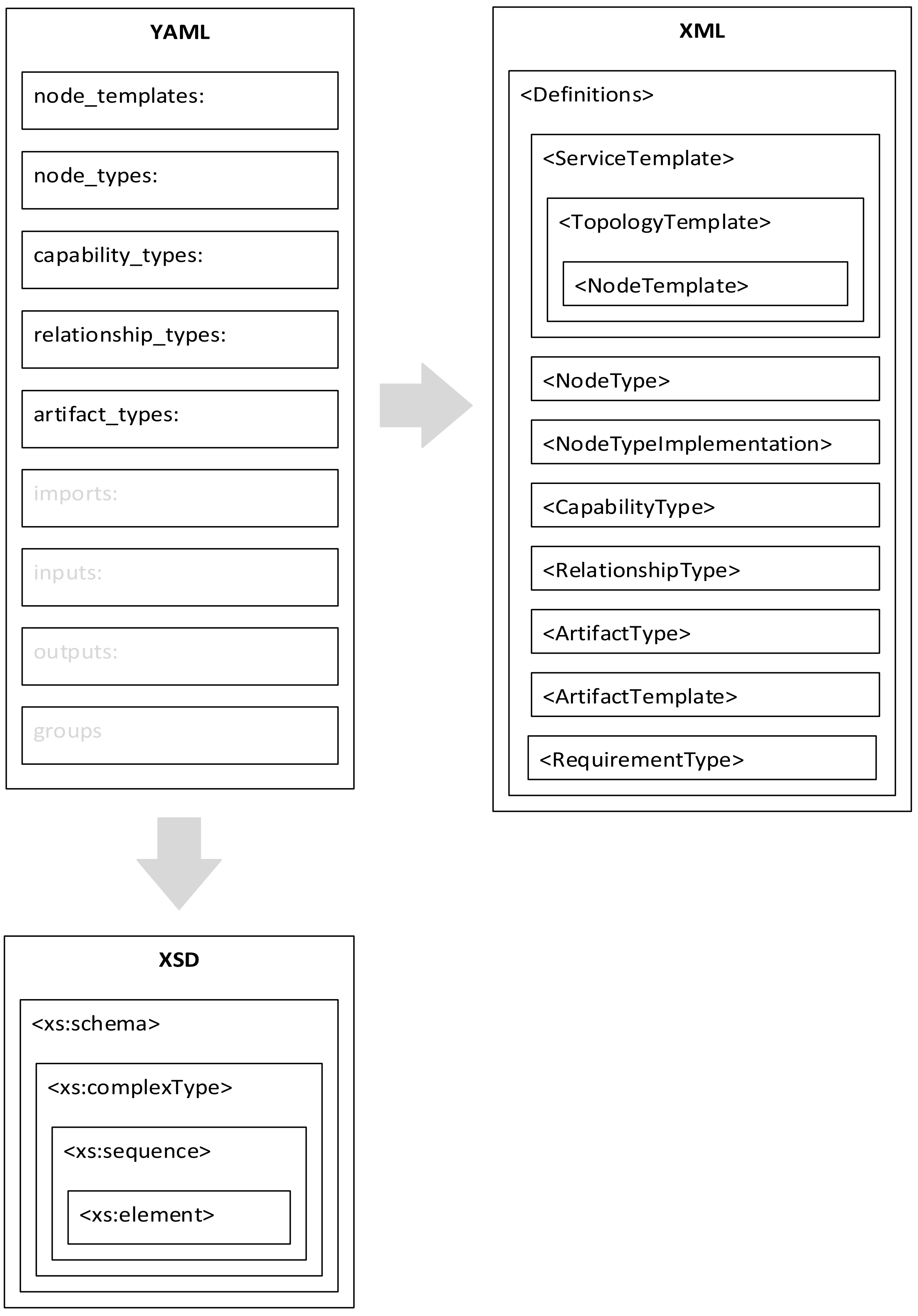
YAML - XML converter

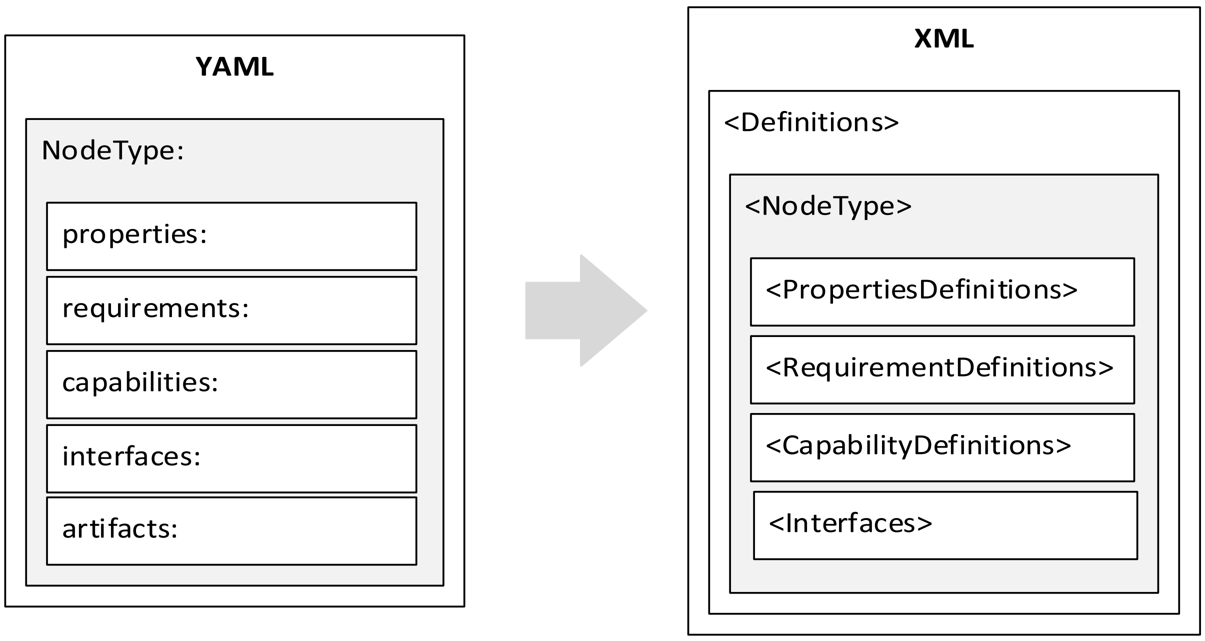
1 Table of content

3 Overview

These are the supported elements from the converter. At the moment imports, inputs, outputs and groups are not supported.



4 Node Type



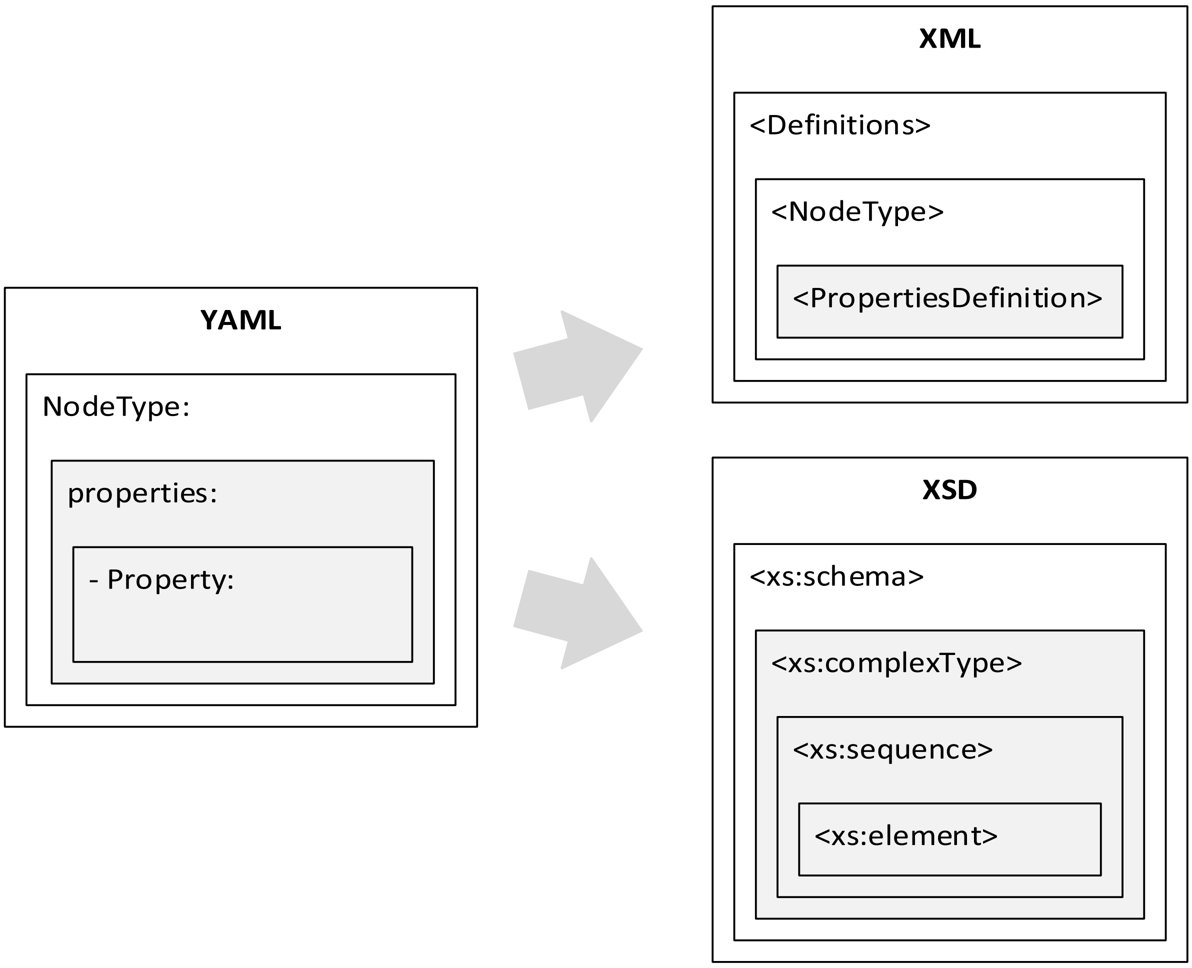
YAML

```
<node_type_name>:
    derived_from: <parent_node_type_name>
    description: <node_type_description>
    properties:
        <property_definitions>
    requirements:
        <requirement_definitions>
    capabilities:
        <capability_definitions>
    interfaces:
        <interface_definitions>
    artifacts:
        <artifact_definitions>
```

XML

```
<NodeType name=" xs:node_type_name " targetNamespace="NamespaceURL”>

    <documentation> node_type_description </documentation>

    <DerivedFrom  typeRef=" xs:parent_node_type_name "/>

    <PropertiesDefinition type="xs: node_type_nameProperties"/>

    <CapabilityDefinitions>

        <CapabilityDefintion …/>

    </CapabilityDefinitions>
    <RequirementDefinitions>

        <RequirementDefinition…/>

    </RequirementDefinitions>

    <Interfaces>

        <Interface…/>

    </Interfaces>

</NodeType>

<ArtifactTemplate>
```

4.1 Property definition



YAML

```
properties:
    <property_name>:
        type: <property_type>
```

```
            description: <property_description>
            default: <default_value>
```

XML

```
<PropertiesDefinition type="xs:[CorrespondingNodeType]Properties"/>
```

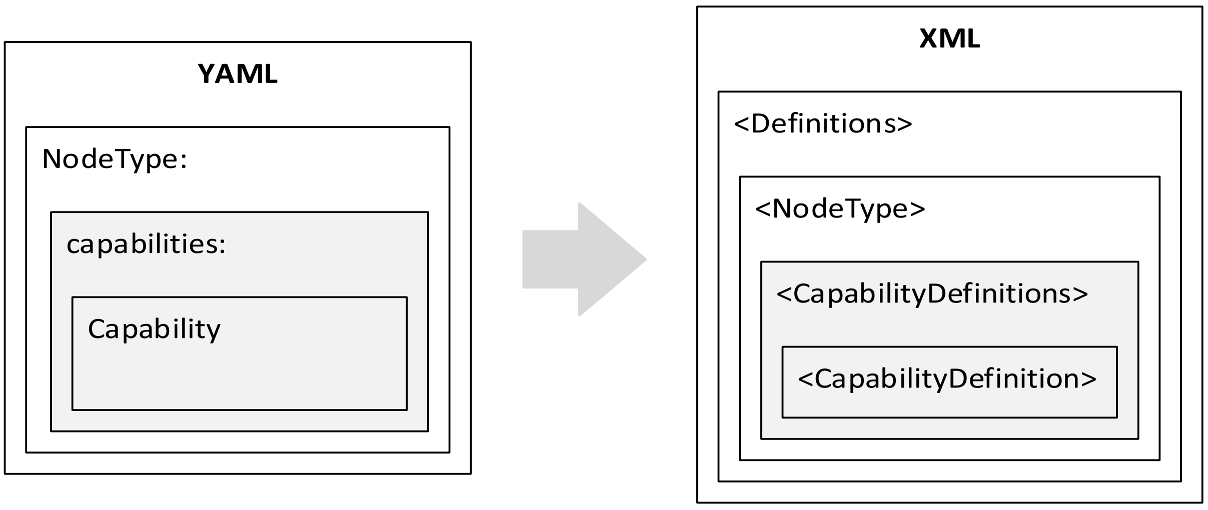
XSD

```
<xs:complexType name="t[CorrespondingNodeType]Properties">

     <xs:sequence>

          <xs:element name="property_name" type="xs: property_type" />

     </xs:sequence

</xs:complexType>

<xs:element name="[CorrespondingNodeType]Properties "
     type="t[CorrespondingNodeType]Properties Properties" />
```

Notes

[*CorrespondingNodeType*]**:** Name of the NodeType where the property corresponds to

4.2 Capability definition



YAML

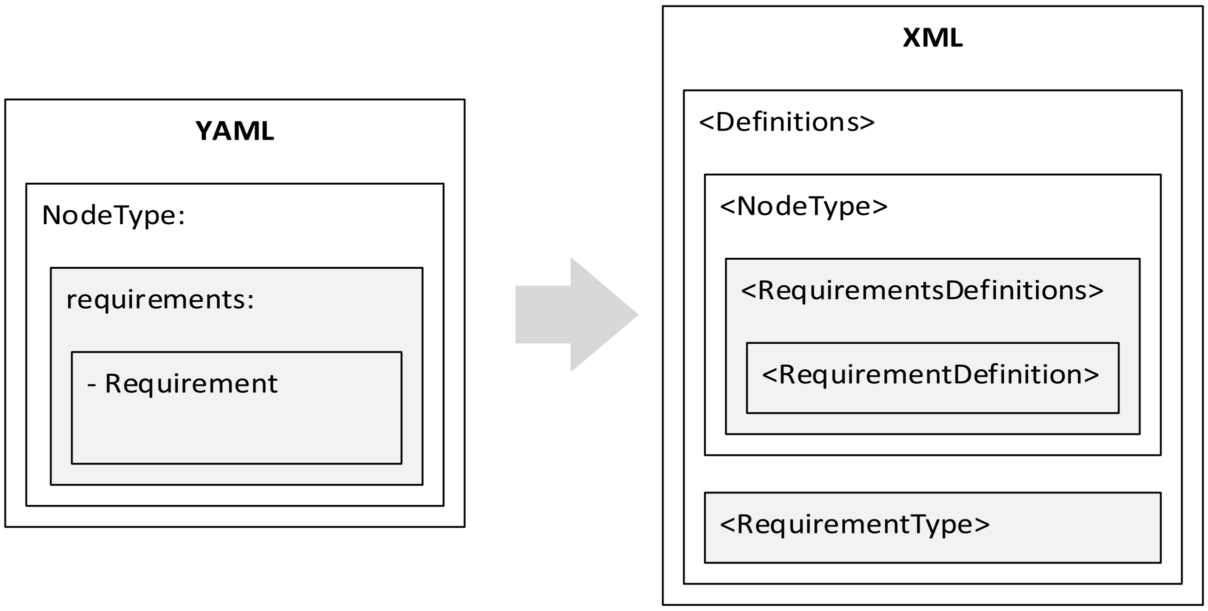
```
capabilities:
     <capability_name>:
     type: <capability_type>
```

XML

```
<CapabilityDefinitions>

    <CapabilityDefinition name="xs: capability_name "   capabilityType="xs: capability_type "/>

</CapabilityDefinitions>
```

4.3 Requirement definition

There are two different notations in YAML for requirements. Node types only support one notation which will create a requirement definition and the corresponding requirement type on XML side. In YAML there is no possibility to create requirement types. So on XML side they will be created automatically when creating a requirement. The other notation which will create a relationship is only supported in node templates.



YAML

```
requirements:
    - <requirement_name>: <capability_type_name>
```

XML

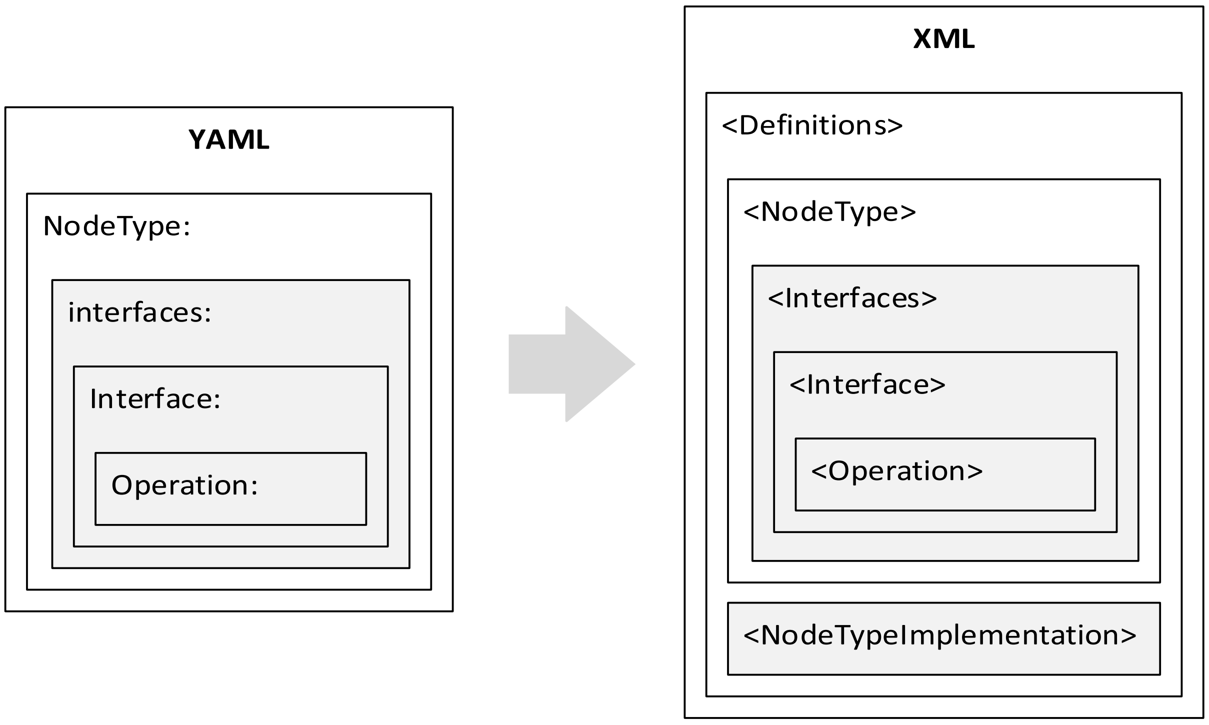
```
<RequirementDefinitions>

    <RequirementDefinition name=" xs:requirement_name " requirementType=
        "xs:capability_type_name [Capability replaced by Requirement] "/>

</RequirementDefinitions>

…
```

```
<RequirementType requiredCapabilityType="xs: capability_type_name " name=
    "xs:capability_type_name [Capability replaced by Requirement] "/>
```

Notes

[Capability replaced by Requirement]: “Capability” in capability_type_name will be replaced by “Requirement”

4.4 Interface definition



YAML

```
interfaces:
    <interface_name>:
        <operation_name>:
            implementation: <implementation_artifact_name>
```

XML

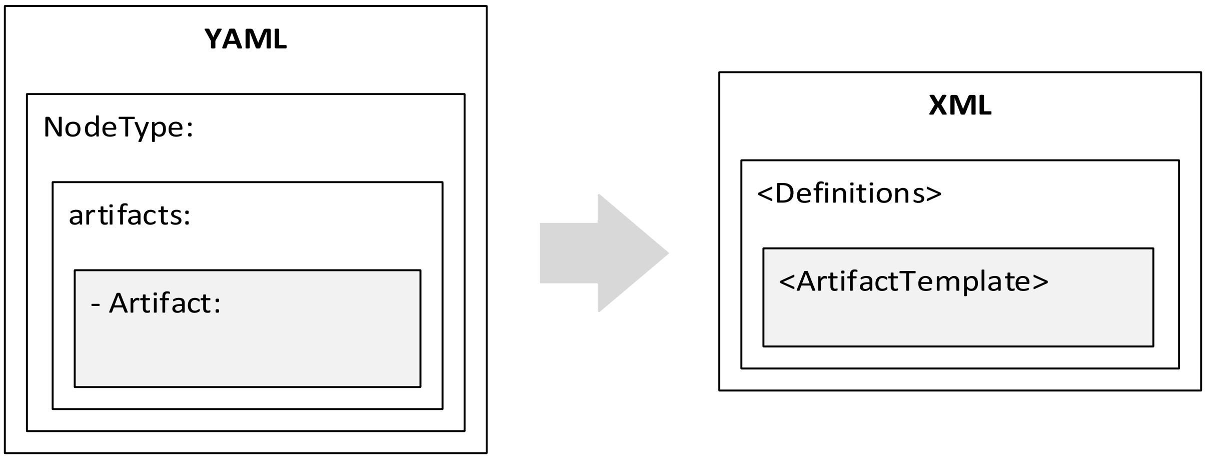
```
<Interfaces>

    <Interface name=" interface_name ">

        <Operation name="operation_name "/>

    </Interface>

</Interfaces>
```

```
…

<NodeTypeImplementation name=" xs:[CorrespondingNodeType]Implementation"
    nodeType="xs:[CorrespondingNodeType]">

    <ImplementationArtifacts>

        <ImplementationArtifact interfaceName="xs: interface_name " operationName=
            "xs:operation_name " artifactType="xs:implementation_artifact_type" artifactRef="xs:
            implementation_artifact_name "/>

    </ImplementationArtifacts>

</NodeTypeImplementation>
```

Notes

[CorrespondingNodeType]: Name of the NodeType where the interface corresponds to
Implementation_artifact_name: Name of the artifact definition
Implementation_artifact_type: Type of the artfifact

4.5 Artifact definition



YAML

```
<artifact_name>: <artifact_file_URI>
type: <artifact_type_name>
description: <artifact_description>
mime_type: <artifact_mime_type_name>
```

XML

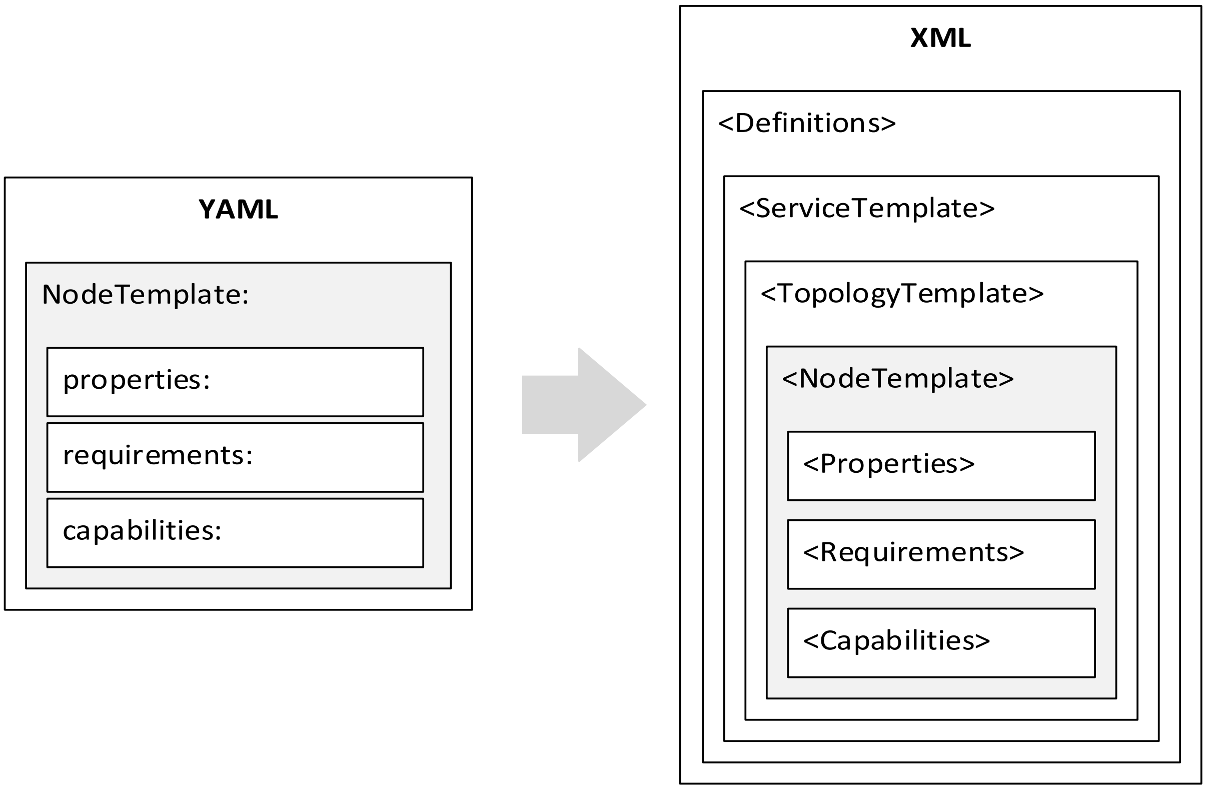
```
<ArtifactTemplate name="xs:artifact_name" id=" xs:artifact_name " type=" xs:artifact_type_name ">
```

```
    <Properties>

        <artifact_type_nameProperties types="http://www.example.org/tosca/yamlgen/types"
        xmlns="http://www.example.org/tosca/yamlgen/types"/>

    </Properties>

    <ArtifactReferences>

        <ArtifactReference reference=" xs:artifact_file_URI  [Folder]">

            <Include pattern="xs: artifact_file_URI  [File]">

        <ArtifactReference>

    <ArtifactReferences>

<ArtifactTemplate>
```

Notes

[*Folder*]: Path of folder where the artifact file is stored.
[*File*]: Name of artifact file.

5 Node Templates



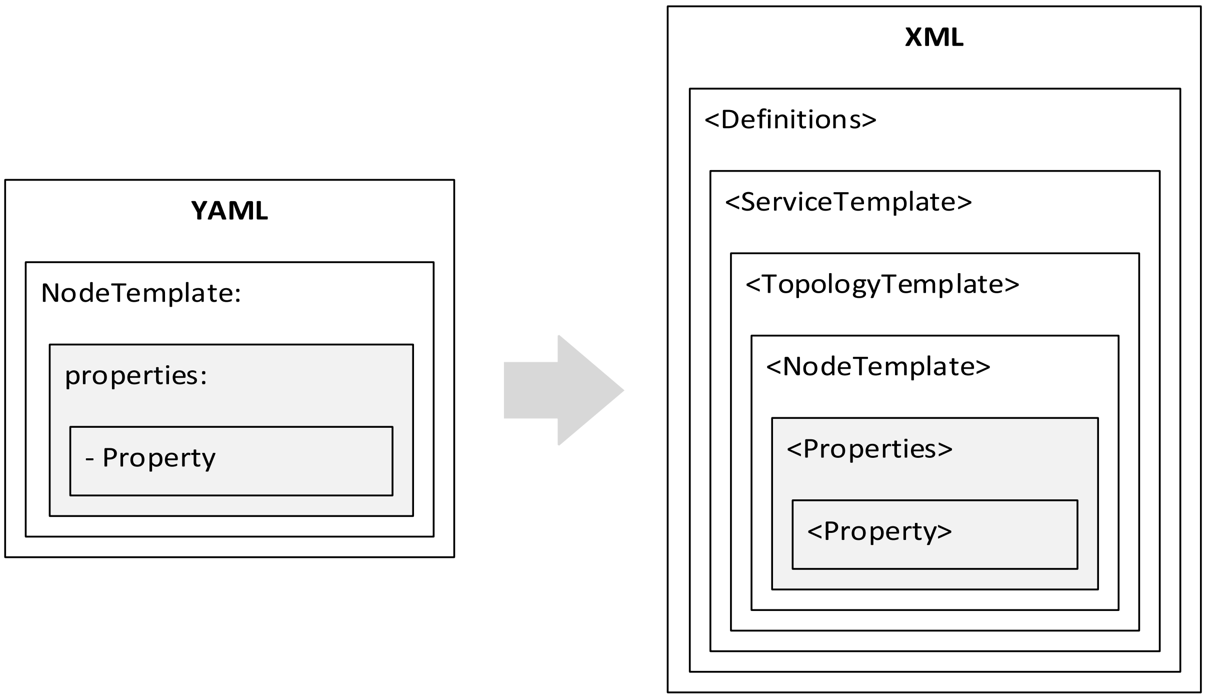
YAML

```
<node_template_name>:
    type: <node_type_name>
    description: <node_template_description>
    properties:
        <property_definitions>
    requirements:
        <requirement_definitions>
    capabilities:
        <capability_definitions>
    interfaces:
        <interface_definitions>
    artifacts:
        <artifact_definitions>
```

XML

```
<NodeTemplate name="xs:node_template_name" id=" xs:node_template_name " type="
xs:node_type_name ">

    <Properties>

        <Property…/>

    </Properties>

    <Capabilities>

        <Capability…/>

    </Capabilities>

    <Requirements>

        <Requirement…/>

    </Requirements>

</NodeTemplate>
```

5.1 Property definition



YAML

```
properties:
    <property_name>: <property_value>
```

XML

```
<xs:[CorrespondingNodeType]Properties>
    < property_name > property_value </ property_name >
</xs:[CorrespondingNodeType]Properties>
```

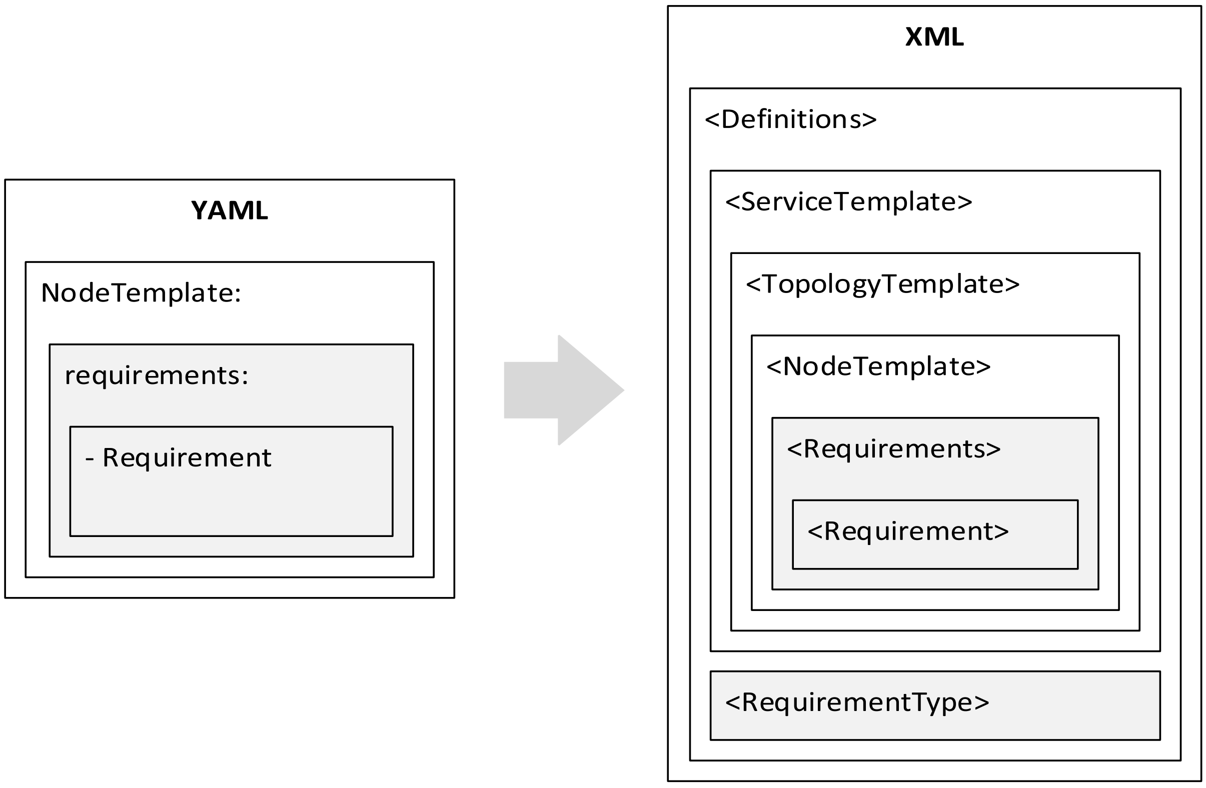
Notes

[*CorrespondingNodeType*]: Name of the assigned NodeType from the NodeTemplate the property corresponds to

5.2 Requirement definition

There are two different notations in YAML for requirements. Node Templates support both notations. First notation creates requirements and corresponding requirement types. The second notation creates relationships.

5.2.1 First notation – XML requirement



YAML

```
requirements:
    - <requirement_name>: <capability_type_name>
```

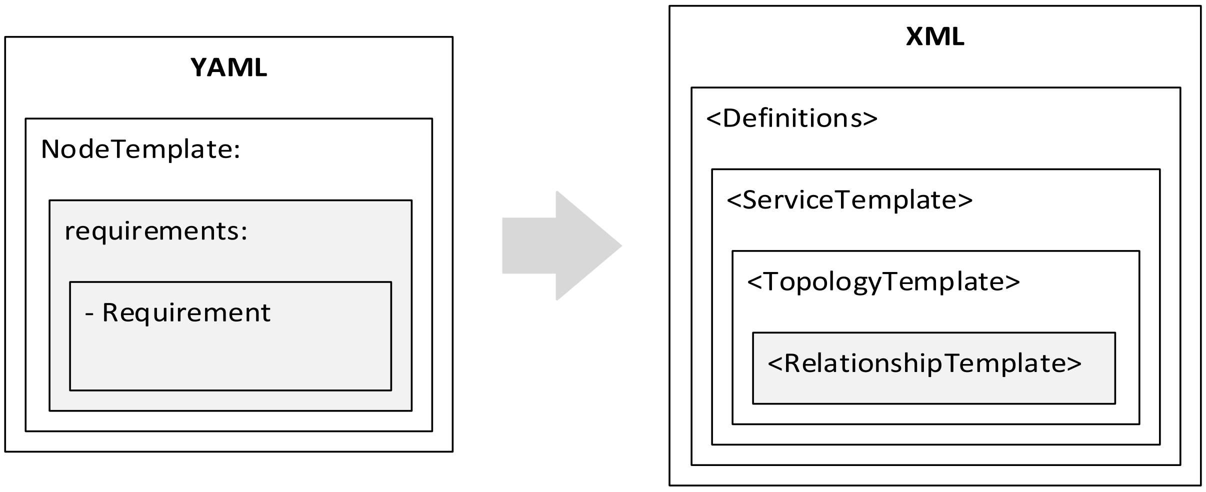
XML

```
<Requirements>

    <Requirement name="xs: requirement_name " requirementType=
        "xs:capability_type_name [Capability replaced by Requirement] "/>

</Requirements>

…

<RequirementType requiredCapabilityType="xs: capability_type_name " name=
    "xs:capability_type_name [Capability replaced by Requirement] "/>
```

Notes

[Capability replaced by Requirement]: “Capability” in capability_type_name will be replaced by “Requirement”

5.2.2 Second notation – XML relationship



YAML

requirements:
- **<requirement_name>**: **<node_name >**
relationship_type: <**relationship_name**>

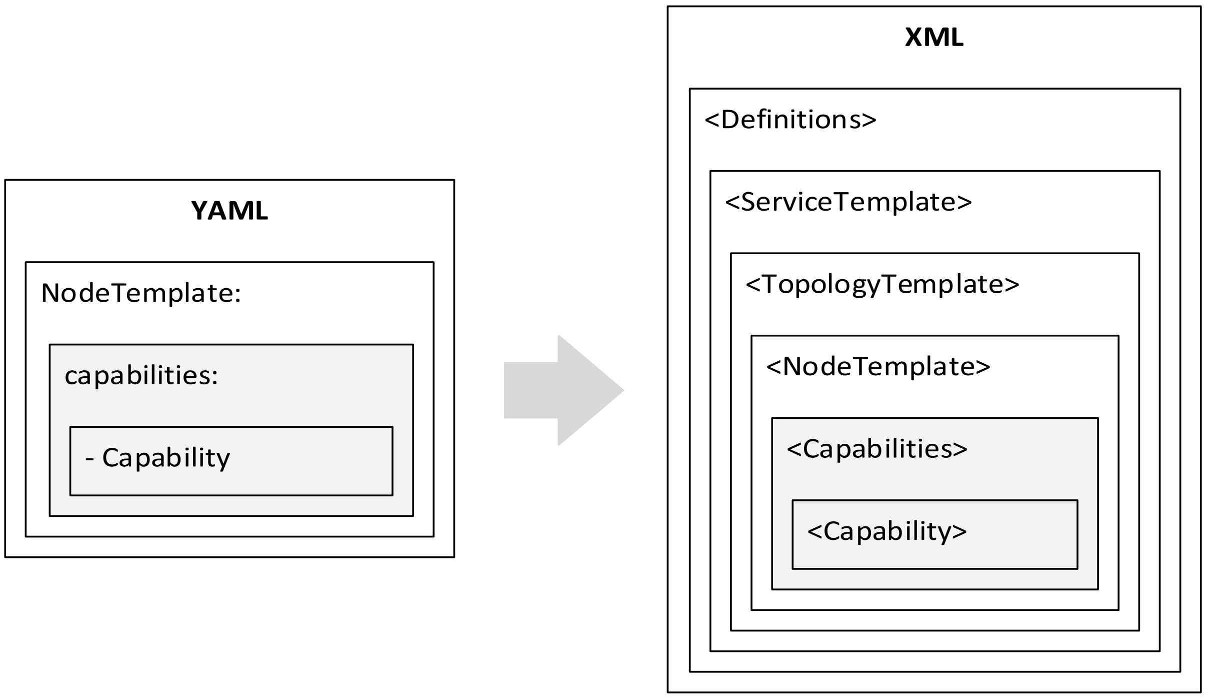
XML

<RelationshipTemplate id=" **requirement_name** " type="xs: **relationship_name** ">
<SourceElement ref="***[CorrespondingNodeTemplate]***"/>
<TargetElement ref=" **node_name** "/>
</RelationshipTemplate>

Notes

[CorrespondingNodeTemplate]: Name of the NodeTemplate the requirement corresponds to. There is only the possibility to relate NodeTemplates.

5.3 Capability definition



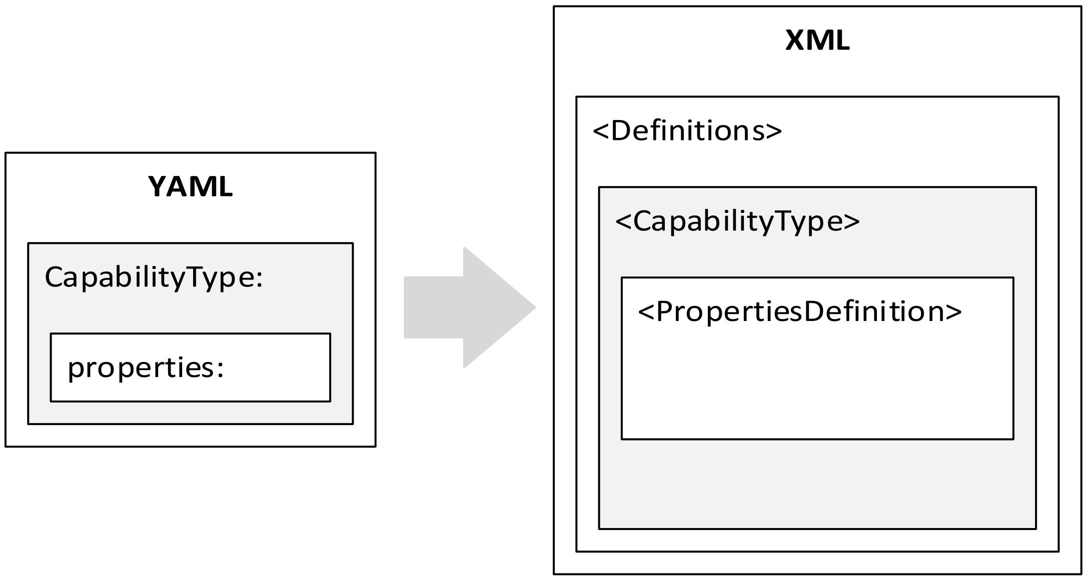
YAML

```
capabilities:
    <capability_name>:
        type:  <capability_type>
```

XML

```
<Capabilities>
    <Capability name="xs:capability_name " id="xs:capability_name " type="xs:capability_type "/>
</Capabilities>
```

6 Capability Type



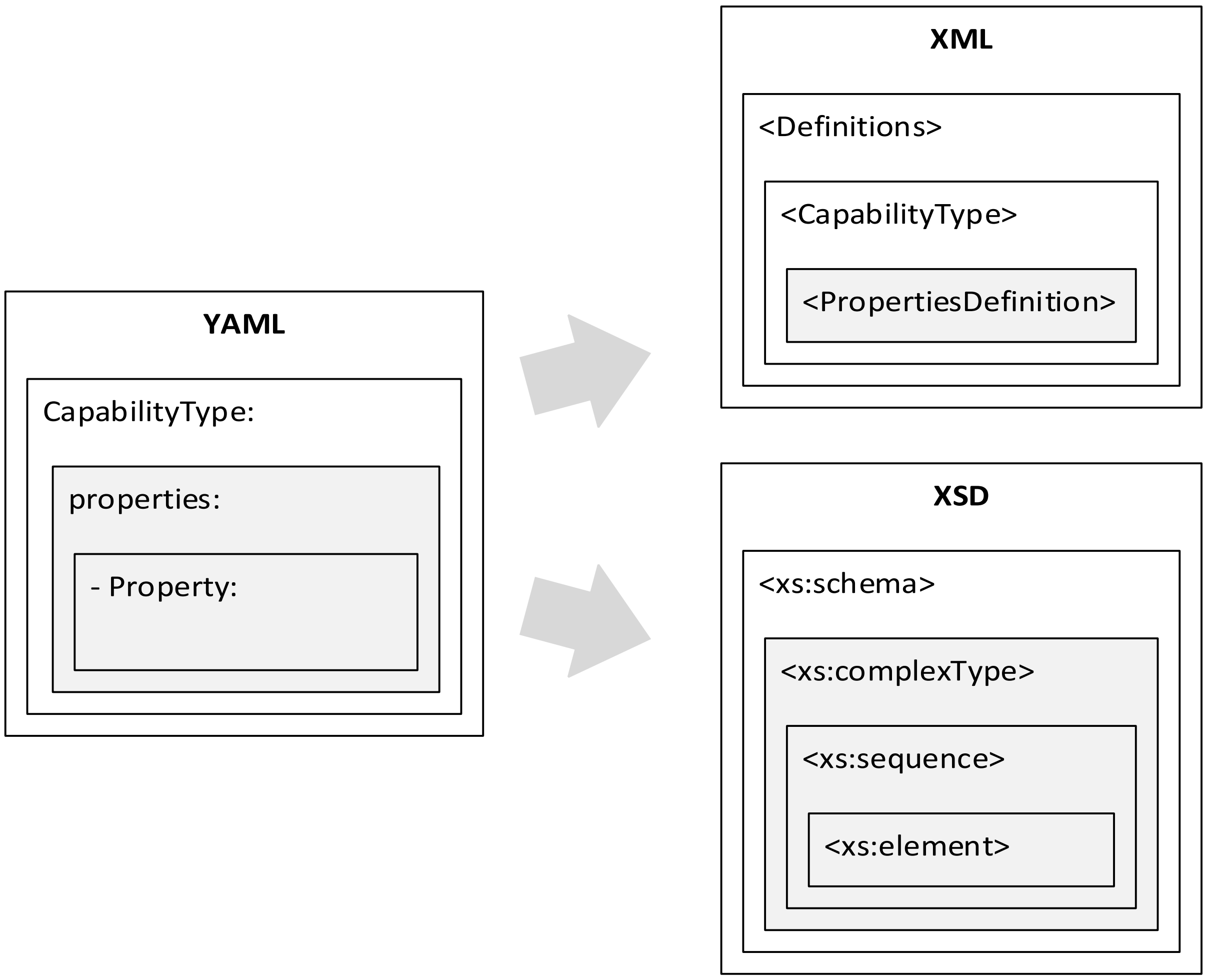
YAML

```
capability_types:
    <capability_type_name>:
        derived_from:  <parent_capability_type_name>
        description:  <capability_description>
         properties:
            <property_definitions>
```

XML

```
<CapabilityType name="xs: capability_type_name ">

    <documentation> capability_description </documentation>

    <DerivedFrom typeRef="xs: parent_capability_type_name "/>

    <PropertiesDefinition type="xs:capability_type_name Properties"/>

</CapabilityType>
```

6.1 Property definition



YAML

```
properties:
    <property_name>:
        type: <property_type>
        description: <property_description>
        default: <default_value>
```

XML

```
<PropertiesDefinition type="xs:[CorrespondingCapabilityType]Properties"/>
```

XSD

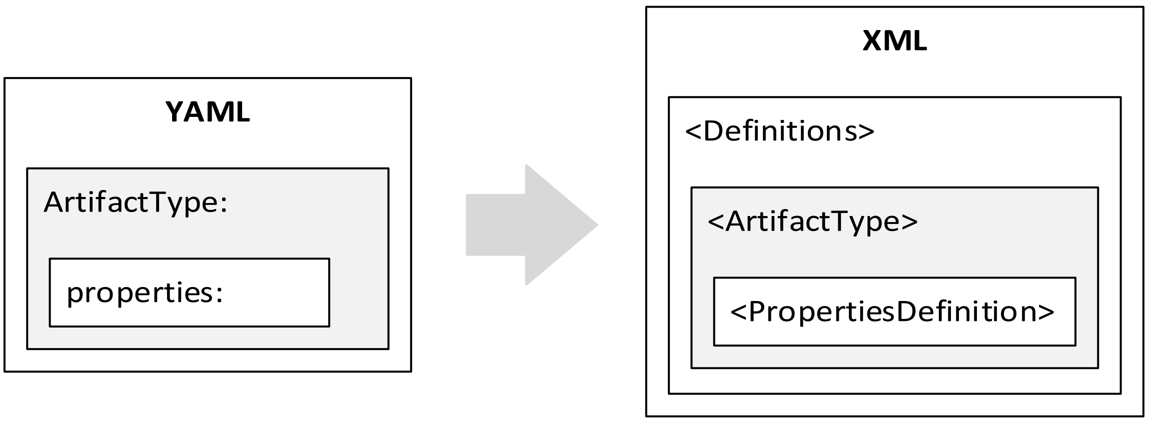
```
<xs:complexType name="t[CorrespondingCapabilityType]Properties">

    <xs:sequence>

        <xs:element name="property_name" type="xs: property_type" />

    </xs:sequence

</xs:complexType>
```

```
<xs:element name="[CorrespondingCapabilityType]Properties "
    type="t[CorrespondingCapabilityType]Properties Properties" />
```

Notes

[CorrespondingCapabilityType]: Name of the CapabilityType the property corresponds to

7 Artifact Type



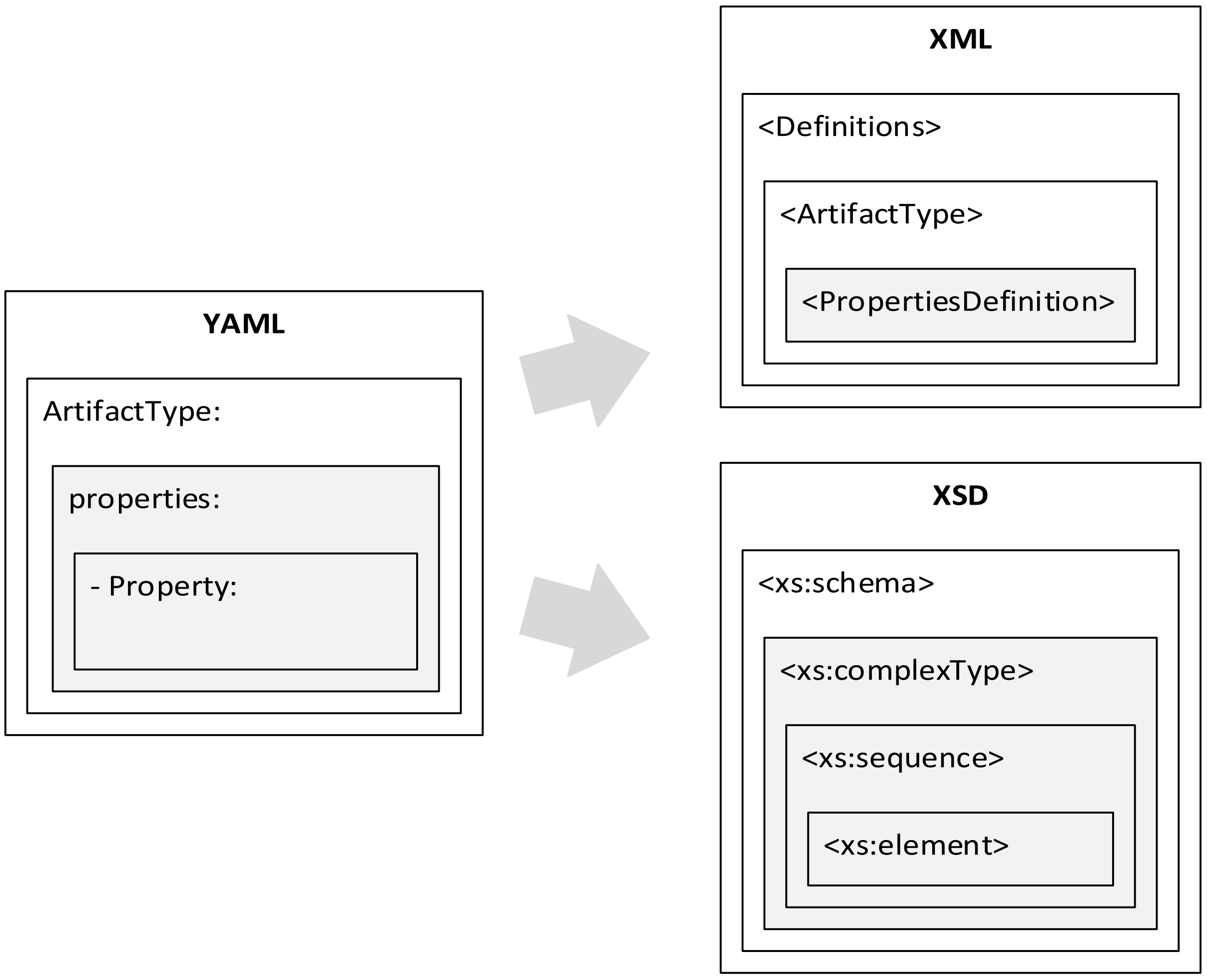
YAML

```
artifact_types:
    <artifact_type_name>:
        derived_from: <parent_artifact_type_name>
        description: <artifact_description>
        mime_type: <mime_type_string>
        file_ext: [<file_extension_1>, ..., <file_extension_n>]
         properties:
            <property_definitions>
```

XML

```
<ArtifactType name="artifact_type_name" targetNamespace="NamespaceURL">

    <PropertiesDefinition type="xs: artifact_type_nameProperties"/>

</ArtifactType>
```

7.1 Property definition



YAML

```
properties:
    <property_name>:
        type: <property_type>
        description: <property_description>
        default: <default_value>
```

XML

```
<PropertiesDefinition type="xs:[CorrespondingArtifactType]Properties"/>
```

XSD

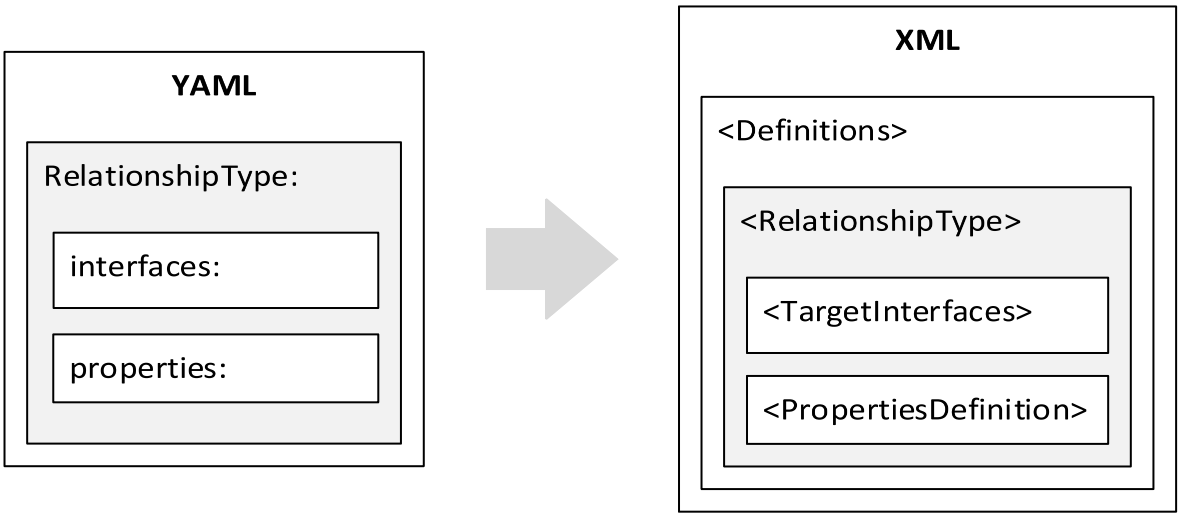
```
<xs:complexType name="t[CorrespondingArtifactType]Properties">

    <xs:sequence>

        <xs:element name="property_name" type="xs: property_type" />

    </xs:sequence

</xs:complexType>
```

```
<xs:element name="[CorrespondingArtifactType]Properties "
    type="t[CorrespondingArtifactType]Properties Properties" />
```

Notes

[CorrespondingArtifactType]: Name of the ArtifactType the property corresponds to

8 Relationship Type



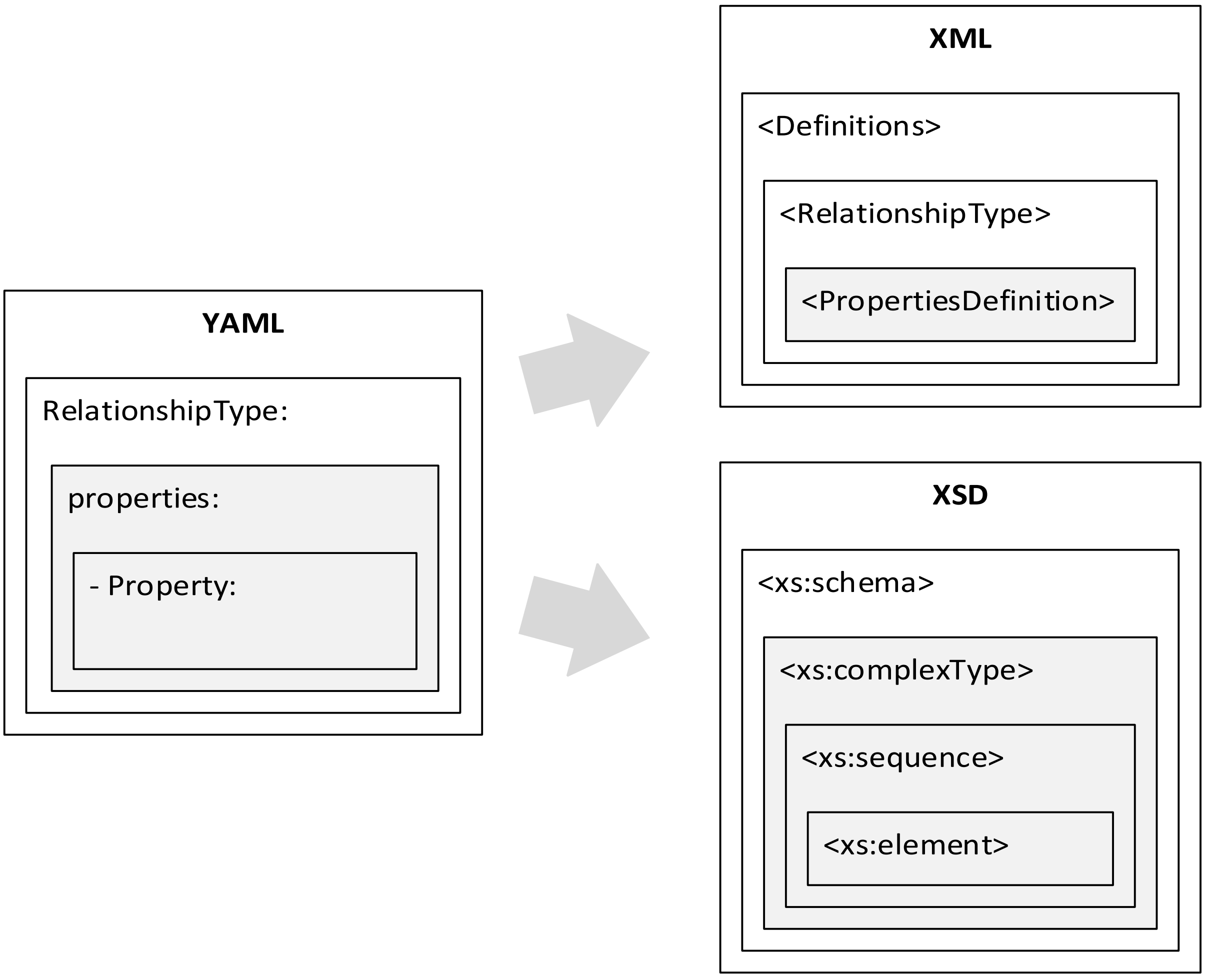
YAML

```
relationship_types:
    <relationship_type_name>:
        derived_from: <parent_relationship_type_name>
        description: <relationship_description>
         properties:
            <property_definitions>
        interfaces:
             <interface_definitions>
        valid_targets: [ <entity_name_or_type>]
```

XML

```
<RelationshipType name="xs: relationship_type_name ">

    <documentation> relationship_description </documentation>

    <DerivedFrom typeRef="xs: parent_relationship_type_name "/>

    <PropertiesDefinition type="xs:[CorrespondingRelationshipType]Properties"/>

    <TargetInterfaces>

        <Interface…/>
    </TargetInterfaces>

    <ValidTarget typeRef="xs: entity_name_or_type "/>

</RelationshipType>
```

8.1 Property definition



YAML

```
properties:
    <property_name>:
        type: <property_type>
        description: <property_description>
        default: <default_value>
```

XML

```
<PropertiesDefinition type="xs:[CorrespondingRelationshipType]Properties"/>
```

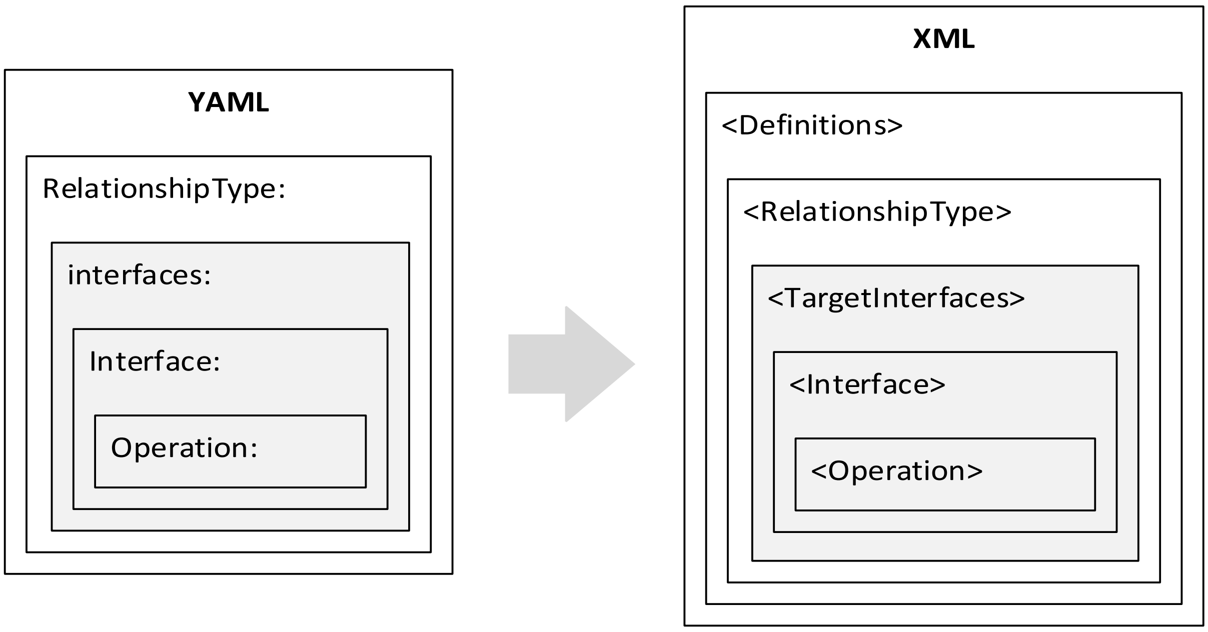
XSD

```
<xs:complexType name="t[CorrespondingRelationshipType]Properties">

    <xs:sequence>

        <xs:element name="property_name" type="xs: property_type" />

    </xs:sequence

</xs:complexType>

<xs:element name="[CorrespondingRelationshipType]Properties "
    type="t[CorrespondingRelationshipType]Properties Properties" />
```

Notes

[CorrespondingRelationshipType]: Name of the RelationshipType the property corresponds to

8.2 Interface definition



YAML

```
interfaces:
    <interface_name>:
        <operation_name>:
            implementation: <implementation_artifact_name>
```

XML

```
<TargetInterfaces>

    <Interface name=" interface_name ">

        <Operation name="operation_name "/>

    </Interface>

</ TargetInterfaces >
```