Consider the following scenario presented by ETSI NFV:

* A network service template (nsd.yaml) imports type definitions for 2 different VNFs (defined in vnf1.yaml and vnf2.yaml respectively)
* vnf1.yaml defines an interface type called MyCompany.nfv.interfaces.VnfIndicator
* vnf2.yaml defines an interface type that is also called MyCompany.nfv.interfaces.VnfIndicator

How is a TOSCA parser expected to handle these two interface type definitions with the same name?

# Answer from OASIS TOSCA:

 TOSCA uses namespaces to handle namespace conflicts. To answer ETSI’s question, we present a brief review of TOSCA namespaces:

* Each TOSCA service template defines a default unnamed namespace. Type definitions in a TOSCA service template are automatically placed in that default unnamed namespace
* When a TOSCA service template imports another template, it has two options:
	+ It can import definitions from the second template into its default unnamed namespace. This option is used by omitting the ‘namespace\_prefix” keyword in the import statement
	+ It can import definitions from the second template into a specific named namespace that is different from the default namespace. That named namespace is specified using the ‘namespace\_prefix’ keyword in the import statement.

# Annex A:

# A.1 Handling the ETSI Scenario based on TOSCA-Simple-Profile-YAML-v1.3

Based on the discussion about TOSCA namespaces, the scenario as presented by ETSI must be flagged by a TOSCA parser as an error, since it will result in duplicate definitions of the MyCompany.nfv.interfaces.VnfIndicator interface type in the default namespace of the nsd.yaml template.

To fix this issue, we should consider two different scenarios:

* The two imported definitions (with the same name) are intended to be different
* The two imported definitions (with the same name) are intended to be the same

## A.1.1 VNFIndicator Type Definitions are Intended to be Different

This case can easily be handled by importing the different VNF definitions into two different namespaces and then using namespace prefixes to identify the correct interface name.

For example. the nsd.yaml could include the following:

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3imports:  - file: vnf1.yaml namespace\_prefix: vnf1 - file: vnf2.yaml namespace\_prefix: vnf2node\_types: TypeForVNF1: ---- interfaces: indicator: type: vnf1:MyCompany.nfv.interfaces.VnfIndicator |

## A.1.2 VNFIndicator Type Definitions are Intended to be The Same

This scenario can be supported by splitting out the VNFIndicator interface type definition into its own TOSCA service template file—e.g. indicator.yaml—and then importing that indicator.yaml file into both vnf1.yaml and vnf2.yaml. Again, there are two scenarios to consider:

### A.1.2.1 indicator.yaml in shared repo

As long as vnf1.yaml and vnf2.yaml can retrieve indicator.yaml from the same CSAR file or from the same repository, a TOSCA parser can determine that the imported indicator.yaml files are the same file, and the definitions in the indicator.yaml file need to be processed only once.

For example, the import chain could look as follows. We first show the indicator.yaml file:

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3metadata: template\_name: indicator.yaml interface\_types: MyCompany.nfv.interfaces.VnfIndicator: description: common VNFIndicator interface type shared by multiple VNFs |

The indicator.yaml file is then imported by both the vnf1.yaml and the vnf2.yaml files as follows. We first show vnf1.yaml

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3metadata: template\_name: vnf1.yaml imports: - indicator.yaml node\_types: # Type definitions for VNF 1 |

Next we show vnf2.yaml

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3metadata: template\_name: vnf2.yaml imports: - indicator.yaml node\_types: # Type definitions for VNF 2 |

The VNF definitions are then imported into the nsd.yaml file as before.

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3imports:  - file: vnf1.yaml namespace\_prefix: vnf1 - file: vnf2.yaml namespace\_prefix: vnf2node\_types: TypeForVNF: ---- interfaces: indicator: type: vnf1:MyCompany.nfv.interfaces.VnfIndicator# OR: type: vnf2:MyCompany.nfv.interfaces.VnfIndicator# Independent of which namespace prefix is used, the same VnfIndicator type is # identified. |

In the nsd.yaml, independent of whether the vnf1 or vnf2 namespace prefix is used, the same VnfIndicator type is identified.

### A.1.2.2 indicator.yaml in different repos

The more challenging scenario occurs when the indicator.yaml file cannot be retrieved from a shared repo. In that case, each VNF can include its own (private) copy of the shared indicator.yaml file (i.e. the CSAR package for VNF1 includes a copy of indicator.yaml and the CSAR package for VNF2 includes a copy of indicator.yaml as well). However, without additional information, a TOSCA parser does not know if these two copies of indicator.yaml are intended to be the same or not.

To address this issue, TOSCA allows service templates to identify themselves by advertising a “well-known name”. In TOSCA Simple Profile in YAML v1.3, the top-level “namespace” keyword is used for this purpose. For example, the indicator.yaml file could advertise a well-known name as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3metadata: template\_name: indicator.yaml **namespace: org.etsi.nfv.indicator:1.0** interface\_types: MyCompany.nfv.interfaces.VnfIndicator: description: common VNFIndicator interface type shared by multiple VNFs |

This approach allows TOSCA parser to mark imported service template as identical as follows:

If an imported service template advertises a well-known name (using the namespace keyword), the parser will check whether any previously-imported templates advertise the same well -know name, and if so, it will mark the two imported template as “the same” and re-use the definitions from the previously-imported template.

Note that the use of the “namespace” keyword is misleading. Namespaces are constructs used by ***importing*** templates and ***imported*** templates should not have information (let alone determine) the name of the namespace into which they are imported. This is why TOSCA v2.0 has renamed this keyword to “profile” as follows:

|  |
| --- |
| tosca\_definitions\_version: tosca\_2\_0metadata: template\_name: indicator.yaml **profile: org.etsi.nfv.indicator:1.0** interface\_types: MyCompany.nfv.interfaces.VnfIndicator: description: common VNFIndicator interface type shared by multiple VNFs |

TOSCA v2.0 also supports “importing by well-known profile name” which makes it even easier to share definitions between different service templates.