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**

Admission Control Installation Guide

Activity 3: *Service Deployment*

WP 3.5: *Admission Control*

Due Date:		M36
Submission Date:		07/06/2013
Start Date of Project:		01/06/2010
Duration of Project:		36 months
Organisation Responsible for the Deliverable:		ICCS/NTUA
Version:		1.0
Status		Final
Author(s):	Sotiris Stamokostas Kleopatra Konstanteli	ICCS/NTUA ICCS/NTUA
Reviewer(s)	George Kousiouris	ICCS/NTUA

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
0.1	2/4/2013	Initial input compiled into the document	Sotiris Stamokostas (ICCS/NTUA)
0.2	20/4/2013	Updated all existing sections with new information	Sotiris Stamokostas (ICCS/NTUA) Kleopatra Konstanteli (ICCS/NTUA)
0.3	30/04/2013	Completed version without ONE extensions	Sotiris Stamokostas (ICCS/NTUA)
0.4	30/05/2013	Included new sections about AC ONE module (interoperability component)	Sotiris Stamokostas (ICCS/NTUA)
0.5	31/05/2013	Ready for submission	Sotiris Stamokostas (ICCS/NTUA)
0.6	07/06/2013	Final version to release	Malena Donato (ATOS)

Table of Contents

Index of Tables

1 Introduction

This document serves as the installation guide for the software component Admission Control Y3 prototype of the OPTIMIS project. In more detail the document at hand focuses on:

- ***Installation guide:*** A description of the functionality and the interfaces of each Admission Control component, including release information, the supported platforms, the software pre-requisites and dependencies as well as detailed, step by step installation instructions, including the interoperability extensions for the OpenNebula environment.
- ***Testing guide:*** Detailed instructions on how to create clients for each AC method are also provided, including detailed instructions for proper configuration and for executing the already existing tests.

It should be noted that detailed instructions on how to use each AC component can be found in the Admission Control User Guide [2] and are not replicated here for the sake of brevity. Furthermore, in the Admission Control Scientific Report [1] one can find a detailed description of the optimization algorithm used behind the Admission Control Component.

1.1 Glossary of Acronyms

Acronym	Definition
AC	Admission Control
Apache_DIR	The path to the Apache Tomcat installation
D	Deliverable
GAMS	General Algebraic Modeling System
GAMS_DIR	The path to the GAMS installation
OS	Operating System
TREC	Trust, Risk, Eco and Cost
WP	Work Package
WS	Web service

2 Admission Control Installation Guide

The Admission Control (AC) component is a set of five service-oriented components (RESTful services) with distinct tasks, that work together to provide the overall admission control functionality in OPTIMIS. Their tasks cover physical host information analysis, Trust-Risk-Eco-Cost (TREC) information retrieval, optimization modelling and admission control, and web service (WS) access to the aforementioned functionality. The services that encapsulate the capabilities mentioned above are the *AC_PhHostsInfo_aaS, the AC_TRECcommon_aaS*, the *TREC Analyzer*, the *Admission Controller* and the *AC Gateway* respectively.

During admission control, the Admission Controller formulates and solves a mathematical optimization problem, as described in detail in the Admission Control Scientific Report [1], for finding the optimum allocation of the services to be deployed inside the IP's resources by communicating with the Model which can be either a Heuristic Solver developed in Python or either another Model which uses General Algebraic Modeling System (GAMS) [5]. The latter is a high-level modelling system for mathematical programming and optimization. GAMS it consists of a language compiler and a stable of integrated high-performance solvers. It is tailored for complex, large scale modeling applications, and allows you to build large maintainable models that can be adapted quickly to new situations.

Therefore, in order to have a fully functional AC, one needs to install all five AC components, as well as Python or GAMS.

In order to use the Admission Control in a standalone mode, i.e. without the CO component, with an OpenNebula infrastructure, the component AC_PhHostsInfo_aaS should be replaced by the AC_ONE component.

2.1 Release information

Component Name	Release Number	Release Date
AC Gateway	1.0-SNAPHOT	31/2013
Admission Controller	1.0-SNAPHOT	31/2013
TREC Analyzer	1.0-SNAPHOT	31/2013
AC_PhHostsInfo_aaS	1.0-SNAPHOT	31/2013
AC_TRECcommon_aaS	1.0-SNAPHOT	31/2013
ACRestClients	1.8-SNAPHOT	31/2013
SMAnalyzer	1.7-SNAPHOT	31/2013
AC_ONE	1.8-SNAPHOT	31/2013

Table 1: AC components release information.

2.2 Minimal System Requirements

No special requirements are needed in order to run Admission Control.

2.3 Platforms Supported

The AC, due to its WS nature, is OS independent; therefore it will run in any OS that is complaint with JDK 1.5 or above and is able to run an application server where a WAR can be deployed.

The AC has been thoroughly tested in Ubuntu 12.04 LTS and Windows XP.

2.4 Software Pre-requisites and Dependencies

The software prerequisites and dependencies of each AC component are listed below:

AC Gateway:

Product	Version	Licence
Java	1.6	GPL V2.0
Apache Tomcat	6.X	Apache License v2.0
Jersey Server	1.6	CDDL 1.1 and GPL 2
Jersey Client	1.6	CDDL 1.1 and GPL 2
Log4j	1.2.14	Apache License v2.0
Junit	3.8.1	Common Public License
SMAnalyzer	1.7-SNAPSHOT	MIT License
service-manifest-api	1.0.8	GNU Lesser General Public License
ACRestClients	1.8-SNAPSHOT	MIT License
DeploymentServiceClient	1.0-SNAPSHOT	GNU Lesser General Public License

Table 2: Software dependencies for AC Gateway

TREC Analyzer:

Product	Version	Licence
Java	1.6	GPL V2.0
Apache Tomcat	6.X	Apache License v2.0
Jersey Server	1.6	CDDL 1.1 and GPL 2
Jersey Client	1.6	CDDL 1.1 and GPL 2
Log4j	1.2.14	Apache License v2.0
Junit	3.8.1	Common Public License
SMAnalyzer	1.7-SNAPSHOT	MIT License

Table 3: Software dependencies for TREC Analyzer

Admission Controller:

Product	Version	Licence
Java	1.6	GPL V2.0
Apache Tomcat	6.X	Apache License v2.0
Jersey Server	1.6	CDDL 1.1 and GPL 2
Jersey Client	1.6	CDDL 1.1 and GPL 2
Log4j	1.2.14	Apache License v2.0
Junit	3.8.1	Common Public License
GAMS (optional)	23.9.5	GAMS Base License
Python	2.7.3	GPL

Table 4: Software dependencies for Admission Controller

AC_TRECcommon_aaS:

Product	Version	Licence
Java	1.6	GPL V2.0
Apache Tomcat	6.X	Apache License v2.0
Jersey Server	1.6	CDDL 1.1 and GPL 2
Jersey Client	1.6	CDDL 1.1 and GPL 2
Log4j	1.2.14	Apache License v2.0
Junit	3.8.1	Common Public License
SMAnalyzer	1.7-SNAPSHOT	MIT License
ACRestClients	1.8-SNAPSHOT	MIT License
CloudOptimizerREST Client	1.0-SNAPSHOT	GNU Lesser General Public License
TrustFrameworkClients	3.0-SNAPSHOT	GNU Lesser General Public License
InfrastructureProviderRisk AssesmentToolRestClient	1.0-SNAPSHOT	GNU Lesser General Public License
EcoEfficiencyToolREST Client	3.0-SNAPSHOT	GNU Lesser General Public License
EconomicFrameworkRest ClientIP	1.0-SNAPSHOT	GNU Lesser General Public License

TRECCommonDBIP	3.0-SNAPSHOT	GNU Lesser General Public License
TRECCommonAPIIP	3.0-SNAPSHOT	GNU Lesser General Public License

Table 5: Software dependencies for AC_TRECCcommon_aaS

AC_PhHostssInfo_aaS:

Product	Version	Licence
Java	1.6	GPL V2.0
Apache Tomcat	6.X	Apache License v2.0
Jersey Server	1.6	CDDL 1.1 and GPL 2
Jersey Client	1.6	CDDL 1.1 and GPL 2
Log4j	1.2.14	Apache License v2.0
Junit	3.8.1	Common Public License
ACRestClients	1.8-SNAPSHOT	MIT License
CloudOptimizerREST Client	1.0-SNAPSHOT	GNU Lesser General Public License
MonitoringResource	0.0.1.-SNAPSHOT	GNU Lesser General Public License
MonitoringRESTClient	0.0.2.-SNAPSHOT	GNU Lesser General Public License

Table 6: Software dependencies for AC_PhHostsInfo_aaS

AC_ONE:

Product	Version	Licence
Java	1.6	GPL V2.0
Apache Tomcat	6.X	Apache License v2.0
Jersey Server	1.6	CDDL 1.1 and GPL 2
Jersey Client	1.6	CDDL 1.1 and GPL 2
Log4j	1.2.14	Apache License v2.0
Junit	3.8.1	Common Public License
ACRestClients	1.8-SNAPSHOT	MIT License
org.opennebula.client	1.0	Apache License v2.0

org.apache.ws.commons.util	1.0	Apache License v2.0
org.apache.xmlrpc	3.1.2	Apache License v2.0

Table 7: Software dependencies for AC_ONE

2.5 Installation Instructions

In order to install the AC, the following steps are needed:

1. Install Apache Tomcat:

   Apache Tomcat version 6 or higher is suggested to be used as a hosting environment for the four AC components. Information about downloading and installing it can be found here [4].

2. Install Python:

   Python version 2.7.3 is required for solving the optimization problem that is formulated by the AC.

   Note that Python must be installed in the same physical host with Admission Controller.

3. Install GAMS(optional):

   GAMS version 23.9.5 is required for solving the optimization problem that is formulated by the AC. Information about downloading and installing it can be found here [5].

   In what follows, it is assumed that GAMS is installed in the path named GAMS_DIR. After GAMS has been installed, a copy of a GAMS Base License must be placed inside the GAMS_DIR.

   Note that GAMS must be installed in the same physical host with Admission Controller.

4. Install Admission Controller:

   The Admission Controller is packaged in WAR format. The installation can be done either by deploying the WAR using the Tomcat application manager GUI or manually by copying it in the desired directory.

5. Install TREC Analyzer:

   The TREC Analyzer is packaged in WAR format. The installation can be done either by deploying the WAR using the Tomcat application manager GUI or manually by copying it in the desired directory.

   Please note that TREC Analyzer must be also deployed in the same physical host as the Admission Controller.

6. Install AC_PhHostssInfo_aaS:

   The AC_PhHostssInfo_aaS is packaged in WAR format. The installation can be done either by deploying the WAR using the Tomcat application manager GUI or manually by copying it in the desired directory.

7. Install AC_ONE:

   The AC_ONE is packaged in WAR format. The installation can be done either by deploying the WAR using the Tomcat application manager GUI or manually by copying it in the desired directory.

8. Install AC_TRECcommon_aaS:

   The AC_TRECcommon_aaS is packaged in WAR format. The installation can be done either by deploying the WAR using the Tomcat application manager GUI or manually by copying it in the desired directory.

9. Install AC Gateway :

   The AC Gateway is packaged in WAR format. The installation can be done either by deploying the WAR using the Tomcat application manager GUI or manually by copying it in the desired directory.

2.6 Getting started

2.6.1 Using the Software

AC Gateway:

In order to use the AC Gateway, one must generate WS clients for invoking the operations described below for performing an admission control test on a given service manifest and for defining the policy with regard to the TREC factors.

Method Name	Description	Input	Output	Owner	Actor
performACTest	Perform admission control test	List of Service manifest	List of Service manifest	AC Gateway	CQoS
setPolicy	Set TREC policy	4 float values	4 csv files	AC Gateway	Cloud Optimizer
perServiceConstraints	Set TREC Constraints for each Service	4 float values	4 csv files	AC Gateway	Cloud Optimizer

Table 8: AC Gateway interfaces

The implementation of method "*performACTest*" is located in the Java class called "ACModelApi" under version "1.0-SNAPSHOT".

The unit tests "ACGatewayRemoteTest" and "ACGatewayTest" under the ACGateway Maven module provide example clients for invoking the method "*performACTest"*.

The implementation of methods "*setPolicy*", "perServiceConstraints" can be found in the Java class "ACDataAPi" under version "1.0-SNAPSHOT".

The unit test "SetPolicyTest" under the ACGateway Maven module provides an example client for invoking the method *"setPolicy"*.

The unit test "PerServiceConstraints Test" under the ACGateway Maven module provides an example client for invoking the method *"*perServiceConstraints*"*.

Important notice:

Please note that in order for method "*performACTest"* to properly work, the method "*setPolicy*" must be invoked first.

AC_TRECcommon_aaS:

In order to use the AC_TRECcommon_aaS, one must generate a client that implements the following interface in order to use it in standalone mode.

Method Name	Description	Input	Output	Owner	Actor
AC_TRECcommon	Returns TREC factors	Service Manifest, Physical Hosts Names	TREC values	AC_TRECcommon_aaS	AC Gateway

Table 9: AC_TRECcommon_aaS interfaces

There is a Maven module called "AC_TRECcommon_aaS" with version "1.0-SNAPSHOT" that provides an easy implementation to use the *"AC_TRECcommon"* method.

AC_PhHostssInfo_aaS:

In order to use the AC_PhHostssInfo_aaS, one must generate a client that implements the following interface in order to use it in standalone mode.

Method Name	Description	Input	Output	Owner	Actor
getXML	Generate Physical Hosts Info	IP address	Physical Hosts Info	AC_PhHostssInfo_aaS	AC Gateway

Table 10: AC_PhHostssInfo_aaS interfaces

There is a Maven module called "AC_PhHostssInfo_aaS" with version "1.0-SNAPSHOT" that provides an easy implementation to use the *"getXML"* method.

AC_ONE:

In order to use the AC_ONE, one must generate a client that implements the following interface in order to use it in standalone mode.

Method Name	Description	Input	Output	Owner	Actor
getXML	Generate Physical Hosts Info	IP address	Physical Hosts Info	AC_ONE	AC Gateway
deployVMs	Deploy VMs in OpenNebula	Allocation offer	String	AC_ONE	AC Gateway

Table 11: AC_ONE interfaces

There is a Maven module called "AC_ONE" with version "1.0-SNAPSHOT" that provides an easy implementation to use the *"getXML"* and "*deployVMs*" methods.

TREC Analyzer:

In order to use the TREC Analyzer, one must generate a client that implements the following interface in order to use it in standalone mode.

Method Name	Description	Input	Output	Owner	Actor
createModel	Generate input for optimization model	Physical Hosts Info, TREC factors, Service Manifest	Input to optimization model (CVS files)	TREC Analyzer	AC Gateway

Table 12: TREC Analyzer interfaces

There is a Maven module called “TRECAnalyzer” with version “1.0-SNAPSHOT” that provides an easy implementation to use the *“createModel”* method.

Admission Controller:

In order to use the Admission Controller in standalone mode, one should generate a WS client that implements the following interface:

Method Name	Description	Input	Output	Owner	Actor
getAdmissionControl	Perform admission control test	GamsDirectory, AllocationInfo Path, IP_Id, Solver Selection	Allocation offers, Allocation Details	Admission Controller	AC Gateway

Table 13: Admission Controller interfaces

The implementation of the “*getAdmissionControl*” method can be found in the Java class “AdmissionController” under version “1.0-SNAPSHOT”.

2.6.2 Testing the Software

1. In order to test the method “*performACTest”* of the AC Gateway one needs to run the “ACGatewayRemoteTest”. The latter has client for examining multiple service manifests at the same time and one for passing only one service manifest.

   When running the aforementioned test, the paths to the service manifest(s) need to be given as input. Make sure that the correct paths and IPs are included in the file config.properties (located at /ACGateway/src/test/resources/).

   The output of the “ACGatewayRemoteTest” is a list of updated service manifests the number of which is equal to the number of the service manifest that was given as input to the test. The exact location of each updated service manifest file is printed in the terminal window.

   The command for invoking the “ACGatewayRemoteTest”:

   mvn –Dtest=ACGatewayRemoteTest test

   which is run at the location of the pom.xml.

   The “ ACGatewayTest” can be run in a similar way. The only difference is that has client for examining only one service Manifest at the same time.

   All the test can be executing with the command:

   mvn test

   Important notice:

   Please note that in order for method “*performACTest”* to properly work, the method “*setPolicy*” must be invoked first.

2. In order to test the “setPolicy”, you need to run the “SetPolicyTest”.

   The command for invoking the “SetPolicyTest” (under the pom.xml location) is the following:

   mvn –Dtest=SetPolicyTest test

3. In order to test the PerServiceConstaints, you need to run “PerServiceConstraintsTest”.

   The command for invoking the “PerServiceConstraintsTest” (under the pom.xml location) is the following:

   mvn –Dtest= PerServiceConstraintsTest test

4. In order to test the AC_PhHostssInfo_aaS, you need to run the command :

   mvn test (under the pom.xml location)

   Before running the test, make sure that the correct paths and IPs are included in the file config.properties (located at / AC_PhHostssInfo_aaS /src/test/resources/).

5. In order to test the AC_TRECcommon_aaS, you need to run the command :

   mvn test (under the pom.xml location)

   Before running the test, make sure that the correct paths and IPs are included in the file config.properties (located at / AC_TRECcommon_aaS /src/test/resources/).

2.6.3 Configuration

The Admission Control configuration can be done with two ways:

- Inside war configuration (Default Values)
- Outside war configuration (Optional)

  The Outside war configuration is optional and **overrides** the Inside war configuration. In case of absence of outside war configuration or in case of absence of certain property from outside war configuration file the Admission Control is auto-configured with the default values from inside war configuration.

  Inside war configuration (Default Values):

  – **Admission Controller:**

    1. In case of use GAMS Solver for solving the optimization problem the file AdmissionController/src/main/resources/gams.properties must contain the correct path to GAMS_DIR:

    executable = GAMS_DIR/gams

    The default path is:

    executable = /usr/gams2395/gams

    2. The Admission Controller IP address must be noted in order to be included in ACGatewayConfig.properties.

  – **ACGateway:**

    The file src/main/resources/ACGatewayConfig.properties must be changed to include the following:

    - The paths to input files for the Admission Controller:

      gams.path, path.gams, path.AllocationInfo

      Three relative paths for getting

      GamsDirectory and AllocationInfoDirectory depending on where Apache Tomcat is installed (Apache_DIR).

GamsDirectory = Apache_DIR/gams.path/path.gams

AllocationInfoDirectory = Apache_DIR/gams.path/path.AllocationInfo

Default values:

path.gams = gams

path.AllocationInfo = AllocationInfo

gams.path = webapps/AdmissionController/WEB-INF/classes/

- The IP and the port where the Admission Controller is listening. Note that the TREC Analyzer must be deployed in the same physical host as the Admission Controller.

  Default values:

  admission.controller.ip = localhost

  admission.controller.port = 8080

- The IP and the port where the AC_TRECcommon_aaS is listening.

  Default values:

  AC_TREC.ip = localhost

  AC_TREC.port = 8080

- The IP and the port where the Risk is listening.

  Default values:

  risk.ip = localhost

  risk.port = 8080

- The IP and the port where the Trust is listening.

  Default values:

  trust.ip = localhost

  trust.port = 8080

- The IP and the port where the Eco is listening.

  Default values:

  eco.ip = localhost

  eco.port = 8080

- The IP and the port where the Cost is listening.

  Default values:

  cost.ip = localhost

  cost.port = 8080

- The IP and the port where the AC_PhHostssInfo_aaS is listening.

  Default values:

  physicalHostsInfo_aaS.ip = localhost

  physicalHostsInfo_aaS.port = 8080

- The solver which will solve the optimization problem. You can choose between Heuristic Solver, and GAMS Solver

  Default Values:

  admission.controller.solver = use_Heuristic

Outside war configuration (Optional):

- **Admission Controller:**

  In the file:

  /opt/optimis/etc/AdmissionControl/gams.properties

  The property executable must set to contain the correct path to GAMS_DIR:

  i.e

  executable = /usr/gams2395/gams

- **ACGateway:**
  - In the file:

    /opt/optimis/etc/AdmissionControl/trec.properties

    could be set the IP and the port of each one of the TREC components:

    eco.ip, eco.port, trust.ip, trust.port, risk.ip, risk.port, cost.ip, cost.port

    i.e

    eco.ip = localhost

    eco.port = 8080
  - In the file:

    /opt/optimis/etc/AdmissionControl/AdmissionControl.properties

    The properties gams.path and AllocationInfo.path are absolute paths.

    The property admission.controller.solver set the Solver to solve the optimization problem. It can also set the IP and the port where the Admission Controller is listening and the IP and the port where the AC_PhHostssInfo_aaS is listening. It can also set the IP_ID.

    i.e

    gams.path = /opt/optimis/GamsDirectory/

    AllocationInfo.path = /opt/optimis/AllocationInfoDirectory/

    admission.controller.ip = localhost

    admission.controller.port = 8080

    physicalHostsInfo_aaS.ip = localhost

    physicalHostsInfo_aaS.port = 8080

    admission.controller.solver = use_GAMS

    IP_ID = ATOS

3 References

[1] Admission Control Scientific Report, ICCS/NTUA and other partners, May 2013.

[2] Admission Control User Guide, ICCS/NTUA and other partners, May 2013.

[3] AdmissionControl_README.txt, ICCS/NTUA and other partners, May 2013.

[4] Apache Tomcat Wiki: http://wiki.apache.org/tomcat/FrontPage.

[5] General Algebraic Modeling System, GAMS, http://www.gams.com/.