were executed for those components of the Melodic platform in which the most substantial changes have been implemented. Parts of the old test cases were archived due to obsolete features, and new test cases referring to new functionality have been created. Table 7 provides a status overview of the regression tests of release 2.0.

Table 7: Status of regression tests executed during release 2.0

Summary	Type	Group	Status
Regression (release 2.0) : TwoComponentApp deployment on One Cloud Provider	Functional	Initial Deployment	Passed
Regression (release 2.0) : TwoComponentApp deployment on Two different Cloud Providers	Functional	Initial Deployment	Passed
Regression (release 2.0) : Utility function FCR	Functional	Reasoning Related	Passed
Regression (release 2.0) : Custom raw metric - FCR	Functional	Metric Management	Passed
Regression (release 2.0) : Temporary unavailability of BPM	Non-functional	Fault Handling	Passed
Regression (release 2.0) : Temporary unavailability of particular components	Non-functional	Fault Handling	Passed
Regression (release 2.0) : High Availability Component configuration	Non-functional	Fault Handling	Passed
Regression (release 2.0) : Global reconfiguration - FCR, time=3600s	Functional	Global Reconfiguration	Passed
Regression (release 2.0) : Counting Resource Overhead of Melodic instance introduced over its host	Non-functional	Performance	Passed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

7 Bugs

In Software testing, when the expected and the actual behavior is not matching, an incident will be raised. An incident may be a *bug*. It is a programmer's fault when a programmer intended to implement a certain behavior, but the code fails to correctly conform to this behavior because of incorrect implementation. It is also known as a *defect*.

7.1 Reported bugs

During release 2.0, 140 new bugs were reported. Table 8 shows the latest status of the corresponding bugs.

Table 8: List of bugs reported in release 2.0

Components	Summary	Priority	Status
Adapter	Missing "Authorization" header error when security is turned off	Highest	Closed
Adapter	NullPointer during adding Process: Response code 400 mapped as failure	High	Closed
Adapter	Job was not configured in Colosseum - schedule cannot be created	Medium	Closed
Adapter	NullPointer after successful deployment (Adapter)	Medium	Closed
Adapter	ApplySolution not received	High	Closed
Adapter	Runtime Exception in MonitorConverter in Adapter	Medium	Closed
Adapter	Error in Adapter while extracting response for monitors	Medium	Closed
Adapter	Wrong applicationId in adapter	Medium	Closed
Adapter	Pre-authorization : 'Adapter' cannot 'DEPLOY' resource 'deployment-model'	High	Closed
Adapter	Error in Adapter during reconfiguration - trying to create new machine instead of deleting	Medium	Closed
Adapter	Problem with creating ProcessTask for node in Adapter	Medium	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Adapter	NPE in the Adapter during deleting monitors	Medium	Closed
Adapter	Error after reconfiguration	Highest	Closed
Adapter	Adapter is continuously waiting for result from queue	Medium	Closed
Adapter	No such vertex exception during reconfiguration	Highest	Closed
Adapter	Wrong adapter task order	Medium	Closed
Camel editor	location.camel is not linked in the camel model	Medium	Closed
Camel editor, CP Generator	Add environment support (support missing)	Medium	Closed
Camel editor	Error by using new Camel - class OCLinEcoreEObjectValidator not found	High	Closed
CDO Server	Error during uploading xmi files (camel version 2)	Medium	Closed
CDO Server	CDO image not pullable	High	Closed
CDO Server	CDO NPE when scaling was triggered	Medium	Closed
CDO Server	CDO NPE	Medium	Closed
CDO Server	‘The string resource_ UI duplicate” cloud not located error	Medium	Closed
CDO Server, CP Generator	CDO Server error - "Too many tables; MySQL can only use 61 tables in a join"	High	Closed
Cloudiator	Unable to create a VM with specified number of disk storage	Medium	Won’t Fix
Cloudiator	Docker-compose.yml file update error	High	Closed
Cloudiator	Lance agent fail to deploy	Low	Closed
Cloudiator	Error during deleting Node: compute service not found	Medium	Closed

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731664

Cloudiator	Error during reading metric in visor	High	In progress
Cloudiator	Openstack: error during connection	Medium	Closed
Cloudiator	MySQL's image problem during docker deployment	Lowest	Postponed
Cloudiator	NPE in monitoring-agent	High	Closed
Cloudiator	Failure during deletion of node	Low	In Progress
Cloudiator	Spark deployment - Duplicate entry 'spark-dummy-id'	Medium	Closed
Cloudiator	FaaS agent compile errors due to changes in common	Medium	Closed
Cloudiator	Cloudiator 2 - Error when retrieving Node Candidates	Medium	Closed
Cloudiator	Checking queue status – wrong process id	High	Closed
Cloudiator	Process ids do not match with corresponding ids from queues	High	Closed
Cloudiator	Unexpected error while processing request userId: "admin"	High	Closed
Cloudiator	Queue status not refreshed for GET /queue	Medium	Closed
Cloudiator	nodeGroups incorrectly updated	High	Closed
Cloudiator	Improper LB comunication (FCRnew application)	High	Closed
Cloudiator	Error sending message on topic NodeCandidateRequestResponse	High	Closed
Cloudiator	Node candidate query - RecordTooLargeException	High	Closed
Cloudiator	preInstallCommand doesn't work	High	Closed
Cloudiator	RequirementAttribute in findNodeCandidate method doesn't respond	High	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloudiator	Various locations for Node Candidates in Cloudiator2 needs to be added	Medium	Closed
Cloudiator	Cannot determine model class of name: <SINGLE>	High	Closed
Cloudiator	Failure during installation of Cloudiator tools	Medium	Closed
Cloudiator	Deployment of docker process fails: "Container reached illegal state CREATION_FAILED while waiting for state READY"	High	Closed
Cloudiator	Cannot deploy docker components: invalid publish opts format	Medium	Closed
Cloudiator	Cannot create Docker Job - "Unknown entity: io.github.cloudiator.persistance.DockerTaskInterfaceModel"	High	Closed
Cloudiator	Node-agent: receiving new node request - NullPointer	Medium	Closed
Cloudiator	Error during starting node – impossible to set up nodes	Highest	Closed
Cloudiator	Docker deployment - unable to resolve host	High	Closed
Cloudiator	Spark arguments deployment – improper naming of parameters	Medium	Closed
Cloudiator	Openstack instance doesn't work properly	Medium	Closed
Cloudiator	Exception during installing Docker Interface	High	Closed
Cloudiator	Spark deployment - application arguments incorrect order	High	Closed
Cloudiator	External IP during starting Node doesn't work	High	Closed
Cloudiator	Dynamic ENV variables not set	High	Closed
Cloudiator	Cloud with id {ID} was not found	High	Closed
Cloudiator	Spark deployment - unknown host exception	High	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloudiator	Script configuration application: error during performing install command - E: dpkg was interrupted	High	Closed
Cloudiator	Unable to install application on Amazon image	High	Closed
Cloudiator	Error during execution of virtual machine creation	Medium	Closed
Cloudiator	Spark application doesn't work with 1,2G data	Medium	Closed
Cloudiator	Security group doesn't open port 80	Medium	Closed
Cloudiator	Error creating lifecycle client - "Connection refused"	Highest	Closed
Cloudiator	NPE in lance - IMPORTANT	Highest	Closed
Cloudiator	Monitors - unable to get	High	Closed
Cloudiator	Exception Could not deploy task TaskImpl for Spark	High	Closed
Cloudiator	Cloudiator v0.2.0 error during deploying an application	High	Closed
Cloudiator	Error during applicationDeploymentNotificationRequest on OpenStack	Medium	Closed
Cloudiator	Incorrect value of cores reported by the Cloudiator 2.0 discovery service	Lowest	Postponed
Cloudiator	Spark deployment - multiple application added for N-workers deployment	Medium	In Progress
Cloudiator	Public-private IPs issue when deploying from UiO to AWS	Highest	Closed
Cloudiator	JMS needs to be added as a different value	Highest	Closed
Cloudiator	Spark app deployment issue	Medium	Closed
Cloudiator	Native application deployment – spark doesn't work correctly	Medium	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloudiator	Docker installation process hangs on docker_retry.sh script	Lowest	Postponed
Cloudiator	Port 8998 is not listen for Spark-agent	High	Closed
Cloudiator	Exception during process removing	High	Closed
Cloudiator	UpdateAction not executed	High	Closed
Cloudiator	Error during creating node lambda – improper naming	Medium	Closed
Cloudiator	Cloudiator 2 default logging causes huge memory usage on hard disk	Medium	Closed
Cloudiator	NodeType is null	Medium	Closed
Cloudiator	Missing notification after removing process	High	Closed
Cloudiator	AddMonitor for the same metric on different process fails	Highest	Closed
Cloudiator	Lance app – improper version in scheduler-agent	High	Closed
Cloudiator	NPE in node-agent	Medium	Closed
Cloudiator	Livy can't deploy pySpark app	Medium	Closed
Cloudiator	Error by killing the process	High	Closed
Cloudiator	Spark-master container not started	Medium	Closed
Cloudiator	Failure during deployment on OpenStack	Medium	Closed
Cloudiator	Installation fails on nodes	Low	Closed
Cloudiator	Error during creating lifecycle client	Medium	Closed
Cloudiator	Creation and updates on nodes very slow	High	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloudiator	Lance agent doesn't execute preInstall script	Medium	Closed
Cloudiator	Cross cloud deployment incorrect value	Medium	Closed
Cloudiator	No communication is attached to this port thus the req is never fulfilled	Medium	Closed
Cloudiator	SQL Constraint Exception with Monitors in monitoring-agent	Highest	Closed
Cloudiator	Instance error (docker container failed) not propagated	Medium	New
Cloudiator	Baguette client is sometimes installed on only one of two instances	Medium	Closed
Cloudiator	Cloudiator doesn't start with initial drestart after fresh installation	Medium	Closed
Cloudiator	Cloudiator first run fails iflux container	Low	Closed
CP Solver, CP Generator	Segmentation fault in CPSolver	Highest	Closed
CP Generator	NullPointer when Camel Model has no application model	Low	Closed
CP Generator	NPE during creation a constraint	Medium	Closed
CP Generator	Error during creating image (docker requirements) - ExampleWithDocker	Medium	Closed
CP Solver	CpSolver crashes	High	Reopen
CP Solver	NullPointer exception during finding the cheapest Node Candidate	High	Closed
CP Solver	Exception in CPSolver when Metric Model is missing	Medium	Closed
DLMS	CDO doesn't see xmi files when DLMS component is integrated	Medium	Closed
DLMS	404 Error during sending notification	High	Closed
DLMS	NullPointerException in DlmsDiffBundle class	Medium	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

DLMS	Incorrect dataModelNotification in the DLMS	Medium	Closed
DLMS	DLMSUtility does not return the utility value during the initial deployment	Medium	Closed
DLMS	Alluxio fails to register S3 data source	Medium	Closed
EMS	Deserializing MonitorsDataResponseImpl object – new value is needed	High	Closed
EMS	ActiveMQ – could not connect to broker URL	Medium	Closed
EMS	Metrics not received at deployed machine's EMS	Medium	Closed
EMS	EMS does not take metrics from the last interval	Medium	Closed
EMS	NPE in EMS logs	Medium	Closed
EMS	Eu.melodic.event.translate.analyze.Model is not implemented	Medium	Closed
SolverTo Deployment	Solver2DDeployment has wrong locale/format	Medium	Closed
EMS	Missing header with token in request with metric to Metasolver	Medium	Closed
Environment	Paused Servers on OpenStack	Highest	Closed
Environment	Connection reset by peer	Low	In progress
Environment	Empty NC exception with AWS	Medium	Closed
Meta Solver	ActiveMQSession in Metasolver is closed	Medium	Closed
Meta Solver	NPE in Metasolver during updating metric values	Medium	Closed
Meta Solver	Metasolver does not update all needed metric values in CP model	High	Closed
Spark	Postman (Spark sample collation) - requests update	Low	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Spark	UiO dualstack network configuration	Medium	Closed
Utility Generator	NullPointerException in DLMSConverter class	Medium	Closed
JIRA	Jira's workflow incomplete	High	Closed
Mule	Mule exception "UnresolvedAddressException"	Medium	Closed
Camel Editor	Camel new-oxygen build failed	Medium	Closed
OpenStack	Issue during deployment on Openstack – improper credentials	Medium	Closed

Table 9 lists bugs reported during earlier releases of Melodic (release 1.0 and release 1.5) and their current statuses as of Melodic Release 2.0.

Table 9: List of bugs reported during previous releases and resolved in release 2.0

Components	Summary	Priority	Status
Meta Solver	Meta solver TimeoutException	Highest	Closed
Cloudiator	Incorrect value of cores reported by the Cloudiator 2.0 discovery service	Medium	Closed
Cloudiator	Unable to create a VM with specified number of disk storage	Medium	Won't fix
Esper	Unable to fully deploy applications on t2.micro machine types	Medium	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

8 Summary

This deliverable presents the Melodic Release 2.0 and the corresponding new test cases for the release. The following information has been provided in the document:

1. An introduction to Melodic release 2.0
2. A high-level overview of the Melodic architecture
3. The testing environments for Melodic release 2.0
4. A Melodic platform installation guide
5. Testing guides on how to execute Test Cases with attached CAMEL Models
6. Test Cases executed for Melodic release 2.0
7. Reported bugs in release 2.0 and regression tests

9 References

[1] F. Zahid et al., "D2.2 Architecture and initial feature definitions", The Melodic H2020 Project Deliverable D2.2, 2018.

[2] P. Skrzypek et al., "D5.03 Security requirements & design", The Melodic H2020 Project Deliverable D5.03, 2018.

[3] M. Jakubczyk et al., "D5.10 Quality Assurance Guide", The Melodic H2020 Project Deliverable D5.10, 2017

[4] K. Materka et al., “D5.06 Test strategy and Environnment”, The Melodic H2020 Project Deliverable D5.06, 2018

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731664

Appendix A – Test Cases Release 2.0

DL.1 [T] Successfully execute data life-cycle event using DLMS Agent on the nodes

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one data source registered in the DLMS.
3. At least one component-data source relation is defined.
4. At least one life-cycle event registered in the DLMS agent.

Steps To Complete:

1. Login to the machine with installed MELODIC.
2. Register an event using the REST interface of the DLMS agent on a commissioned VM.

Expected results:

1. Script defined in the life-cycle event executed.

U.5 [T] Successfully retrieve utility value at utility generator based on cost algorithm

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least two cloud providers used in the deployment.
3. At least one data source registered in the DLMS.
4. At least one component-data source relation is defined.

Steps To Complete:

1. Login to the machine with installed MELODIC.

Expected results:

1. A double value representing the utility received.

U.4 [T] Successfully retrieve utility value at utility generator based on total utility

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least five historical deployments.
3. Total utility metric configured.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed MELODIC.

Expected results:

1. A double value representing the utility received.

U.3 [T] Successfully retrieve utility value at utility generator based on data-aware algorithm

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one data source registered in the DLMS
3. At least one component-data source relation is defined
4. Dynamic data size using CAMEL / Metadata schema

Steps To Complete:

1. Login to the machine with installed MELODIC
2. Deploy PeopleFlow.xmi (attached)

Expected results:

1. A double value representing the utility received

U.2 [T] Successfully retrieve utility value at utility generator based on dc-aware algorithm

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least two locations used in the deployment.

Steps To Complete:

1. Login to the machine with installed MELODIC.

Expected results:

1. A double value representing the utility received.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

U.1 [T] Successfully retrieve utility value at utility generator based on affinity-aware algorithm

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one data source registered in the DLMS.
3. At least one component-data source relation is defined.

Steps To Complete:

1. Login to the machine with installed MELODIC.
2. Deploy PeopleFlow.xmi (attached).

Expected results:

1. A double value representing the utility received.

N.1 [T] Successfully mount data sources under alluxio cluster

Input Conditions:

1. Melodic Configured and Running.
2. At least one data source registered in the DLMS.
3. Alluxio has the directory melodic. Otherwise, create it with command:=> alluxio fs mkdir /melodic.

Steps To Complete:

1. Login to the machine with installed MELODIC.
2. Deploy PeopleFlow.xmi (attached).

Expected results:

1. The data source is successfully mounted.
2. Log has the following information: Mounted s3a://datasourcetest at /melodic/S3BucketData
3. The command "alluxio fs ls /melodic/S3BucketData" shows the following: DIR /melodic/S3BucketData/test

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

M.3[T] Successfully read DataModel from CAMEL and register data sources in the DLMS

Input Conditions:

1. Melodic Configured and Running.
2. At least one data source registered in the DLMS.
3. Alluxio has the directory melodic. Otherwise, create it with command:=> alluxio fs mkdir /melodic.

Steps To Complete:

1. Login to the machine with installed MELODIC.
2. Deploy the attached PeopleFlow2.xmi.
3. Use the following URL to check the list of stored data sources: <url>/ds

Expected results:

1. The data source is successfully mounted.
2. Log has the following information: Mounted s3a://datasourcetest at /melodic/S3BucketData

DEP4 [T] Deploy dockerized application from private registry

Input Conditions:

1. Image in private registry present.

Steps To Complete:

1. Create a Job with the Docker-Interface and specify the private registry with valid credentials to retrieve image.

Expected results:

1. Application successfully deployed from private registry, running and usable.

DEP3 [T] Deploy application with native and dockerized component

Input Conditions:

1. Scripts present.
2. Image present.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Create a job which contains a task based on the LanceInterface and a second component based on the DockerInterface.
2. Check DEP1 and DEP2 for examples.

Expected results:

1. Application and both components successfully deployed, running and usable

DEP2 [T] Deploy dockerized application

Input Conditions:

1. Image present in public Docker repository.

Steps To Complete:

1. Create a Job with the DockerInterface, examples need to be added. *http://cloudiator.org/rest-swagger/#operation/addJob*

Sample body

```
// job

{
    "name": "TwoCompDocker_JOB",
    "tasks": [{
            "name": "Component_MySql",
            "executionEnvironment": "DOCKER",
            "taskType": "SERVICE",
            "ports": [
                    {
                        "port":3306,
                        "type":"PortProvided",
                        "name":"ComponentMySqlPort"
                    }
                ],
            "interfaces": [{
                    "type": "DockerInterface",
                    "dockerImage": "mariadb",
                    "environment":{
                        "MYSQL_ROOT_PASSWORD": "admin",
                        "MYSQL_USER": "melodic",
                        "MYSQL_PASSWORD": "testpwd",
                        "MYSQL_DATABASE": "wordpress",
                        "port": "3306:3306"
                    }
                }
            ]
        },
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
        {
            "name": "Component_Wordpress",
            "executionEnvironment": "DOCKER",
            "taskType": "SERVICE",
            "ports": [
                    {
                        "isMandatory":true,
                        "type":"PortRequired",
                        "name":"DBRequiredPort"
                    },
                    {
                        "port":80,
                        "type":"PortProvided",
                        "name":"ComponentWordpressPort"
                    }
                ],
            "interfaces": [{
                    "type": "DockerInterface",
                    "dockerImage": "wordpress",
                    "environment":{
                        "WORDPRESS_DB_HOST":
"$PUBLIC_DBRequiredPort",
                        "WORDPRESS_DB_USER": "melodic",
                        "WORDPRESS_DB_PASSWORD": "testpwd",
                        "WORDPRESS_DB_NAME": "wordpress",
                        "port":"80:80"
                    }
                }
            ]
        }
    ],
    "communications": [
        {
            "portRequired":"DBRequiredPort",
            "portProvided":"ComponentMySqlPort"
        }

 ]
}
```

Expected results:

1. Application successfully deployed, running and usable.

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 731664

DEP1 [T] Deploy native application

Input Conditions:

1. UULM OpenStack configured as Cloud.
2. Node Candidates retrieved successfully.
3. Scripts present.

Steps To Complete:

1. Create (native) job via LanceInterface.

Sample body

```
{
 "name":"mediawiki",
 "tasks":[ {
  "name":"loadbalancer",
  "executionEnvironment":"SPARK",
  "taskType":"SERVICE",
  "ports":[ {
   "type":"PortProvided",
   "name":"LBPROV",
   "port":80
   },
   {
   "type":"PortRequired",
   "name":"LOADBALANCERREQWIKI",
   "isMandatory":"false",
   "updateAction":"./mediawiki-tutorial/scripts/lance/nginx.sh configure"
   }
  ],
  "interfaces":[ {
   "containerType":"DOCKER",
   "type":"LanceInterface",
   "preInstall":"sudo apt-get -y update && sudo apt-get -y install git && git clone
https://github.com/dbaur/mediawiki-tutorial.git",
    "install":"./mediawiki-tutorial/scripts/lance/nginx.sh install",
   "start":"./mediawiki-tutorial/scripts/lance/nginx.sh startBlocking"
   }
  ],
  "requirements":[ {
   "constraint":"nodes->forAll(hardware.cores <= 2)",
   "type":"OclRequirement"
  },
  {
   "constraint":"nodes->forAll(hardware.ram <= 2048)",
   "type":"OclRequirement"
  }
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
 ] },
 {
  "name":"wiki",
  "executionEnvironment":"SPARK",
  "taskType":"SERVICE",
  "ports":[ {
   "type":"PortRequired",
   "name":"WIKIREQMARIADB",
   "isMandatory":"true"
   },
   {
   "type":"PortProvided",
   "name":"WIKIPROV",
   "port":80
  } ],
  "interfaces":[ {
   "type":"LanceInterface",
   "containerType":"DOCKER",
   "preInstall":"sudo apt-get -y update && sudo apt-get -y install git && git clone
https://github.com/dbaur/mediawiki-tutorial.git",
   "install":"./mediawiki-tutorial/scripts/lance/mediawiki.sh install",
   "postInstall":"./mediawiki-tutorial/scripts/lance/mediawiki.sh configure",
   "start":"./mediawiki-tutorial/scripts/lance/mediawiki.sh startBlocking"
  } ],
  "requirements":[ {
   "constraint":"nodes->size() >= 5",
   "type":"OclRequirement"
   },
   {
   "constraint":"nodes->forAll(hardware.cores <= 2)",
   "type":"OclRequirement"
   },
   {
   "constraint":"nodes->forAll(hardware.ram <= 2048)",
   "type":"OclRequirement"
  } ]
 },
 {
  "name":"wordCount",
  "executionEnvironment":"SPARK",
  "taskType":"SERVICE",
  "ports":[ {
   "type":"PortRequired",
   "name":"SPARKREQMARIADB",
   "isMandatory":"true"
  } ],
  "interfaces":[ {
   "type":"SparkInterface",
   "file":"http://example.com",
```

```
    "className":"MainClass"
   } ],
   "requirements":[ {
    "constraint":"nodes->forAll(hardware.cores <= 2)",
    "type":"OclRequirement"
    },
    {
    "constraint":"nodes->forAll(hardware.ram <= 2048)",
    "type":"OclRequirement"
   } ]
   }, {
   "name":"database",
   "executionEnvironment":"SPARK",
   "taskType":"SERVICE",
   "ports":[ {
    "type":"PortProvided",
    "name":"MARIADBPROV",
    "port":3306
   } ],
   "interfaces":[ {
    "type":"LanceInterface",
    "containerType":"DOCKER",
    "preInstall":"sudo apt-get -y update && sudo apt-get -y install git && git clone
https://github.com/dbaur/mediawiki-tutorial.git",
    "install":"./mediawiki-tutorial/scripts/lance/mariaDB.sh install",
    "postInstall":"./mediawiki-tutorial/scripts/lance/mariaDB.sh configure",
    "start":"./mediawiki-tutorial/scripts/lance/mariaDB.sh startBlocking"
   } ],
   "requirements":[ {
    "constraint":"nodes->forAll(hardware.cores <= 2)",
    "type":"OclRequirement"
    }, {
    "constraint":"nodes->forAll(hardware.ram <= 2048)",
    "type":"OclRequirement"
   } ]
  } ],
  "communications":[ {
   "portRequired":"WIKIREQMARIADB",
   "portProvided":"MARIADBPROV"
   }, {
   "portRequired":"LOADBALANCERREQWIKI",
   "portProvided":"WIKIPROV"
   }, {
   "portRequired":"SPARKREQMARIADB",
   "portProvided":"WIKIPROV"
  } ]
}
```


This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 731664

Expected results:

1. Application successfully deployed, running and usable.

DLMS1 [T] Deploy working DLMS Agent

Input Conditions:

1. Node was created (see VM creation).

Steps To Complete:

1. Trigger DLMS Agent installation for the created node via *http://cloudiator.org/rest-swagger/#operation/installTools*

Expected results:

1. SSH into node and verify that DLMS agent is running.

BD5 [T] Execute Java based Spark Job

Input Conditions:

1. UULM OpenStack configured as Cloud.
2. Node Candidates retrieved successfully.
3. Node Candidates are configured via API for the use with Spark.

Steps To Complete:

1. Execute Java create job:

Sample body

```
// Java - create job
{
  "name": "JAVA_JOB",
  "tasks": [{
        "name": "Component_Worker",
        "executionEnvironment": "SPARK",
        "taskType": "BATCH",
        "ports": [],
        "interfaces": [{
              "type": "SparkInterface",
              "file": "https://s3-eu-west-
1.amazonaws.com/melodic.testing.data/mdfs/spark_java/SparkPi/sparkpi_2.10-1.0.jar",
```

```
            "className": "SparkPi",
            "arguments": [10],
            "sparkArguments":{},
            "sparkConfiguration":{}
          }
        ]
    }
  ],
  "communications": [
 ]
}
```


Expected results:

1. Java app successfully executed, which can be verified in the Spark Master dashboard at: *http://MELODIC_PLATFORM_IP:8181* and Livy Server *http://MELODIC_PLATFORM_IP:8998*

VM3 [T] Create VM over two different provider's resources

Input Conditions:

1. Profitbricks configured as Cloud.
2. Node Candidates retrieved successfully.
3. UULM OpenStack configured as Cloud.
4. Node Candidates retrieved successfully.

Steps To Complete:

1. Deploy application, for example FCRnew.xmi or TwoComponentApp.xmi, using two Cloud Providers: OpenStack and Profitbricks.

Expected results:

1. Two VMs are successfully created and available.

VM2 [T] Create VM on UULM OpenStack resource

Input Conditions:

1. UULM OpenStack configured as Cloud
2. Node Candidates retrieved successfully

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To complete:

1. Deploy application using OpenStack Cloud Provider, for example FCRnew.xmi or TwoComponentAppnew.xmi.

Expected results:

1. VM is successfully created and available.

BD4 [T] Execute Python based Spark Job

Input Conditions:

1. UULM OpenStack configured as Cloud
2. Node Candidates retrieved successfully
3. Node Candidates are configured via API for the use with Spark

Steps To Complete:

1. Execute Python based Spark Job:

Sample body

```
// pi.py
{
  "name": "PI_JOB",
  "tasks": [{
        "name": "Component_Worker",
        "executionEnvironment": "SPARK",
        "taskType": "BATCH",
        "ports": [],
        "interfaces": [{
              "type": "SparkInterface",
              "file": "https://s3-eu-west-1.amazonaws.com/melodic.testing.data/mdfs/pi.py",
              "className": "",
              "arguments": ["10"],
              "sparkArguments":{},
              "sparkConfiguration":{}
           }
        ]
     }
  ],
  "communications": [

 ]
}
```

Expected results:

1. Python app successfully executed, which can be verified in the Spark Master dashboard *at http://MELODIC_PLATFORM_IP:8181* and Livy Server: *http://MELODIC_PLATFORM_IP:8998*

VM1 [T] Create VM on ProfitBricks resource

Input Conditions:

1. Profitbricks configured as Cloud
2. Node Candidates retrieved successfully

Steps To Complete:

1. Deploy application, for example FCRnew.xmi or TwoComponentApp.xmi, using Profitbricks Cloud Provider.

Expected results:

1. VM is successfully created and available.

NC3 [T] Successfully retrieve Node Candidates for ProfitBricks

Input Conditions:

1. ProfitBricks Cloud configured.

Steps To Complete:

1. Configure Profitbricks Cloud with credentials against $IP:9000/clouds:

Sample body

```
{
	"cloudType": "PUBLIC",
	"api": {
		"providerName": "profitbricks-rest"
	},
	"credential": {
		"user": "censoredUser",
		"secret": "censoredPassword"
	},
	"cloudConfiguration": {
		"nodeGroup": "cloudiator",
		"properties": null
	}
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2. Retrieve configured clouds that match the previously configured one.
3. Retrieve node candidates with constraints:

Sample body

```
[
  {
    "constraint": "nodes->forAll(hardware.cores >= 2)",
    "type": "OclRequirement"
  },
  {
    "constraint": "nodes->forAll(hardware.cores <= 4)",
    "type": "OclRequirement"
  },
  {
    "constraint": "nodes->forAll(hardware.ram = 2048)",
    "type": "OclRequirement"
  }
]
```

4. Verify that no node candidates are returned.

Expected results:

1. A list of nodes for the provided identity is returned.

NC2 [F] Assure that wrong cloud configuration forbids retrieval of Node Candidates

Input Conditions:

1. UULM OpenStack Cloud configured with wrong password.

Steps To Complete:

1. Configure erroneous cloud with credentials against **$IP:9000/clouds**:

Sample body

```
{
        "endpoint": "https://omistack.e-technik.uni-ulm.de:5000/v3/",
        "cloudType": "PRIVATE",
        "api": {
                "providerName": "openstack4j"
        },
        "credential": {
                "user": "default:74f1d8b1b33c413faca7603c83362a74:doe",
                "secret": "347232"
        },
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
        "cloudConfiguration": {
                "nodeGroup": "cloudiator",
                "properties": null
        },
        "id": "91e344c475c8173749952538b544e213"
}
```


2. Retrieve configured clouds that match the previously configured one.
3. Retrieve node candidates with constraints:

Sample body


```
[
        {
            "constraint": "nodes->forAll(hardware.cores >= 2)",
            "type": "OclRequirement"
        },
        {
            "constraint": "nodes->forAll(hardware.ram >= 2048)",
            "type": "OclRequirement"
        }
]
```


4. Verify that no node candidates are returned.

Expected results:

1. Empty result list.

NC1 [T] Successfully retrieve Node Candidates for UULM OpenStack

Input Conditions:

1. UULM OpenStack Cloud configured.

Steps To Complete:

1. Configure UULM OpenStack with credentials against **$IP:9000/clouds.**
2. Retrieve configured clouds that match the previously configured one.
3. Retrieve node candidates with constraints:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Sample body

```
[
        {
            "constraint": "nodes->forAll(hardware.cores >= 2)",
            "type": "OclRequirement"
        },
        {
            "constraint": "nodes->forAll(hardware.ram >= 2048)",
            "type": "OclRequirement"
        }
]
```

4. Verify that the returned node candidates meet constraints.

Expected results:

1. A list of nodes for the provided identity is returned.

BD3 [T] Use maximum resources available for Spark Job

Input Conditions:

1. UULM OpenStack configured as Cloud.
2. Node Candidates retrieved successfully.
3. Node Candidates are configured via API for use with Spark.

Steps To Complete:

1. Execute Spark Job with no specific resource requirements, e.g:

Sample body

```
{
  "name": "pi","executionEnvironment": "SPARK","taskType": "BATCH",
  "ports": [
  ],
  "interfaces": [ {
    "type": "SparkInterface",
    "file": "https://omi-gitlab.e-technik.uni-ulm.de/cloudiator/apache-livy-
container/raw/master/examples/pi.py",
    "className": "bla", "arguments": ["10"], "sparkArguments":{
     },
    "sparkConfiguration":{ }
  } ],
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. Spark job was successfully executed and Spark Master UI shows the usage of the maximum available resources at: *http://MELODIC_PLATFORM_IP:8181*

BD2 [T] Execute Spark Job with concrete Spark Job requirements

Input Conditions:

1. UULM OpenStack configured as Cloud.
2. Node Candidates retrieved successfully.
3. Node Candidates are configured via API for the use with Spark.

Steps To Complete:

1. Create Spark Process Request: http://cloudiator.org/rest-swagger/#operation/createProcess with concrete requirements, e.g.

Sample body

```
{
  "name": "pi",
  "executionEnvironment": "SPARK",
  "taskType": "BATCH",
  "ports": [
  ],
  "interfaces": [ {
    "type": "SparkInterface",
    "file": "https://omi-gitlab.e-technik.uni-ulm.de/cloudiator/apache-livy-container/raw/master/examples/pi.py",
    "className": "bla",
    "arguments": ["10"],
    "sparkArguments":{
      "driverCores" : "1",
      "numExecutors": "1"},
    "sparkConfiguration":{}
  } ],
}
```

2. Get queued task: http://cloudiator.org/restswagger/#operation/getQueuedTasks
3. Create Schedule: http://cloudiator.org/rest-swagger/#operation/addSchedule

Expected results:

1. Spark job was successfully executed and Spark Master UI shows the correct resources at: http://MELODIC_PLATFORM_IP:8181

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

BD1 [T] Deploy Spark Cluster for 2 Spark Workers

Input Conditions:

1. UULM OpenStack configured as Cloud.
2. Node Candidates retrieved successfully.
3. Node Candidates are configured via API for the use with Spark.

Steps To Complete:

1. Create Spark Process Request: http://cloudiator.org/rest-swagger/#operation/createProcess

Sample body


```
{
  "name": "spark-conf-test-5",
  "tasks": [ {
    "name": "pi",
    "executionEnvironment": "SPARK",
    "taskType": "BATCH",
    "ports": [
    ],
    "interfaces": [ {
      "type": "SparkInterface",
      "file": "https://omi-gitlab.e-technik.uni-ulm.de/cloudiator/apache-livy-container/raw/master/examples/pi.py",
      "className": "bla",
      "arguments": ["10"],
      "sparkArguments":{
         "driverCores" : "1",
         "numExecutors": "1" },
      "sparkConfiguration":{}
    } ],
    "requirements": [ {
      "constraint": "nodes->forAll(hardware.providerId = 'df09fccf-1a81-42ab-b16b-6932f371a1c8')",
      "type": "OclRequirement"
      }, {
      "constraint": "nodes->forAll(location.providerId = 'nova')",
      "type": "OclRequirement"
      }, {
      "constraint": "nodes->forAll(image.providerId = 'f688f98d-7e62-4404-a672-1fc054fcfa6c')",
      "type": "OclRequirement"
    } ]
  } ],
  "communications": [
  ]
}
```

2. Get queued task: http://cloudiator.org/rest-swagger/#operation/getQueuedTasks
3. Create Schedule: http://cloudiator.org/rest-swagger/#operation/addSchedule

Expected results:

1. Spark Cluster is available and accessible via the Spark Master Dashboard which shows the two worker nodes at: http://MELODIC_PLATFORM_IP:8181

AC-13[F] At least one node of Deployment Instance Model (given to Adapter) has less than 16 GB RAM

Input Conditions:

1. Upperware installed and configured.
2. CAMEL model for a 2-component app, including a constraint that a VM has a maximum of 8 GB RAM.
3. Pre-authorization Policy that permits models where each node has >16 GB RAM.

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-12+13.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Rejected.

AC-12[T] The nodes of Deployment Instance Model (given to Adapter) have 16 GB RAM or more (each)

Input Conditions:

1. Upperware installed and configured.
2. CAMEL model for a 2-component app, including a constraint that every VM has at least 16 GB RAM.
3. Pre-authorization Policy that permits models where each node has >16 GB RAM.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-12+13.xml?at=RC2.0
3. Upload model of 2-component app to CDO.
4. Start deployment process.

Expected results:

1. Deployment plan being Accepted.

AC-11[F] Deployment Instance Model (given to Adapter) has a total number of GB RAM (across all nodes) higher than 64

Input Conditions:

1. Upperware installed and configured.
2. CAMEL model for a 2-component app, including a constraint that there will be at least 2 VM instances per component and every VM instance has at least 32 GB RAM.
3. Pre-authorization Policy that permits models where total RAM sums up to 64 GB RAM.

Steps To Complete:

1. Login to the machine with installed Melodic.
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-10+11.xml?at=RC2.0
3. Upload model of 2-component app to CDO.
4. Start deployment proces.

Expected results:

1. Deployment plan being Rejected.

AC-10[T] Deployment Instance Model (given to Adapter) has a total number of GB RAM (across all nodes) lower or equal than 64

Input Conditions:

1. Upperware installed and configured.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2. CAMEL model for a 2-component app.
3. Pre-authorization Policy that permits models where total RAM sums up to 64 GB RAM.

Steps To Complete:

1. Login to the machine with installed Melodic.
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-10+11.xml?at=RC2.0
3. Upload model of 2-component app to CDO.
4. Start deployment process.

Expected results:

1. Deployment plan being Accepted.

AC-9[F] Deployment Instance Model (given to Adapter) contains at least one node with NO location information

Input Conditions:

1. Upperware installed and configured.
2. CAMEL model for a 2-component app:
   (1) with no location constraints
   (2) with such a constraint that it will lead to selecting a node candidate that (we know) has no location information (eg. BYON candidate).
3. Pre-authorization Policy that permits models where all nodes are located in DE.

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-07+08+09.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Rejected.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

AC-8[F] Deployment Instance Model (given to Adapter) contains at least one node NOT located in DE

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app, including a constraint requiring that at least one VM is NOT located in DE
3. Pre-authorization Policy that permits models where all nodes are located in DE

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-07+08+09.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Rejected

AC-7[T] Deployment Instance Model (given to Adapter) contains node located in DE (all of them)

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app, including constraints requiring that every VM is located in DE
3. Pre-authorization Policy that permits models where all nodes are located in DE

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-07+08+09.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. Deployment plan being Accepted

AC-6[F] At least one node of Deployment Instance Model (given to Adapter) has exactly 1 core

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app, including a constraint that every VM has 1 core
3. Pre-authorization Policy that permits models where all nodes have >1 cores each

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-05+06.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Rejected

AC-5[T] The nodes of Deployment Instance Model (given to Adapter) have more than 1 cores (each)

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app, including a constraint that every VM has at least 2 cores
3. Pre-authorization Policy that permits models where all nodes have >1 cores each

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-05+06.xml?at=RC2.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Accepted

AC-4[F] Deployment Instance Model (given to Adapter) has a total number of cores (across all nodes) higher than 4.

Input Conditions:

1. Melodic installed and configured
2. CAMEL model for a 2-component app, including a constraint that every VM has at least 3 cores
3. Pre-authorization Policy that permits models where total number of cores is lower or equal to 4 cores

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-03+04.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Rejected

AC-3[T] Deployment Instance Model (given to Adapter) has a total number of cores (across all nodes) lower or equal than 4

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app
3. Pre-authorization Policy that permits models where total number of cores sums up to 4 cores

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-03+04.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. Deployment plan being Accepted

AC-2[F] Successfully connect to Authorization Server and get a Negative response

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app
3. Pre-authorization Policy that permits all requests

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-02.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. See log of authorization server that deployment plan being Rejected (i.e. get a negative response)

AC-1[T] Successfully connect to Authorization Server and get a Positive response

Input Conditions:

1. Upperware installed and configured
2. CAMEL model for a 2-component app
3. Pre-authorization Policy that permits all requests

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed Melodic
2. Apply pre-authorization policy from link: https://bitbucket.7bulls.eu/projects/MEL/repos/security/browse/authorization-service/server/src/main/resources/config/policies/test-cases/tc-ac-01.xml?at=RC2.0
3. Upload model of 2-component app to CDO
4. Start deployment process

Expected results:

1. See log of authorization server that deployment plan being Pre-authorized (i.e. get a positive response)

T1.5a[T] Installation and deployment of FCR application in Docker containers on two different Cloud Providers

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least two cloud providers integrated with the Melodic platform, where the user has provided his/her own credentials for both of them (included in the CAMEL model cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to the given Cloud Providers.
5. Complete CAMEL model of the FCR application (which includes the definition of these two components and their installation/maintenance scripts). The two components are installed as Docker containers in two VM instances of different VM offerings.
6. FCR CAMEL model can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/FCR?at=refs%2Fheads%2FRC2.0
7. There should be a requirement in the application to use different Cloud Providers (e.g., a location requirement in the virtual machine requirement set in the user CAMEL model).

Steps To Complete:

1. Login to the machine with installed MELODIC platform
2. Upload models into "models" directory
3. Upload Camel model FCR.xmi into "models" directory
4. Using SoapUI tool execute following steps:
5. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
6. Using POST method start process
7. Start deploying of application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

For each step, the status of the executed action should be positive.

Expected results:

1. Two VM instances should be created using the selected Cloud Provider.
2. The application should be installed on those VM instances (where each application component is deployed and installed on a different VM instance).
3. The application should be run properly (The login page of DAM application should be displayed properly).

T1.5b[T] Installation and deployment of FCR application, where one component is installed in a Docker container and another on a normal VM on two different Cloud Providers

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider integrated with the Melodic platform, where the user has provided his/her own credentials for this provider (included in CAMEL model cloud credentials).
3. Meta solver configured to use CP solver for that case.
4. Cloudiator properly connected to given Cloud Providers.
5. Complete CAMEL model of FCR application (which includes the definition of just one component along with its installation/management scripts). One application container will be installed as a Docker container, another as a unix process. Each component is to be deployed on different cloud providers.
6. FCR CAMEL model can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/FCR?at=refs%2Fheads%2FRC2.0
7. There should be a proper configuration of the virtual machine both in the CAMEL Providers model and on the Cloud Providers sides.

Steps To Complete:

1. Login to the machine with installed MELODIC platform
2. Upload models into "models" directory
3. Upload Camel model FCR.xmi into "models" directory
4. Using SoapUI tool execute following steps:
5. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
6. Using POST method start process
7. Start deploying of application

For each step, the status of the executed action should be positive.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. Two virtual machine instances should be created, one instance per Cloud Provider.
2. The application should be installed on those VM instances (the instance of the first component should be installed on one VM instance and the instance of the second to the other VM instance).
3. The application should be run properly (The login page of the DAM application should be displayed).
4. The specific feature defined in the input CAMEL model is properly applied.

T1.4[T] Installation and deployment of FCR application in Docker container on one Cloud Provider

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider integrated with the Melodic platform, where the user has provided his/her own credentials for this provider (included in CAMEL model cloud credentials).
3. Meta solver configured to use CP solver for that case.
4. Cloudiator properly connected to given Cloud Providers.
5. Complete CAMEL model of FCR application (which includes the definition of just one component along with its installation/management scripts).
6. FCR CAMEL model can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/FCR?at=refs%2Fheads%2FRC2.0

Steps To Complete:

1. Login to the machine with installed Melodic
2. Upload models into "models" directory
3. Upload Camel model FCR.xmi into "models" directory
4. Using SoapUI tool execute following steps:
5. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
6. Using POST method start process
7. Start deploying of application

For each step, the status of the executed action should be positive.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. Virtual machine on the selected Cloud Provider should be created.
2. FCR application should be installed on that machine.
3. The application should be run properly (Website should be displayed properly).

T7.2[F] ESB Authentication

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. ESB configuration to require authentication (e.g. oAuth2.0)
3. ESB connected to a properly configured directory service (e.g. OpenLDAP)
4. All Melodic components (connecting to ESB) should be configured to provide the appropriate credentials to ESB (the first time they connect)
5. Provisioned credentials for Adapter should be invalid
6. At least one cloud provider integrated with the Melodic platform; the user credentials for this provider should have also been supplied (included in the CAMEL model cloud credentials).
7. Meta solver configured to use CP solver for that case.
8. Cloudiator properly connected to the given Cloud Provider.
9. Complete CAMEL model of two component application (which includes the definition of the application components and their installation/maintenance scripts). CAMEL model of two component application can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/TwoComponentApp?at=refs%2Fheads%2FRC2.0
7. CAMEL model of given Cloud Provider prepared and registered in the Melodic platform with at least one virtual machine offer provided.
10. There should be a proper configuration of the virtual machine both in the CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the machine with installed Melodic
2. Download and run cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
3. Upload Camel model Two_ComponentApp.xmi_ into the "/home/models" directory
4. Using SoapUI tool execute following steps:
5. Create new REST project with URL: http://5.249.145.169:8088/api/frontend/deploymentProcess
6. Using POST method start process
7. Receive authentication failure when adapter will try to connect to ESB

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

For each step, the corresponding components should be successfully authenticated (up to Adapter) and the status of the executed action should be positive. At Adapter, the authentication to ESB fails and the workflow terminates after having reported the failure.

Expected results:

1. Authentication success per component (until Adapter) is logged
2. Authentication failure for Adapter is logged
3. No virtual machine should be created

T7.1[T] ESB Authentication

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. ESB configuration to require authentication (e.g. oAuth2.0)
3. ESB connected to a properly configured directory service (e.g. OpenLDAP)
4. All Melodic components (connecting to ESB) should be configured to provide the appropriate credentials to ESB (the first time they connect)
5. Provisioned credentials for Adapter should be valid
6. At least one cloud provider integrated with the Melodic platform; the user credentials for this provider should have also been supplied (included in the CAMEL model cloud credentials).
7. Meta solver configured to use CP solver for that case.
8. Cloudiator properly connected to the given Cloud Provider.
9. Complete CAMEL model of two component application (which includes the definition of the application components and their installation/maintenance scripts). CAMEL model of two component application can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/TwoComponentApp?at=refs%2Fheads%2FRC2.0
5. CAMEL model of given Cloud Provider prepared and registered in the Melodic platform with at least one virtual machine offer provided.
10. There should be a proper configuration of the virtual machine both in the CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the machine with installed Melodic
2. Download and run cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
3. Upload Camel model TwoComponentAppnew.xmi_ into "/home/models" directory
4. Using SoapUI tool execute following steps:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5. Create new REST project with URL: http://<VM's IP>:8088/api/frontend/deploymentProcess
6. Using POST method start process
7. Start deploying of application

Sample body
"applicationId": "TwoComponentAppnew", "username": "user1", "password":"melodic", "cloudDefinitions":

For each step, the corresponding component should be successfully authenticated and the status of the executed action should be positive.

Expected results:

1. Virtual machine on the selected Cloud Provider should be created.
2. Authentication success per component is logged.
3. The component of the application should be installed on the machine.
4. The application should run properly (Apache web page is properly displayed).

T7.4[F] New plan deployment authorisation

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider has been integrated with the Melodic platform; the user credentials for this provider should also have been supplied.
3. Cloudiator properly connected to the relevant Cloud Providers
4. Complete CAMEL model of FCR application. The CAMEL model can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases
5. Pre-authorization Policy that permits models where **total cores sums <= 2**
   This can be changed in: ~/conf/policies$ vi sample-PREAUTHORIZATION-policy.xml

```
<!-- ... ... total-number-of-cores <= 2-->
 <xacml3:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
 <xacml3:Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than"/>
 <xacml3:AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#integer">2</xacml3:AttributeValue>
 <xacml3:AttributeDesignator AttributeId="total-number-of-cores"
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
              DataType="http://www.w3.org/2001/XMLSchema#integer" MustBePresent="true"/>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed Melodic
2. Proceed to the ~/conf/policies and edit *sample-PREAUTHORIZATION-policy.xml_OFF* change name of *sample-PREAUTHORIZATION-policy.xml_OFF to sample-PREAUTHORIZATION-policy.xml*
3. Upload model of FCRnew.xmi app to CDO
4. Start deployment process
5. Check adapter.log

For each step, the status of the executed action should be positive except from the authorization step.

Expected results:

1. DENY decision is logged.
2. An authorization deny event is sent to ESB.
3. No virtual machine should be created.

T7.3[T] New plan deployment authorisation

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider has been integrated with the Melodic platform; the user credentials for this provider should have also been supplied.
3. Cloudiator properly connected to the relevant Cloud Providers.
4. Complete CAMEL model of FCR application.
5. CAMEL model can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/TwoComponentApp?at=refs%2Fheads%2FRC2.0
6. Pre-authorization Policy that permits models where total cores <= 3
   This can be changed in: ~/conf/policies$ vi sample-PREAUTHORIZATION-policy.xml

```
<!-- ... ... total-number-of-cores <= 3-->
 <xacml3:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
   <xacml3:Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than"/>
   <xacml3:AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#integer">3</xacml3:AttributeValue>
   <xacml3:AttributeDesignator AttributeId="total-number-of-cores"
        Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
        DataType="http://www.w3.org/2001/XMLSchema#integer" MustBePresent="true"/>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed Melodic
2. Proceed to the ~/conf/policies and edit *sample-PREAUTHORIZATION-policy.xml_OFF* change name of *sample-PREAUTHORIZATION-policy.xml_OFF to sample-PREAUTHORIZATION-policy.xml*
3. Upload model of FCRnew.xmi app to CDO
4. Start deployment process
5. Check adapter.log

For each step, the status of the executed action should be positive.

Expected results:

1. PERMIT authorization decision is logged.
2. An authorization permit event is sent to ESB.
3. A certain virtual machine on the selected Cloud Provider should be created.
4. Two component application should be installed on that virtual machine.
5. The application should run properly (web page is properly displayed).

T6.6[T] Dynamic scalability (using a single public Cloud location)

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. Each of the following components should be installed with single instances:
   CP Generator, CDO Server, Meta solver, CP Solver, Solver to deployment and Adapter.
3. At least one Cloud provider integrated with the Melodic platform; the user credentials for this provider should have also been supplied (included in the CAMEL model Cloud credentials).
4. Meta solver configured to use CP solver for that case.
5. Cloudiator properly connected to a given Cloud Provider.
6. Complete CAMEL model of an application which increases or decreases it's VM requirements (randomly). Such an application does not only change the VM number but also the types.
7. CAMEL model of given Cloud provider prepared with the number of VM offerings included.
8. CAMEL model of given Cloud Provider prepared and registered in the Melodic platform with at least one virtual machine offer provided. There should be a proper configuration of the virtual machine both in the CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. The goal of the Test Case is to verify the execution times of each component while scaling the application within one Cloud Provider.
2. Login to the machine with installed Melodic by using: ssh melodic@<VM's IP>
3. Download and run cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
4. Upload the models: cpGenerator-functionTypes.xmi, cpGenerator-locations.xmi, cpGenerator-providerTypes.xmi, cpGenerator-operatingSystems.xmi into "/home/user/models" directory
5. Upload Provider AmazonEC2.xmi into the "/home/models/upperware-models/fms" directory
6. Upload Camel model OneComponentApp.xmi into the "/home/models" directory
7. Using SoapUI tool execute following steps:
8. Create new REST project with URL: http://<VM's IP>:8088/api/frontend/deploymentProcess
9. Using POST method start process
10. Start deploying of application
11. Check solver logs

For each step, the status of the executed action should be positive.

Expected results:

1. The virtual machines on the selected Cloud Provider should be created.
2. The sole component of the simple application should be installed on that machine.
3. Correctly installed and working application.
4. The application should run properly.
5. Error messages due to component inaccessibility or any other issues should be logged.
6. Another log file storing the list of scaled up and down VMs (including their IP, trigger time and also total boot time (time taken by Cloud service provider to start a VM)).

T6.3[T] Temporary unavailability of Cloud Provider

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. One cloud provider has been integrated with the Melodic platform, while the user has supplied his/her credentials for this provider.
3. Meta solver configured to use CP solver for that case.
4. Cloudiator properly configured with given Cloud Provider

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5. No network connection to the Cloud Provider - the lack of network connection will be simulated by changes of routing table or firewall rules configuration on the machine when Cloudiator is installed.
6. Complete CAMEL model of a TwoComponentApp.
7. CAMEL model of given Cloud Provider prepared with at least one virtual machine offer included.
8. There should be a proper configuration of the virtual machine both in the CAMEL Provider model (see step 2) and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the machine with installed Melodic by using: ssh melodic@<IP VM>
2. Upload models into the "models" directory
3. Upload Cloud Provider into the "models/upperware-models/fms" directory
4. Upload Camel model "TwoComponentApp.xmi into the "models" directory
5. Using SoapUI tool execute following steps:
6. Create new REST project with URL: http://<IP VM>:8088/api/frontend/deploymentProcess
7. Using POST method start process
8. Check in logs that Adapter starts to invoke Executionware
9. Resume the network connection to the Cloud Provider after first deployment attempt on the cloud of this provider has failed

For each step, the status of the executed action should be positive.

Expected results:

1. Proper error message with information about Cloud Provider inaccessibility should be logged.
2. A VM (instance) on the selected Cloud Provider should be created.
3. The simple application should be installed on that VM (instance).
4. The application should be run properly (Apache web server's web page should be displayed properly).

T4.9[T]Global reconfiguration - FCR, time > 3600s

Input Conditions:

1. Installed and configured Melodic platform without any application related artefacts.
2. At least one cloud provider integrated with the Melodic platform for which the respective user credentials have been supplied.
3. Cloudiator properly connected to given Cloud Providers.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4. There exists at least one cloud provider with at least two offering satisfying the requirements posed by the user.
5. Complete CAMEL model of an FCRnew application. The CAMEL model should include definition of events and metrics needed to execute the particular test case, parameter of time > 3600s. The model *.xmi* can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/

Steps To Complete:

1. Login to the machine with installed Melodic by using: ssh melodic@<VM IP>
2. Upload Camel model *FCRnew.xmi* into "models" directory
3. Using SoapUI tool execute following steps:
4. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
5. Using POST method start process
6. Deploy application

For each step, the status of the executed action should be positive.

Expected results:

1. Application should be reconfigured according to defined SLOs.
   Time > 3600s there should be less workers deployed.

```
slo NotFinished constraint FCRConstraintModel.NotFinishedOnTime
constraint model FCRConstraintModel{
    variable constraint WorkerCoresGreaterThanMinimumCores :
      FCRMetricModel.WorkerCoresGreaterThanMinimumCores > 0.0
    metric constraint NotFinishedOnTime :
      [FCRMetricModel.NotFinishedOnTimeContext] >= 0.0
}
```

2. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

T4.8[T] Global reconfiguration - FCR, time < 3600s

Input Conditions:

1. Installed and configured Melodic platform without any application related artefacts.
2. At least one cloud provider integrated with the Melodic platform for which the respective user credentials have been supplied.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3. Cloudiator properly connected to given Cloud Providers.
4. There exists at least one cloud provider with at least two offering satisfying the requirements posed by the user.
5. Complete CAMEL model of a FCRnew application. The CAMEL model should include definition of events and metrics needed to execute the particular test case, parameter of time < 3600s. Model *.xmi* can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/

Steps To Complete:

1. Login to the machine with installed Melodic by using: ssh melodic@<VM IP>
2. Upload the Camel model FCRnew.xmi into the "models" directory
3. Using SoapUI tool execute following steps:
4. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
5. Using POST method start process
6. Deploy application

For each step, the status of the executed action should be positive.

Expected results:

1. Application should be reconfigured according to the defined SLOs.
   Time < 3600s there should be more workers deployed.

```
slo NotFinished constraint FCRConstraintModel.NotFinishedOnTime
constraint model FCRConstraintModel{
    variable constraint WorkerCoresGreaterThanMinimumCores :
      FCRMetricModel.WorkerCoresGreaterThanMinimumCores > 0.0
    metric constraint NotFinishedOnTime :
      [FCRMetricModel.NotFinishedOnTimeContext] >= 0.0
}
```

2. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

T4.6[T] Global reconfiguration - GenomWithSpark, time < 3600s

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. AWS Cloud Provider integrated with the Melodic platform for which the respective user credentials have been supplied.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3. Cloudiator properly connected to given Cloud Providers.
4. Complete CAMEL model of a GenomWithSpark application. The CAMEL model should include definition of events and metrics needed to execute the particular test case, parameter of time should be < 3600s.
5. Model .xmi can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/

Steps to complete:

1. Login to machine with installed Melodic
2. Upload Camel model GenomWithSpark.xmi into the "models" directory
3. Using the Postman tool execute the following steps:
4. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
5. Using POST method start process

Sample body

```
//GenomWithSpark AWS body deployment
{
 "applicationId": "GenomWithSpark",
 "username": "user1",
 "password":"melodic",
 "cloudDefinitions":
 [ {
     "endpoint": "",
     "cloudType": "PUBLIC",
     "api": { "providerName": "aws-ec2" },
     "credential": {
        "user": "", //add user
        "secret": "" //add password },
     "cloudConfiguration": {
       "nodeGroup": "edyta",
       "properties":
       {  "sword.ec2.ami.cc.query": "image-id=ami-08a0a7bee3f024aeb",
          "sword.ec2.ami.query": "image-id=ami-08a0a7bee3f024aeb"
       }
     }
 } ],
 "watermark": {
    "user": "edyta",
    "system": "UI",
    "date": "2016-02-28T16:41:41+0000",
    "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002AAA"
 }
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6. Deploy application
7. Check metasolver/ems logs
8. Check VM instances (should appear in cloud provider console)

For each step, the status of the executed action should be positive.

Expected results:

1. Application should be reconfigured according to defined SLOs.
   For time < 3600s more workers should be deployed.
2. EMS properly delivers events to Metasolver (produced according to CAMEL model specifications).
3. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

T4.5[T] Global reconfiguration - GenomWithSpark, time > 3600s

Input Conditions:

1. Installed and configured Melodic platform without any application related artefacts.
2. At least one cloud provider integrated with the Melodic platform for which the respective user credentials have been supplied.
3. Cloudiator properly connected to given Cloud Providers.
4. There exists at least one cloud provider with at least two offering satisfying the requirements posed by the user.
5. Complete CAMEL model of a GenomWithSpark application. The CAMEL model should include definition of events and metrics needed to execute the particular test case, parameter of time > 3600s. Model *.xmi* can be downloaded from https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/

Steps to complete:

1. Login to the machine with installed Melodic by using: ssh melodic@<VM IP>
2. Upload the Camel model GenomWithSpark.xmi_ into the "models" directory
3. Using the SoapUI tool execute the following steps:
4. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
5. Using POST method start process
6. Deploy application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

For each step, the status of the executed action should be positive.

Expected results:

1. Application should be reconfigured according to the defined SLOs.
   For time > 3600s there should be less workers deployed.
   ```
   slo NotFinished constraint GenomConstraintModel.NotFinishedOnTime
   constraint model GenomConstraintModel{
       variable constraint WorkerCoresGreaterThanMinimumCores :
         GenomMetricModel.WorkerCoresGreaterThanMinimumCores > 0.0
       metric constraint NotFinishedOnTime :
         [GenomMetricModel.NotFinishedOnTimeContext] >= 0.0
   }
   ```
2. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

T4.4b[T] Global reconfiguration - GenomWithSpark deployed on OpenStack Cloud Provider

Input Conditions:

1. Installed and configured Melodic platform without any application related artefacts.
2. OpenStack cloud provider integrated with the Melodic platform for which the respective user credentials have been supplied.
3. Cloudiator properly connected to given Cloud Providers.
4. There exists at least one cloud provider with at least two offering satisfying the requirements posed by the user.
5. Complete CAMEL model of a GenomWithSpark application. The CAMEL model should include definition of events and metrics needed to execute the particular test case, parameter of time = 3600s. Model *.xmi* can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/

Steps to complete:

1. Login to the machine with installed Melodic by using: ssh melodic@<VM IP>
2. Upload Camel model GenomWithSpark.xmi_ into the "models" directory
3. Using SoapUI tool execute following steps:
4. Create new REST project with URL:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

http://<VM IP>:8088/api/frontend/deploymentProcess

5. Using POST method start process
6. Deploy application

For each step, the status of the executed action should be positive.

Expected results:

1. Application should be reconfigured according to the defined SLOs.
   One worker created at first, then a second one should be added.

```
slo NotFinished constraint GenomConstraintModel.NotFinishedOnTime
constraint model GenomConstraintModel{
    variable constraint WorkerCoresGreaterThanMinimumCores :
      GenomMetricModel.WorkerCoresGreaterThanMinimumCores > 0.0
    metric constraint NotFinishedOnTime :
      [GenomMetricModel.NotFinishedOnTimeContext] >= 0.0
}
```

2. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

T4.4a[T] Global reconfiguration - GenomWithSpark deployed on AWS Cloud Provider

Input Conditions:

1. Installed and configured Melodic platform without any application related artefacts.
2. AWS Cloud Provider integrated with the Melodic platform for which the respective user credentials have been supplied.
3. Cloudiator properly connected to given Cloud Providers.
4. Complete CAMEL model of a GenomWithSpark application. The CAMEL model should include definition of events and metrics needed to execute the particular test case; parameter of time should be = 3600s. Model *.xmi* can be downloaded from: https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/

Steps to complete:

1. Login to the machine with installed Melodic
2. Upload Camel model GenomWithSpark.xmi into "models" directory
3. Using the Postman tool execute the following steps:
4. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5. Using POST method start process

Sample body

```
//GenomWithSpark AWS body deployment
{
 "applicationId": "GenomWithSpark",
"username": "user1",
"password":"melodic",
"cloudDefinitions":
[
    {
    "endpoint": "",
     "cloudType": "PUBLIC",
     "api": {
        "providerName": "aws-ec2"
     },
     "credential": {
        "user": "", //add user
        "secret": "" //add password
    },
    "cloudConfiguration": {
       "nodeGroup": "edyta",
       "properties":
       {
          "sword.ec2.ami.cc.query": "image-id=ami-08a0a7bee3f024aeb",
          "sword.ec2.ami.query": "image-id=ami-08a0a7bee3f024aeb"
       }
    }
  }
],
"watermark": {
   "user": "edyta",
   "system": "UI",
   "date": "2016-02-28T16:41:41+0000",
   "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002AAA"

}
}
```

6. Deploy application
7. Check Metasolver/ems logs
8. Check VM instances (should appear in cloud provider console)

For each step, the status of the executed action should be positive

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. Application should be reconfigured according to defined SLOs;
   Have an additional application VM deployed, i.e. two or more VM active at the same time (Again a record of the new VM must appear in cloud provider console, and an Apache web server will respond at the corresponding IP address).
2. EMS properly delivers events (produced according to CAMEL model specifications) to Metasolver.
3. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

UC-CAS-9 [T] Reconfiguration Correctly Handles Load Balancer Configuration When Removing Instance(s)

Input Conditions:

1. Deployed and functional application 'SmartWe'.

Steps to complete:

1. Login to both components of 'SmartDesign' via ssh
2. Increase RAM load to 80% with 'createRamLoad.sh' on both instances of component 'SmartDesign'

```
total=$(free | awk '{print $2}'| head -2| tail -1); echo "Free:"$total;
used=$(free | awk '{print $3}'| head -2| tail -1); echo "Used:"$used;
avail=$(free | awk '{print $7}'| head -2| tail -1); echo "Avail"$avail;

target=$((total / 100 * $1));
echo "Targeted:"$target;
target_n=$((target - used));
target_m=$((target_n/1000))
echo "Targeted_MB:"$target_m

if [[ $target_m =~ ^[\-0-9]+$ ]] && (( $target_m > 0)); then
  stress --vm-bytes "$target_m"M --vm-keep -m 1
else
  echo "Memory load already higher than "$1"%"
fi
```

3. Wait until three or more instances were added by reconfiguration mechanism of MELODIC
4. Assure newly added instance is available

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5. Direct access to VM IP on SmartDesign port
6. Assure configuration of HA Proxy contains newly added machine
7. Stop script 'createRamLoad.sh'

Expected results:

1. Newly added instance is removed
2. Direct access to VM IP on SmartDesign port is not working any longer
3. Configuration of HA Proxy does not contain IP anymore

UC-CAS-8 [F] Deployment According To Scurity Rule/s

Input Conditions:

1. Installed and configured Melodic platform without any application related artefacts.

Steps to complete:

1. Have application's CAMEL model file 'smartwe-docker.xmi' ready
2. change model in a way that causes deployment of one or more components outside of Europe
3. Upload CAMEL model file to CDO MELODIC (CDO Server)
4. Initiate deployment by POST to deploymentProcess endpoint with valid credentials for AWS provider

Expected results:

1. Deployment fails due to violation of security rule.

UC-CAS-6 [T] Application Is Deployed On 1&1 IONOS

Input Conditions:

1. Available MELODIC platform and valid credentials for 1&1 IONOS (formerly ProfitBricks)

Steps To Complete:

1. Upload application's CAMEL model 'smartwe-docker.xmi' to MELODIC
2. Issue deployment on endpoint by providing application name and credentials for provider

Expected results:

1. Application and all components are available and functional.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

UC-CAS-5 [T] Application Is Deployed On NordicStack

Input Conditions:

1. Available MELODIC platform and valid credentials for NordicStack

Steps To Complete:

1. Upload application's CAMEL model 'smartwe-docker.xmi' to MELODIC
2. Issue deployment on endpoint by providing application name and credentials for provider

Expected results:

1. Application and all components are available and functional.

UC-CAS-4 [T] Application Is Deployed On OMI

Input Conditions:

1. Available MELODIC platform and valid credentials for OMI

Steps To Complete:

1. Upload application's CAMEL model 'smartwe-docker.xmi' to MELODIC
2. Issue deployment on endpoint by providing application name and credentials for provider

Expected results:

1. Application and all components are available and functional.

UC-CAS-3 [T] Application Is Deployed On AWS

Input Conditions:

1. Available MELODIC platform and valid credentials for Amazon AWS

Steps To Complete:

2. Upload application's CAMEL model 'smartwe-docker.xmi' to MELODIC
3. Issue deployment on endpoint by providing application name and credentials for provider

Expected results:

1. Application and all components are available and functional.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

UC-CAS-2 [T] Reconfiguration Happens Within Bounds Based On SLOs and UF

Input Conditions:

1. Deployed and functional application 'SmartWe'.

Steps To Complete:

1. Login to both components of 'SmartDesign' via ssh
2. Increase RAM load to 80% with 'createRamLoad.sh' on both instances of component 'SmartDesign'

Sample body

```
total=$(free | awk '{print $2}'| head -2| tail -1); echo "Free:"$total;
used=$(free | awk '{print $3}'| head -2| tail -1); echo "Used:"$used;
avail=$(free | awk '{print $7}'| head -2| tail -1); echo "Avail"$avail;

target=$((total / 100 * $1));
echo "Targeted:"$target;

target_n=$((target - used));
target_m=$((target_n/1000))
echo "Targeted_MB:"$target_m

if [[ $target_m =~ ^[\-0-9]+$ ]] && (( $target_m > 0)); then
  stress --vm-bytes "$target_m"M --vm-keep -m 1
else
  echo "Memory load already higher than "$1"%"
fi
```

Expected results:

1. Melodic performed a reconfiguration and added one or more instances of the component 'SmartDesign'
2. The new instance of 'SmartDesign' was made available in the LoadBalancer component and is used for new sessions (configuration file contains IP ad new node)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

UC-CAS-1 [T] Multi-Component App Is Initially Correctly Deployed on OMI Stack

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.

Steps To Complete:

1. Have application's CAMEL model file 'smartwe-docker.xmi' ready
2. Upload CAMEL model file to CDO MELODIC (CDO Server)
3. Initiate deployment by POST to deploymentProcess endpoint with valid credentials for the OMI stack provider

Expected results:

1. All components are deployed and functional and within the range of their initial instance count.

UC-CET-8 [F] Deploy pySpark app on AWS and download packages via pip

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model

Steps To Complete:

1. Upload CAMEL model with pyFiles argument for Livy server
    - pyFiles is missing some packages
    - Missing packages are downloaded via pip
        1. Import pip
        2. pip.main()
2. Trigger deployment

Expected results:

1. Instance is deployed, however Livy server fails to deliver desired output, because Livy can't utilize ex-post downloaded packages.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

UC-CET-7 [F] Deploy pySpark app on AWS with additional compiled libraries for different platform

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model

Steps To Complete:

1. Upload CAMEL model with pyFiles argument for Livy server. pyFiles include compiled packages like numpy/pandas precompiled to different platform as requested for Spark workers
    1. Workers: Ubuntu 16.04
    2. pyFiles package precompiled for Windows
2. Trigger deployment

Expected results:

1. Instance is deployed, however Livy server fails to deliver desired output.

UC-CET-6 [T] Deploy pySpark app on AWS with additional compiled libraries

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model

Steps To Complete:

1. Upload CAMEL model with pyFiles argument for Livy server. pyFiles include compiled packages like numpy/pandas precompiled to same platform as requested for Spark workers (both Ubuntu)
2. Trigger deployment

Expected results:

1. Instance is deployed. Livy server serves stdout with results.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

UC-CET-5 [T] Deploy pySpark app on AWS with additional libraries

Input Conditions:

1. Available and tested Melodic platform
3. Valid credentials for AWS
4. CAMEL model

Steps To Complete:

1. Upload CAMEL model with pyFiles argument for Livy server
2. Trigger deployment

Expected results:

1. Instance is deployed. Livy server serves stdout with results.

UC-CET-4 [T] Deploy Spark app on AWS

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model

Steps To Complete:

1. Upload CAMEL model
2. Trigger deployment

Expected results:

1. Instance is deployed. Livy server serves stdout with results.

UC-CET-3 [T] Deploy instance with GB RAM >= 8

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model where we request 8 or more GB of RAM

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Upload CAMEL model
2. Trigger deployment

Expected results:

1. Instance is deployed with 8+ GB RAM

UC-CET-2 [T] Deploy app on AWS in United Kingdom

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model where we request deployment in UK

Steps To Complete:

1. Upload CAMEL model
2. Trigger deployment

Expected results:

1. Instance is deployed in UK on AWS

UC-CET-1 [F] Request VM with exactly 3 cores at AWS

Input Conditions:

1. Available and tested Melodic platform
2. Valid credentials for AWS
3. CAMEL model with cores = 3

Steps To Complete:

1. Upload CAMEL model where we request 3 cores
2. Trigger deployment

Expected results:

1. Deployment fails as of today (2019-02-13) there are no EC2 VMs with 3 cores.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664