Sharing Types between Profiles

October 6, 2020

This document presents simple examples that motivate the need for sharing types between profiles.

# Valid Target Types Use Case

## Profile Version 1

Assume I create version 1 of a profile that defines (among other things) a Host capability type and a corresponding HostedOn relationship type as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0profile: org.profile.v1capability\_types: Host: description: Hosting capabilityrelationship\_types: HostedOn: valid\_target\_types: [ Host ] |

## Platform Profile

Now let’s assume I create a platform-specific profile that defines (among other things) a Platform node type. The Platform node type defines a capability of type Host (as defined in the org.profile.v1 profile):

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0imports: - profile: org.profile.v1 namespace: p1node\_types: Platform: capabilities: host: type: p1:Host |

## Profile Version 2

At some later point of time, I create version 2 of the org.profile. This profile just adds a Credential data type (in addition to defining the Host capability type and the HostedOn relationship type), as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0profile: org.profile.v2capability\_types: Host: description: Hosting capabilityrelationship\_types: HostedOn: valid\_target\_types: [ Host ]data\_types: Credential: properties: key: type: string |

## Platform Service

Now let’s assume I create a template for a service that is to be hosted on the platform defined in the org.platform profile. The template introduces a Service node type that has a requirement for the platform’s Host capability. It also has a credential property of type Credential as defined in org.profile.v2:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0imports: - profile: org.profile.v2 namespace: p2 - profile: org.platform namespace\_prefix: plnode\_types: Service: properties: credential: type: p2:Credential requirements: - host: capability: p2:Host relationship: p2:HostedOntopology\_template: node\_templates: service: type: Service properties: credential: key: password requirements: - host: platform platform: type: pl:Platform |

This service template will fail validation, since the platform node template does not define a capability of a type that is compatible with the valid\_target\_types specified by the host requirement in the service node template. TOSCA grammar extensions are needed to specify that the Host capability type defined in org.profile.v2 is the same as the Host capability type defined in org.profile.v1

# Derivation Rules Use Case

## Profile Version 1

Assume I create version 1 of a profile that defines (among other things) a Person data type as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0profile: org.profile.v1data\_types: Person: properties: name: type: string sex: type: string constraints: - valid\_values: [ male, female ] |

## Participant Version 1

Now let’s assume I create service-specific node types based on Version 1 of this profile. These types are defined in a file called participant.yaml, which defines a Participant node type as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0imports: - profile: org.profile.v1 namespace: p1node\_types: Participant: properties: person: type: p1:Person |

## Profile Version 2

At some later point of time, I create version 2 of the same profile. This profile defines the same Person data type, but for convenience it also defines Man and Woman derived data types as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0profile: org.profile.v2data\_types: Person: properties: name: type: string sex: type: string constraints: - valid\_values: [ male, female ] Woman: derived\_from: Person properties: sex: female Man: derived\_from: Person properties: sex: male |

## Participant Version 2

Now let’s assume I create an updated version of my service. This updated version re-uses the service-specific node types defined in Version 1 by importing the old participant.yaml file, but it also adds MaleParticipant and FemaleParticipant node types. For example:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0imports: - profile: org.profile.v2 namespace: p2 - file: participant.yaml namespace: panode\_types: FemaleParticipant: derived\_from: pa:Participant properties: person: type: p2:Woman |

In this example, the FemaleParticipant node type uses a property refinement to “refine” the person property defined in the base Participant node type. However, this refinement will fail validation, since the Woman data type defined in org.profile.v2 will not be recognized as a derived type of the Person data type defined in org.profile.v1.

TOSCA language support is need to allow profile designers to indicate that the Person datatype defined in org.profile.v2 is the same as the Person data type defined in org.profile.v1.