GUIDELINE FOR IMPLEMENTATION OF SCHEMAS

(Version 0.2 - Revision)

INTRODUCTION

The purpose of this guideline is to provide the mechanisms and steps recommended to create XML Schemas compliant to the Standards and fitting the needs of an Office or message.

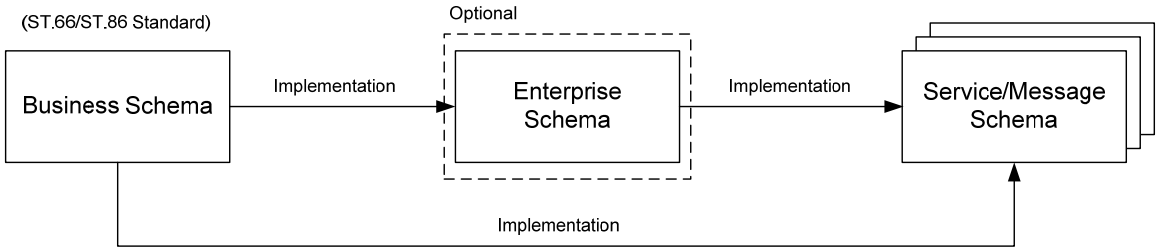
The document is intended for technical audience who are familiarized with XML and W3C's XML Schema languages.

The principal benefit of this guideline is to improve the efficiency and quality of implementing schemas by following a series of steps according to best practices and assuring the compliance with the Standards.

The Standards' schemas are delivered as business schemas providing generic XML building blocks in the trademark and design business domains. This document explains, how to build a schema derived from a Standard's schema restricting it in such a way, that XML instances will still conform to the Standard definition (Part One), and how to add extensions in such a way, that compliance to the Standard is preserved (Part Two).

The building blocks, consisting of types, elements, attributes and enumeration values, are used for schema implementations mainly within 2 levels:

- Enterprise schema: (Usually defined globally for an organisation. This schema contains elements for all data required in the context of the organisation)
- Service/Message schemas: typically strongly reduced versions of the schema, which only comprise information needed in message exchange. They are used to parse the information from a message.



RESTRICTING A SCHEMA

The following paragraph explains the steps which have to be performed in order to restrict the schema to more precisely capture an office's needs. The resulting schema will still be conformant to the Standard.

In the header, the xsi:schemaLocation attribute should be specified in XML instances to provide hints as the physical location of the XML schema which may be used for validation.

Example of XML instance heading:

```
<?xml version="1.0" encoding="UTF-8"?>
<Transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.wipo.int/standards/XMLSchema/trademarks/sampledata sampleSchema.xsd">
   xmlns="http://www.wipo.int/standards/XMLSchema/trademarks/sampledata">
```

The Standard's schema may be restricted in implementation schemas by following the next steps:

Step 1: Create a new schema by including the required types taken from the Standard.

Step 2: Remove the line <xs:any namespace="###other" …> at the end of each complex type. This line is only useful to open the type for extension.

Step 3: Remove the unnecessary element(s), attribute(s) and enumeration value(s).
Note: Mandatory elements and attributes must not be removed. There are only a few.

Step 4: If needed, limit or fix the minimum and maximum occurrence for elements in the instances by using the minOccurs and maxOccurs attributes of the element definition. They must be set within the limits defined for this element in the Standard.

Example:
The definition of the element ContactInformationDetails in the Standard's schema allows the Phone element to occur any

number of times (also 0). Below, we show the office implementation which changes the occurrence of this element in such a way that two phone entries have to be supplied.

Standard's schema definition:

```
<xs:element name="ContactInformationDetails" minOccurs="0">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Phone" type="PhoneType" minOccurs="0" maxOccurs="unbounded"/>
         [...]
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Office implementation:

```
<!-- Optional and unbounded Phone element defined in implementation schema as mandatory and
with a maximum of two -->
<xs:element name="ContactInformationDetails" minOccurs="0">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Phone" type="PhoneType" maxOccurs="2"/>
         [...]
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Step 5: If required, specify more constraints on the value space of data types for elements and attributes with XML facets.
The set of applicable facets depends on the base type:

- Facets for numerals may constrain the lower and upper bounds of the value space.
- Facets for strings may constrain the string length or the lexical space to literals which match a specific pattern given by a regular expression.

Example 1:
This example demonstrates how to change the definition of RepresentationSizeType in such a way, that the height value is restricted to be in the range [0-1024]

Standard's schema definition:

```
<xs:complexType name="RepresentationSizeType">
   <xs:sequence>
          <xs:element name="Height" type="xs:integer" minOccurs="0"/>
      [...]
   </xs:sequence>
</xs:complexType>
```

Office implementation

```
<xs:complexType name="RepresentationSizeType">
   <xs:sequence>
      <xs:element name="Height" minOccurs="0"/>
         <xs:simpleType>
            <xs:restriction base:"xs:integer">
               <xs:minInclusive value="0">
               <xs:maxInclusive value="1024">
            </xs:restriction>
         </xs:simpleType>
      </xs:element>
      [...]
   </xs:sequence>
</xs:complexType>
```

Example 2:
This example shows how the Email element definition can be changed from any string to only those strings following the typical email formatting.

Standard's schema definition:

```
<xs:element name="Email" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

Office implementation:

```
<xs:element name="Email" minOccurs="0" maxOccurs="unbounded">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[\w-\.]+@([\w-]+\.)+[\w-]{2,4}"/>
    </xs:restriction>
  </xs:simpleType>
```

```
</xs:element>
```

Step 6: Extend the set of values for enumeration types according to the application needs. Most of enumerations in the Standard are defined in union with their base type enabling extensions of them in implementation schemas. Remove also the union element from the enumeration type definition.

Example:
The example removes the enumeration values 'Other' and 'Undefined' and adds the value 'Videophone'. It also removes the union with xs:token, preventing values apart from the specified ones for PhoneKindType.

Standard's schema definition:

```
<xs:simpleType name="PhoneKindType">
       <xs:union memberTypes="xs:token">
              <xs:simpleType>
                     <xs:restriction base="xs:token">
                            <xs:enumeration value="Fixed"/>
                            <xs:enumeration value="Mobile Phone"/>
                            <xs:enumeration value="Other"/>
                            <xs:enumeration value="Undefined"/>
                     </xs:restriction>
              </xs:simpleType>
       </xs:union>
</xs:simpleType>
```

Office implementation

```
<xs:simpleType name="PhoneKindType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Fixed"/>
    <xs:enumeration value="Mobile Phone"/>
    <xs:enumeration value="Videophone"/>
  </xs:restriction>
</xs:simpleType>
```

EXTENDING A SCHEMA

The following steps explain how an office implementation can extend the schema in such a way that it still complies with the Standard. The implementation is compliant, if that part of the schema limited to the contents defined in the Standard conforms to the Standard and if all extensions are defined in specific namespaces and according to the design rules of the Standard.

Implementation schemas may extend the Standard with office-specific elements that are defined in specific namespaces. It is recommended to propose elements which are not entirely specific to a particular office for inclusion in future versions of the Standard.

To extend the schema with specific elements and types follow the next steps:

Step 1: Define the required new elements and types in a separate schema file following the Standard design rules.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns="http://www.tm-office.int/XMLSchema"
       targetNamespace="http://www.tm-office.int/XMLSchema"
       version="1.0"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">

  <!-- Define office-specific element -->
  <xs:element name="OfficeSpecificElement" type="OF_OfficeSpecificType"/>

  <!-- Define office-specific types -->
  <xs:complexType name="OF_OfficeSpecificType">
    <xs:sequence>
      <xs:element name="AnotherOfficeSpecificElement" type="xs:string" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Step 2: In the implementation schema, declare the namespace of this file in the <xs:schema> element. As specified in the Standard, the namespace prefix for an Office corresponds to the two-letter office code as given in WIPO Standard ST.3. For other organizations and companies the namespace prefix should be composed of three or four uppercase letters.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.wipo.int/standards/XMLSchema/trademarks/sampledata"
  targetNamespace="http://www.wipo.int/standards/XMLSchema/trademarks/sampledata"
  xmlns:OF="http://www.organization.xx/XMLSchema"
  version="1.0"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
```

Step 3: Import the office-specific schema by using the import element.

Example:

```
<xs:import namespace="http://www.organization.xx/XMLSchema"
schemaLocation="Extension.xsd"/>
```

Step 4: Locate the complex type definition in which the new elements are going to be included. Remove the “any” element at the end of the type definition and introduce references to the new elements by using the “ref” element. Set the allowable minimum and maximum number of occurrences of the elements if different from the default values.

Example:

```
<xs:complexType name="TradeMarkType">
    <xs:sequence>
      ……
      <!-- Remove the xs:any element at the end of type declaration and reference office-
        specific elements  -->
      <xs:element ref="OF:OfficeSpecificElement" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

In the XML instance, also define the namespace of the office-specific schema file in the root element.

Example:

```
<Transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.wipo.int/standards/XMLSchema/trademarks/sampledata sampleSchema.xsd"
  xmlns="http://www.wipo.int/standards/XMLSchema/trademarks/sampledata"
  xmlns:OF="http://www.organization.xx/XMLSchema">
```

In the XML instance, give values to the office-specific elements.

Example:

```
<TradeMark>
  …
  <!-- Give values to office-specific elements -->
  <OF:OfficeSpecificElement>
    <OF:AnotherOfficeSpecificElement>A String Value</OF:AnotherOfficeSpecificElement>
  </OF:OfficeSpecificElement>
</TradeMark>
```