SRIMS TOSCA comments

At a workshop 21/8/19 we identified some areas where the current SRIMS TOSCA documents were not conformant to the TOSCA specification but contained features required for operation. This document contains proposals to resolve this.

Paul Jordan 23/8/19 Issue 1

GUI parameter definitions invalid

## Description

topology\_template/inputs requires additional parameters which are used by the runtime GUI to control the form which gathers parameter data from the planner, the current parameter names are defaultValue:

displayName:

editable:

display:

group:

As of 22/08/2019 SRIMS adds these parameters are declared at the same level and the parameter type. The problem is that TOSCA does not allow arbitrary custom extensions at this level.

## Analysis

In TOSCA 1.1 Inputs are a parameter\_defintion\_list. Parameter definitions allow keywords of type and value plus all the key names of Property definition which are description, required, default, status, constraints and entry\_schema.   
TOSCA 1.2 adds external-schema and metadata. puccini functionality is incorrect as it rejects metadata at this level  
TOSCA 1.3 does not add anything here  
  
One solution is to move these parameter definitions into a metadata section. TOSCA says that metadata should not affect runtime behaviour but this would still be true of the orchestrator as it would only affect the GUI restage. The advantage is improved standards conformance but in this case the benefit is limited until the validator can be corrected. The disadvantage is that the type of editable and display moves from the required boolean to the less restrictive string. If the GUI encounters a string value which want not true or false, the GUI operation would be unpredictable. The disadvantage does not seem to be worth the benefit.

## Proposal

The proposed solution is to add an explicit prefix to all parameter definitions which are BT specific. This would make it simple to visually and programmatically identify and filter such extensions where necessary. e.g.  
topology\_template:  
 inputs:  
 in\_orderID:  
 type: string  
 required: true   
 description:   
 constraints:  
 - category: ProvideVPRN  
 value: "@getValue(fetchOrderId)"  
 bt\_displayName:   
 bt\_editable:  
 bt\_display:  
 bt\_group:  
This would not prevent moving these parameters into a metadata clause at a later date if required.

GUI group is defined bottom up

## Description

topology\_template/inputs/input\_name includes a BT specific keyword of group. This is used by the runtime GUI to visually group input parameter fields. As currently defined every parameter definition has a group string. A more common way of defining group membership is to define the group name and then list the members of the group. This is easier to maintain, allows groups to contain groups and mitigates the risk that the group value at parameter level is misspelt.

## Analysis

An option is to create a new datatype for GUI\_group. The inputs section will contain entries of type GUI\_group, the value of GUI\_group would be a list comprising other inputs including other GUI\_group instances. (The bt extension parameters of editable and display name could be included but would carry no information because GUI\_group items would always be displayed with their name and would not be editable. display is still relevant as TOSCA input names cannot contain spaces.  
This solution allows top down definition of GUI groups, and allows groups to contain groups the risk of mistyping is reduced because we could check that all listed group members are defined inputs.  
In TOSCA 1.1 the gui\_group would have to be a string with some delimiter which is clunky and not recommended but   
TOSCA 1.2 allows the GUI\_group to be a list  
The datatype can be defined as required false as the orchestrator does not need a value

## Proposal

Define and use a GUI\_group datatype as a list in TOSCA 1.2 e.g.  
  
data\_types:  
 GUI\_group:   
 derived\_from: list  
 description: A datatype for visually grouping input fields on the runtime GUI  
 required: false

topology\_template:  
 inputs:  
 VPRN\_section:  
 type: GUI\_group  
 value:   
 - VPRN\_Attributes\_subsection  
  
 VPRN\_Attributes\_subsection:  
 type: GUI\_group  
 value:   
 - in\_vrfType  
 - in\_vprnAutonomousSystem

Operations cannot have outputs

## Description

Node definitions contains operations which are invoked by workflow. The operations may have defined input parameters, but the spec does not allow outputs to be defined. Output values are sometimes needed as inputs to the next stage

## Analysis

This appears to be an omission in the TOSCA 1.2 spec. It has been fixed in the TOSCA 1.3 which now looks stable. That version defines outputs form node definitions as;

outputs:

<attribute mappings>

where <attribute mappings> is a map

and where each entry is of the form:

output\_name: [ SELF, <optional\_capability\_name>, <attribute\_name>, <nested\_attribute\_name\_or\_index\_1>, ..., <nested\_attribute\_name\_or\_index\_or\_key\_n> ]

There must be an attribute defined in the node to hold the result of every output. Since SRIMS appears to return an error code and error message with every result it might be convenient to ensure they are returned as a structure and define a TOSCA datatype to use when defining the attributes like this which is valid in 1.2 even with the

data\_types:

# map of datatype definitions

# some are defined in the spec, this is for refining those or creating new ones

errorDetail:

description: data structure for use when returning from operations

properties:

errorCode:

type: integer

errorMessage:

type: string

operationResult:

derived\_from: errorDetail

properties:

returnValue:

type: string

This type can be used in the attributes section of the node definition

attributes:

# map of <attribute\_definitions>

MyOperationResultHolder:

type: operationResult

so that it can be used to hold the operation output:

outputs:

output\_name: [ SELF, MyOperationResultHolder ]

Otherwise just repeat the line above for the errorCode and errorMessage as shown in the proposal

## Proposal

Use the format defined in the TOSCA 1.3 spec. For node interfaces like that in VPRN\_requirements the changes would be

1. Define an attribute to hold each output

attributes:

# map of <attribute\_definitions>

out\_deviceInterfaceObjectID:

type: string

out\_vprnServiceName:

type: string

out\_lastOperationErrorCode:

type: integer

out\_lastOperationErrorMessage:

type: string

I assume that the error responses would be read immediately and so it is ok to overwrite them, if not use a structure as explained in the analysis or more specific names for them.

1. Change the format of outputs to:

interfaces:

placeVPRN:

outputs:

DeviceInterfaceObjectID: [SELF, out\_deviceInterfaceObjectID]

errorCode: [SELF, out\_lastOperationErrorCode]

errorMessage: [SELF, out\_lastOperationErrorMessage]

configure:

outputs:

vprnServiceName: [SELF, out\_vprnServiceName]

errorCode: [SELF, out\_lastOperationErrorCode]

errorMessage: [SELF, out\_lastOperationErrorMessage]