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**

Service Deployment Optimizer (SDO) User Guide

Activity 3: *Service Deployment*

WP 3.1: *Service Deployment*

Due Date:		M34
Submission Date:		31/03/2013
Start Date of Project:		01/06/2010
Duration of Project:		36 months
Organisation Responsible for the Deliverable:		UMU
Version:		1.0
Status		Under review
Author(s):	Wubin Li (Viali)	UMU
Reviewer(s)		

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
1.0	2013-03-09		Wubin Li (Viali)

Table of Contents

Index of Tables

1 Introduction

This document includes the user guide for the software component *Service Deployment Optimizer (SDO).*

1.1 Glossary of Acronyms

Acronym	Definition
D	Deliverable
WP	Work Package
DO	Deployment Optimizer
SD	Service Deployer
DS	Deployment Service
DSC	Deployment Service Client

2 Service Deployment Optimizer (SDO) User Guide

2.1 Release information

Component Name	Release Number	Release Date
Service Deployment Optimizer (SDO)	3.0	2013-03-31

2.2 Introduction

The purpose of the SDO is two-fold; it generates optimal deployment solutions for a service, and coordinates the deployment process in order to provision a service according to the deployment plan. The SDO comprises of four components, the Service Deployer (SD), the Deployment Optimizer (DO), the Deployment Service (DS), and the Deployment Service Client (DSC). The DO receives a placement request from the SD and calculates an optimal placement based on the service manifest and the chosen optimization goal, Risk or Cost. The SD then orchestrates the deployment, by calling other components in the Optimis toolkit, i.e. the DataManager (DM) and the Service Manager (SM), as needed. The DO is a pure decision making component while the SD provides the utility functionality. The DS is a web application based on the SD and the DO. The DSC is a friendly REST-Style client for users/developers to access DS.

2.3 Functionalities

Once the artefact is installed, the Deployment Service is started. Users can use the service using one of the two options:

Option 1: GUI:

1. Open web browser and access SDO service via SDO endpoint url.
    a. If SDO is installed using **Stand-alone Option**, the url is 'http://localhost:8087/'.
    b. If SDO is installed using **Container Option**, the url is 'http://localhost:8080/sdo-gui/'.

Option 2: SDO Client API usage:

1. SDO service available via a REST-style client api, there are two options for the usage:

    Option a. add the following dependency locally:

```
<dependency>
    <groupId>eu.optimis.sdo</groupId>
    <artifactId>DeploymentServiceClient</artifactId>
    <version>3.0</version>
</dependency>
```

    Option b. Download the DSC artifact from the artifact repository:

http://optimis-artifactory.atosorigin.es/artifactory/libs-release-local/eu/optimis/sdo/DeploymentServiceClient/3.0/DeploymentServiceClient-3.0.jar

add this jar file to the local classpath.

2. Then users can use it as follows:

String sdoURL ="http://optimis-spvm.atosorigin.es:8087";//where the SDO is running.

DeploymentServiceClient sdoClient = new DeploymentServiceClient(sdoURL);

2.3.1 Data VM federation

This functionality provides the feasibility of data VM federation. The interface returns an IP name/Id, which is the best choice for federating a data VM.

- suggestFederatedIP
- suggestFederatedIP(String manifestXML), returns an IP name/id.
- Example:

String manifestXML=""; "; //Manifest in XML format

String ip = sdoClient. suggestFederatedIP(manifestXML)

2.3.2 Service Deployment

This is the most common used functionality, which is used for deploying a service.

- deploy
- deploy(String manifestXML, String objective), returns *true* if it's a successful deployment, otherwise *false*;
- Example:

String manifestXML=""; "; //Manifest in XML format

String objective = Objective.COST.toString();

Boolean success = sdoClient. deploy(manifestXML, objective)

2.3.3 Service Undeployment

This functionality is used to undeploy a service.

- undeploy
- undeploy(String serviceId, String agreementEPR, boolean keepData), agreementEPR represents the agreement endpoint info for a deployed service. This API undeploys a service, and removes data in the IP side if the parameter *keepData* is true.
- Example:

String serviceId=" ecd4ce61-b3bc-4a55-847b-0428531a2cd8";

String agreementEPR = "XXXX";//service agreement endpoint info

sdoClient. undeploy(serviceId, agreementEPR,false)

2.3.4 Query deployment status

This API is designed for status query during a service deployment process.

- queryDeploymentStatus
- queryDeploymentStatus(String serviceId), use service Id as an input.

  It will return a json object represents the deployment status of service with id {serviceId}. The json object contains two key-values.

  The 1st key is "TYPE", its value could be "ERROR" or "PROGRESS".

  The 2nd key is "MESSAGE", its value could be percent completion or an error message if the deployment failed. For example:

  {"TYPE":"ERROR", "MESSAGE":"No service deployed with id serviceId1234"}

  {"TYPE":"ERROR", "MESSAGE":"Failed to get an optimal solution due to XXXX!"}

  {"TYPE":"PROGRESS", "MESSAGE":"80%"}

- Example:

  String serviceId=" ecd4ce61-b3bc-4a55-847b-0428531a2cd8";

  String status =sdoClient. queryDeploymentStatus (serviceId);

2.3.5 Query deployment result

This API is designed for result query after a service deployment process.

- *queryDeploymentResult*
- *queryDeploymentResult(String serviceId),* uses service Id as an input.

  it will return a json object represents the deployment result of service with id {serviceId}. The json object contains two key-values.

  The 1st key is "TYPE", its value could be "ERROR" or "RESULT".

  The 2nd key is "MESSAGE" or "IPS",

  its value could be an error message if the deployment result is not available, or a list of IPs where the service is deployed.

  for example:

  {"TYPE":"ERROR", "MESSAGE":"Placement Result is NOT available (YET)."} {"TYPE":"RESULT", "IPS":"atos;umea;"}

- Example:

  String serviceId=" ecd4ce61-b3bc-4a55-847b-0428531a2cd8";

  String status =sdoClient. *queryDeploymentResult* (serviceId)

2.3.6 Deployment time federation

This functionality is used for deployment time federation in IP side.

- outSourceVMs
- outSourceVMs(String manifestXML, String objective) takes manifest and objective as inputs, and will return a XML String object represents an Offer.

- Example:

  String manifestXML = "XXX"; //Manifest in XML format

  String objective = "COST";

  String result = sdoClient.outSourceVMs(manifestXML, objective);

2.4 Getting Started

2.4.1 Using the Software

Please find a user guide at

https://bscw.scai.fraunhofer.de/bscw/bscw.cgi/330000 .

2.4.2 Testing the Software

Various test cases were identified and designed to fulfil the verification and validation of SDO. To test the software, just check out the source code from the svn repository, and run 'mvn test'.

2.4.3 Configuration

A default configuration file should be located at "/opt/optimis/etc/sdo/sdo.properties". Meanwhile, users can specify their own configuration to run SDO. In this configuration file, require endpoints for TREC, Service Manager, IP discovery, etc., should be specified. More detail is listed below:

Field	Default value	Semantics
configuration.refreshperiod	30000	Configuration file monitor interval (in ms)
ip.discovery.host	localhost	IP Discovery endpoint host name
ip.discovery.port	8080	IP Discovery endpoint host port
vmc.config.file.path		Configuration file location for VM Contextualization component
service.manager.host	localhost	Service Manager endpoint host name
service.manager.port	8080	Service Manager endpoint host port
is.service.bursting.case	false	Specify if SDO is running for Cloud bursting case
manifest.repo.url	http://localhost:8080/manifest-	Manifest Registry Endpoint

	registry/RegistryService	url
eu.optimis.localprovider.name	atos	Specify the name of local provider
is.check.legal.ignored	false	Specify if CheckLegal operation would be ignored during the deployment process.
is.trec.call.skip	false	Specify if TREC calls operation would be ignored during the deployment process.
sp.trec.db.url	jdbc:mysql://optimis-database:3306/sptrecdb	Database url for TREC DB SP
sp.trec.db.username	trecdb_usr	Username for TREC DB SP
sp.trec.db.password	L84VA8cStVf5bV7q	Password for TREC DB SP

Table 1: Configuration details.

2.5 FAQ

None.

2.6 Other information

2.6.1 Source Code Information

The SDO comprises of multiple components, their source code can be accessed via

http://pandora.atosorigin.es/svn/optimis/branches/OptimisY3/DeploymentOptimizer

http://pandora.atosorigin.es/svn/optimis/branches/OptimisY3/ServiceDeployer

http://pandora.atosorigin.es/svn/optimis/branches/OptimisY3/DeploymentService

2.6.2 Directory Structure

For each sub-component, the source code directory contains the following subdirectories or files:

- pom.xml: maven description of this project.

- cc-build.xml: cruise control system description file.

- LICENCE.txt: licence statement file.

- src: java source code.

- doc: description of this component.

2.6.3 Contributors

Wubin Li (Viali), Petter Svärd