#########INSTALLATION#################

**1. REQUIREMENTS**

Tools needed to compile and run RL:

*a) JAVA JRE 1.8.x or higher*

*b) maven*

*c) NETBEANS 8.x or higher (optional)*

*d) MySQL 8.0 server*

*e) JDBC SQL connector*

*f) MySQL Workbench (optional)*

**2. COMPILATION PROCEDURE**

**2.1 COMPILE DEPENDENCY LIBRARY**

Each library is a maven project that can be compiled via command line or via a Java IDE

**2.1.1 COMPILATION VIA CLI**

Enter in the "rl/dependency folder" directory

For each library:

1. enter in the lib directory (“pa-client”, “mon-client”, “sbi-client”)

b) execute "*mvn install package*"

**2.1.2 COMPILATION VIA NETBEANS**

Open maven projects in Netbeans (under File->Open Project)

Compile the projects (click on build button)

**2.2 COMPILE RL CORE**

Enter in the "rl/rl" directory

Execute "*mvn install package*"

**2.2.1 COMPILATION VIA CLI**

Compile the rl

a) execute "*mvn install package*"

**2.2.2 COMPILATION VIA NETBEANS**

Open maven “rl” project in Netbeans (under File->Open Project)

Compile the project (click on build button)

#########RUNNING#################

**1. PREPARATION**

RL needs some sql scripts to configure specific aspects. Below is described how to configure properly the MySQL server and how to prepare the sql script.

**1.1 MySQL DB Server**

Install MySQL DB server (refer to MySQL online wiki for installation according the Operative System)

**1.1.1 Configure MySQL DB Server access**

RL is configured to access the MySQL DB with login "mtp" and pwd "mtp").

The server installation provides a "root@localhost" access, with a generated temporary password; so after the installation you need to change the root password to "mtp" (please refer to MySQL wiki for this operation as it depends from the Operative System)

Under directory “rl/rl/dbscripts/utility” you can find a script to change access credential (“create\_mtp\_user.sql”)

**1.1.2 Prepare interdomainlinks sql files**

RL needs a file containing all interdomain links, i.e. the physical links connecting VIM, WIM, Radio, MEC and federated domains. “interdomainlinks.sql file include the connection between VIM/WIM/MEC/Radio, “fed\_interdomainlink.sql” include the connections for federated domains. Each link has the following format:

mtpdomdb.interdomainlink

(srcDomId, /\* Source domain Id of the link \*/

dstDomId, /\* Destination domain Id of the link \*/

srcGwId, /\* Source node Id of the link \*/

dstGwId, /\* Destination node Id of the link \*/

srcGWIp, /\* Source interface IP of the link \*/

dstGwIp, /\* Destination interface IP of the link \*/

delay, /\* Delay associated to the link \*/

availableBandwidth, /\* Available BW of the link \*/

reservedBandwidth, /\* Reserved BW of the link \*/

totalBandwidth, /\* Total BW of the link \*/

allocatedBandwidth) /\* Allocated BW of the link \*/

An example of interdomainlinks.sql and fed\_interdomainlinks.sql script is already present. The file represents the interdomain links of the reference topology that is used for test.

**1.1.3 Prepare computeFlavour.sql file**

RL needs a file containing all compute flavours available for each NfviPop that is used to scale available compute resources in the abstraction. Each flavour has the following format (refer to IFA005 Section 8.4.2 for the description of the fields):

mtpdomdb.computeflavour

(computeFlavourId, /\* Unique key to identify the flavor (used by SO) \*/

flavourId, /\* Datacenter local identifier of the flavor \*/

accelerationCapability, /\* See IFA005 \*/

NfviPopId) /\* Identifier of the NfviPop using the flavor \*/

mtpdomdb.virtualstoragedata

(typeOfStorage, /\* See IFA005 \*/

sizeOfStorage, /\* See IFA005 \*/

computeFlavourId) /\* Key used in mtpdomdb.computeflavour table \*/

mtpdomdb.virtualcpu

(cpuArchitecture, /\* See IFA005 \*/

numVirtualCpu, /\* See IFA005 \*/

cpuClock, /\* See IFA005 \*/

virtualCpuOversubscriptionPolicy, /\* See IFA005 \*/

computeFlavourId) /\* Key used in mtpdomdb.computeflavour table \*/

mtpdomdb.virtualmemorydata

(virtualMemSize, /\* See IFA005 \*/

virtualMemOversubscriptionPolicy, /\* See IFA005 \*/

numaEnabled, /\* See IFA005 \*/

computeFlavourId) /\* Key used in mtpdomdb.computeflavour table \*/

VALUES (100,"policy1",true, 1);

mtpdomdb.virtualnetworkinterfacedata

(networkId, /\* See IFA005 \*/

networkPortId, /\* See IFA005 \*/

typeVirtualNic, /\* See IFA005 \*/

typeConfiguration, /\* See IFA005 \*/

bandwidth, /\* See IFA005 \*/

accelerationCapability, /\* See IFA005 \*/

metadata, /\* See IFA005 \*/

computeFlavourId) /\* Key used in mtpdomdb.computeflavour table \*/

VALUES (0,0,0,"",100,"","",1);

An example of computeFlavour.sql script is already present. The file represents the flavours of the NfviPoPs of the reference topology that is used for test.

**1.2 Domain configuration file**

as input RL needs to know the list of VIM, WIM, MEC, Radio domains that can controls with all the information to how to connect to it. The syntax is:

<?xml version="1.0" encoding="UTF-8"?>

<Domains>

<Domain >

<Type>type</Type>

<Name>name</Name>

<Id>id</Id>

<MecId>mecid </MecId>

<Ip>127.0.0.1</Ip>

<Port>10000</Port>

</Domain>

.....

<Domain >

<Type>type</Type>

<Name>name</Name>

<Id>id</Id>

<MecId>mecid </MecId>

<Ip>127.0.0.1</Ip>

<Port>10000</Port>

</Domain>

</Domains>

XML file is a list of "Domain" entries where each entry represent the information for a specific domain. Specifically:

<Type>: identifies the domain type ("T-WIM" or "VIM")

<Name>: identifies the domain name (same reported in IFA005)

<Id>: identifies the domain id (same reported in IFA005)

<MecId>: identifies the associated MEC id (valid only for VIM domain)

<Ip>: identifies the IP of the server HTTP (use for REST call)

<Port>: identifies the port of the server HTTP (use for REST call)

**1.3 Federated Domain configuration file**

as input RL needs to know information about the federated POP in a XML

<?xml version="1.0" encoding="UTF-8"?>

<MTPConfig>

<NfviPoPs>

<NfviPoP>

<Id>1</Id>

<FederatedVimId>Provider3</FederatedVimId>

<networkConnectivityEndpoints>192.168.10.10;192.168.10.11</networkConnectivityEndpoints>

</NfviPoP>

</NfviPoPs>

</MTPConfig>

The information contains the id of the POP, id of the VIM and the connectivity endpoints. Such info is used to built the graph for abstraction

**2 RUN RL IN STUB MODE**

RL can run in two ways:

- Normal mode: It expects to have for each domain a corresponding RL plugin to contact (detail of contact described in xml file)

- Stub mode: The domains are simulated as Stub threads (useful for test and debug).

The mode is enabled by a System properties ("STUB\_ENABLE") configured at the startup (see below). If not specified the default behavior is to have the “stub mode” disabled.

**3 RUN RL**

Output of the compilation is a self-contained jar file. So to run it, just type "*java -D"STUB\_ENABLE=<yes/no>" -jar <jarfile> <xmlfilename> <ip> <port>*"

where:

*- <jarfile>* is the name of the jar file (with the local path to reach it)

*- <xmlfilename>* is the the file xml describing in Section 1.3

*- <ip>* is the ip address where RL is listening for HTTP REST calls (used for SO communication)

*- <port>* is the port number where RL is listening for HTTP REST calls (used for SO communication)

*- STUB\_ENABLE=<yes/no>* is a flag that enable/disable the stub mode