TOSCA Enhancements for Operation Outputs

June 4, 2018

# Introduction

TOSCA allows service template designers to define the *signature* of interface operations by specifying the set of *inputs* (and their types) that need to be provided when invoking an operation. However, no equivalent support currently exists for defining the expected return values for interface operations.

We recommend that TOSCA should be extended as follows:

* Allow service template designers to define *operation* *outputs*: named values that are expected to be returned by interface operations.
* Provide a mechanism for reflecting output values returned by interface operations into *attributes on nodes or relationships.*

Adding support for output definitions addresses two shortcomings in the current grammar:

* It completes the *contract* between the orchestrator and the artifacts that provide operation implementations (the “plug-ins”) by specifying the set of output values that must be returned in order for the artifact to be a valid implementation of the operation.
* Without operation output functionality, there is no mechanism for a TOSCA orchestrator to obtain values for attributes specified in TOSCA node templates.

This document presents a proposal for providing output definition functionality.

# Output Definitions and Assignments

Our proposed approach leverages the *property mapping* grammar that is part of substitution mappings to specify:

* The set of named output values that must be returned by operation implementations
* Property mappings that specify node or relationship attributes into which the returned output value must be stored.

These proposed extensions result in the following updated *operation definition grammar* that is to be used in node type or relationship type definitions:

| Keyname | Required | Type | Description |
| --- | --- | --- | --- |
| description | no | description | The optional description string for the associated named operation. |
| implementation | no | [operation implementation definition](#TYPE_YAML_STRING) | The optional definition of the operation implementation  |
| inputs | no | list of parameter definitions | The optional list of input property definitions (i.e., parameter definitions) for operation definitions that are within TOSCA Node or Relationship Type definitions. This includes when operation definitions are included as part of a Requirement definition in a Node Type. |
| outputs | no | list of property mappings | The optional list of property mappings that specify named operation output values and their mappings onto attributes of the node\_type or relationship that contains the interface within which the operation is defined. |

# Output Definition Grammar

The proposal above results in the following grammar for operation output definitions:

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

The various entities in this grammar are defined as follows:

| Parameter | Required | Type | Description |
| --- | --- | --- | --- |
| SELF | SOURCE | TARGET  | yes | string | For operation outputs in interfaces on node templates, the only allowed keyname is SELF: output values must always be stored into attributes that belong to the node template that has the interface for which the output values are returned.For operation outputs in interfaces on relationship templates, allowable keynames are SELF, SOURCE, or TARGET. |
| <optional\_capability\_name> | no | string | The optional name of the capability within the specified node template that contains the named attribute into which the output value must be stored. |
| <attribute\_name>  | yes | string | The name of the attribute into which the output value must be stored. |
| <nested\_attribute\_name\_or\_index\_or\_key\_\*>  | no | string| integer | Some TOSCA attributes are complex (i.e., composed as nested structures). These parameters are used to dereference into the names of these nested structures when needed. Some attributes represent list or map types. In these cases, an index or key may be provided to reference a specific entry in the list or map (as named in the previous parameter) to return.  |

# Examples

The service template below shows an example of the operation output syntax. The template is used to create a compute node. The config operation of the Standard lifecycle returns both the private and the public IP addresses of the config node. The *property mappings grammar* is used to reflect these addresses into the appropriate Compute node attributes:

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2\_0description: Template for creating compute node topology\_template: node\_templates: node: type: tosca.nodes.Compute interfaces: Standard: configure: outputs: ip1: [ SELF, private\_address ] ip2: [ SELF, public\_address ] |

Some operation outputs may need to be reflected into attributes of capabilities of nodes, rather than in attributes of the nodes themselves.

The following example shows how an IP address returned by a config operation is stored in the ip\_address attribute of the endpoint capability of a Compute node:

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2\_0description: Template for creating compute nodetopology\_template: node\_templates: compute: type: tosca.nodes.Compute interfaces: Standard: config: outputs: ip1: [ SELF, endpoint, ip\_address ] |

# Additional Notes

It is possible for multiple operations to define outputs that map onto the same attribute value. For example, a *create* operation could include an output value that sets an attribute to an initial value, and the subsequence *configure* operation could then update that same attribute to a new value.

It is also possible that a node template assigns a value to an attribute that has an operation output mapped to it (including a value that is the result of calling an intrinsic function). Orchestrators could handle this scenario in one of the following two ways:

* The orchestrator could flag such templates as invalid.
* Alternatively, the orchestrator could use the assigned value for the attribute as its initial value. After the operation runs that maps an output value onto that attribute, the orchestrator must then use the updated value, and the value specified in the node template will no longer be used.

Further discussion is required to decide on the best approach.