#### 7.6.6.6 Effect of the *minOccurs* and *maxOccurs* attributes on the mapping

When either or both the *minOccurs* and/or the *maxOccurs* attributes of the *sequence* compositor specify a different value than "1", the following rules shall apply:

a) First, the *sequence* compositor shall be mapped to a TTCN-3 **record** field (as opposed to ignoring it in the previous clauses, when both *minOccurs* and *maxOccurs* equal to 1) with the name "sequence".

b) The encoding instruction "untagged" shall be attached to the field corresponding to *sequence*.

c) The procedures in clause 7.1.4 shall be applied to this **record** field.

NOTE: As the result of applying clause 7.1.4, the type of the field may be changed to **record of record** and in parallel the name of the field may be changed to "sequence\_list".

d) Finally, clause 5.2.2 shall be applied to the name of the resulted field and the field shall be added to the enframing TTCN-3 **record** (see clauses 7.6 and 7.6.6) or **union** field (see clause 7.6.5).

EXAMPLE 1: Mapping an optional *sequence*:

<xsd:complexType name="e36b">

 <xsd:sequence minOccurs="0">

 <xsd:element name="foo" type="xsd:integer"/>

 <xsd:element name="bar" type="xsd:float"/>

 </xsd:sequence>

</xsd:complexType>

// Is mapped to

**type** **record** E36b **{**

 **record** **{**

 XSD.Integer foo,
 XSD.Float bar

 **}** sequence **optional**

**}**

**with** **{**

 **variant** "name as uncapitalized";

 **variant** (sequence) "untagged";
**}**

EXAMPLE 2: Sequence nesting an optional sequence:

<xsd:complexType name="e40c">

 <xsd:sequence>

 <sequence minOccurs="0">

 <xsd:element name="foo" type="xsd:string"/>

 <xsd:element name="bar" type="xsd:string"/>

 </xsd:sequence>

 <xsd:choice>

 <xsd:element name="foo1" type="xsd:string"/>

 <xsd:element name="bar1" type="xsd:string"/>

 </xsd:choice>

 <xsd:element name="ding" type="xsd:string"/>

 </xsd:sequence>

</xsd:complexType>

Will be mapped to e.g.:

**type record** E40c **{**

 **record** **{**

 XSD.String foo,

 XSD.String bar

 **}** sequence **optional**,

 **union** **{**

 XSD.String foo1,

 XSD.String bar1

 **}** choice,

 XSD.String ding

**}**

**with {**

 **variant** "name as uncapitalized";

 **variant**(sequence, choice) "untagged";

**}**

EXAMPLE 3: Sequence nesting a sequence of multiple recurrence:

<xsd:complexType name="e40d">

 <xsd:sequence>

 <xsd:sequence minOccurs="0" maxOccurs="unbounded">

 <xsd:element name="foo" type="xsd:string"/>

 <xsd:element name="bar" type="xsd:string"/>

 </xsd:sequence>

 <xsd:element name="ding" type="xsd:string"/>

 </xsd:sequence>

</xsd:complexType>

Will mapped to e.g.:

**type record** E40d **{**

 **record** **of record** **{**

 XSD.String foo,

 XSD.String bar

 **}** sequence\_list,

 XSD.String ding

**}**

**with {**

 **variant** "name as uncapitalized";

 **variant**(sequence\_list) "untagged";

**}**

EXAMPLE 4: Decoding an empty XML *element* when the optional *sequence* contains optional elements only

<xsd:element name="optionals\_in\_optional">

 <xsd:complexType>

 <xsd:sequence minOccurs="0">

 <xsd:element name="elem1" type="xsd:string" minOccurs="0"/>

 <xsd:element name="elem2" type="xsd:integer" minOccurs="0"/>

 <xsd:element name="elem3" type="xsd:decimal" minOccurs="0"/>

 <xsd:element name="elem4" type="xsd:dateTime" minOccurs="0"/>

 <xsd:element name="elem5" type="xsd:duration" minOccurs="0"/>

 </xsd:sequence>

 </xsd:complexType>

</xsd:element>

Will be mapped to e.g.:

**type** **record** Optionals\_in\_optional

{

 **record** {

 XSD.String elem1 **optional**,

 XSD.Integer elem2 **optional**,

 XSD.Decimal elem3 **optional**,

 XSD.DateTime elem4 **optional**,

 XSD.Duration elem5 **optional**

 } sequence **optional**

}

**with** {

 **variant** "name as uncapitalized";

 **variant** "element";

 **variant** (sequence) "untagged";

};

And an incoming empty element, e.g. <optionals\_in\_optional></optionals\_in\_optional> will be decoded to the short TTCN‑3 value (see clause B.3.21):

{ Optionals\_in\_optional := { sequence := **omit** }};

## B.3.21 Untagged elements

***Syntactical structure(s)***

 **variant** """ untagged """

***Applicable to (TTCN-3)***

Structured type definitions and structured type fields.

***Description***

Without this attribute the names of the structured type fields (as possible modified by a **name as** and **namespace** encoding instructions) or, in case of TTCN-3 type definitions corresponding to global XSD element declarations the name of the TTCN-3 type (as possible modified by a **name as** and **namespace** encoding instructions) are used as the local part of the start and end tags of XML elements at encoding. If the **untagged** encoding instruction is applied to a TTCN-3 type or structured type field, the name of the type or field shall not produce an XML tag when encoding the value of that type or field (in other words, the tag that would be produced without the untagged attribute shall be suppressed during encoding and shall not be expected during decoding). The **untagged** encoding instruction shall only have effect on the TTCN-3 language element to which it is directly applied; e.g. if applied to a structured type, the type itself shall not result a starting and end tag in the encoded XML document but the fields of the structured type shall be encoded using starting and end tags (provided no **untagged** attribute is applied to the fields).

At decoding no XML starting and end tags shall be present in the encoded XML document. In the specific case, when the **untagged** encoding instruction is applied to an optional **record** field, which includes merely optional fields, an empty XML *element* shall be decoded to an omitted enframing **record** field (see example in clause 7.6.6.6).

Shall not be applied to TTCN-3 components generated for XSD attribute elements (neither global nor local).

For typical use cases for extending or restricting simple content see clauses 7.6.1.1 and 7.6.1.2, for optional sequences see clause 7.6.6.6 and for model groups see clause 7.9.

NOTE: Please note, that using the **untagged** encoding instruction in other cases than specified in the present document, may result in an undecodable XML document.