Report

Complete Mechanism for IBN

Reporter
Mehmood Asif
AM20176804

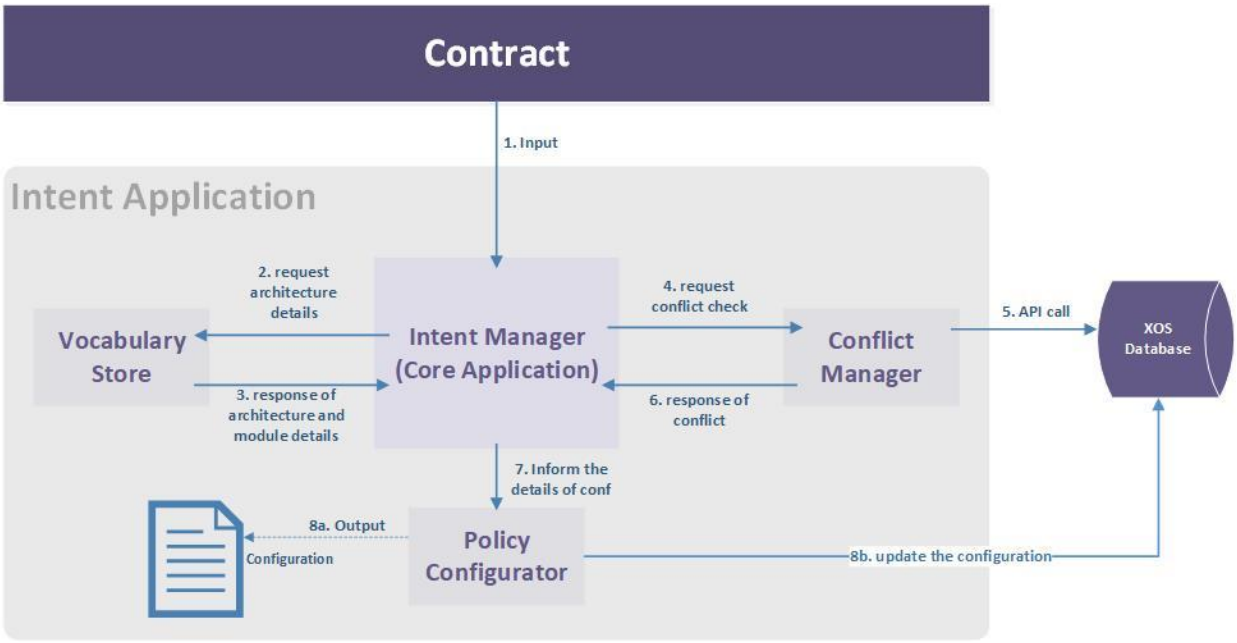
Dated: 15th October, 2018

Introduction

We can show contracts as an example of input to the Intent-Application, we propose. This will contain the modules named as:

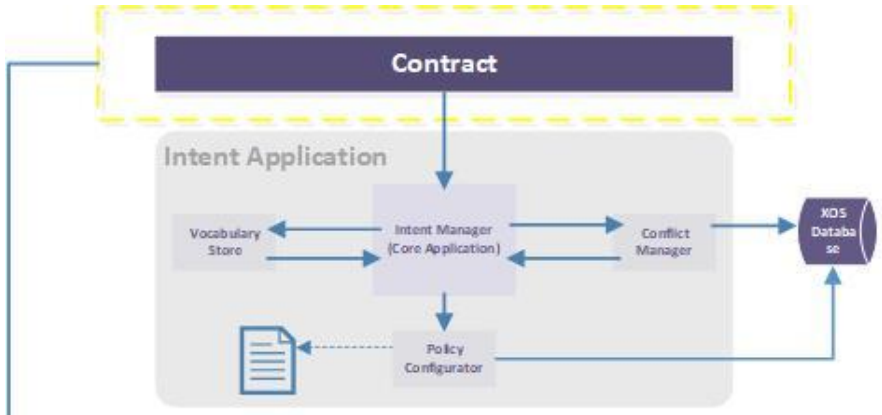
1. Intent Manager (core part)
2. Vocabulary Store (contains the architectures supported)
3. Conflict Manager (responsible to check the conflicts)
4. Policy Configurator (generates configuration for any Orchestrator)

The architectural-view for the following system is:



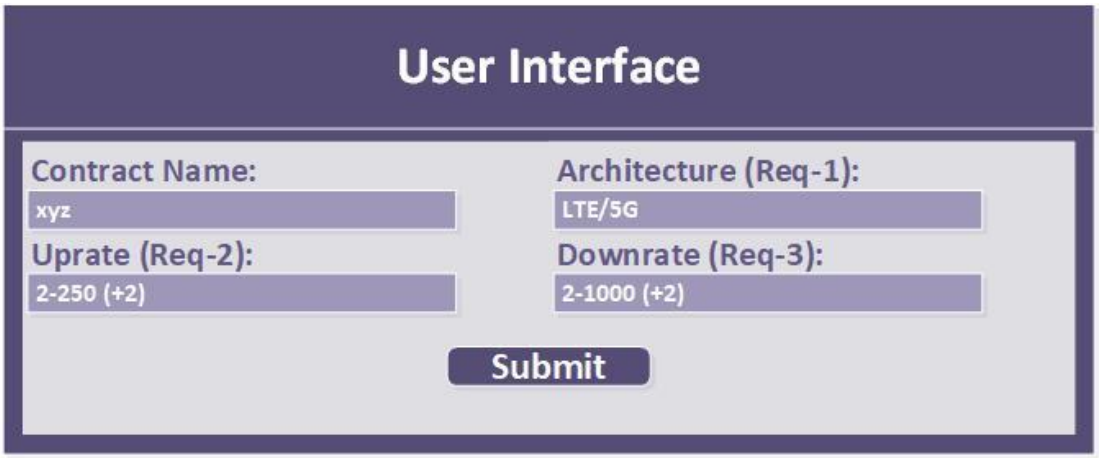
Contract

The **database** for contracts contains the following information shown in the below mentioned diagram. These contracts are further inputted to the Intent Application. Contracts look like this:



ID	Name	Architecture ID	Uprate	Downrate
1	xyz	1 (LTE)	50	100

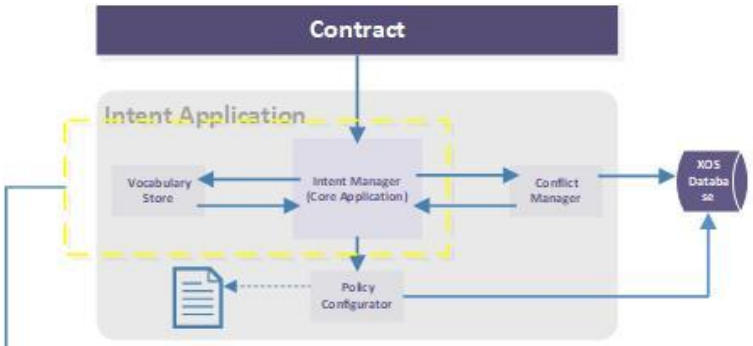
The above contracts in database are stored which the user defines through a **GUI** interface provided by us. GUI for the user looks like:



Intent-manager interaction **Vocabulary-store**

The intent manager interacts with the vocabulary-store (highlighted in yellow), using the below part of mentioned algorithm (details of algorithm are given in the Section of Algorithm).

The main process is to get the details of the architectures supported by the system. This helps the Intent Manager to validate/verify the configuration that is required to be implemented. The figure is as followed:

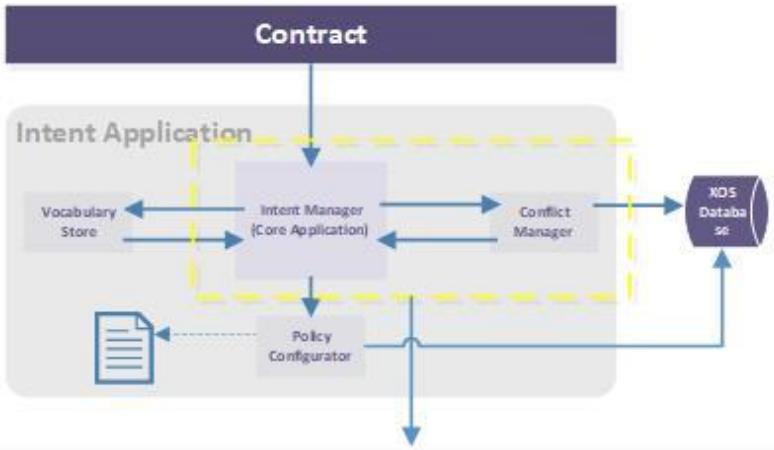



```
① object = initialize_architecture_details(contracts);
② modules = fetch_nodes(object);
③ modules = modules.sortby(sequence_number);
④ graph = NULL;
⑤ i = INITIAL_START_INDEX;
   while ( i < modules.length; i++ )
 a.    if (modules[i].sequence_number == 1)
    01.    graph.push(modules[i]);
 b.    Else
    01.    module_exists = graph.module_exists(modules[i]);
    02.    if(module_exists)
        001.    module_relations = fetch_noderelations(modules[i].id);
        002.    j = INITIAL_START_INDEX;
        003.    while( j < module_relations.length; j++)
            0001.    if(graph.link_does_not_exists(module_relations[j].mod_id
            0002.      , module_relations[j].relation_id))
                00001.    graph.link(modules[i].id, module_relations[j], 1);
    03.    Else
        001.    module_relations = fetch_noderelations(modules[i].id);
        002.    j = INITIAL_START_INDEX;
        003.    while( j < module_relations.length; j++)
            0001.    graph.link(modules[i].id, module_relations[j], 1);
⑥ forward datastructure( graph );
```

Intent-manager interaction **Conflict-manager**

The intent manager interacts with the conflict-manager (highlighted in yellow), using the below part of mentioned algorithm (details of algorithm are given in Section of Algorithm).

This highlighted area gives the Intent Manager a conflict result. Then further if the conflict is found, the default slice detail is provided as a configuration to the policy-configurator by the intent-manager. The figure is as follows:




```
① forward datastructure( graph );
② i = INITIAL_START_INDEX;
   while ( i < graph.length; i++ )
  i.    if ( graph[i].sequenceNumber is not ok )
    01.    problemInOrder = true;
③ while ( end of relations for each graph )
  i.    if (graph[i].relationalNode)
    01.    conflictValue = true;
④ return conflictValue;
```

Database Overview

The database part of vocabulary-store contains the necessary information that is required to show what the current **architecture supports**.

Table Name: Architectures_Supported			
***(PK)* id**	**architecture_name**	**description**	**modules**
1	LTE	some text	json string value (*)
2	5G	…	…
3	…	…	…
Note:			

This is the table that shows the module-details for architecture separately:

Table Name: Modules_Architecture					
***(PK)* id**	***(FK)* arch_id**	**node_name**	**sequence_number**	**is_end_node**	**relational_nodes**
1	1 (i.e. LTE)	UE	1	1	2
2	1	eNB	2	0	1, 3 (1)
3	1	vMME	3	0	2, 4, 5, 6, 7
4	1	NSSF	4	1	3
5	1	vHSS	5	1	3
6	1	vSPGWC	6	0	3, 7
7	1	vSPGWU	7	1	2, 3,6
Note(s)		1. *For example, 1, 3, 7 are the relational nodes for eNB. This refers that eNB has 1, 3, 7 connected services to it:* a. *1 is referred to UE* b. *3 is referred to vMME*			

Table Name: Modules_Relations				
***(PK)* id**	***mod_id***	**node_name**	**relation_id**	**link_specs**
1	1	UE	2	1
2	2	eNB	1	1
3	2	eNB	3	1
4	3	vMME	2	1
5	3	vMME	4	1
6	3	vMME	5	1
7	3	vMME	6	1
8	3	vMME	7	1
9	4	NSSF	3	1
10	5	vHSS	3	1
11	6	vSPGW-C	3	1
12	6	vSPGW-C	7	1
13	7	vSPGW-U	2	1
14	7	vSPGW-U	3	1
15	7	vSPGW-U	6	1
Note(s)		*This table helps the intent manager to draw a service graph for the topology to be created*		

Algorithm

1. Part-1: intent-manager interacting with vocabulary-store

This part explains the interaction between

- Intent Manager
- Vocabulary Store

Steps:

① Initialize the Architecture details

② Fetch tabular information from vocabulary-store

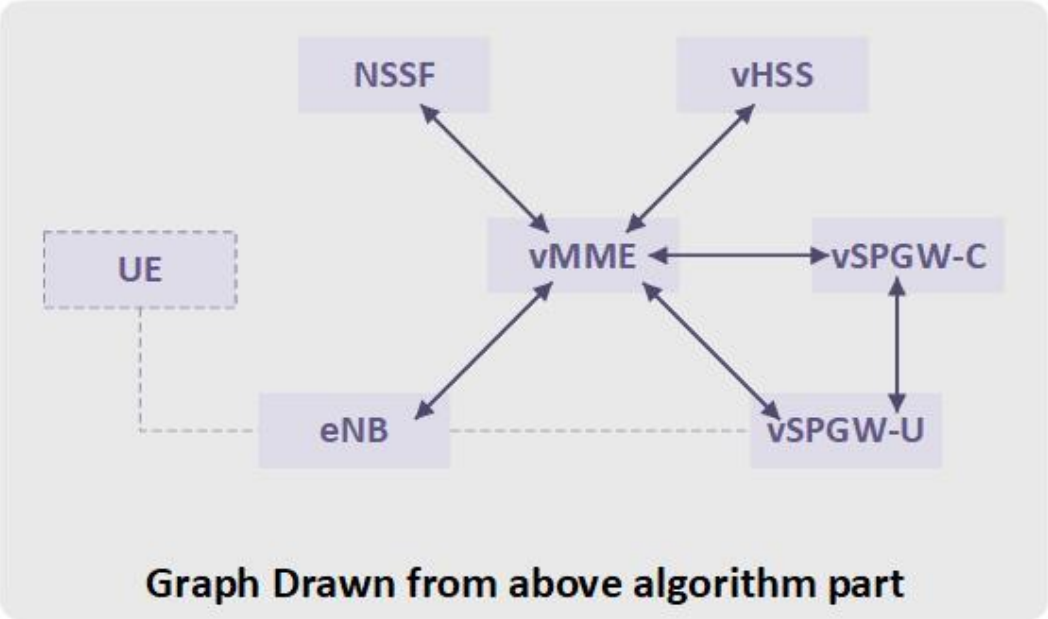
③ Define relations between the Nodes

④ Forward the information to conflict-manager

Code:

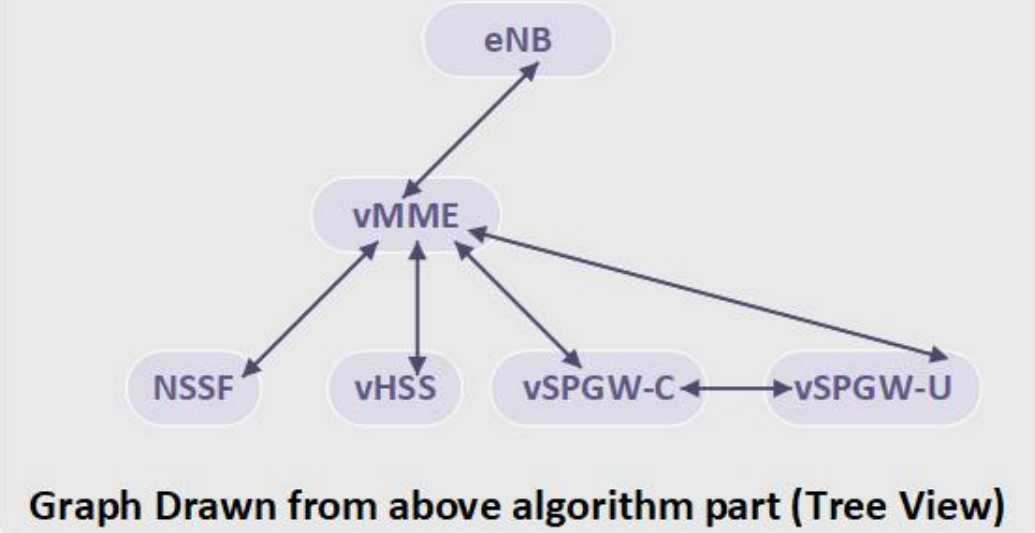
```
① object = initialize_architecture_details(contracts);
② modules = fetch_nodes(object);
③ modules = modules.sortby(sequence_number);
④ graph = NULL;
⑤ i = INITIAL_START_INDEX;
  while ( i < modules.length;  i++ )
 a.    if (modules[i].sequence_number  == 1)
    01.    graph.push(modules[i]);
 b.    Else
    01.    module_exists = graph.module_exists(modules[i]);
    02.    if(module_exists)
        001.    module_relations = fetch_noderelations(modules[i].id);
        002.    j = INITIAL_START_INDEX;
        003.    while( j < module_relations.length;  j++)
            0001.    if(graph.link_does_not_exists(module_relations[j].mod_id
            0002.      , module_relations[j].relation_id))
                00001.    graph.link(modules[i].id,  module_relations[j], 1);
    03.    Else
        001.    module_relations = fetch_noderelations(modules[i].id);
        002.    j = INITIAL_START_INDEX;
        003.    while( j < module_relations.length;  j++)
            0001.    graph.link(modules[i].id,  module_relations[j], 1);
⑥ forward datastructure( graph );
```

Graph drawn with the help of above algorithm (with the help of above two tables named as Modules_Architecture and Module_Relations) is as follows:



Graph Drawn from above algorithm part

This graph is drawn as a tree data-structure. Which is then passed onto the conflict-manager. The job of the conflict manager is



Graph Drawn from above algorithm part (Tree View)

2. Part-2: intent-manager interacting with conflict-manager

This part explains the interaction between:

- Intent Manager
- Conflict Manager

Steps:

① Forward the information to conflict-manager

② Check the order validity for each of the pushed nodes

③ Check the relational mapping validity for each pushed node

④ Respond conflict-value (true/false) to the intent-manager

Code:

```
① forward datastructure( graph );
② i = INITIAL_START_INDEX;
   while ( i < graph.length;  i++ )
  i.    if ( graph[i].sequenceNumber  is not ok )
     01.    problemInOrder = true;
③ while ( end of relations for each graph )
  i.    if (graph[i].relationalNode)
     01.    conflictValue = true;
④ return conflictValue;
```

References

1. Paper - Policy Framework for the Next Generation Platform as a Service
2. Presentation - Policy Framework for the Next Generation Platform as a Service