Project Acronym:	**OPTIMIS**
Project Title:	**Optimized Infrastructure Services**
Project Number:	**257115**
Instrument:	**Integrated Project**
Thematic Priority:	**ICT-2009.1.2 – Internet of Services, Software and Virtualisation**

Ecoefficiency Tool User Guide

Activity 4: *Basic Service Operation*

WP 4.2: *Cloud Runtime Optimization*

Due Date:		M36
Submission Date:		30/05/2013
Start Date of Project:		01/06/2010
Duration of Project:		36 months
Organisation Responsible for the Deliverable:		Barcelona Supercomputing Centre (BSC)
Version:		1.0
Status		Final
Author(s):	Josep Subirats	BSC

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
2.0	2013-03-15	Updated document.	Josep Subirats
1.0	2012-04-05	Created document.	Josep Subirats

Table of Contents

1 INTRODUCTION 6

1.1 GLOSSARY OF ACRONYMS 6

2 ECOEFFICIENCY TOOL USER GUIDE 7

2.1 RELEASE INFORMATION 7
2.2 INTRODUCTION 7
2.3 FUNCTIONALITIES 7
2.3.1 Functionalities at IP level 7
2.3.2 Functionalities at SP level 15
2.4 KNOWN LIMITATIONS 16
2.4.1 Unavailability of energy data in some testbeds 16
2.5 GETTING STARTED 16
2.5.1 Using the Software 16
2.5.2 Testing the Software 18
2.5.3 Configuration 19
2.6 FAQ 19
2.7 OTHER INFORMATION 19
2.7.1 Source Code Information 19
2.7.2 Directory Structure 20
2.7.3 Contributors 20

3 REFERENCES 21

Index of Figures

Figure 1 - TREC GUI displaying energy efficiency evolution for node "optimis1" during operation time. 17

Index of Tables

Table 1 Ecoefficiency Tool release information 7

Table 2 - IP Functionalities 7

Table 3 - SP Functionalities 15

1 Introduction

This document includes the user guide for the software component Ecoefficiency Tool.

Section 2 details the component's release information and provides an overview of its functionality and provided methods. It also details its limitations, as well as its code information and directory structure. Its usage and testing procedures are also provided. Finally, Section 3 contains references for further information.

1.1 Glossary of Acronyms

Acronym	Definition
D	Deliverable
IP	Infrastructure Provider
JAR	Java ARchive
JDK	Java Development Kit
SP	Service Provider
WP	Work Package

Table 1- Acronyms table

2 Ecoefficiency Tool User Guide

2.1 Release information

Component Name	Release Number	Release Date
Ecoefficiency Tool	3.0	2013-03-30

Table 2 Ecoefficiency Tool release information

2.2 Introduction

The Ecoefficiency Tool component is in charge of assessing energy-related aspects, such as the energy efficiency (amount of useful work per Watt consumed) and the eco efficiency (amount of useful work per Kg of carbon emitted to the atmosphere) of a given cloud infrastructure. Given that these terms are closely related, we will refer indistinctively to either of them using the term eco-efficiency. To measure the eco-efficiency, the Ecoefficiency Tool component uses the information collected by the respective monitoring tool of the OPTIMIS Basic toolkit.

The Ecoefficiency Tool can both evaluate the current eco-efficiency as well as predict it for a future potential status, even in the presence of events such as a VM cancellation, deployment or migration, as well as service deployment. Note that the forecasts performed by the eco-efficiency component upon service deployment are fully conditioned by ecological-aware (legal) constraints consisting in having different certifications, such as the LEED's certification, specified in service manifest.

Once this forecast has been performed, it is used by self-management resource scheduling policies that operate in accordance to provider's energy and ecological requirements as well as its high-level objectives (i.e. BLOs).

For more information about this component, refer to [1].

2.3 Functionalities

The Ecoefficiency Tool component provides different functionalities at IP and SP level, as detailed in the following subsections.

2.3.1 Functionalities at IP level

In this subsection you can find the available functionalities provided at IP level which can be accessed using the EcoEfficiencyToolRESTClientIP.java class. In order to learn how to use them from your Java code, refer to Section 2.5.1 of this document. Possible tests are provided in Section 2.5.2.

Table 3 - IP Functionalities

Operation	Input	Output	Description
startAssessment(timeout)	Parameter 1: timeout between successive ecoefficiency automatic assessments. A	-	Starts the automatic eco-assessment of an IP. The IP's eco-efficiency will be

	default timeout value will be used if this parameter is not specified.		evaluated periodically using the provided timeout value.
stopAssessment()	-	-	Stops the automatic eco-assessment of an IP.
ecoEfficiencyScore assessIPEcoEfficiency(type)	Parameter 1: type of eco-efficiency assessment: energy (“energy”) or ecological (“ecological”) efficiency.	ecoEfficiencyScore: eco-efficiency evaluation of the IP.	Assesses IP’s overall eco-efficiency.
ecoEfficiencyScore forecastIPEcoEfficiency(timespan, type)	Parameter 1: timespan specifies the amount of time in the future in which the prediction will be made. Parameter 2: type of eco-efficiency forecast: energy (“energy”) or ecological (“ecological”) efficiency.	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP’s overall eco-efficiency.
ecoEfficiencyScore forecastIPEcoEfficiencyServiceDeployment(manifest, timeSpan, type)	Parameter 1: service manifest XML descriptor. Parameter 2: timespan specifies the amount of time in the future in which the prediction will be made. Parameter 3: type of eco-efficiency forecast: energy (“energy”) or ecological (“ecological”) efficiency.	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP’s overall eco-efficiency upon a service deployment.
ecoEfficiencyScore forecastIPEcoEfficiencyVMCancellation(vmId, timeSpan, type)	Parameter 1: VM identifier of the VM to be cancelled Parameter 2: timespan specifies the amount of time in the future in which the prediction will be made. Parameter 3: type of eco-efficiency forecast: energy (“energy”) or	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP’s overall eco-efficiency upon a VM cancellation (VM to be shut down).

	ecological ("ecological") efficiency.		
ecoEfficiencyScore forecastIPEcoEfficiencyVMDeploymentKnownPlacement(ovfDom, destNode, List activeNodes, type, timeSpan)	Parameter 1: OVF descriptor of the VM to be deployed, generated using EMOTIVE's OVFWrapper (see Installation Guide). Parameter 2: physical node where the VM will be deployed. Parameter 3: list of active physical nodes. Parameter 4: timespan specifies the amount of time in the future in which the prediction will be made. Parameter 5: type of eco-efficiency forecast: energy ("energy") or ecological ("ecological") efficiency.	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP's overall eco-efficiency upon a VM deployment in a known physical host.
ecoEfficiencyScore forecastIPEcoEfficiencyVMMigrationKnownPlacement(vmId, destNode, List activeNodes, type, timeSpan)	Parameter 1: VM identifier of the VM to be migrated. Parameter 2: physical node where the VM will be migrated to. Parameter 3: list of active physical nodes. Parameter 4: type of eco-efficiency forecast: energy ("energy") or ecological ("ecological") efficiency. Parameter 5: timespan specifies the amount of time in the future in which the prediction will be made.	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP's overall eco-efficiency upon a VM migration to a known physical host.
ecoEfficiencyScore forecastIPEcoEfficiencyVMDeploymentUnknownPlacement(ovfDom, type, timeSpan)	Parameter 1: OVF descriptor of the VM to be deployed, generated using EMOTIVE's OVFWrapper (see	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP's overall eco-efficiency upon a VM deployment in a yet unknown physical host.

	Installation Guide). Parameter 2: type of eco-efficiency forecast: energy (“energy”) or ecological (“ecological”) efficiency. Parameter 3: timespan specifies the amount of time in the future in which the prediction will be made.		
ecoEfficiencyScore forecastIPEcoEfficiencyVMMigrationUnknownPlacement(vmId, type, timeSpan)	Parameter 1: VM identifier of the VM to be migrated. Parameter 2: type of eco-efficiency forecast: energy (“energy”) or ecological (“ecological”) efficiency. Parameter 3: timespan specifies the amount of time in the future in which the prediction will be made.	ecoEfficiencyScore: eco-efficiency forecast of the IP.	Predicts IP’s overall eco-efficiency upon a VM migration to a yet unknown physical host.
ecoEfficiencyScore assessNodeEcoEfficiency (nodeID, type)	Parameter 1: Physical node identifier. Parameter 2: type of eco-efficiency assessment: energy (“energy”) or ecological (“ecological”) efficiency.	ecoEfficiencyScore: eco-efficiency evaluation of the node.	Assesses the eco-efficiency of a node. The IP is implicit, since it is the one which owns the node and executes this method.
List<ecoEfficiencyScore> assessMultipleNodesEcoEfficiency (List<nodeIDs>, type)	Parameter 1: list containing the nodes’ IDs whose ecoefficiency wants to be evaluated. Parameter 2: type of eco-efficiency assessment: energy (“energy”) or ecological (“ecological”) efficiency.	List<ecoEfficiency Score>: eco-efficiency evaluation of the set of requested nodes.	Assesses the eco-efficiency of a set of nodes. The IP is implicit, since it is the one which owns the nodes and executes this method.
ecoEfficiencyScore forecastNodeEcoEfficiency(nodeId, List ovfs, type, timeSpan) {	Parameter 1: Physical node identifier. Parameter 2: List of OVF descriptors of the VMs to deploy (if not existing in the	ecoEfficiencyScore: eco-efficiency forecast of the node.	Predicts the eco-efficiency of a node of the IP. The IP is implicit, since it is the one which owns the node and

	node) or undeploy (if existing in the node), generated using EMOTIVE's OVFWrapper (see Installation Guide). Parameter 3: type of eco-efficiency forecast: energy ("energy") or ecological ("ecological") efficiency. Parameter 4: timespan specifies the amount of time in the future in which the prediction will be made.		executes this method.
List<ecoEfficiencyScore> forecastMultipleNodesEcoEfficiency (List<nodeIDs>, type)	Parameter 1: list containing the nodes' IDs whose ecoefficiency wants to be predicted. Parameter 2: type of eco-efficiency forecast: energy ("energy") or ecological ("ecological") efficiency.	List<ecoEfficiency Score>: eco-efficiency prediction of the set of requested nodes.	Predicts the eco-efficiency of a set of nodes. The IP is implicit, since it is the one which owns the nodes and executes this method.
startServiceAssessment(serviceId, timeout)	Parameter 1: Service identifier. Parameter 2: timeout between successive ecoefficiency automatic assessments. A default timeout value will be used if this parameter is not specified.	-	Starts the automatic eco-assessment of a service.
stopServiceAssessment(serviceId)	Parameter 1: Service identifier.	-	Stops the automatic eco-assessment of a service.
ecoEfficiencyScore *assessServiceEcoEfficiency(serviceId, type)*	Parameter 1: Service identifier. Parameter 2: type of eco-efficiency assessment: energy ("energy") or ecological ("ecological") efficiency.	ecoEfficiencyScor e: eco-efficiency evaluation of the service.	Evaluates the current eco-efficiency of a service.

[performance, power] getServicePerformancePowerAndCO2(serviceId)	Parameter 1: Service identifier.	Performance: the current performance being delivered to the service. Power: the current power being consumed by the service.	Returns the current performance being delivered to the service and its associated power consumption. This method is only used by the EcoEfficiency Tool for SP providers.
ecoEfficiencyScore *forecastServiceEcoEfficiency(manifest, timeSpan, type)*	Parameter 1: service manifest XML descriptor. Parameter 2: timespan specifies the amount of time in the future in which the prediction will be made. Parameter 3: type of eco-efficiency forecast: energy ("energy") or ecological ("ecological") efficiency.	ecoEfficiencyScore: eco-efficiency forecast of the service (to be deployed if it doesn't exist).	Predicts the service's eco-efficiency upon its deployment if it doesn't exist in the system, or it just predicts its future eco-efficiency otherwise.
[energyEfficiency, ecologicalEfficiency] forecastServiceEnEcoEff(manifest, timeSpan)	Parameter 1: service manifest XML descriptor. Parameter 2: timespan specifies the amount of time in the future in which the prediction will be made.	energyEfficiency: energy efficiency forecast of the service (to be deployed if it doesn't exist). ecologicalEfficiency: ecological efficiency forecast of the service (to be deployed if it doesn't exist).	Predicts the service's energy and ecological efficiency upon its deployment if it doesn't exist in the system, or it just predicts its future energy and ecological efficiency otherwise.
ecoEfficiencyScore *assessVMEcoEfficiency(vmId, type)*	Parameter 1: VM identifier of the VM to be evaluated. Parameter 2: type of eco-efficiency assessment: energy ("energy") or ecological ("ecological") efficiency.	ecoEfficiencyScore: eco-efficiency evaluation of the VM.	Evaluates the current eco-efficiency of a VM.
ecoEfficiencyScore *forecastVMEcoEfficiency(vmId, type, timeSpan)*	Parameter 1: VM identifier of the VM to be evaluated. Parameter 2: type of eco-efficiency forecast: energy ("energy") or ecological ("ecological")	ecoEfficiencyScore: eco-efficiency evaluation of the VM.	Predicts the future eco-efficiency of a VM.

	efficiency. Parameter 3: timespan specifies the amount of time in the future in which the prediction will be made.		
setInfrastructureEcoThreshold(energyEfficiencyTH, ecologicalEfficiencyTH)	Parameter 1: Energy efficiency threshold. Parameter 1: Ecological efficiency threshold.	-	Specifies the minimum eco-efficiency thresholds at infrastructure level. If any of them are surpassed, a proactive notification will be generated. Note that setting a value of -1.0 to any of the thresholds will disable them (separately).
setNodeEcoThreshold(nodeId, energyEfficiencyTH, ecologicalEfficiencyTH)	Parameter 1: Physical node identifier. Parameter 2: Energy efficiency threshold. Parameter 3: Ecological efficiency threshold.	-	Specifies the minimum eco-efficiency thresholds at node level. If any of them are surpassed, a proactive notification will be generated. Note that setting a value of -1.0 to any of the thresholds will disable them (separately).
setServiceEcoThreshold(serviceId, energyEfficiencyTH, ecologicalEfficiencyTH)	Parameter 1: Service identifier. Parameter 2: Energy efficiency threshold. Parameter 3: Ecological efficiency threshold.	-	Specifies the minimum eco-efficiency thresholds at service level. If any of them are surpassed, a proactive notification will be generated. Note that setting a value of -1.0 to any of the thresholds will disable them (separately).
setVMEcoThreshold(vmId, energyEfficiencyTH, ecologicalEfficiencyTH)	Parameter 1: VM identifier. Parameter 2: Energy efficiency threshold. Parameter 3: Ecological efficiency threshold.	-	Specifies the minimum eco-efficiency thresholds at VM level. If any of them are surpassed, a proactive notification will be generated. Note that setting a value of -1.0 to any of the

			thresholds will disable them (separately).
getNodeMaxEco(nodeId, type)	Parameter 1: Physical node identifier. Parameter 2: type of eco-efficiency value: energy ("energy") or ecological ("ecological") efficiency.	Returns the maximum eco-efficiency of a given node.	Returns the maximum eco-efficiency of a given node.
getCPUMeanPerformance()	-	Returns the mean performance of a single CPU at 100% of CPU utilisation, considering all the CPUs in the datacenter.	Returns the mean performance of a single CPU at 100% of CPU utilisation, considering all the CPUs in the datacenter.
getCPUNumber(nodeId)	Parameter 1: Physical node identifier.	Returns the number of CPUs of a node.	Returns the number of CPUs of a node.
getCPUPerformance(nodeId)	Parameter 1: Physical node identifier.	Returns the performance of a single CPU at 100% of CPU utilisation.	Returns the performance of a single CPU at 100% of CPU utilisation.
getMaxPerformance(nodeId)	Parameter 1: Physical node identifier.	Returns the performance of the node at 100% of CPU utilisation.	Returns the performance of the node at 100% of CPU utilisation.
getNodeUsedCPUs(nodeId)	Parameter 1: Physical node identifier.	Returns the number of used CPUs in the node.	Returns the number of used CPUs in the node.
getPidle(nodeId)	Parameter 1: Physical node identifier.	Returns the node minimum power	Returns the node minimum power
getPMax(nodeId)	Parameter 1: Physical node identifier.	Returns the node maximum power	Returns the node maximum power
getPIncr(nodeId)	Parameter 1: Physical node identifier.	Returns the incremental power per cpu	Returns the incremental power per cpu
getAllDeploymentMessages()	-	Returns the last service deployment messages.	Returns the last service deployment messages.

2.3.2 Functionalities at SP level

In this subsection you can find the available functionalities provided at SP level which can be accessed using the EcoEfficiencyToolRESTClientSP.java class. In order to learn how to use them from your Java code, refer to Section 2.5.1 of this document. Possible tests are provided in Section 2.5.2.

Table 4 - SP Functionalities

Operation	Input	Output	Description
startAssessment(serviceId, timeout)	Parameter 1: Service identifier. Parameter 2: timeout between successive ecoefficiency automatic assessments. A default timeout value will be used if this parameter is not specified.	-	Starts the automatic eco-assessment of a service.
stopAssessment(serviceId)	Parameter 1: Service identifier.	-	Stops the automatic eco-assessment of a service.
ecoEfficiencyScore *assessServiceEcoEfficiency(serviceId, type)*	Parameter 1: Service identifier. Parameter 2: type of eco-efficiency assessment: energy (“energy”) or ecological (“ecological”) efficiency.	ecoEfficiencyScore: eco-efficiency evaluation of the service.	Evaluates the current eco-efficiency of a service.
ecoEfficiencyScore *forecastServiceEcoEfficiency(providerId, manifest, timeSpan, type)*	Parameter 1: IP provider identificator, where the service is (to be) deployed. Parameter 2: service manifest XML descriptor. Parameter 3: timespan specifies the amount of time in the future in which the prediction will be made. Parameter 4: type of eco-efficiency forecast: energy (“energy”) or ecological	ecoEfficiencyScore: eco-efficiency forecast of the service (to be deployed if it doesn’t exist).	Predicts the service’s eco-efficiency upon its deployment if it doesn’t exist in the system, or it just predicts its future eco-efficiency otherwise.

	("ecological") efficiency.		
[energyEfficiency, ecologicalEfficiency] forecastServiceEnEcoEff(providerId, manifest, timeSpan)	Parameter 1: IP provider identificator, where the service is (to be) deployed. Parameter 2: service manifest XML descriptor. Parameter 3: timespan specifies the amount of time in the future in which the prediction will be made.	energyEfficiency: energy efficiency forecast of the service (to be deployed if it doesn't exist). ecologicalEfficiency: ecological efficiency forecast of the service (to be deployed if it doesn't exist).	Predicts the service's energy and ecological efficiency upon its deployment if it doesn't exist in the system, or it just predicts its future energy and ecological efficiency otherwise.
getAllDeploymentMessages()	-	Returns the last service deployment messages.	Returns the last service deployment messages.

2.4 Known limitations

The following limitation is known to exist in this release of the software:

2.4.1 Unavailability of energy data in some testbeds

In some testbeds, energy data might not be available. In this case, the adopted workaround is to use an offline-generated power model which will estimate the real power consumption based on the CPU utilization of a given node.

2.5 Getting Started

2.5.1 Using the Software

The client part of the component is packaged as two different client classes included in a JAR file, which can be used by any user (one to interact with the SP-side of the Ecoefficiency Tool and the other to interact with the IP-side one). The only thing that one needs to do is importing such JAR file. You can alternatively add the following dependency in your pom.xml (if using Maven) file and import the appropriate class (IP or SP client):

```
<dependency>
      <groupId>eu.optimis</groupId>
      <artifactId>EcoEfficiencyToolRESTClient</artifactId>
```

```
        <version>1.0-SNAPSHOT</version>
</dependency>
```

The Ecoefficiency Tool's generated information can also be visualized by means of the TREC GUI, which is available both for SP and IP providers through its respective dashboards.

The Ecoefficiency Tool displays the following information for IP providers in the TREC GUI:

- Service deployment eco-efficiency forecast evaluations (for the latest deployment evaluations).
- Service-level eco-efficiency information at operation time.
- Infrastructure-level eco-efficiency information at operation time.
- Physical-node-level eco-efficiency information at operation time.
- VM-level eco-efficiency information at operation time.
- Eco-efficiency information at operation time for all of the nodes of the infrastructure simultaneously.

Whereas the following information can be visualized by SP providers in their respective TREC GUI:

- Service deployment eco-efficiency forecast evaluations (for the latest deployment evaluations).
- Service-level eco-efficiency information at operation time.

For each of the operation time visualizations, the initial and final dates and times to visualize must be introduced by the user, as well as the metric type (energy efficiency, ecological efficiency, performance, power and CO2 emission rate) and, given the case, an identifier (service identifier, node identifier or VM identifier, when operating at these levels).

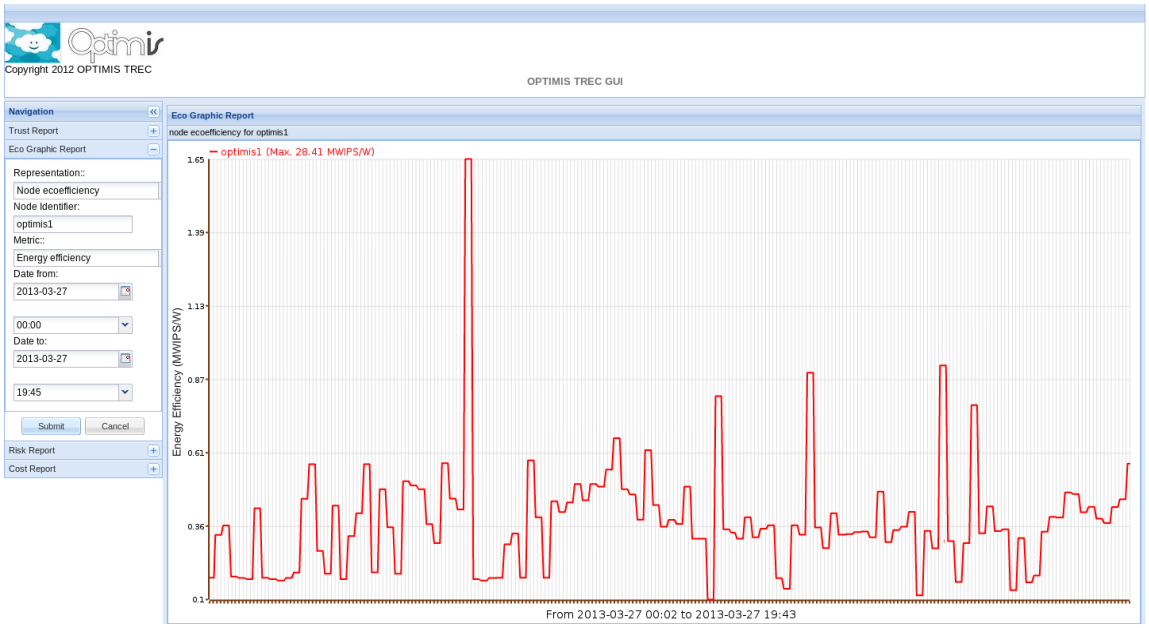

Figure 1 - TREC GUI displaying energy efficiency evolution for node "optimis1" during operation time.

2.5.2 Testing the Software

This component provides similar functionalities both at IP and SP level. Its SP side functionality needs a service to be up and running in order to be tested. Most IP side functionalities can be tested standalone in conjunction with the Cloud Optimizer. Nevertheless, a service needs to be up and running too in order to test eco-efficiency service assessments and forecasts.

All the provided tests can be easily executed by using the "mvn test" command on the root folder of the EcoEfficiencyToolRESTClient source code directory (see Section 2.7.1) with the appropriate arguments, as it is detailed below.

Download the code of the Ecoefficiency Tool from the subversion repository using the following command:

svn co http://pandora.atosorigin.es/svn/optimis/branches/OptimisY3/EcoEfficiencyTool EcoEfficiencyTool/

At this step, you can perform the following tests by executing the following commands:

Tests for Service Providers

- Eco-efficiency assessments and forecasts at service level

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolSP_Service_Test test

- Insertion of eco-efficiency assessments in the Service Provider common TREC database.

  cd EcoEfficiencyToolDBSP

  mvn -Dtest=TREC_DDBB_SP_Test test

Tests for Infrastructure Providers

- Eco-efficiency assessments and forecasts at infrastructure level

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolIP_Infrastructure_Test test

- Energy efficiency assessments and forecasts at node level

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolIP_Node_Test test

- Energy efficiency assessments and forecasts at service level

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolIP_Service_Test test

- Energy efficiency assessments and forecasts at VM level

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolIP_VM_Test test

- Test of the different certificate fields included in the Service Manifest

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolIP_ServiceManifest_Test test

- Conditional forecast functions used by the Holistic Management component.

  cd EcoEfficiencyToolRESTClient

  mvn -Dtest=EcoEfficiencyToolIP_HMNewFunctions_Test test

- Insertion of energy efficiency assessments in the Infrastructure Provider common TREC database.

  cd EcoEfficiencyToolDBIP

  mvn -Dtest=TREC_DDBB_IP_Test test

2.5.3 Configuration

See the Installation Guide.

2.6 FAQ

N/A

2.7 Other information

2.7.1 Source Code Information

The Ecoefficiency Tool source code is divided in seven directories:

- *EcoEfficiencyToolCoreIP*: this core part of the component comprises the main functionality of this tool for IP providers.
- *EcoEfficiencyToolCoreSP*: this core part of the component comprises the main functionality of this tool for SP providers.
- *EcoEfficiencyToolRESTIP*: it is composed by the server side RESTful class interfaces for IPs, and provides access to the core functionalities.
- *EcoEfficiencyToolRESTSP*: it is composed by the server side RESTful class interfaces for SPs, and provides access to the core functionalities.
- *EcoEfficiencyToolRESTClient*: this software part should be used by other components using the Ecoefficiency Tool. It is composed by two Java classes that act as clients of the RESTful methods offered by the SP-side and the IP-side Ecoefficiency tool.
- *EcoEfficiencyToolDBIP*: includes the classes that allow the Eco-Efficiency Tool to store and retrieve records to/from the TREC Common DDBB IP.

- *EcoEfficiencyToolDBSP*: includes the classes that allow the Eco-Efficiency Tool to store and retrieve records to/from the TREC Common DDBB SP.

2.7.2 Directory Structure

The Ecoefficiency Tool uses two directories:

- *Configuration directory*: it contains the configuration files of the Ecoefficiency Tool. It is located at $OPTIMIS_HOME/etc/EcoEfficiencyToolIP for the IP providers version of the tool, whereas it can be found at $OPTIMIS_HOME/etc/EcoEfficiencyToolSP for the SP providers version. An overview of the configuration files available is provided at the Installation Guide of the tool.
- *Output logs directory*: it contains the log files generated by the Ecoefficiency tool. It is placed either at $OPTIMIS_HOME/var/log/EcoEfficiencyToolIP for IP providers or at $OPTIMIS_HOME/var/log/EcoEfficiencyToolSP for SP providers. There are three log files available (note: * can be either “I” or “S” depending on the provider type):
    - EcoEfficiencyTool*P.log: contains all the messages emitted by the Ecoefficiency Tool, being “DEBUG” the minor level of all of them, including all the necessary details, such as time, message type, emitting class, thread, line, etc. To be used for low-level debugging purposes.
    - EcoEfficiencyTool*P_Simplified.log: it is a simplified version of the previous file, also containing from “DEBUG” messages upwards, but only displaying the time and the message type.
    - EcoEfficiencyTool*P_SimplifiedINFO.log: it only contains from “INFO” messages upwards, and only displays the time and message type of each of them.

2.7.3 Contributors

Josep Subirats (BSC)

J. Oriol Fitó (BSC)

Jordi Guitart (BSC)

3 References

[1] Self-managed Cloud Runtime Detailed Design, Deliverable ID4.2.1 of OPTIMIS project.